第10章　sed 用 法 介 绍

sed是一个非交互性文本流编辑器。它编辑文件或标准输入导出的文本拷贝。标准输入可能是来自键盘、文件重定向、字符串或变量，或者是一个管道的文本。 sed可以做些什么呢？别忘了， Vi也是一个文本编辑器。 sed可以随意编辑小或大的文件，有许多 sed命令用来编辑、删除，并允许做这项工作时不在现场。 sed一次性处理所有改变，因而变得很有效，对用户来讲，最重要的是节省了时间。

本章内容有：

- 抽取域。
- 匹配正则表达式。
- 比较域。
- 增加、附加、替换。
- 基本的sed命令和一行脚本。

可以在命令行输入sed命令，也可以在一个文件中写入命令，然后调用 sed，这与awk基本相同。使用sed需要记住的一个重要事实是，无论命令是什么， sed并不与初始化文件打交道，它操作的只是一个拷贝，然后所有的改动如果没有重定向到一个文件，将输出到屏幕。

因为sed是一个非交互性编辑器，必须通过行号或正则表达式指定要改变的文本行。

本章介绍sed用法和功能。本章大多编写的是一行命令和小脚本。这样做可以慢慢加深对sed用法的了解，取得宝贵的经验，以便最终自己编出大的复杂 sed脚本。

和grep与awk一样， sed是一种重要的文本过滤工具，或者使用一行命令或者使用管道与grep与awk相结合。

10.1　sed怎样读取数据

sed从文件的一个文本行或从标准输入的几种格式中读取数据，将之拷贝到一个编辑缓冲区，然后读命令行或脚本的第一条命令，并使用这些命令查找模式或定位行号编辑它。重复此过程直到命令结束。

10.2　调用sed

调用sed有三种方式：在命令行键入命令；将 sed命令插入脚本文件，然后调用 sed；将sed命令插入脚本文件，并使sed脚本可执行。

使用sed命令行格式为：

sed [选项] **sed**命令　输入文件。

记住在命令行使用sed命令时，实际命令要加单引号。 sed也允许加双引号。

使用sed脚本文件，格式为：

sed [选项]　**-f**　**sed**脚本文件　输入文件

要使用第一行具有sed命令解释器的sed脚本文件，其格式为：

sed脚本文件 [选项] 输入文件

不管是使用shell命令行方式或脚本文件方式，如果没有指定输入文件， sed从标准输入中接受输入，一般是键盘或重定向结果。

sed选项如下：

n 不打印；sed不写编辑行到标准输出，缺省为打印所有行（编辑和未编辑）。p命令可以用来打印编辑行。

c 下一命令是编辑命令。使用多项编辑时加入此选项。如果只用到一条 sed命令，此选项无用，但指定它也没有关系。

f 如果正在调用sed脚本文件，使用此选项。此选项通知 sed一个脚本文件支持所有的 sed命令，例如：sed -f myscript.sed input_file，这里myscript.sed即为支持sed命令的文件。

10.2.1 保存sed输出

由于不接触初始化文件，如果想要保存改动内容，简单地将所有输出重定向到一个文件即可。下面的例子重定向 sed命令的所有输出至文件‘ myoutfile ’，当对结果很满意时使用这种方法。

```
$ sed 'some-sed-commands' input-file > myoutfile
```

10.2.2 使用sed在文件中查询文本的方式

sed浏览输入文件时，缺省从第一行开始，有两种方式定位文本：

1) 使用行号，可以是一个简单数字，或是一个行号范围。

2) 使用正则表达式，怎样构建这些模式请参见第 7章。

表10-1给出使用sed定位文本的一些方式。

表10-1 使用sed在文件中定位文本的方式

x	x为一行号，如1
x,y	表示行号范围从x到y，如2，5表示从第2行到第5行
/pattern/	查询包含模式的行。例如/disk/或/[a-z]/
/pattern/pattern/	查询包含两个模式的行。例如/disk/disks/
pattern/,x	在给定行号上查询包含模式的行。如/ribbon/,3
x,/pattern/	通过行号和模式查询匹配行。 3./vdu/
x,y!	查询不包含指定行号x和y的行。 1,2!

10.2.3 基本sed编辑命令

表10-2列出了Sed的编辑命令。

表10-2 sed编辑命令

p	打印匹配行
=	显示文件行号
a\	在定位行号后附加新文本信息
i\	在定位行号后插入新文本信息
d	删除定位行
c\	用新文本替换定位文本

(续)

s	使用替换模式替换相应模式
r	从另一个文件中读文本
w	写文本到一个文件
q	第一个模式匹配完成后推出或立即推出
l	显示与八进制ASCII代码等价的控制字符
{}	在定位行执行的命令组
n	从另一个文件中读文本下一行，并附加在下一行
g	将模式2粘贴到/pattern n/
y	传送字符
n	延续到下一输入行；允许跨行的模式匹配语句

如果不特别声明，sed例子中使用下述文本文件quote.txt。

```
$ pg quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

10.3 sed和正则表达式

sed识别任何基本正则表达式和模式及其行匹配规则。记住规则之一是：如果要定位一特殊字符，必须使用（\）屏蔽其特殊含义，如有必要请参照第 7章正则表达式。第7章使用的所有正则表达式在sed中都是合法的。

10.4 基本sed编程举例

下面通过例子实际检验一下sed的编辑功能。

10.4.1 使用p（rint）显示行

print命令格式为[address[，address]P。显示文本行必须提供sed命令行号。

```
$ sed '2p' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

错误在哪儿？原意只打印第二行，但是却打印了文件中所有行，为此需使用 -n选项，显示打印定位（匹配）行。

```
$ sed -n '2p' quote.txt
It was an evening of splendid music and company.
```

10.4.2 打印范围

可以指定行的范围，现打印1到3行，用逗号分隔行号。

```
$ sed -n '1,3p' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
```

10.4.3 打印模式

假定要匹配单词Neave，并打印此行，方法如下。使用模式/pattern/格式，这里为/Neave/。

```
$ sed -n '/Neave/'p quote.txt
The local nurse Miss P.Neave was in attendance.
```

10.4.4 使用模式和行号进行查询

为编辑某个单词浏览一个文件时， sed返回包含指定单词的许多行。怎样使返回结果更精确以满足模式匹配呢？可以将行号和模式结合使用。下面这个例子，假定要改动文件 quote.txt 最后一行中的单词the，使用sed查询the，返回两行：

```
$ sed -n '/The/'p quote.txt
The honeysuckle band played all night long for only $90.
The local nurse Miss P.Neave was in attendance.
```

使用模式与行号的混合方式可以剔除第一行，格式为 line_number,/pattern/。逗号用来分隔行号与模式开始部分。为达到预期结果，使用 4,/the/。意即只在第四行查询模式 the，命令如下：

```
$ sed -n '4,/The/'p quote.txt
The local nurse Miss P.Neave was in attendance.
```

10.4.5 匹配元字符

匹配元字符$前，必须使用反斜线\屏蔽其特殊含义。模式为 /\$/ p。

```
$ sed -n '/\$/'p quote.txt
The honeysuckle band played all night long for only $90.
```

10.4.6 显示整个文件

要打印整个文件，只需将行范围设为第一行到最后一行 1,$。$意为最后一行。

```
$ sed -n '1,$p' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

10.4.7 任意字符

匹配任意字母，后跟任意字母的 0次或多次重复，并以ing结尾，模式为/.*ing/。可以使用这个模式查询以ing结尾的任意单词。

```
$ sed -n '/.*ing/'p quote.txt
It was an evening of splendid music and company.
```

10.4.8 首行

要打印文件第一行，使用行号：

```
$ sed -n '1p' quote.txt
The honeysuckle band played all night long for only $90.
```

10.4.9 最后一行

要打印最后一行，使用$。$是代表最后一行的元字符。

```
$ sed -n '$p' quote.txt
The local nurse Miss P.Neave was in attendance.
```

10.4.10 打印行号

要打印行号，使用等号=。打印模式匹配的行号，使用格式/pattern/=。

```
$ sed  -e '/music/=' quote.txt
The honeysuckle band played all night long for only $90.
2
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

整个文件都打印出来，并且匹配行打印了行号。如果只关心实际行号，使用 -e选项。

```
$ sed  -n '/music/=' quote.txt
2
```

如果只打印行号及匹配行，必须使用两个 sed命令，并使用e选项。第一个命令打印模式匹配行，第二个使用=选项打印行号，格式为sed -n -e /pattern/p -e /pattern/=。

```
$ sed -n -e '/music/p' -e '/music/=' quote.txt
It was an evening of splendid music and company.
2
```

10.4.11 附加文本

要附加文本，使用符号a\，可以将指定文本一行或多行附加到指定行。如果不指定文本放置位置，sed缺省放在每一行后面。附加文本时不能指定范围，只允许一个地址模式。文本附加操作时，结果输出在标准输出上。注意它不能被编辑，因为 sed执行时，首先将文件的一行文本拷贝至缓冲区，在这里 sed编辑命令执行所有操作（不是在初始文件上），因为文本直接输出到标准输出，sed并无拷贝。

要想在附加操作后编辑文本，必须保存文件，然后运行另一个 sed命令编辑它。这时文件的内容又被移至缓冲区。

附加操作格式如下：

```
[address]a\
text\
text\
...
text
```

地址指定一个模式或行号，定位新文本附加位置。 a\ 通知sed对a\后的文本进行实际附加操作。观察格式，注意每一行后面有一斜划线，这个斜划线代表换行。 sed执行到这儿，将创建一新行，然后插入下一文本行。最后一行不加斜划线， sed假定这是附加命令结尾。

当附加或插入文本或键入几个 sed命令时，可以利用辅助的 shell提示符以输入多行命令。这里没有这样做，因为可以留给使用者自己编写，并且在一个脚本文件中写这样的语句更适宜。现在马上讲述sed脚本文件。另外，脚本可以加入空行和注释行以增加可读性。

10.4.12 创建sed脚本文件

要创建脚本文件append.sed，输入下列命令：

```
$ pg append.sed
#!/bin/sed -f
/company/ a\
Then suddenly it happened.
```

保存它，增加可执行权限：

```
$ chmod u+x append.sed
```

运行，

```
$ append.sed quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Then suddenly it happened.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

如果返回‘ file not found ’，试在脚本前加入.\。

现在查看其具体功能。第一行是sed命令解释行。脚本在这一行查找sed以运行命令，这里定位在/bin。

第二行以/company/开始，这是附加操作起始位置。 a\通知sed这是一个附加操作，首先应插入一个新行。第三行是附加操作要加入到拷贝的实际文本。

输出显示附加结果。如果要保存输出，重定向到一个文件。

10.4.13 插入文本

插入命令类似于附加命令，只是在指定行前面插入。和附加命令一样，它也只接受一个地址。下面是插入命令的一般格式。地址是匹配模式或行号：

下面例子在以attendance结尾的行前插入文本utter confusion followed。

运行结果是：

也可以使用行号指定文本插入位置，插入位置在模式或指定行号 4之前。脚本如下：

```
#!/bin/sed -f
4 i\
Utter confusion followed.
```

10.4.14 修改文本

修改命令将在匹配模式空间的指定行用新文本加以替代，格式如下：

```
[address[,address] c\
text\
text\
text\
...
text
```

将第一行The honeysuckle band played all night long for only $90替换为The office Dibble band played well。首先要匹配第一行的任何部分，可使用模式‘ /Honeysuckle/ ’。sed脚本文件为change.sed。内容如下：

```
$ pg change.sed
#!/bin/sed -f
# change.sed
/honeysuckle/ c\
The Office Dibble band played well.
```

运行它，不要忘了给脚本增加可执行权限。 chmod u+x change.sed。

```
$ change.sed quote.txt
The Office Dibble band played well.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

像插入动作一样，可以使用行号代替模式，两种方式完成相同的功能。

```
#!/bin/sed -f
3 c\
The Office Dibble band played well.
```

可以对同一个脚本中的相同文件进行修改、附加、插入三种动作匹配和混合操作。

下面是一个带有注释的脚本例子。

```
$ pg mix.sed
#!/bin/sed -f
# this is a comment line, all comment starts with a #
# name: mix.sed


# this is the change on line 1
1 c\
The Dibble band were grooving.
# let's now insert a line
/evening/ i\
They played some great tunes.
# change the last line, a $ means last line
$ c\
Nurse Neave was too tipsy to help.
```

```
# stick in a new line before the last line
3 a\
Where was the nurse to help?
```

运行它，结果如下：

```
$ mix.sed quote.txt
The Dibble band were grooving.
They played some great tunes.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
Where was the nurse to help?
Nurse Neave was too tipsy to help.
```

10.4.15 删除文本

sed删除文本格式：

```
[address[, address]]d
```

地址可以是行的范围或模式，让我们看几个例子。

删除第一行；1d意为删除第一行。

```
$ sed '1d' quote.txt
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

删除第一到第三行：

```
$ sed '1,3d' quote.txt
The local nurse Miss P.Neave was in attendance.
```

删除最后一行：

```
$ sed '$d' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
```

也可以使用正则表达式进行删除操作。下面的例子删除包含文本‘ Neave ’的行。

```
$ sed '/Neave/d' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
```

10.4.16 替换文本

替换命令用替换模式替换指定模式，格式为：

```
[address[, address]] s/ pattern-to-find /replacement-pattern/[g p w n]
```

s选项通知sed这是一个替换操作，并查询 pattern-to-find，成功后用replacement-pattern替换它。

替换选项如下：

g 缺省情况下只替换第一次出现模式，使用 g选项替换全局所有出现模式。

p 缺省sed将所有被替换行写入标准输出，加 p选项将使-n选项无效。 -n选项不打印输出结果。

w 文件名 使用此选项将输出定向到一个文件。

让我们看几个例子。替换 night为NIGHT，首先查询模式 night，然后用文本NIGHT替换它。

```
$ sed 's/night/NIGHT/' quote.txt
The honeysuckle band played all NIGHT long for only $90.
```

要从 $90中删除 $符号（记住这是一个特殊符号，必须用 \屏蔽其特殊含义），在replacement-pattern部分不写任何东西，保留空白，但仍需要用斜线括起来。在 sed中也可以这样删除一个字符串。

```
$ sed 's/\$//' quote.txt
The honeysuckle band played all night long for only 90.
```

要进行全局替换，即替换所有出现模式，只需在命令后加 g选项。下面的例子将所有 The替换成Wow！。

```
$ sed 's/The/Wow!/g' quote.txt
Wow! honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Too bad the disco floor fell through at 23:10.
Wow! local nurse Miss P.Neave was in attendance.
```

将替换结果写入一个文件用 w选项，下面的例子将 splendid替换为SPLENDID的替换结果写入文件sed.out：

```
$ sed s/splendid/SPLENDID/w sed.out' quote.txt
```

注意要将文件名括在sed的单引号里。文件结果如下：

```
$ pg sed.out
It was an evening of SPLENDID music and company.
```

10.5 使用替换修改字符串

如果要附加或修改一个字符串，可以使用（ &）命令， &命令保存发现模式以便重新调用它，然后把它放在替换字符串里面。这里给出一个修改的设计思路。先给出一个被替换模式，然后是一个准备附加在第一个模式后的另一个模式，并且后面带有 &，这样修改模式将放在匹配模式之前。例如， sed语句s/nurse/"Hello"&/p 的结果如下：

```
$ sed -n 's/nurse/"Hello" &/p' quote.txt
The local "Hello" nurse Miss P.Neave was in attendance.
```

原句是文本行The local nurse Miss P.Neave was in attendance。

记住模式中要使用空格，因为输出结果表明应加入空格。

还有一个例子：

```
$ sed -n 's/played/from Hockering &/p' quote.txt
The honeysuckle band from Hockering played all night long for only $90.
```

原句是The honeysuckle band played all night long for only $90。相信这种修改动作已经讲解得很清楚了。

10.6 将sed结果写入文件命令

像使用>文件重定向发送输出到一个文件一样，在 sed命令中也可以将结果输入文件。格式有点像使用替换命令：

```
[address[, address]]w filename
```

‘w’选项通知sed将结果写入文件。filename是自解释文件名。下面有两个例子。

```
$ sed '1,2 w filedt' quote.txt
```

文件quote.txt输出到屏幕。模式范围即1，2行输出到文件filedt。

```
$ pg filedt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
```

下面例子中查询模式Neave，匹配结果行写入文件filedht。

```
$ sed '/Neave/ w dht' quote.txt

$ pg dht
The local nurse Miss P.Neave was in attendance.
```

10.7 从文件中读文本

处理文件时，sed允许从另一个文件中读文本，并将其文本附加在当前文件。此命令放在模式匹配行后，格式为：

```
address r filename
```

这里r通知sed将从另一个文件源中读文本。filename是其文件名。

现在创建一个小文件sedex.txt，内容如下：

```
$ pg sedex.txt
Boom boom went the music.
```

将sedex.txt内容附加到文件quote.txt的拷贝。在模式匹配行/company/后放置附加文本。本例为第三行。注意所读的文件名需要用单引号括起来。

```
$ sed '/company./r sedex.txt' quote.txt
The honeysuckle band played all night long for only $90.
It was an evening of splendid music and company.
Boom boom went the music.
Too bad the disco floor fell through at 23:10.
The local nurse Miss P.Neave was in attendance.
```

10.8 匹配后退出

有时需要在模式匹配首次出现后退出sed，以便执行其他处理脚本。退出命令格式为：

```
address q
```

下面的例子假定查询模式/.a.*/，意为任意字符后跟字符a，再跟任意字符0次或任意多次。查看文本文件，然后在下列行产生下列单词：

Line 1. band
Line 2. bad
Liner3. was
Line 4. was

查询首次出现模式，然后退出。需要将q放在sed语句末尾。

```
$ sed '/.a.*/q' quote.txt
The honeysuckle band played all night long for only $90.
```

10.9 显示文件中的控制字符

当从其他系统下载文件时，有时要删除整个文件的控制字符（非打印字符），从菜单中捕获一个应用的屏幕输出有时也会将控制字符输出进文件，怎样知道文件中是否有控制字符？使用cat -v filename命令，屏幕会乱叫，且到处都是一些垃圾字符，这可以确知文件中包含有控制字符，如果有兴趣可以观察一下这些字符以便于更加确认它们是控制字符。

一些系统中使用cat filename而不是cat-v来查看非打印字符。

sed格式为：

```
[address, [address]]l
```

‘ l ’意为列表。

一般情况下要列出整个文件，而不是模式匹配行，因此使用 l要从第一到最后一行。模式范围1，$即为此意。

如果cat一个文件，发现实际上包含有控制字符。

```
$ cat -v func.txt
This is is the F1 key:^[OP
This is the F2 key:^[OQ
```

现在运行sed命令，观察输出结果。

```
$ sed -n '1,$l' func.txt
This is is the F1 key:\033OP$
This is the F2 key:\033OQ$
```

sed找到并显示了两个控制字符。\033代表退格键，OP为F1键值，放在退格键后。第二行也是如此。

各系统控制字符键值可能不同，主要取决于其映射方式（例如使用 terminfo或termcap）。如果要在文本文件中插入控制字符F1键，使用vi查看其键值，操作如下：

- 启动vi。
- 进入插入模式。
- 按下<Ctrl>键，然后按<v>键 （出现a^）。
- 释放上述两个键。
- 按下F1键（显示[OP]。
- 按下<ESC>键（显示F1键值）。

10.10 使用系统sed

前面已经讲述了sed的基本功能，但是在脚本或命令行中使用 sed真正要做的是修改或删除文件或字符串中文本。下面运用前面学过的知识讲述这一点。

10.10.1 处理控制字符

使用sed实现的一个重要功能是在另一个系统中下载的文件中剔除控制字符。

下面是传送过来的文件（dos.txt）的部分脚本。必须去除所有可疑字符，以便于帐号所有者使用文件。

```
$ pg dos.txt
```

```
12332##DISO##45.12^M
00332##LPSO##23.11^M
01299##USPD##34.46^M
```

可采取以下动作：

1) 用一个空格替换所有的（##）符号。

2) 删除起始域中最前面的0（00）。

3) 删除行尾控制字符（^M）。

一些系统中，回车符为^@和^L，如果遇到一些怪异的字符，不必担心，只要是在行尾并且全都相同就可以。

按步执行每一项任务，以保证在进行到下一任务前得到理想结果。使用输入文件 dos.txt。

任务1。 删除所有的#字符很容易，可以使用全局替换命令。这里用一个空格替换两个或更多的#符号。

```
$ sed 's/##*//g' dos.txt
12332 DISO 45.12^M
00332 LPSO 23.11^M
01299 USPD 34.46^M
```

任务2。 删除所有行首的0。使用^符号表示模式从行首开始， ^0*表示行首任意个0。模式s/^0*//g设置替换部分为空，即为删除模式，正是要求所在。

```
$ sed 's/^0*//g' dos.txt
12332##DISO##45.12^M
332##LPSO##23.11^M
1299##USPD##34.46^M
```

任务3。 最后去除行尾 ^M符号，为此需做全局替换。设置替换部分为空。模式为：‘ s/^m//g ’，注意‘ ^M ’，这是一个控制字符。

要产生控制字符（^M），需遵从前面产生F1键同样的处理过程。步骤如下；键入 sed s/，然后按住<Ctrl>键和v键，释放v键，再按住^键，并保持<Ctrl>键不动，再释放两个键，最后按<return>键。下面命令去除行尾^M字符。

```
$ sed 's/^M//g' dos.txt
```

分步测试预想功能对理解整个过程很有帮助。用 sed在移到下一步前测试本步功能及结果很重要。如果不这样，可能会有一大堆包含怪异字符的意料外的结果。

将所有命令结合在一起，使用管道将 cat命令结果传入一系列 sed命令， sed命令与上面几步精确过滤字符的sed相同。

```
$ cat dos.txt | sed 's/^0*//g' | sed 's/^M//g' | sed 's/##*/ /g'
12332 DISO 45.12
332 LPSO 23.11
1299 USPD 34.46
```

现在文件对帐号管理者可用。

可以将命令放在文件里，然后运行它。下面即为转换脚本。

```
$ pg dos.sed
#!/bin/sed -f
# name: dos.sed
# to call: dos.sed dos.txt

# get rid of the hash marks
```

```
s/##//g

# now get rid of the leading zeros
s/^0*//g
# now get rid of the carriage return
# the ^M is generated like we did for the F1 key.
s/\^M//g
```

通过仅指定一个sed命令可以将命令行缩短，本书后面部分介绍脚本中 sed的用法。

10.10.2 处理报文输出

当从数据库中执行语句输出时，一旦有了输出结果，脚本即可做进一步处理。通常先做一些整理，下面是一个sql查询结果。

```
Database        Size (MB)     Date Created
--------------------------------------------
GOSOUTH         2244          12/11/97
TRISUD          5632          8/9/99

(2 rows affected)
```

为了使用上述输出信息做进一步自动处理，需要知道所存数据库名称，为此需执行以下操作：

1) 使用s/-*//g删除横线------。
2) 使用/^$/d删除空行。
3) 使用$d删除最后一行
4) 使用1d删除第一行。
5) 使用awk {print $1}打印第一列。

命令如下，这里使用了 cat，并管道传送结果到sed命令。

```
$ cat sql.txt | sed 's/--*//g' | sed '/^$/d' | sed '$d' | sed '1d' | awk
'{print $1}'
GOSOUTH
TRISUD
```

10.10.3 去除行首数字

对接下来卸载的这个文件实施的操作是去除行首所有数字，每个记录应以 UNH或UND开头，而不是UNH或UND前面的数字。文件如下：

```
$ pg UNH.txt
12345UND SPLLFC 234344
9999999UND SKKLT   3423
1UND SPLLY  434
...
```

使用基本正则表达式完成这个操作。 [0-9]代表行首任意数字，替换部分为空格是为了确保删除前面的匹配模式，即数字。

```
$ sed 's/^[0-9]//g' UNH.txt
UND SPLLFC 234344
UND SKKLT   3423
UND SPLLY  434
```

10.10.4 附加文本

当帐户完成设置一个文件时，帐号管理者可能要在文件中每个帐号后面加一段文字，下面是此类文件的一部分：

```
$ pg ok.txt
AC456
AC492169
AC9967
AC88345
```

任务是在每一行末尾加一个字符串‘ passed ’。

使用$命令修改各域会使工作相对容易些。首先需要匹配至少两个或更多的数字重复出现，这样将所有的帐号加进匹配模式。

```
$ sed 's/[0-9][0-9]*/& Passed/g' ok.txt
AC456 Passed
AC492169 Passed
AC9967 Passed
AC88345 Passed
```

10.10.5 从shell向sed传值

要从命令行中向sed传值，值得注意的是用双引号，否则功能不执行。

```
$ NAME="It's a go situation"
$ REPLACE="GO"
$ echo $NAME | sed "s/go/$REPLACE/g"
It's a GO situation
```

10.10.6 从sed输出中设置shell变量

从sed输出中设置 shell变量是一个简单的替换过程。运用上面的例子，创建 shell变量NEW-NAME，保存上述sed例子的输出结果。

```
$ NAME="It's a go situation"
$ REPLACE="GO"
$ NEW_NAME=`echo $NAME | sed "s/go/$REPLACE/g"`
$ echo $NEW_NAME
It's a GO situation
```

10.11 快速一行命令

下面是一些一行命令集。（[]表示空格，[ ]表示tab键）

‘ s/\.$//g ’	删除以句点结尾行
‘ -e /abcd/d ’	删除包含abcd的行
‘ s/[][]*/[]/g ’	删除一个以上空格，用一个空格代替
‘ s/^[][]*//g ’	删除行首空格
‘ s/\.[][]*/[]/g ’	删除句点后跟两个或更多空格，代之以一个空格
‘ /^$/d ’	删除空行
‘ s/^.//g ’	删除第一个字符
‘ s/COL\(...\)//g ’	删除紧跟COL的后三个字母
‘ s/^\///g ’	从路径中删除第一个\
‘ s/[]/[]//g ’	删除所有空格并用tab键替代
‘ S/^[]//g ’	删除行首所有tab键
‘ s/[]*//g ’	删除所有tab键

's/\(COL\)\(...\)/\1/g'

's/ //g' 删除空格

在结束这一章前，看看一行脚本的一些例子。

1. 删除路径名第一个\符号

将当前工作目录返回给sed，删除第一个\：

```
cd /usr/local
$ echo $PWD| sed 's/^\///g'
$ usr/local
```

2. 追加/插入文本

将"Mr Willis"字串返回给sed并在Mr后而追加"Bruce"。

```
$ echo "Mr Willis" | sed 's/Mr /& Bruce/g'
Mr Bruce Willis
```

3. 删除首字符

sed删除字符串“ accounts.doc ”首字符。

```
$ echo "accounts.doc" | sed 's/^.//g'
$ ccounts.doc
```

4. 删除文件扩展名

sed删除“ accounts.doc ”文件扩展名。

```
$ echo "accounts.doc" | sed 's/.doc//g'
accounts
```

5. 增加文件扩展名

sed附加字符串“ .doc ”到字符串“ accounts ”。

```
$ echo "accounts" | sed 's/$/.doc/g'
accounts.doc
```

6. 替换字符系列

如果变量x含有下列字符串：

```
$ x="Department+payroll%Building G"
$ echo $x
$ Department+payroll%Building G
```

如果要实现下列转换：

+ to ‘of’
% to ‘located’

sed命令是：

```
$ echo $x | sed 's/\+/ of /g' | sed 's/\%/ Located at /g'
Department of payroll Located Building G
```

10.12 小结

sed是一个强大的文本过滤工具。使用 sed可以从文件或字符串中抽取所需信息。正像前面讲到的，sed不必写太长的脚本以取得所需信息。本章只讲述了 sed的基本功能，但使用这些功能就可以执行许多任务了。

如果使用sed对文件进行过滤，最好将问题分成几步，分步执行，且边执行边测试结果。经验告诉我们，这是执行一个复杂任务的最有效方式。