目 录

1 sed 编辑器基础知识 2
1.1 sed 编辑器的介绍 . . . 2
1.1.1 在命令行中定义编辑器命令 . . . 2
1.1.2 在命令行中使用多个编辑器命令 . . . 2
1.1.3 从文件读取编辑器命令 . . . 3
1.2 s 命令的替换选项 . . . 3
1.2.1 替换标记 . . . 3
1.2.2 替换字符 . . . 3
1.3 行寻址 . . . 4
1.3.1 数字式行寻址 . . . 4
1.3.2 使用文本模式来行寻址 . . . 4
1.4 删除行命令 d . . . 5
1.5 插入文本命令 i . . . 5
1.6 附加文本命令 a . . . 5
1.7 更改行命令 c . . . 5
1.8 变换命令 y . . . 6
1.9 打印命令 . . . 6
1.10 写文件命令 w . . . 6
1.11 附加文件命令 r . . . 6

2 多行命令 7
2.1 next 命令 . . . 7
2.2 多行删除命令 . . . 7
2.3 多行打印命令 . . . 8

3 保留空间 8

4 否定命令 9

5 更改命令流 9
5.1 分支命令 . . . 9
5.2 测试命令 . . . 10

6 模式替换 **10**
6.1 与号 & 10
6.2 替换个别单词 11

7 在脚本中使用 sed **11**
7.1 使用包装器 11
7.2 重定向 sed 输出 11

1 sed 编辑器基础知识

1.1 sed 编辑器的介绍

sed 编辑器又叫做流编辑器，它根据在编辑器处理数据之前事先提供的规则集来编辑数据流。

sed 编辑器可以根据输入命令行的命令或者存储在命令文本文件中的命令处理数据。它每次从输入读取一行数据，将该数据与所提供的编辑器命令进行匹配，根据命令修改数据流中的数据，然后将新数据输出到 STDOUT。在流编辑器将全部命令和一行数据匹配完之后，它读取下一行数据，并重复上述过程。当处理完数据流中的全部数据行之后，流编辑器停止。

使用 sed 命令的格式如下所示：

```
sed [options] <script> <file>
```

script 参数指定要应用于流数据的单个命令。如果需要多个命令，可以借助 options 参数项。

options 包含的命令选项如下所示：

-e script	将多条命令添加到处理输入时执行的命令中
-f file	将文件中指定的命令添加到处理输入时执行的命令中
-n	不需要为每个命令产生输出，但要等待打印命令

需要注意的是，sed 编辑器并不修改文本文件中的数据，它只是讲修改后的文本发送到 STDOUT。

1.1.1 在命令行中定义编辑器命令

使用单个命令处理数据的例子如下所示：

```
# s命令将用第二个文本字符串替换第一个文本字符串模式
echo "This is a test" | sed 's/test/big test/'
# 输出为: This is a big test
```

1.1.2 在命令行中使用多个编辑器命令

如果要从 sed 命令行中执行多个命令，只需要使用-e 选项，如下例所示：

```
# data1是一个文件
sed -e 's/brown/green/; s/dog/cat/' data1
```

需要注意的是，使用多个命令时，命令必须用分号隔开，而且在命令结尾和分号之间不能有任何空格。

1.1.3 从文件读取编辑器命令

如果有太多的 sed 命令要处理，可以把它们保存在一个独立的文件中，然后在 sed 命令中使用-f 选项指定添加文件中的命令，例子如下所示：

```
# script1是存放sed命令的文件
sed -f script1 data1
```

需要注意的是，文件中每一行包含一个单独的一个命令，而且不需要在每个命令后放一个分号。

1.2 s 命令的替换选项

1.2.1 替换标记

替换命令在默认情况下仅 iguana 各行中首次出现的文本，如果想要使用不同的替换模式，需要使用替换标记，格式如下：

```
s/patter/replacement/flags
```

可用的替换标记如下所示：

- 数字，表示要替换的第几个模式。
- g，表示用新文本替换现有文本的全部实例。
- p，表示打印原始行的内容。
- w file，表示将替换的结果写入文件。

例子如下：

```
sed 's/test/trial/g' data1
```

1.2.2 替换字符

如果要替换路径时，需要考虑到正斜杠的影响，正斜杠必须要反斜杠转义，如下例所示：

```
sed 's/\/bin\/bash/\/bin\/csh/' /etc/passwd
```

sed 编辑器允许为替换命令中的字符串定界符选择一个不同的字符，如下例所示：

```
# 将“!”作为字符串定界符
sed 's!/bin/bash!/bin/csh!' /etc/passwd
```

1.3 行寻址

如果想将 sed 编辑器的命令应用于某一特定的文本数据行或一组文本数据行，需要使用行寻址。行寻址形式有：行的数值范围和筛选行的文本模式。

1.3.1 数字式行寻址

数字式行寻址的形式如下所示：

```
[address]command
```

也可以将多个命令组合在一起，应用于一个特定的地址，如下所示：

```
address {
    command1
    command2
    command3
}
```

指定特定的行如下所示：

```
# 指定第2行
sed '2s/dog/cat/' data1
```

指定某一个范围的行如下所示：

```
# 指定第2到第5行，从1开始
sed '2,5/dog/cat/' data1
```

指定某一行到文本结束如下所示：

```
sed '2,$s/dog/cat/' data1
```

1.3.2 使用文本模式来行寻址

使用文本模式来行寻址，格式如下：

```
/pattern/command
```

例子如下：

```
sed '/rich/s/bash/csh/' /etc/passwd
```

1.4 删除行命令 d

d 命令用于删除数据，一般与行寻址相配合，如下例所示：

```
# 数字式行寻址
sed '3,$d' data1
# 文本模式行寻址
sed '/number 1/d' data1
```

1.5 插入文本命令 i

插入命令 i 在指定行之前添加新的一行，格式如下：

```
sed '[address]i\
new line' data1
```

如果没有指定行号，数据将添加在文本的最前面。

1.6 附加文本命令 a

附加命令 a 在指定行之后添加新的一行，格式如下：

```
sed '[address]a\
new line' data1
```

如果没有指定行号，数据将添加在文本的最后面。

1.7 更改行命令 c

更改行命令 d 格式如下：

```
sed '[address]c\
new line' data1
```

例子如下所示：

```
# 数字式行寻址
sed '3c\
This is a changed line of text' data1
# 文本模式行寻址
sed '/number 3/c\
```

```
6    This is a changed line of text' data1
```

1.8 变换命令 y

变换命令 y 是唯一对单个字符进行操作的 sed 编辑器命令，格式如下：

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

变换命令将 inchars 和 outchars 的值进行一对一映射，例子如下所示：

```
1    echo "This 1 is a test of 1 try." | sed 'y/123/456/'
```

1.9 打印命令

打印命令 p 就是打印数据的原始版本，例子如下所示：

```
1    sed '2,3p' data1
```

等号命令 = 用于打印当前行的行号，例子如下所示：

```
1    sed '=' data1
```

列表命令 l 用于打印数据流中的文本和不可打印的 ASCII 字符，例子如下所示：

```
1    sed -n 'l' data1
```

1.10 写文件命令 w

写命令 w 如下所示：

```
1    [address]w filename
```

例子如下：

```
1    sed '1,2w test' data1
```

1.11 附加文件命令 r

附加文件命令用于将某一个文件的内容附加在文件中的某一行后。附加文件命令 r 如下所示：

```
1    [address]r filename
```

例子如下所示：

```
sed '3r data1' data6
```

2 多行命令

2.1 next 命令

n 命令将 sed 编辑器移动到文本的下一行，也就是将数据流的下一行移动到 sed 编辑器的处理空间中，例子如下：

```
sed '/header/{
    n
    d
}' data1
```

N 将数据流的下一行添加到已经存在于模式空间的文本中，这个命令适用于搜索数据文件中跨行短语的情况，如下例所示：

```
sed '
    N
    s/System\nAdministrator/Desktop\nUser/
    s/System Administrator/Desktop User/
    ' data1
```

需要注意的是，sed 编辑器到达文本最后一行时，N 命令因为没有下一行可以读入，所以会导致 sed 编辑器停止。所以应该将单行命令移动到 N 命令之前，使得多行命令在 N 命令之后，例子如下：

```
sed '
    s/System Administrator/Desktop User/
    N
    s/System\nAdministrator/Desktop\nUser/
' data1
```

2.2 多行删除命令

单行删除命令 d 会删除模式空间中的两行，多行删除命令 D 不同。在多行删除命令中，sed 编辑器只删除模式空间中的第一行，例子如下：

```
sed '
    N
    /System\nAdministrator/d
' data1

```

```
sed '
    N
    /System\nAdministrator/D
'
```

2.3 多行打印命令

单行打印命令 p 会打印模式空间中的两行，而多行打印命令 P 不同。在多行打印命令中，sed 编辑器只打印模式空间中的第一行，例子如下：

```
sed -n '
    N
    /System\nAdministrator/P
' data1
```

3 保留空间

模式空间是 sed 编辑器处理命令时保留的文本，而保留空间是在处理模式空间中的其他行时，可以使用保留空间暂时保留文本行。

sed 编辑器保留空间相关的命令：

h	将模式空间复制到保留空间
H	将模式空间追加到保留空间
g	将保留空间复制到模式空间
G	将保留空间追加到模式空间
x	将模式空间和保留空间的内容变换

需要注意的是，不管是复制还是追加，都不会把源数据空间的内容清空。

例子如下：

```
sed -n '/first/{
    h
    p
    n
    p
    g
    p
}'
```

4 否定命令

感叹号命令! 用于否定命令，也就是在命令被激活的地方不激活命令，在命令不被激活的地方激活命令，例子如下：

```
sed -n '/header/!p' data1
```

否定命令可以用于饭庄数据流中的文本行的顺序，算法如下：

1. 将一行放到保留空间中。
2. 将文本的下一行放到模式空间中。
3. 将保留空间追加到模式空间。
4. 将模式空间放到保留空间。
5. 重复第 2 步到第 4 步，直至所有行以相反的顺序放到保留空间。
6. 检索行并打印它们。

5 更改命令流

5.1 分支命令

分支命令可以指定地址不去执行脚本命令。分支命令的格式如下：

```
[address]b [label]
```

address 参数决定哪个行或哪些行激活分支命令，label 参数定义了分支的标签，使用标签可以跳过与分支地址匹配的命令，而仍然执行脚本中的其他命令。如果 label 参数不存在，则分支命令将继续执行到脚本的结尾，也就是整个脚本的命令都不执行。例子如下：

```
sed '{
    /first/b jump1
    s/ is/ might be/
    s/line/test/
    :jump1
    s/data/text/
}' data1
```

需要注意的是，如果没有恰当地指定 address 参数，可能会导致死循环，例子如下：

```
echo "This, is, a, test, to, remove, commas." | sed -n '{
    :start
    s/,//1p
    b start
}'
```

5.2 测试命令

测试命令是基于替换命令的结果跳转到标签。如果替换命令成功匹配并替换了一个模式，则测试命令分支到指定的标签。如果替换命令不匹配指定的模式，则测试命令不分支。测试命令的格式如下:

```
[address]t [label]
```

address 指定测试命令作用的地址范围。如果不指定标签 label，测试成功时将分支到脚本的最后，也就是不执行整个脚本。测试命令可以实现 if-then 语句功能，如果一个替换命令成功，则另一个替换命令则不需要在执行，例子如下:

```
sed '{
    s/first/starting/
    t
    s/line/test/
}' data1
```

测试命令还可以实现 for 循环的功能，例子如下:

```
echo "This, is, a, test, to, remove, commas." | sed -n '{
    :start
    s/,//1p
    t start
}'
```

6 模式替换

现在面临的情况是，我们在使用替换命令时，使用通配符可以匹配任意单词，但是替换字符串不能以通配符的值替换匹配的单词。例子如下:

```
echo "The cat sleeps in his hat." | sed 's/.at/".at"/g'
```

6.1 与号 &

与号 & 表示替换命令中的匹配模式，例子如下:

```
echo "The cat sleeps in his hat." | sed 's/.at/"&"/g'
```

6.2 替换个别单词

模式匹配中匹配的是整个字符串，但有时候我们只想要匹配模式中的子集。sed 编辑器使用圆括号定义替换模式中的子字符串，然后用特定的符号引用子字符串元素。特定的符号由反斜杠和数字组成，第一个元素分配为字符 \1，第二个元素分配为字符 \2，依次类推。需要注意的是，在替换命令中使用圆括号时，必须在圆括号前加转义符号。例子如下：

```
echo "The System Administrator manual" | sed 's/\(System\) Administrator/\1
    User/'
```

7 在脚本中使用 sed

7.1 使用包装器

在 shell 脚本中，可以在 sed 编辑器脚本中使用常规的 shell 变量和参数，例子如下：

```
#!/bin/bash
# 文件名是reverse
sed -n '{
    1!G
    h
    $p
}' "$1"
# $1是命令行参数
```

使用这个脚本和使用正常的 shell 脚本一样，例子如下：

```
# reverse是sed编辑器脚本的文件名
./reverse data1
```

7.2 重定向 sed 输出

可以使用反引号将 sed 编辑器命令的输出重定向到一个变量，例子如下：

```
#!/bin/bash

factorial=1
counter=1
number=$1

```

```
7     while [ $counter -le $number ]
8     do
9         factorial=$[ $factorial * $counter ]
10        $counter=$[ $counter + 1 ]
11    done
12
13    result=`echo $factorial | sed '{
14    :start
15    s/\(.*[0-9]\)\([0-9]\{3\}\)/\1,\2/
16    t start
17    }'`
18
19    echo "The result is $result"
```