sed进阶

正则表达式练习题

##计算PATH目录下的文件数

PATH目录下的都是二进制命令文件

```
1.查看PATH值
[root@gitlab01 ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin

2.获取每个目录的字符串，利用sed替换功能
[root@node02 ~]# echo $PATH|sed 's/:/ /g'
/usr/local/sbin /usr/local/bin /usr/sbin /usr/bin /root/bin

3.分离目录后，可以for循环遍历取值了
[root@node02 ~]# my_path=$(echo $PATH|sed 's/:/ /g')
[root@node02 ~]# for dir in $my_path;do echo $dir;done
/usr/local/sbin
/usr/local/bin
/usr/sbin
/usr/bin
/root/bin

4.最终计算文件脚本
[root@node02 tmp]# cat test.sh
#!/bin/bash
# 超哥带你学shell编程

my_path=$(echo $PATH|sed 's/:/ /g')
count=0
for dir in $my_path
do
  check_dir=$(ls $dir)
  for item in $check_dir
  do
    count=$[ $count + 1 ]
  done
  echo "$dir ---- $count"
  count=0 # 归零，否则影响全局的count数量
done

[root@node02 tmp]# bash test.sh
/usr/local/sbin ---- 0
/usr/local/bin ---- 0
/usr/sbin ---- 540
/usr/bin ---- 1139
/root/bin ---- 0
```

##验证电话号码

```
# 对于一些数据表单的输入，例如电话号码，输入错误，需要进行检测
以下是美国电话号码格式
(123)456-7890
(123) 456-7890
123-456-7890
123.456.7890

这也就表明了电话输入的格式，可以有四种，因此你的正则必须完美的校验上述电话形式。

1.正则构建，最好从左边开始，先判断是否有空格
^\(?    脱字符后面跟着转义符和括号,问号表示括号，可有可无

2.紧接着是三位区号，美国区号从2开始，没有0或1，最大到9结束
[2-9][0-9]{2}  这表示第一个数字是2-9，第二个是0-9，第三个同样

3.区号后面的括号，也是可有可无的，因此
\)?

4.从上述的电话看出，区号后面，可能有单破折线，或者空格，或者什么都没有，或者小数点，可以这么写
(| |-|\.)
如此的写法表示，首先括号分组，后面也是同样的规则
然后用管道符进行表示，四种状态，空，空格，横杠，小数点

5.再接着是三位电话交换机号码，数字即可
[0-9]{3}

6.在电话交换机号码后面，必须有一个空格，一个单破折现，或者一个点。（这次不存在没有空格的情况），因此
( |-|\.)

7.最后就是尾部匹配4位本地电话分机号
[0-9]{4}$
```

总结

```
完整的匹配模式
^\(?[2-9][0-9]{2}\)?(| |-|\.)[0-9]{3}( |-|\.)[0-9]{4}$
```

此时可以集合sed，awk进行对电话号码过滤验证。

```
[root@node02 tmp]# cat phone_list
000-000-0000
123-456-7890
212-555-1234
(317)555-1234
(202) 555-9876
33523
1234567890
234.123.4567
[root@node02 tmp]#
[root@node02 tmp]#
```

```
[root@node02 tmp]#
[root@node02 tmp]# cat phone_list | awk '/^\(?[2-9][0-9]{2}\)?(| |-|\.)[0-9]{3}( |-|\.)[0-9]
{4}$/{print $0}'
212-555-1234
(317)555-1234
(202) 555-9876
234.123.4567
```

##解析邮件地址

邮件地址是手机号之外的另一大通信方式，也存在千奇百怪的格式。

> username@hostname
>
> 点号
>
> 单破折线
>
> 加号
>
> 下划线

有效的邮件用户名里，这些字符都有可能存在组合。

> 邮件地址hostname格式是 一个或多个域名和一个服务器名字组成，也有一定的规则。
>
> 点号
>
> 下划线


```
1.左侧的用户名正则，用户名里可以有多个有效字符
^([a-zA-Z0-9_\-\.\+]+)@. 中括号里的字符，匹配一次或者任意多次
这个分组指定了用户名中允许出现的字符，正则加号，确保至少有一个字符

2.hostname的匹配规则
([a-zA-Z0-9_\-\.]+)

这样的模式，可以匹配，例如
server
server.subdomain
server.subdomain.subdomain

3.顶级域名的匹配规则，也就是例如.cn .com .org此类
顶级域名只能是字母，不少于二个字符，长度不搞过五个字符。
\.([a-zA-Z]{2,5})$

4.因此整个的匹配邮箱的规则，可以是
^([a-zA-Z0-9_\-\.\+]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$
```

案例

```
使用该正则，就可以过滤掉不正确的邮件地址
[root@node02 tmp]# cat email_list
yuchao@163.com
yy@163.com.
yy@ee.n
yy.city@163.now
yy_city@163.cn
yy#city@163.cc
yy+city@163.cc
yy*city@163.org
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# cat email_list | awk '/^([a-zA-Z0-9_\-\.\+]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]
{2,5})$/{print $0}'
yuchao@163.com
yy.city@163.now
yy_city@163.cn
yy+city@163.cc
```

sed进阶

sed编辑器可以满足大多数日常文本需求，这里超哥在给大家讲讲些高级用法。

多行命令

sed命令的特点是单行数据操作，基于换行符的位置吧数据分成行。然后sed一行一行的处理，重复过程。

如果需要跨行对数据处理，这就麻烦了。

比如文本里找一个短语 `Linux System Administrators Group`，这个长语句可能出现在两行，默认的sed就无法识别这种短语。

sed开发者也考虑到了这个情况，提供了处理方案。

> sed特殊指令
>
> N: 将数据流中的下一行加进来，创建一个多行组处理，multiline group
>
> D：删除多行组的一行
>
> P：打印多行组的一行

next命令

sed小写的n命令会告诉sed编辑器移动到数据流中的下一行文本。

sed编辑器再移动到下一行文本前，会在当前行执行完毕所有定义好的命令。单行next命令改变了这个流程。

看下案例。

```
# 数据文件
[root@node02 tmp]# cat data.txt
This is an apple.

This is a boy.

This is a gril.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '/^$/d' data.txt
This is an apple.
This is a boy.
This is a gril.

这种写法是删除空行
```

若是我们想要指定删除某个语句后面的空行，可以用n指令。

```
# 删除apple下一行的空格
[root@node02 tmp]# cat data.txt
This is an apple.

This is a boy.

This is a gril.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '/apple/{n;d}' data.txt
This is an apple.
This is a boy.

This is a gril.
```


此时sed编辑器匹配到apple这一行后，通过n指令，让sed编辑器移动到文本的下一行，也就是空行，然后通过d指令，删除了该行。

此时sed执行完毕命令后，继续重复查找apple，然后尝试删除apple的下一行。

如果找不到apple字符串，也就不会执行任何动作了。

合并文本行

刚才使用小写的n将文本的下一行移动到sed的`模式空间`，属于是单行处理。

大写的N指令将下一行文本添加到模式空间中已经有的文本的后面，实现多行文本处理。

这个作用是将数据流的两个文本行合并在同一个模式空间里处理。

```
[root@node02 tmp]# cat data2.txt
This is the header line.
This is the first data line.
This is the second data line.
This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '/first/{N;s/\n/ /}' data2.txt
This is the header line.
This is the first data line. This is the second data line.
This is the last line.
```

这里sed找到first一行后，用N指令把下一行合并到first该行，并且执行s替换指令，结果是如上合并了一行。

> 案例：在数据文件里，查找替换一个可能分散两行的文本短语

```
[root@node02 tmp]# cat data.txt
On Tuesday, the Linux System
Administrator's group meeting will be held.
All System Administrators should attend.
Thank you for your attendance.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed 's/System Administrator/Desktop User/' data.txt
On Tuesday, the Linux System
Administrator's group meeting will be held.
All Desktop Users should attend.
Thank you for your attendance.
[root@node02 tmp]#
```


这里发现没有变化，是因为该短语出现在了两行，替换命令无法识别

用N命令解决

```
[root@node02 tmp]# sed 'N;s/System.Administrator/Desktop User/' data.txt
On Tuesday, the Linux Desktop User's group meeting will be held.
All Desktop Users should attend.
Thank you for your attendance.
```

这里注意，是使用了两个特点，一是N指令，二是`通配符.`来匹配空格和换行符的情况。

> 当匹配出现了换行符，它就从字符串里删除了换行符，缺导致了两行合并

想要解决这个办法，可以如此：

```
[root@node02 tmp]# cat data.txt
On Tuesday, the Linux System
Administrator's group meeting will be held.
All System Administrators should attend.
Thank you for your attendance.
[root@node02 tmp]#
```

```
[root@node02 tmp]#
[root@node02 tmp]# sed '
> s/System Administrator/Desktop User/
> N
> s/System\nAdministrator/Desktop\nUser/
> ' data.txt
On Tuesday, the Linux Desktop
User's group meeting will be held.
All Desktop Users should attend.
Thank you for your attendance.
```

用这种写法，可以保证，先进行单行替换处理，即使在最后一行也可以工作，因为N指令在最后一行时会停止工作。

###多行删除

sed基础中说了删除命令(d)，sed编辑器用于删除模式空间里的当前行，和N一起使用的时候，要小心点了。

```
#这里是坑了
[root@node02 tmp]# cat data.txt
On Tuesday, the Linux System
Administrator's group meeting will be held.
All System Administrators should attend.
Thank you for your attendance.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed 'N;/System\nAdministrator/d' data.txt
All System Administrators should attend.
Thank you for your attendance.
```


这里会发现直接在模式空间里删除了两行。

sed提供了大写的D，只删除模式空间里的第一行，该指令会删除到换行符位置的所有字符。

```
[root@node02 tmp]# sed 'N;/System\nAdministrator/D' data.txt
Administrator's group meeting will be held.
All System Administrators should attend.
Thank you for your attendance.
```

这里就只删除了第一行。

用这个方式可以删除文件的开头空白行。

```
[root@node02 tmp]# cat data2.txt

This is the header line.
This is the first data line.
This is the second data line.
This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '/^$/{N;/header/D}' data2.txt
```

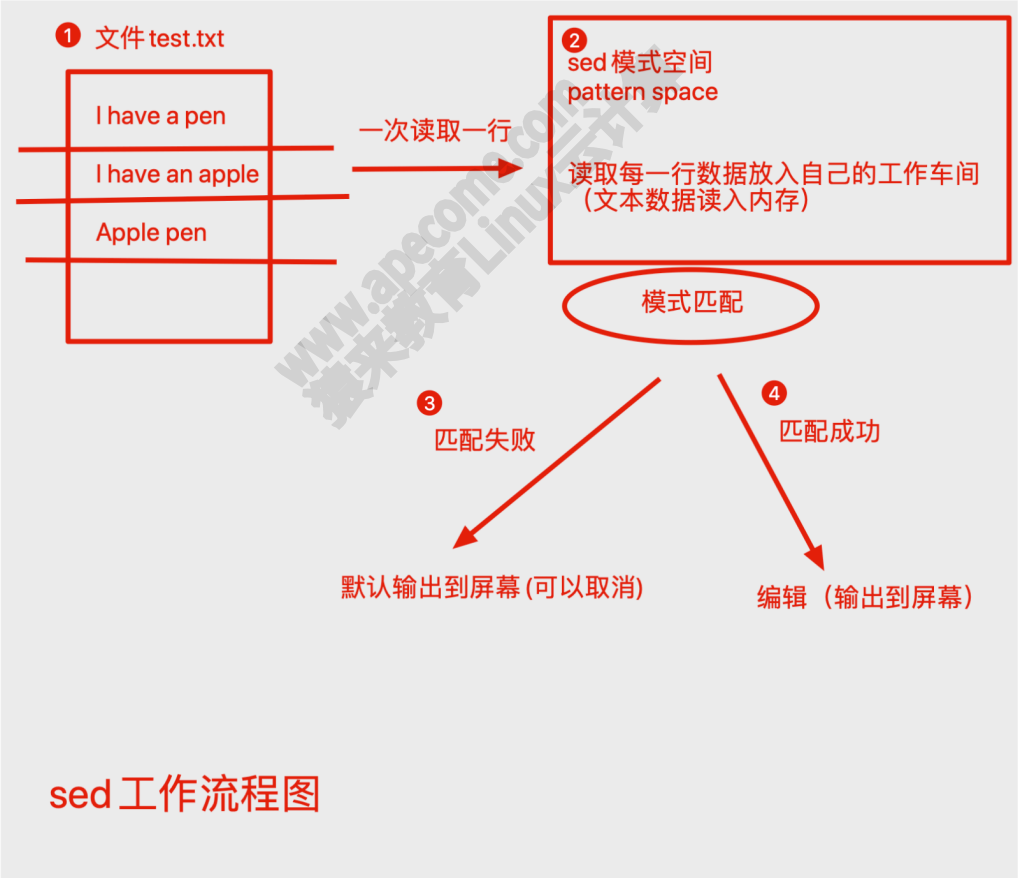
```
This is the header line.
This is the first data line.
This is the second data line.
This is the last line.
[root@node02 tmp]#
```

sed编辑器会查找空白行，然后用N命令把下一行的文本添加到模式空间，如果新的模式空间里有单词header，D指令就删除模式空间的第一行，也就删除了空白行。

> 记住sed是按行处理，如果不结合N和D指令，很难做到不删除其他空白行情况下，只删除第一个空白行。

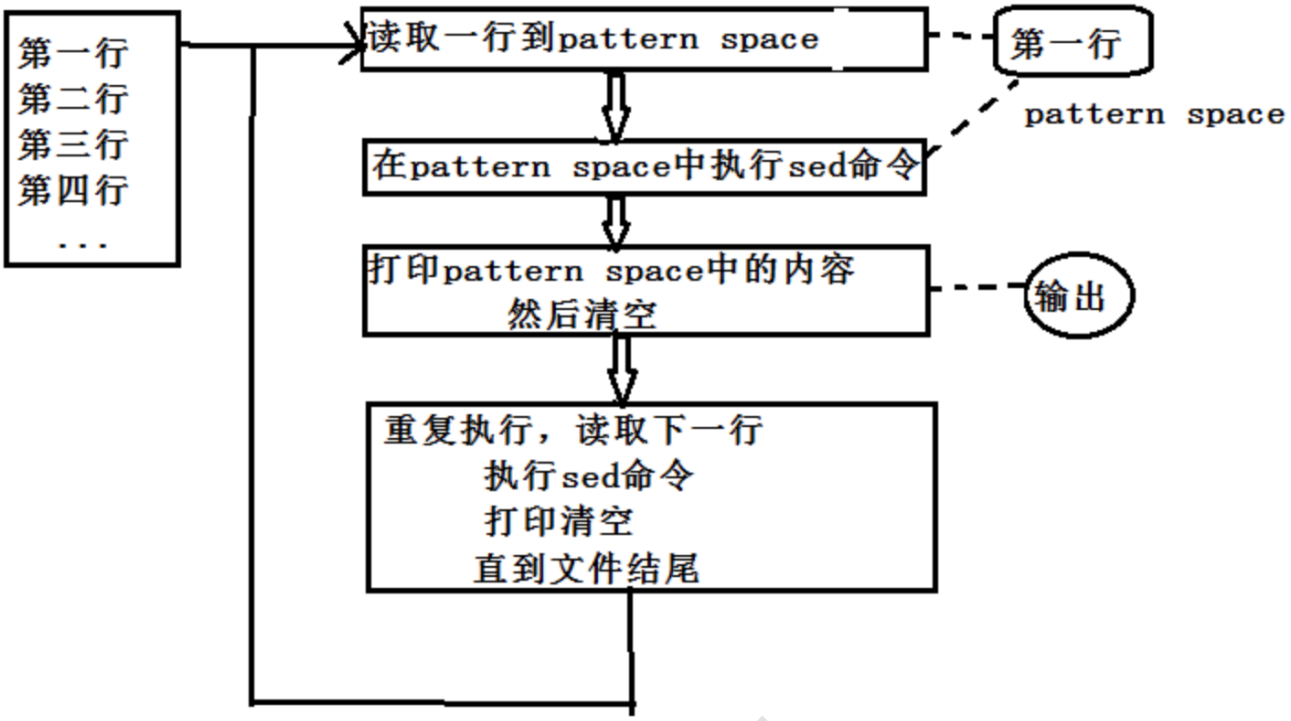
保持空间

sed我们已知有一块模式空间（pattern space）用于sed编辑器执行命令的时候，保存待检查的文本。



sed还有一块空间叫做保持空间的缓冲区域（hold space）。

> sed之所以能以行为单位的编辑或修改文本，其原因在于它使用了两个空间：一个是活动的“模式空间（pattern space）”，另一个是起辅助作用的“保持空间（hold space）这2个空间的使用。
>
> 模式空间：可以想成工程里面的流水线，数据之间在它上面进行处理。
> 保持空间：可以想象成仓库，我们在进行数据处理的时候，作为数据的暂存区域。
>
> 正常情况下，如果不显示使用某些高级命令，保持空间不会使用到！

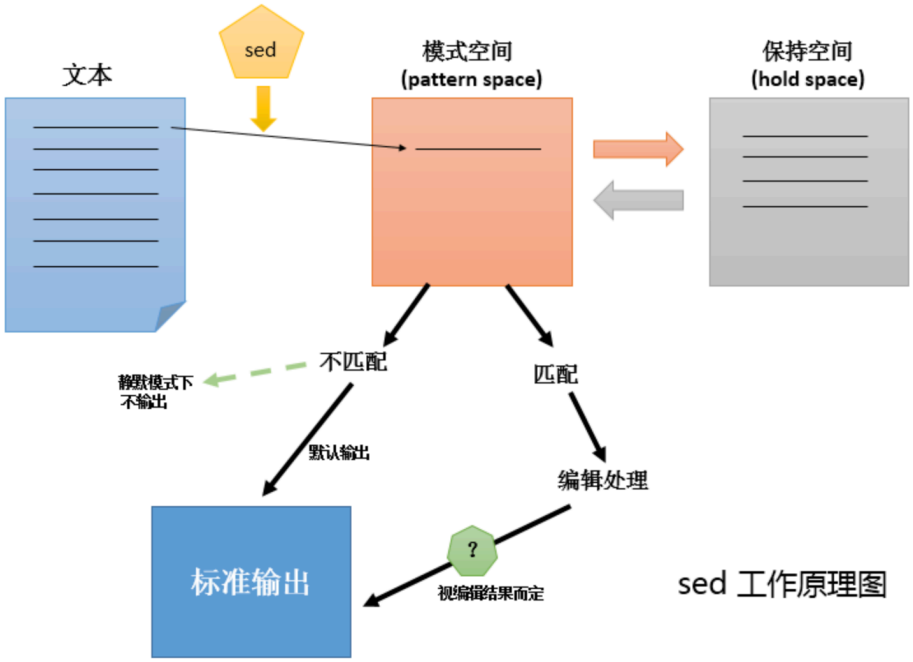


一般情况下，数据的处理只使用模式空间（pattern space），按照如上的逻辑即可完成主要任务。但是某些时候，通过使用保持空间（hold space），还可以带来意想不到的效果。

sed编辑器的保持空间命令。

命　令	描　述
h	将模式空间复制到保持空间
H	将模式空间附加到保持空间
g	将保持空间复制到模式空间
G	将保持空间附加到模式空间
x	交换模式空间和保持空间的内容

这些命令可以将文本从模式空间复制到保持空间。



这些保持空间的命令用于将文本从模式空间复制到保持空间。这样就可以清空模式空间加载其他需要处理的字符串。

案例，理解如何用h和g命令将数据在两个缓冲空间移动。

```
[root@node02 tmp]# cat data2.txt

This is the header line.
This is the first data line.
This is the second data line.
This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed -n '/first/{h;p;n;p;g;p}' data2.txt
This is the first data line.
This is the second data line.
This is the first data line.
```

我们看下这个案例

1. sed过滤含有first的单词的行
2. 当出现含有first单词的行，h指令将该行复制到保持空间
3. p命令此时打印模式空间第一行的数据，也就是first的行
4. n命令提取数据流的下一行，并且也放到了模式空间
5. p命令再次打印模式空间的内容，也就是打印了second那一行。
6. g命令此时将保持空间的内容放回模式空间，替换当前文本
7. p命令再次打印模式空间的内容，又打印了first内容了。

此时可以看出，保持空间的指令，可以来回移动文本行，再看一个案例

```
[root@node02 tmp]# sed -n '/first/{h;n;p;g;p}' data2.txt
This is the second data line.
This is the first data line.
```

若是这里去掉一个p打印，则会看出不同的结果了。

排除命令

sed编辑会将一些处理命令应用到数据流中的每一个文本行，单个行，或者一些区间行。也支持排除某个区间。

sed支持用`感叹号!`来排除命令，让某个命令不起作用。

```
[root@node02 tmp]# cat data2.txt

This is the header line.
This is the first data line.
This is the second data line.
This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed -n '/header/p' data2.txt
This is the header line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed -n '/header/!p' data2.txt

This is the first data line.
This is the second data line.
This is the last line.
```


sed实战

##向文本中插入空白行

G 获得内存缓冲区的内容，并追加到当前模板块文本的后面。

```
[root@node02 tmp]# cat data2.txt
This is the header line.
This is the first data line.
This is the second data line.
This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed 'G' data2.txt
This is the header line.

This is the first data line.
```

```
This is the second data line.

This is the last line.
```

这里的技巧在于，sed `G命令` 会简单的保持空间内容附加到模式空间后。

当sed启动时候，保持空间默认只有一个空行，将它附加到已有行后面，就是上述的效果了。

##向文本中插入空白行，去掉最后一行的空白

```
# 找出最后一行
[root@chaogelinux sed_awk]# sed '$p' data2.txt  -n
This is the last line.

# 使用排除符号！和尾行符号$确保sed不会在最后一行添加空白行
[root@node02 tmp]# sed '$!G' data2.txt
This is the header line.

This is the first data line.

This is the second data line.

This is the last line.
```

这里的技巧在于，只要不是最后一行，G命令就会附加保持空间的内容，且忽略最后一行。

##对可能存在空白行的文件，加倍行间距

如果文件已经有了一些空白行，但是你想要给所有的行加倍间距怎么办？上面的案例没发用，因为会导致有些区域空白太多。

```
[root@node02 tmp]# cat data2.txt
This is the header line.
This is the first data line.

This is the second data line.
This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '$!G' data2.txt
This is the header line.

This is the first data line.



This is the second data line.
```

```
This is the last line.
```

想要解决的办法是，先删除数据流所有的空白行，然后再用G加入新行。

答案如下

```
[root@node02 tmp]# sed '/^$/d' data2.txt
This is the header line.
This is the first data line.
This is the second data line.
This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '/^$/d;$!G' data2.txt
This is the header line.

This is the first data line.

This is the second data line.

This is the last line.
```


给文件中行编号

sed可以使用 `=` 命令打印当前行号。

```
[root@node02 tmp]# sed '=' data2.txt
1
This is the header line.
2
This is the first data line.
3
This is the second data line.
4
This is the last line.
```

> 这样太丑，你应该会想到可以用 `N` 命令合并行处理。

```
[root@node02 tmp]# sed '=' data2.txt | sed 'N;s/\n/ /'
1 This is the header line.
2 This is the first data line.
3 This is the second data line.
4 This is the last line.
```

你可能会想到其他有些linux命令也会显示行号，但是可能不那么合适

```
[root@node02 tmp]# nl data2.txt
     1  This is the header line.
```

```
     2  This is the first data line.
     3  This is the second data line.
     4  This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# cat -n data2.txt
     1  This is the header line.
     2  This is the first data line.
     3  This is the second data line.
     4  This is the last line.

[root@node02 tmp]# grep '.' data2.txt -n
1:This is the header line.
2:This is the first data line.
3:This is the second data line.
4:This is the last line.
```

> 因此sed会是一个很合适的小工具。

删除行

若是用sed删除所有的空白行很简单，但是选择性的删除空白行，则麻烦了些

```
[root@node02 tmp]# cat data2.txt
This is the header line.

This is the first data line.

This is the second data line.

This is the last line.

[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '/^$/d' data2.txt
This is the header line.
This is the first data line.
This is the second data line.
This is the last line.
```

删除连续的空白行

有些文件里会有讨厌的多个空白行，删除连续的空白行是用地址区间检查数据流。

删除连续的空白行的关键在于创建一个`非空白行`和`空白行`的地址区间，sed碰到该区间，不删除，其他的空白行区间则删除。

```
sed语法
区间是/./到/^$/
sed '/./,/^$/!d'    !d这表示不删除该区间
这就好比sed '1,3p' 打印1到3行一样
```

案例

```
[root@node02 tmp]# cat data2.txt
This is the header line.




This is the first data line.



This is the second data line.

This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '/./,/^$/!d' data2.txt
This is the header line.

This is the first data line.

This is the second data line.

This is the last line.
```


无论数据行之间有多少个空白行，都会只保留一个空白行了。

删除开头的空白行

数据文件经常也会存在空白行，若是导入数据库也会生成空项，较为麻烦。

```
sed命令
/./,$!d
该sed命令表示不删除有益内容，删除开头空白行。
```

案例

```
[root@node02 tmp]# cat data2.txt



This is the header line.
```

```
This is the first data line.



This is the second data line.

This is the last line.
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '/./,$!d' data2.txt
This is the header line.




This is the first data line.



This is the second data line.

This is the last line.
[root@node02 tmp]#
```


删除HTML标签

现在从网站上下载html并且保存使用的场景还是较多，例如爬虫等场景，HTML的标签较多，如何筛选出有益的信息。

```
[root@node02 tmp]# cat data2.txt
<html>
<head>
<title>This is the page title</title> </head>
<body>
<p>
This is the <b>first</b> line in the Web page.
This should provide some <i>useful</i>
information to use in our sed script.
</body>
</html>
```

对HTML标签的删除大部分是成对的删除，例如

```
<b> </b>
```

对于标签的删除正则，要小心，否则会删错，例如这样的正则

```
s/<.*>//g 这样的正则是有问题的
```

```
[root@node02 tmp]# sed 's/<.*>//g' data2.txt




This is the  line in the Web page.
This should provide some
information to use in our sed script.
```

这里是有问题，发现titile标签整行被删除了，以及加粗，斜体的文本都不见了。

sed认为的是在大于号、小于号之间的文本都要被替换为空。

正确的正则改成如下

```
# 正确的思路应该是让sed编辑器忽略掉，嵌入在原始标签里的大于号，排除写法[^>]
[root@node02 tmp]# cat data2.txt
<html>
<head>
<title>This is the page title</title> </head>
<body>
<p>
This is the <b>first</b> line in the Web page.
This should provide some <i>useful</i>
information to use in our sed script.
</body>
</html>
[root@node02 tmp]#
[root@node02 tmp]# sed 's/<[^>]*>//g;/^$/d' data2.txt
This is the page title
This is the first line in the Web page.
This should provide some useful
information to use in our sed script.
```


sed基本练习题

以行为单位添加/删除

> 将/etc/passwd的内容输出并且打印行号，同时在第1行之前插入两行文本，第一行内容为"How are you?"，第二行内容为"How old are you?"：

```
[root@node02 tmp]# nl /etc/passwd | sed '1i How are you?\nHow old are you?' | head -5
How are you?
How old are you?
     1  root:x:0:0:root:/root:/bin/bash
     2  bin:x:1:1:bin:/bin:/sbin/nologin
     3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

在/etc/passwd第2行之后追加文本"Drink tea?"：

```
[root@node02 tmp]# nl /etc/passwd|sed '2a Drink Tea?'  | head -5
     1  root:x:0:0:root:/root:/bin/bash
     2  bin:x:1:1:bin:/bin:/sbin/nologin
Drink Tea?
     3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
     4  adm:x:3:4:adm:/var/adm:/sbin/nologin
```

删除/etc/passwd第2行至第5行内容：

```
[root@node02 tmp]# nl /etc/passwd|sed '2,5d' | head -5
     1  root:x:0:0:root:/root:/bin/bash
     6  sync:x:5:0:sync:/sbin:/bin/sync
     7  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
     8  halt:x:7:0:halt:/sbin:/sbin/halt
     9  mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
```

sed将2-5行匹配到模式空间，然后d命令删除了模式空间的内容，因此这几行没输出

题：把文本第二行之后的空白行删除

这里注意要用到next命令

```
[root@node02 tmp]# cat data2.txt
<html>

<head>
<title>This is the page title</title> </head>
<body>



<p>


This is the <b>first</b> line in the Web page.


This should provide some <i>useful</i>
information to use in our sed script.
</body>
```

```
</html>
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed '2n;/^$/d' data2.txt
<html>

<head>
<title>This is the page title</title> </head>
<body>
<p>
This is the <b>first</b> line in the Web page.
This should provide some <i>useful</i>
information to use in our sed script.
</body>
</html>
```

##{}提示

对某一行执行多次处理，用`{}`将命令扩起来，花括号内每个命令用分号分割。

以行为单环替换/打印

> 将/etc/passwd第三行替换为文本"This is the third line."：
>
> 提示：
>
> c \TEXT：将指定行的内容替换为文本TEXT；


```
[root@node02 tmp]# nl /etc/passwd | sed '3c This is the third line'
     1  root:x:0:0:root:/root:/bin/bash
     2  bin:x:1:1:bin:/bin:/sbin/nologin
This is the third line
     4  adm:x:3:4:adm:/var/adm:/sbin/nologin
     5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

> 使用编辑命令y实现对应转换字符：
>
> 提示
>
> y：用于(对应)转换字符；
>
> 例如 a > A b > B，注意字符数对应

```
[root@node02 tmp]#
[root@node02 tmp]# cat data.txt
On Tuesday, the Linux System
Administrator's group meeting will be held.
All System Administrators should attend.
Thank you for your attendance.
```

```
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed 'y/abc/ABC/' data.txt
On TuesdAy, the Linux System
AdministrAtor's group meeting will Be held.
All System AdministrAtors should Attend.
ThAnk you for your AttendAnCe.
```

显示/etc/passwd前十行

提示

q：读取匹配到的行后退出；

```
# 注意，一个文件若是非常大，sed用p处理会处理每一行，因此用q节省cpu资源。
[root@node02 tmp]# nl /etc/passwd | sed '10q'
     1  root:x:0:0:root:/root:/bin/bash
     2  bin:x:1:1:bin:/bin:/sbin/nologin
     3  daemon:x:2:2:daemon:/sbin:/sbin/nologin
     4  adm:x:3:4:adm:/var/adm:/sbin/nologin
     5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
     6  sync:x:5:0:sync:/sbin:/bin/sync
     7  shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
     8  halt:x:7:0:halt:/sbin:/sbin/halt
     9  mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
    10  operator:x:11:0:operator:/root:/sbin/nologin
```

搜索/etc/passwd中root用户对应的一行，将'bash'改为'blueshell'，再输出该行：

```
# -n 不自动打印模式空间的内容
[root@node02 ~]# nl /etc/passwd | sed '/^root/{s/bash/blueshell/;p}' /etc/passwd -n
root:x:0:0:root:/root:/bin/blueshell

# 替换后立即退出
[root@node02 ~]# nl /etc/passwd | sed '/^root/{s/bash/blueshell/;q}' /etc/passwd
root:x:0:0:root:/root:/bin/blueshell
```

在两个数字之间添加：符号

```
# -r, --regexp-extended  在脚本中使用扩展正则表达式
[root@node02 tmp]# cat num.txt
789

345
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]#
[root@node02 tmp]# sed -r 's/([0-9])([0-9])([0-9])/\1:\2:\3/' num.txt
7:8:9

3:4:5
```

> 输出ens33网卡ip

```
# 正则锚点用法
[root@node02 tmp]#  ifconfig ens33 | sed -n '/inet\>/p' | sed -e 's/^.*inet //' -e
's/netmask.*$//'
172.18.0.69
```

> 将/etc/passwd第1到第5行中shell为/bin/bash的用户的shell改为'/bin/greenshell'，再输出：

```
# sed执行多次命令的花括号用法
[root@node02 tmp]# sed '1,5{s@/bin/bash@/bin/greenshell@;p}' /etc/passwd -n
root:x:0:0:root:/root:/bin/greenshell
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
```