sed命令_Linux sed命令：替换、删除、更新文件中的内容

Python小白入门到精通8天训练营 点我报名
授课内容：Python入门+办公自动化+爬虫 免费领Python纸质书籍+50G学习大礼包+5G面试礼包

C语言中文网推出辅导班啦，包括「C语言辅导班、C++辅导班、算法/数据结构辅导班」，全部都是一对一教学：一对一辅导 + 一对一答疑 + 布置作业 + 项目实践 + 永久学习。QQ在线，随时响应！

sed 是 stream editor 的缩写，中文称之为“流编辑器”。

sed 命令是一个面向行处理的工具，它以“行”为处理单位，针对每一行进行处理，处理后的结果会输出到标准输出（STDOUT）。你会发现 sed 命令是很懂礼貌的一个命令，它不会对读取的文件做任何贸然的修改，而是将内容都输出到标准输出中。

我们来看看 sed 的命令格式：

```
sed command file
```

- command 部分：针对每行的内容所要进行的处理（这部分很重要很重要）。
- file 部分：要处理的文件，如果忽略 file 参数，则 sed 会把标准输入作为处理对象。

sed 的工作原理是什么

刚才我们说了，sed 命令是面向“行”进行处理的，每一次处理一行内容。处理时，sed 会把要处理的行存储在缓冲区中，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。这个缓冲区被称为“模式空间”（pattern space）。

如前面所说，在这个处理过程中，sed 命令并不会对文件本身进行任何更改。

我们来一起看一个最最简单的 sed 命令的例子，让大家对它有个感性的认识。

```
#我们先来看看原文件的内容
[roc@roclinux ~]$ cat roc.txt
test 1
test2
testtest
XtestX
BBtest

#我们想用sed命令来删除文件中带字符“2”的行
[roc@roclinux ~]$ sed '/2/d' roc.txt
test 1
testtest
XtestX
BBtest
```

此例子是利用 sed 来删除 roc.txt 文件中含有字符“2”的行。看到了吧，例子很简单，这个命令的 command 部分是 `/2/d`，而且它是用单引号括起来的。你也一定要学着这样做，用到 sed，别忘了用单引号将 command 部分括起来。

`/2/d` 中的 d 表示删除，意思是说，只要某行内容中含有字符 2，就删掉这一行。（sed 所谓的删除都是在模式空间中执行的，不会真正改动 roc.txt 原文件。）

想用 sed 命令实现 cut 命令的效果

假如我们想实现类似于 cut-d：-f 1/etc/passwd 的效果，也就是以冒号为间隔符提取第 1 个域，用 sed 命令应该怎么操作呢?

```
#先来一起看看/etc/passwd文件的内容
[roc@roclinux ~]$ head -n 5 /etc/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

#我们这回用sed命令来提取文件中每行的第一个域，间隔符是冒号
[roc@roclinux ~]$ head -n 5 /etc/passwd|sed 's/:.*$//'
root
bin
daemon
adm
lp
```

看到了吧，我们将 command 部分指定成了 `'s/: .*$//'`，表示我们要把每一行的第一个冒号到结尾的部分都清空，这样留下的便是第一个冒号前的内容啦。

sed 都有哪些好用的选项

说到 sed 命令的选项，就不得不提 `-n` 选项，想把这个选项介绍清楚，还是要费一些脑子和笔墨的。

前面提到，sed 会将模式空间里的行经过处理后输出到标准输出，这是默认的处理方式。也就是说，除非你使用“d”来删除此行，否则经过“模式空间”处理的行都是会被输出到标准输出（屏幕）上的。我们一起来看下面的例子：

```
#还是先来看看原文件的内容
[roc@roclinux ~]$ cat roc.txt
1
2
3
4
5

#仔细看，输出中出现了两个“4”
[roc@roclinux ~]$ sed ‘/4/p’ roc.txt
1
2
3
4
4
5
```

看，所有的原始文件内容都被输出来了，而且含有字符4的行被输出了两遍。

但这就是 sed 命令的工作原理，它会不问青红皂白地把经过处理的行先输出出来，然后再执行后面的动作。（在这里我们设定了 p，表示打印此行。）这明显不符合我们的初衷，我们只是想让 sed 命令找到含有 4 的行再输出。

这时候，不妨加上 `-n` 选项试一试，你会发现，结果变得如你所愿了。

```
[roc@roclinux ~]$ sed -n '/4/p' roc.txt
4
```

`-n` 选项会很严肃地警告 sed 命令：除非是明确表明要输出的行，否则不要给我胡乱输出。`-n` 选项经常和 p 配合使用，其含义就是，输出那些匹配的行。

command 部分花样很多

还记得我们在前面介绍过的，sed 命令的命令格式是这样的：

```
$ sed command file
```

其中，command 部分是 sed 命令的精髓，对 command 部分的掌握程度决定了你是不是 sed 高手。

command 部分可以分为两块知识：一块是范围设定，一块是动作处理。

范围设定，可以采用两种不同的方式来表达：
1. 指定行数：比如‘3，5’表示第 3、第 4 和第 5行；而‘5，$’表示第 5 行至文件最后一行。
2. 模式匹配：比如/^[^dD]/表示匹配行首不是以 d 或 D 开头的行。

而动作处理部分，会提供很丰富的动作供你选择，下面就来介绍几个最常用的动作吧：
- d：表示删除行。
- p：打印该行。
- r：读取指定文件的内容。
- w：写入指定文件。
- a：在下面插入新行新内容。

展示 sed 实力的时候到了

事实胜于雄辩，我们打算通过四个例子，让大家感受到 sed 命令真的是一位“实力派”选手。

第一个例子，我们来显示 test 文件的第 10 行到第 20 行的内容：

```
#我们采用了刚才提到的指定行区间的方法
[roc@roclinux ~]$ sed -n '10,20p' test
```

第二个例子，我们尝试将所有以 d 或 D 开头的行里的所有小写 x 字符变为大写 X 字符：

```
[roc@roclinux ~]$ sed '/^[dD]/s/x/X/g' test
```

这个用法值得一讲，我们在 command 部分采用了 /AA/s/BB/CC/g 的语法格式，这表示我们要匹配到文件中带有 AA 的行，并且将这些行中所有的 BB 替换成 CC。

第三个例子，我们要删除每行最后的两个字符：

```
#点号表示一个单个字符，两个点号就表示两个单个字符
[roc@roclinux ~]$ sed 's/..$//' test
```

有人可能会问，用 sed‘/..$/d’test 为什么不行呢，d 不是表示删除么？用 d 是不行的，这是因为 d 表示删除整行内容，而非字符。`'/..$/d'` 表示的是匹配所有末尾含有两个字符的行，然后删除这一整行内容，显然这和我们的初衷是相悖的。

第四个例子，我们希望删除每一行的前两个字符：

```
[roc@roclinux ~]$ sed 's/..//' test
```

通过这四个例子，相信大家对 sed 命令的最常见用法已经有了很直观的认识了，不妨在自己的实际工作中把这些知识用起来吧。

& 符号的妙用

我们仍然通过一个场景来讲解这个知识点。

```
#按照惯例，先展示文件的内容
[roc@roclinux ~]$ cat mysed.txt
Beijing
London

#我们使用到了&符号，大家试着猜一猜它的作用
[roc@roclinux ~]$ sed 's/B.*/&2008/' mysed.txt
Beijing2008
London
```

不卖关子，答案揭晓啦，这个命令的作用是将包含‘B.*’的字符串后面加上 2008 四个字符。这个命令中我们用到了 & 字符，在 sed 命令中，它表示的是“之前被匹配的部分”，在我们的例子中当然就是 Beijing 啦！

我们再通过一个例子强化一下大家对&符号的理解：

```
#这个例子或许更易理解
[roc@roclinux 20160229]$ sed 's/Bei/&2008/' mysed.txt
Bei2008jing
London
```

sed 中的括号有深意

在 sed 命令中，其实小括号‘（）’也是有深意的。我们开门见山，通过一个例子，让大家见识一下小括号的威力：

```
[roc@roclinux ~]$ echo "hello world" | sed 's/\(hello\).*/world \1/'
world hello
```

我们看到，原本是“hello world”，经过 sed 的处理，输出变成了“world hello”。

这个例子中就用到了小括号的知识，我们称之为“sed 的预存储技术”，也就是命令中被“（”和“）”括起来的内容会被依次暂存起来，存储到 \1、\2…里面。这样你就可以使用‘\N’形式来调用这些预存储的内容了。

来继续看一个例子，我们希望只在每行的第一个和最后一个 Beijing 后面加上 2008 字符串，言下之意就是，除了每行的第一个和最后一个 2008 之外，这一行中间出现的 Beijing 后面就不要加 2008 啦。这个需求，真的是很复杂很个性化，但 sed 命令仍然可以很好地满足：

```
#先看下文件内容，第一行中出现了4个Beijing
[roc@roclinux ~]$ cat mysed.txt
Beijing Beijing Beijing Beijing
London London London London

#效果实现啦，可是，命令真的好复杂
[roc@roclinux ~]$ sed 's/\(Beijing\)\(.*\)\(Beijing\)/\12008\2\32008/' mysed.txt
Beijing2008 Beijing Beijing Beijing2008
London London London London
```

这个命令确实足够复杂，用流行的语言说就是“足够虐心”。这个例子中我们再次使用了预存储技术，存储了三项内容，分别代表第一个 Beijing、中间的内容、最后的 Beijing。而针对 `\1` 和 `\3`，我们在其后面追加了 2008 这个字符串。

更聪明的定位行范围

实践是学习知识最好的方法，相信大家看了这个例子后，就明白如何更好地定位行范围了：

```
#文件内容展示一下
[roc@roclinux ~]$ cat mysed.txt
Beijing 2003
Beijing 2004
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008
Beijing 2007

#我们想展示匹配了2005的行和2007的行之间的内容
[roc@roclinux ~]$ sed -n ‘/2005/,/2007/p’ mysed.txt
Beijing 2005
Beijing 2006
Beijing 2007
```

我们使用 /2005/ 来匹配行范围的首行，用 /2008/ 来匹配行范围的尾行。可以看到，在匹配尾行时，只要遇到第一个符合要求的行，就会停止，而不会再继续向后匹配了。所以，sed 命令只是匹配到了第一个 2007，并没有匹配到第二个 2007。

用 -e 选项来设置多个 command

还记得 command 部分吧，现在有一个好消息要告诉你，那就是 sed 命令可以包含不只一个 command。如果要包含多个 command，只需在每个 command 前面分别加上一个-e选项即可。

```
#我们通过2个-e选项设置了两个command
[roc@roclinux ~]$ sed -n -e ‘1,2p’ -e ‘4p’ mysed.txt
Beijing 2003
Beijing 2004
Beijing 2006
```

有一个地方值得大家注意，那就是 `-e` 选项的后面要立即接 command 内容，不允许再夹杂其他选项。

`-e` 选项支持设置多个 command，这原本是一件好事情，让我们可以更方便地实现一些替换效果。但是，这也给我们带来了幸福的烦恼，假如我们设定了很多个 command，那它们的执行顺序是怎样的呢?

如果这一点不搞明白，`-e` 选项带来的或许只有混乱而非便捷。我们来一起看看下面的例子：

```
#先看看文件的内容
[roc@roclinux ~]$ cat mysed.txt
Beijing 2003
Beijing 2004
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008

#我们设置了两个command
[roc@roclinux ~]$ sed -e ‘s/Beijing/London/g’ -e ‘/Beijing/d’ mysed.txt
London 2003
London 2004
London 2005
London 2006
London 2007
London 2008
```

前一个 command 表示将 Beijing 替换为 London，而后一个 command 表示要删除包含了 Beijing 字符串的行，但是最后的结果却是输出了所有行，并没有发现被删除的行。这是因为第一个 command 已经将 Beijing 都替换成了 London，所以怪第二个 command 找不到 Beijing 了。

我们再来把上面例子中的 command 颠倒一下位置，看看效果如何：

```
#我们先指定删除动作，再指定替换动作
[roc@roclinux 20160229]$ sed -e '/Beijing/d' -e 's/Beijing/London/g' mysed.txt
[roc@roclinux 20160229]$
```

通过这两个小例子，我们可以很清晰地看到，多个 command 之间，是按照在命令中的先后顺序来执行的。

用 -f 选项设定 command 文件

如果你的 sed 命令的 command 部分很长，那么可以将内容写到一个单独的文件中，然后使用 `-f` 选项来指定这个文件作为我们 sed 命令的 command 部分：

```
#这是我们事先写好的文件
[roc@roclinux ~]$ cat callsed
/2004/,/2006/p

#我们用-f选项来指定command文件
[roc@roclinux ~]$ sed -n -f callsed mysed.txt
Beijing 2004
Beijing 2005
Beijing 2006
```

很好理解吧，`-f` 选项并不难，而且我会经常使用，因为一些比较常用的匹配规则，我都会存到单独的文件中，不用再费脑子记忆啦。

内容插入

sed 命令远比你想象的要强大，它不仅可以处理本行内容，还可以在这一行的后面插入内容：

```
#我们将要插入的内容保存到一个单独的文件中
[roc@roclinux ~]$ cat ins.txt
====China====

#展示一下我们要处理的文件
[roc@roclinux ~]$ cat mysed.txt
Beijing 2003
Beijing 2004
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008

#看，我们使用r来实现插入
[roc@roclinux ~]$ sed ‘/2005/r ins.txt’ mysed.txt
Beijing 2003
Beijing 2004
Beijing 2005
====China====
Beijing 2006
Beijing 2007
Beijing 2008
```

通过效果可以看出来，我们在文件中的含有 2005 字符串的行的下面一行插入了 ins.txt 文件的内容。

除了可以通过指定文件来插入外，其实还可以使用 a\ 在特定行的“下面”插入特定内容：

```
#文件内容
[roc@roclinux ~]$ cat new.txt
Beijing 2004
Beijing 2005
Beijing 2006

#我们希望在2004的下一行插入China
[roc@roclinux ~]$ sed ‘/2004/a\China’ mysed.txt
Beijing 2003
Beijing 2004
China
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008
```

可以看到，我们只要使用 `a\` 然后加上要插入的内容就可以轻松实现啦。

有些同学会问，既然可以在一行的下面插入内容，那是否可以在一行的上面插入内容呢？答案是当然可以，那就是使用 `i\` 动作：

```
[roc@roclinux ~]$ sed ‘/2004/i\China’ mysed.txt
Beijing 2003
China
Beijing 2004
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008
```

说一说 y 动作

在介绍 y 动作之前，我们先来看看它可以实现什么效果：

```
#原文件内容
[roc@roclinux ~]$ cat mysed.txt
Beijing 2003
Beijing 2004
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008

#y就是按照字符顺序，实现前后的替换
[roc@roclinux 20160229]$ sed 'y/ei/ie/' mysed.txt
Biejeng 2003
Biejeng 2004
Biejeng 2005
Biejeng 2006
Biejeng 2007
Biejeng 2008
```

这个例子其实已经很清楚了，我们希望将所有的 e 和 i 互换。

有些同学会问，`y///` 和 `s///` 有什么区别呢？主要有以下两点：
- y 的语法格式是 y/source/dest/，表示将 source 中的字符对位替换为 dest 中的字符。而 s 的语法格式是 s/regexp/replacement/，表示通过正则匹配到的内容替换为 replacement 部分。
- y 只是简单的逐字替换，没有很多花样。s 支持 & 符号和预存储等特性，可以实现更多灵活的替换效果。

这时，一些 GEEK 或许会想到一种情况，那就是 y/ee/ei/ 会产生什么效果呢？因为这里面出现了两个同样的字符，我们还是通过例子来看一下：

```
[roc@roclinux 20160229]$ sed 'y/ee/ie/' mysed.txt
Biijing 2003
Biijing 2004
Biijing 2005
Biijing 2006
Biijing 2007
Biijing 2008
```

看到了吧，如果 source 部分出现了重复的字符，则只有第一次出现的对位替换会产生效果，后面的并不会起作用。或许下面这个例子更加清晰些：

```
#原文件内容
[roc@roclinux ~]$ cat mysed.txt
Beijing 2003
Beijing 2004
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008

#iji到iba的替换中，只有j到b起到了效果
[roc@roclinux 20160229]$ sed 'y/iji/iba/' mysed.txt
Beibing 2003
Beibing 2004
Beibing 2005
Beibing 2006
Beibing 2007
Beibing 2008
```

通过 n 动作来控制行的下移

有时我们希望实现隔行处理的效果，比如只需对偶数行做某个替换，这时候，我们就需要 n 动作的帮忙啦：

```
#原文件内容
[roc@roclinux ~]$ cat mysed.txt
Beijing 2003
Beijing 2004
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008

#我们同时使用了n动作和y动作
[roc@roclinux ~]$ sed ‘/200/{n;y/eijng/EIJNG/;}’ mysed.txt
Beijing 2003
BEIJING 2004
Beijing 2005
BEIJING 2006
Beijing 2007
BEIJING 2008
```

你会发现，大写的 BEIJING 是隔行出现的。这就是 `n` 选项在起作用，它的真实作用是将下一行内容放到处理缓存中，这样，就让当前这一行躲避开了替换动作，是不是有点像小时候玩游戏时通过左右键躲避开 BOSS 的大招，哈哈。

将指定行写入到特定文件中

文章要进入尾声了，我们最后再教大家一个非常实用的动作，那就是 w 动作，它可以将匹配到的内容写入到另一个文件中，即用来实现内容的筛选与保存：

```
#将包含2004、2005、2006的行保存到new.txt文件中
[roc@roclinux ~]$ sed ‘/200[4-6]/w new.txt’ mysed.txt
Beijing 2003
Beijing 2004
Beijing 2005
Beijing 2006
Beijing 2007
Beijing 2008

#我们要的内容已经乖乖到碗里来了
[roc@roclinux ~]$ cat new.txt
Beijing 2004
Beijing 2005
Beijing 2006
```

好了，sed 的流艺术系列到这里就全部结束啦，相信你对 sed 已经有了初步的认识，可以在实战中露两手了。

编程帮，一个分享编程知识的公众号。跟着站长一起学习，每天都有进步。
通俗易懂，深入浅出，一篇文章只讲一个知识点。
文章不深奥，不需要钻研，在公交、在地铁、在厕所都可以阅读，随时随地涨姿势。
文章不涉及代码，不烧脑细胞，人人都可以学习。
当你决定关注「编程帮」，你已然超越了90%的程序员！

微信扫描二维码关注

所有教程

C语言入门 | C语言编译器 | C语言项目案例 | 数据结构 | 多线程 | 链接库 | 算法 | C++ | STL | C++11 | socket | GCC
GDB | Makefile | OpenCV | Qt教程 | Unity 3D | UE4 | 游戏引擎 | Python | Python并发编程 | TensorFlow | Django
NumPy | 机器学习算法 | Python爬虫 | Pandas | Matplotlib | Linux | Shell | Java教程 | 设计模式 | Java Swing
Servlet教程 | JSP教程 | JSTL | Struts2 | Maven | Nexus | Spring | Spring MVC | Spring Boot | Spring Cloud | Hibernate
Mybatis | SQL | MySQL教程 | MySQL函数 | NoSQL | Redis | Redis常用命令手册 | HBase | MongoDB | Go语言 | C#
MATLAB | JavaScript | Bootstrap | HTML | CSS | PHP | 汇编语言 | TCP/IP | vi命令 | Android教程 | 区块链 | Docker
大数据 | 云计算 | 推荐阅读 | 编程笔记 | 资源下载 | VIP视频 | 一对一答疑 | 关于我们

精美而实用的网站，分享优质编程教程，帮助有志青年。千锤百炼，只为大作；精益求精，处处斟酌；这种教程，看一眼就倾心。
关于网站 | 关于站长 | 如何完成一部教程 | 联系我们 | 网站地图

Copyright ©2012-2020 biancheng.net, 陕ICP备15000209号