

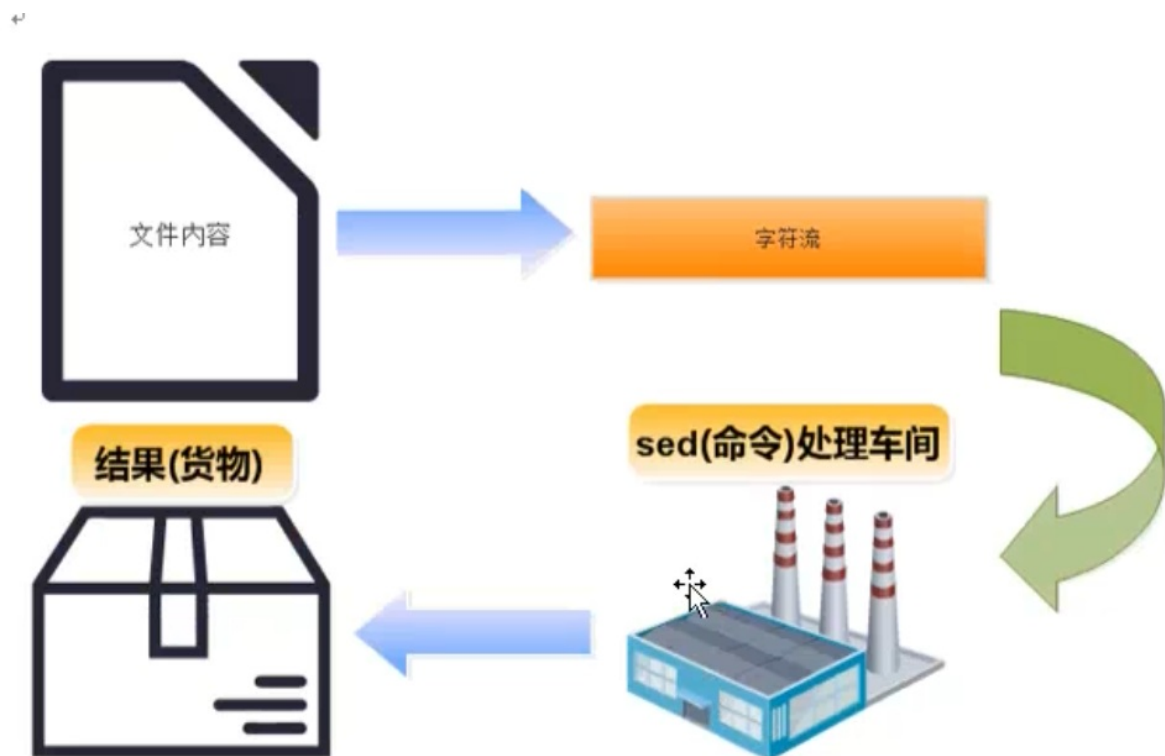
云计算第一册第八节补充内容-sed命令精讲

青鸟云计算教学笔记-Mr.chen

---私人课件，禁止传播

1, 前言

- 我们都知道，在Linux中一切皆文件，比如配置文件，日志文件，启动文件等等。如果我们相对这些文件进行一些编辑查询等操作时，我们可能会想到一些vi,vim,cat,more等命令。但是这些命令效率不高，这就好比一块空地准备搭建房子，请了10个师傅拿着铁锹挖地基，花了一个月的时间才挖完，而另外一块空地则请了个挖土机，三下五除二就搞定了，这就是效率。而在linux中的“挖土机”有三种型号：顶配awk，中配sed，标配grep。使用这些工具，我们能够在达到同样效果的前提下节省大量的重复性工作，提高效率。
- 接下来我们就看一下sed的详细说明
- sed 是Stream Editor（字符流编辑器）的缩写，简称流编辑器。什么是流？大家可以想象以下流水线，sed就像一个车间一样，文件中的每行字符都是原料，运到sed车间，然后经过一系列的加工处理，最后从流水线下下来就变成货物了。



- 当然上图中的文件内容可以是来自文件，也可以直接来自键盘或者管道等标准输入，最后的结果默认

情况下是显示到终端的屏幕上，但是也可以输出到文件中。

- 以前工厂中没有流水线时，生产一件商品需要十几个工种互相配合，这样下来利润太低，后来就有了流水线，生产一件商品虽然还是有十几道工序，但都是机器化生产，工人只是辅助作用，这样利润就大大提高了，产量也大大提高了。
- 编辑文件也是这样，以前我们修改一个配置文件，需要移动光标到某一行，然后添加点文字，然后又移动光标到另一行，注释点东西.....可能修改一个配置文件下来需要花费数十分钟，还有可能改错了配置文件，又得返工。这还是一个配置文件，如果数十个数百个呢？因此当你学会了sed命令，你会发现利用它处理文件中的一系列修改是很有用的。只要想到在大约100多个文件中，处理20个不同的编辑操作可以在几分钟之内完成，你就会知道sed的强大。

2，软件功能与版本

- Sed命令是操作，过滤和转换文本内容的强大工具。常用功能有增删改查（增加，删除，修改，查询），其中查询的功能中最常用的2大功能是过滤（过滤指定字符串），取行（取出指定行）。
- 我们现在准备学习的sed版本是GNU开源版本的，我的实验环境是CentOS6.8系统，内核版本是2.6.32-642.el6.x86_64

```
1. [root@chensiqil ~]# cat /etc/redhat-release
2. CentOS release 6.8 (Final)
3. [root@chensiqil ~]# uname -r
4. 2.6.32-642.el6.x86_64
5. [root@chensiqil ~]# sed --version # 查看sed软件版本
6. GNU sed version 4.2.1
```

3,语法格式

```
1. sed [options] [sed -commands] [input -file]
2. sed [选项] 【sed命令】 【输入文件】
```

说明：

- 1，注意sed软件以及后面选项，sed命令和输入文件，每个元素之间都至少有一个空格。
- 2，为了避免混淆，本文称呼sed为sed软件。sed -commands(sed命令)是sed软件内置的一些命令选项，为了和前面的options（选项）区分，故称为sed命令
- 3，sed -commands 既可以是单个sed命令，也可以是多个sed命令组合。
- 4，input -file (输入文件)是可选项，sed还能够从标准输入如管道获取输入。

4，命令执行流程

概括流程：

Sed软件从文件或管道中读取一行，处理一行，输出一行；再读取一行，再处理一行，再输出一行....

小知识：

一次一行的设计使得sed软件性能很高，sed在读取非常庞大的文件时不会出现卡顿的想象。大家都用过vi命令，用vi命令打开几十M或更大的文件，会发现卡顿现象，这是因为vi命令打开文件是一次性将文件加载到内存，然后再打开，因此卡顿的时间长短就取决于从磁盘到内存的读取速度了。而且如果文件过大的话还会造成内存溢出现象。Sed软件就很好的避免了这种情况，打开速度非常快，执行速度也很快。

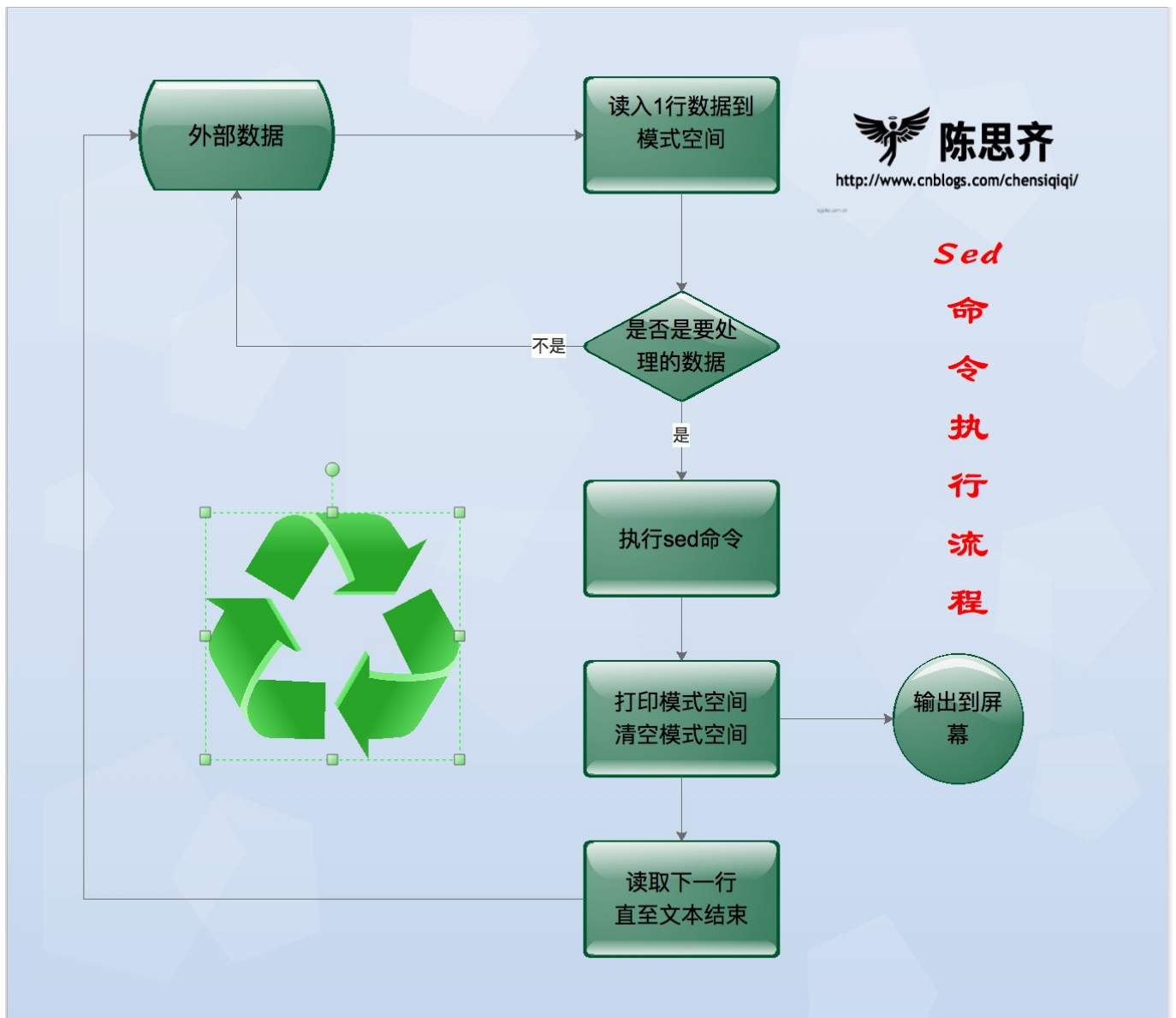
详细流程：

现有一个文件person.txt,共有五行文本，sed命令读入文件person.txt的第一行“101，chensiqi,CEO”，并将这行文本存入模式空间（sed软件在内存中的一个临时缓存，用于存放读取到的内容，比喻为工厂流水线的传送带。）

文件person.txt在模式空间的完整处理流程

- 1，判断第1行是否需要处理的行，如果不是要处理的行就重新从文件读取下一行，如果是要处理的行，则接着往下走。
- 2，对模式空间的内容执行sed命令，比如a（追加），i（插入），s（替换）...
- 3,将模式空间中经过sed命令处理后的内容输出到屏幕上，然后清空模式空间
- 4，读取下一行文本，然后重新执行上面的流程，直到文件结束

上面的流程概括如下图所示：



Sed软件有两个内置的存储空间：

- 模式空间（pattern space）：是sed软件从文本读取一行文本然后存入的缓冲区（这个缓冲区是在内存中的），然后使用sed命令操作模式空间的内容。
- 保持空间（hold space）：是sed软件另外一个缓冲区，用来存放临时数据，也是在内存中，但是模式空间和保持空间的用途是不一样的。Sed可以交换保持空间和模式空间的数据，但是不能在保持空间上执行普通的sed命令，也就是说我们可以在保持空间存储数据。

5，选项说明

option[选项]	解释说明（带*的为重点）
-n	取消默认的sed软件的输出，常与sed命令的p连用。*
-e	一行命令语句可以执行多条sed命令
-f	选项后面可以接sed脚本的文件名
-r	使用扩展正则表达式，默认情况sed只识别基本正则表达式*
-i	直接修改文件内容，而不是输出到终端，如果不使用-i选项sed软件只是修改在内存中的数据，并不会影响磁盘上的文件*

sed - commands[sed命令]	解释说明（带*的为重点）
a	追加，在指定行后添加一行或多行文本*
c	取代指定的行
d	删除指定的行*
D	删除模式空间的部分内容，直到遇到换行符\n结束操作，与多行模式相关
i	插入，在指定行前添加一行或多行文本*
h	把模式空间的内容复制到保持空间
H	把模式空间的内容追加到保持空间
g	把保持空间的内容复制到模式空间
G	把保持空间的内容追加到模式空间
x	交换模式空间和保持空间的内容
l	打印不可见的字符
n	清空模式空间的内容并读入下一行
N	不清空模式空间，并读取下一行数据并追加到模式空间*
p	打印模式空间内容，通常p会与选项-n一起使用*
P（大写）	打印模式空间的内容，直到遇到换行符\n结束操作
q	退出Sed
r	从指定文件读取数据
s	取代，s#old#new#g==>这里g是s命令的替代标志，注意和g命令区分。*

sed - commands[sed命令]	解释说明（带*的为重点）
w	另存，把模式空间的内容保存到文件中
y	根据对应位置转换字符
: label	定义一个标签
b label	执行该标签后面的命令
t	如果前面的命令执行成功，那么就跳转到t指定的标签处，继续往下执行后续命令。否则，仍然继续正常的执行流程

特殊符号	解释说明（带*的为重点）
!	对指定行以外的所有行应用命令*
=	打印当前行行号
~	“First ~ step” 表示从First行开始，以步长Step递增
&	代表被替换的内容
:	实现一行命令语句可以执行多条sed命令*
{}	对单个地址或地址范围执行批量操作
+	地址范围中用到的符号，做加法运算

6，使用范例

6.1 统一实验文本

为了更好的测试sed命令的用法，我们需要准备好下面的测试文件。

```
1. [root@chensiqi1 ~]# cat >person.txt<<KOF
2. > 101,chensiqi,CEO
3. > 102,zhangyang,CTO
4. > 103,Alex,COO
5. > 104,yy,CFO
6. > 105,feixue,CIO
7. > KOF #KOF必须成对出现，表示终止输入
8. 命令说明：使用一条cat命令创建多行文本，文件包含上面的内容，后面的操作都会使用这个文件。
```

6.2 常用功能-增删改查

接下来我们就开始学习sed命令的“四斧头”-->增，删，改，查。大家也不要着急，我们一个一个的学过去，热豆腐要慢慢的吃。

6.2.1 增

- 接下来我来教大家第一式招法-->**往文件指定位置追加或插入指定文本。**
- 这个功能非常有用，比如我们平时往配置文件写入几行文本，最常用的是vi或vim命令，但是这2个命令是一种交互式的命令，还需要我们在vi/vim编辑器界面输入字符串然后保存退出，操作有些繁琐但是还能用。但是当我们学会了Shell脚本后，我们就会发现在脚本中不能正常使用vi或vim命令，为什么呢？同学们请自行体验。
- 我们学习Shell脚本主要是为了解放我们的双手，执行一个脚本，然后自动往文件中写入数据，不需要我们再动手。因此我们想到了sed软件，它能够帮助我们实现目的。
- 这里我们需要用到2个sed命令，分别是：
 - “a” :追加文本到指定行后，记忆方法：a的全拼是append，意思是追加。
 - “i ”：插入文本到指定行前，记忆方法：i的全拼是insert，意思是插入。

6.2.1.1 单行增

首先我们看一下单行增加的用法，说白了就是在文件中增加一行文本，我们以前学过echo命令可以在文件的末尾追加文本，比较简单，但是我们还有其他的复杂需求，比如在第10行插入一行数字等等，这里就需要sed出马了。我们来看一下下面的例子，同学也跟着例子一起练习，命令只有多练才会熟悉，光看永远是学不会的。

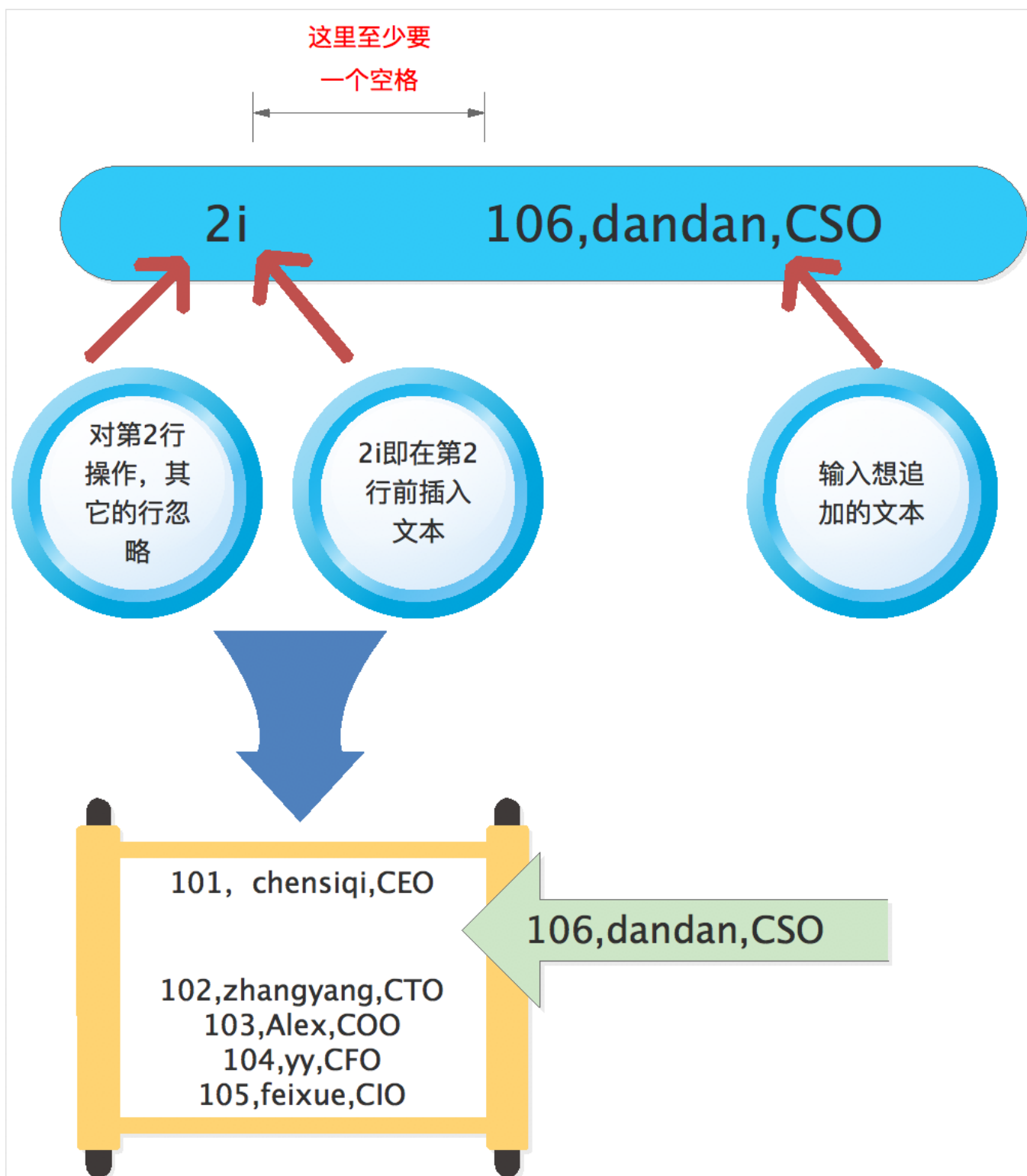
```
1. [root@chensiqil ~]# sed '2a 106,dandan,CSO' person.txt
2. 101,chensiqi,CEO
3. 102,zhangyang,CTO
4. 106,dandan,CSO    #这就是新增那句
5. 103,Alex,COO
6. 104,yy,CFO
7. 105,feixue,CIO
```

命令行详解：

首先我们看一下命令的结果，我们可以看到原来的第二行“102,zhangyang,CTO”前面插入了新的一行为“106,dandan,CSO”，原来的第2行变成第3行了。

接下来我们解读一下sed语句的结构，sed打头，然后接上空格（空格个数不限，但至少要是有一个！）在空格后面，我们先敲上一对单引号（"）然后退格在单引号中协商‘106,dandan,CSO’。

- 2代表指定对第2行操作，其他的行忽略
- i代表插入的意思，2i即在第2行前插入文本
- 2i后面加上空格，然后跟上你想要插入的文本即可



最后接上你想要处理的文件person.txt

6.2.1.2 引号的区别总结

在教学过程中，有同学问我在sed中到底使用单引号还是双引号？这里给大家详细说说引号的区别。

1. 双引号：把双引号的内容输出出来；如果内容中有命令，变量等，会先把命令，变量解析出结果，然后再输

出最终内容来。双引号内命令或变量的写法为`命令或变量`或\$(命令或变量)

2. 单引号：所见即所得，将单引号内的内容原样输出，阻止所有字符的转义
3. 不加引号：不会将含有空格的字符串视为一个整体输出，如果内容中有命令，变量等，会先把命令，变量解析出结果，然后再输出最终内容来，如果字符串含有空格等特殊字符，则不能完整输出，则需改加双引号。
4. 倒引号（反引号Esc键下方）：进行命令的替换，在倒引号内部的shell命令将会被执行，其结果输出代替用倒引号括起来的文本。

Sed为何用单引号？

```
1. [root@chensiqi1 ~]# cat person.txt
2. 101,chensiqi,CEO
3. 102,zhangyang,CTO
4. 103,Alex,COO
5. 104,yy,CFO
6. 105,feixue,CIO
7. [root@chensiqi1 ~]# sed '2i $PATH' person.txt #单引号--文本内容原封不动插入
8. 101,chensiqi,CEO
9. $PATH
10. 102,zhangyang,CTO
11. 103,Alex,COO
12. 104,yy,CFO
13. 105,feixue,CIO
14. [root@chensiqi1 ~]# sed 2i $PATH person.txt #不加引号，linux无法辨认空格，不会把有空格的命令当成一条命令来执行
15. sed: -e expression #1, char 2: expected \ after `a', `c' or `i'
16. [root@chensiqi1 ~]# sed "2i $PATH" person.txt #双引号--变量$PATH被解析以后在当作文本进行插入
17. 101,chensiqi,CEO
18. /usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
19. 102,zhangyang,CTO
20. 103,Alex,COO
21. 104,yy,CFO
22. 105,feixue,CIO
```

结论一：

- 如果引号里面是普通字符串的话，你可以任意使用单引号或者双引号：
- 如果引号里面是变量或者带反引号的命令的话，你想要变量解析的结果或者命令执行的结果，那就使用双引号；你想要引号内的原样字符串，那就使用单引号。
- 只要大家理解上面的用法，明白自己想要什么，那么你想用什么引号就用什么引号。
- 其实就二选一，非此即彼，这个不行就试试那个。

结论二：

1. 很多时候，总结的结论未必适合所有情况。但是，总结小结论，便于我们学习知识和对知识的深刻理解，因此推荐大家多做总结。

2. 不要太纠结于结论，结论毕竟不是真理，只是我们学习过程的产物。就像数学里的平行线一样，在初中定义就是不相交的两条直线就是平行线，到了高中结论就是错的了。

6.2.1.3 多行增

前面学习的内容已经实现了往文件追加或插入单行文本，但是还有插入多行文本的需求，我们也学习过cat命令能够往文件中追加多行文本。

示例准备，请执行如下完整命令生成test.txt文件内容：

```
1. [root@chensiqi1 ~]# cat > test.txt <<EOF
2. > welcome to my blog.http://www.cnblogs.com/chensiqiqi/
3. >
4. > if you like my blog\'s contents,pls support me.
5. >
6. >
7. >
8. > bye!boys and girls.
9. > EOF
10. 命令说明：以上就是cat的生产环境常用的生成或为文件追加内容的方法，请不要忽略上文中的空行。另外，如果内容中有单引号，$符号等特殊符号作为内容时要用\"转义。
```

以上命令执行过程及结果：

```
1. [root@chensiqi1 ~]# cat > test.txt <<EOF
2.
3. > welcome to my blog.http://www.cnblogs.com/chensiqiqi/
4. > #此行是空行
5. > if you like my blog\'s contents,pls support me.
6. > #此行是空行
7. > #此行是空行
8. > #此行是空行
9. > bye!boys and girls.
10. > EOF #-->这里要敲回车才能结束，另外，EOF必须成对出现，但也可以用别的成对标签替换，例如：chensiqi
    字符标签。
11. 命令说明：上面命令结果最左侧的">"是Linux系统自动生成的，不是我们手动敲的。
```

- ok,我们前面已经简单的回顾了cat命令的用法，如果大家想详细了解cat命令的详细用法，请持续关注我的博客。
- 前面我们实现了用cat命令往文件中追加多行文本，但是cat命令和echo命令有同样的缺点，只能是往文末添加内容，并不能够在指定行操作，因此还是需要我们的sed命令出马。
- 首先我们需要了解一个特殊符号“\n”，这个符号叫做换行符，顾名思义换行。我们看几个例子来认识一下这个符号。

```
1. [root@chensiqi1 ~]# echo "chensiqi";echo "chensiqi"
```

```
2. chensiqi
3. chensiqi
4.
5. 命令说明：上面的命令用分号“;”连接了两条命令，然后输出结果为2行chensiqi，但是有没有更简单的方法呢？
6. [root@chensiqi1 ~]# echo -e "chensiqi\nchensiqi"
7. chensiqi
8. chensiqi
9. 命令说明：这里就“\n”派上用场了，行与行之间是以“\n”作为分隔符的，所以“chensiqi\nchensiqi”就等效于
    2行chensiqi。接下来我们用echo命令实验一下，其中-e参数表示字符串中如果出现以下特殊字符（\n代表换行
    ，\t代表Tab键等），则加以特殊处理，而不会将它当成一般文字输出。如果大家想详细了解echo命令的详细用法
    ，请持续关注我的博客。
```

学习完换行符，我们来看一下sed命令如何使用

！

```
1. [root@chensiqi1 ~]# sed '2a 106,dandan,CSO\n107,bingbing,CCO' person.txt
2. 101,chensiqi,CEO
3. 102,zhangyang,CTO
4. 106,dandan,CSO    #同时追加多行
5. 107,bingbing,CCO  #同时追加多行
6. 103,Alex,COO
7. 104,yy,CFO
8. 105,feixue,CIO
9. EOF
```

命令行分析：

- 我们首先粗略扫一眼这个命令语句，可以发现命令结构是和单行增加文本是几乎没有区别的。
- 然后我们看一下命令的结果，我们可以看到原来的第二行“102，zhangyang，CTO”后面追加了2行文本“106，dandan，CSO”和“107，bingbing，CCO”。
- 接下来我们解读一下sed语句的结构，sed软件打头，然后接上空格，在空格后面，我们先敲上一对单引号（“”），然后退一格在单引号中写上‘2a 106,dandan,CSO\n107,bingbing,CCO’。

1. 2代表指定对第2行操作，其他的行忽略；
2. a代表追加的意思，2a即在第2行后追加文本；
3. 2a后面加上空格，然后跟上你想要插入的多行文本即可。这里的每行文本使用“\n”连接就可以写成一行了。

这里至少要
一个空格

2a

106,dandan,CSO\n107,bingbing,CCO

对第2行
操作，其
它的行忽
略

2a即在第
2行后插
入文本

输入想追
加的文本

101, chensiqi,CEO
102,zhangyang,CTO

103,Alex,COO
104,yy,CFO
105,feixue,CIO

106,dandan,CSO\n107,bingbing,CCO

在终端输出时遇
到换行符会自动
换行

101, chensiqi,CEO
102,zhangyang,CTO

103,Alex,COO
104,yy,CFO
105,feixue,CIO

106,dandan,CSO
107,bingbing,CCO

最后接上你想要处理的文件person.txt

当然还有另外一种方法添加多行文本，但这种方法并没有“\n”方便，所以在这里就简单说一下。这种方法利用了“\”，它也有换行的意思。如果大家在执行一行很长的命令时候，如果都写在一行，那太难看了也难以理解，因此就利用到了这个符号可以将一条完整的命令分成多行，举个例子：

```
1. [root@chensiqi1 ~]# echo chensiqi
2. chensiqi
3. [root@chensiqi1 ~]# echo \
4. > chensiqi
5. chensiqi
6. [root@chensiqi1 ~]# echo \
7. > chensiqi \
8. > is \
9. > me
10. chensiqi is me
11.
12. 命令说明：一条命令可以通过\符号进行多行输出
```

接着说一下sed命令使用反斜线，首先输入完“sed '2a 106,dandan,CSO \”，然后敲回车键，这样窗口会显示一个符号“>”，我们在这个符号后面接着写命令的剩余部分“107,bingbing,CCO ‘person.txt”。

```
1. [root@chensiqi1 ~]# sed '2a 106,dandan,CSO \
2. > 107,bingibng,CCO' \
3. > person.txt
4. 101,chensiqi,CEO
5. 102,zhangyang,CTO
6. 106,dandan,CSO
7. 107,bingibng,CCO
8. 103,Alex,COO
9. 104,yy,CFO
10. 105,feixue,CIO
11. EOF
12. 命令说明：结果和第一种方法是一样的，但是我们也发现这种方法比较麻烦，因此建议大家使用第一种方法。
```

- sed软件使用命令i插入多行文本和命令和a的用法是一样的，因此这里不再详细列出，大家可以练习一下，就是把前面命令的“a”换成“i”。
- 学习了这么多内容，总要有用武之地，因此大家不要看答案先独立完成下面的企业案例！

企业案例1:优化SSH配置（一键完成增加若干参数）

在我们学习CentOS6系统优化时，有一个优化点：更改ssh服务远程登录的配置。主要的操作是在ssh的配置文件/etc/ssh/sshd_config加入下面5行文本。（下面参数的具体含义见其他课程。）

```
1. Port 52113
2. PermitRootLogin no
```

```
3. PermitEmptyPasswords no
4. UseDNS no
5. GSSAPIAuthentication no
```

当然我们可以使用vi/vim命令编辑这个文本，但是这样就比较麻烦，现在想用一条命令增加5行文本到第13行前？

注意：修改前别忘了备份配置文件：cp /etc/ssh/sshd_config{,.bak}

上面的命令知识要是不懂请看通配符章节

这道企业面试题可以用我们学过的sed命令多行追加功能就可以搞定。

```
1. [root@chensiqil ~]# sed -i '13i Port 52113\nPermitRootLogin no\nPermitEmptyPasswords
2. no\nUseDNS no\nGSSAPIAuthentication no' /etc/ssh/sshd_config
3. 命令说明：题目要求在第13行前插入，那就需要使用命令13i。有同学做个题目时，是这样想的，在13行前，那不
   就是12行后吗，12a也是可以的。是的，这样也是没错的，这可以算是第二种方法。
4. 最后插入的5行内容使用"\n"就可以变成一行了。
5. 上面还有一个没讲过的选项"-i"，这个选项能够实际的修改文件内容，大家练习时可以去掉，防止改掉了配置文件。
   如果使用了-i，可以用备份文件还原。当然，在生产环境修改配置文件那就需要用-i选项了。
6. [root@chensiqil ~]# sed -n '13,17p' /etc/ssh/sshd_config
7. Port 52113
8. PermitRootLogin no
9. PermitEmptyPasswords no
10. UseDNS no
11. GSSAPIAuthentication no
12.
13. 命令说明：查看增加的文本内容，选项-n与命令p的具体用法见后文的6.2.4查。
```

6.2.2 删

- 前面给大家演示了第一招，是不是很好用。如果没看懂，请再看一遍。如果学会了，那就跟着我来学习第2招--->删除指定行文本。
- 这个功能也是非常得有用，比如我们想删除文件中的某些行，以前最常用的是vi或vim命令，但现在我们知道了sed命令，就应该使用这个高逼格的命令完成任务了。
- 这里我们需要用到1个sed命令；
- “d”：删除文本，记忆方法：d的全拼是delete，意思是删除。
- 因为删除功能比较简单，因此我们结合地址范围一起说明。我们前面学过sed命令可以对一行文本为目标进行处理（在单行前后增加一行或多行文本），接下来我们看一下如何对多行文本为目标操作。

6.2.2.1指定执行的地址范围

sed软件可以对单行或多行文本进行处理。如果在sed命令前面不指定地址范围，那么默认会匹配所有行。

用法：n1[,n2]{sed -commands}

地址用逗号分隔开，n1,n2可以用数字，正则表达式，或者二者的组合表示。

地址范围	含义
10{sed-commands}	对第10行操作
10, 20{sed-commands}	对10到20行操作，包括第10，20行
10, +20{sed-commands}	对10到30 (10+20) 行操作，包括第10，30行
1 ~ 2{sed-commands}	对1, 3, 5, 7.....行操作
10, \${sed-commands}	对10到最后一行 (\$代表最后一行) 操作，包括第10行
/chensiqi/{sed-commands}	对匹配chensiqi的行操作
/chensiqi/,/Alex/{sed-commands}	对匹配chensiqi的行到匹配Alex的行操作
/chensiqi/\${sed-commands}	对匹配chensiqi的行到最后一行操作
/chensiqi/,10{sed-commands}	对匹配chensiqi的行到第10行操作， 注意：如果前10行没有匹配到chensiqi， sed软件会显示10行以后的匹配chensiqi的行
1,/Alex/{sed-commands}	对第1行到匹配Alex的行操作
/chensiqi/,+2{sed-commands}	对匹配chensiqi的行到其后的2行操作

下面用具体的例子演示一下，测试文件还是person.txt

1. [root@chensiqi1 ~]# sed 'd' person.txt
2. [root@chensiqi1 ~]#
- 3.
4. 命令说明：如果在sed命令前面不指定地址范围，那么默认会匹配所有行，然后使用d命令删除功能就会删除这个文件的所有内容

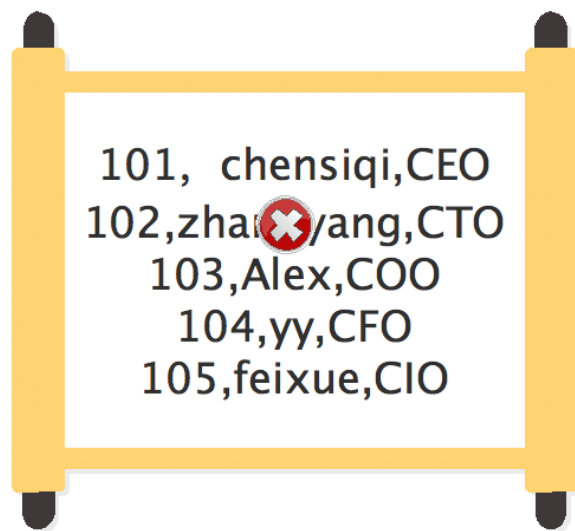


默认等效于第1行
到最后一行，因此
全部删除

'd'

```
1. [root@chensiqi1 ~]# sed '2d' person.txt
2. 101,chensiqi,CEO
3. 103,Alex,COO
4. 104,yy,CFO
5. 105,feixue,CIO
6.
```

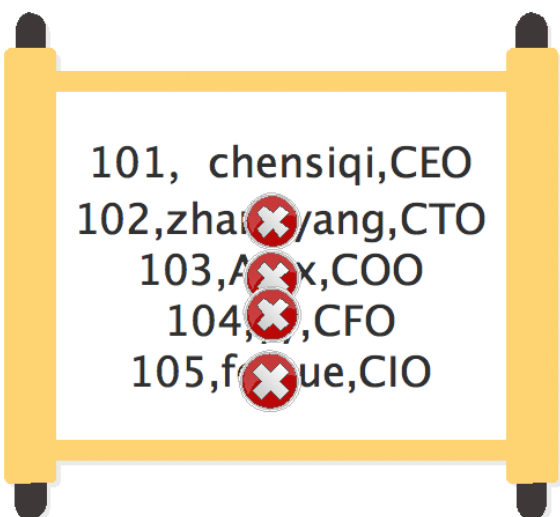
7. 命令说明：这个单行删除想必大家能理解，指定删除第2行的文本102, zhangyang, CTO



指定删除第2行的
文本

'2d'

1. [root@chensiqi1 ~]# sed '2,5d' person.txt
2. 101,chensiqi,CEO
- 3.
4. 命令说明：'2, 5d' 指定删除第2行到第5行的内容，d代表删除操作。

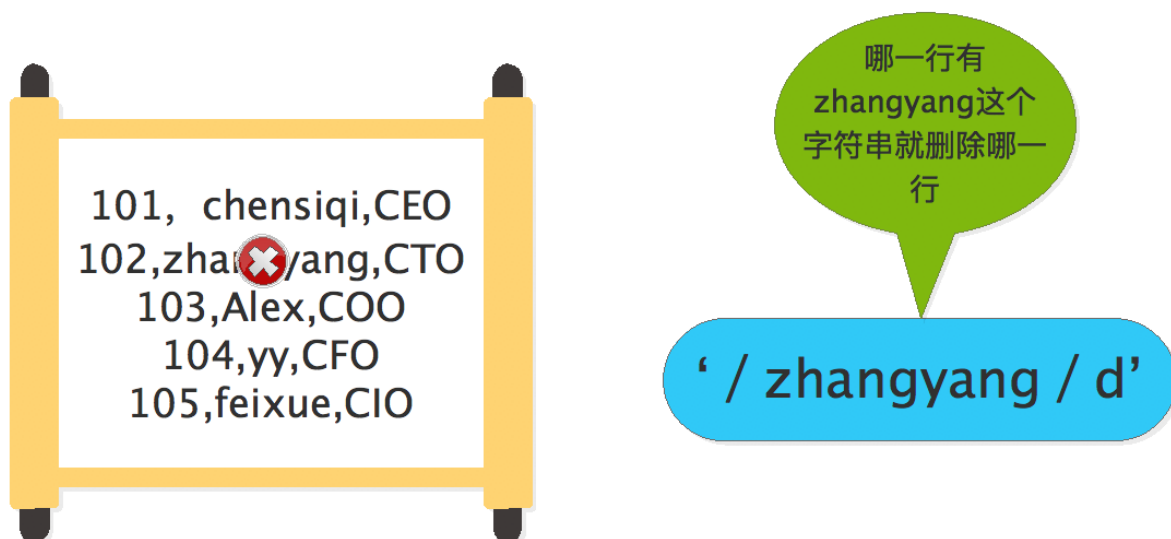


指定删除第2行到
第5行的文本

'2, 5d'

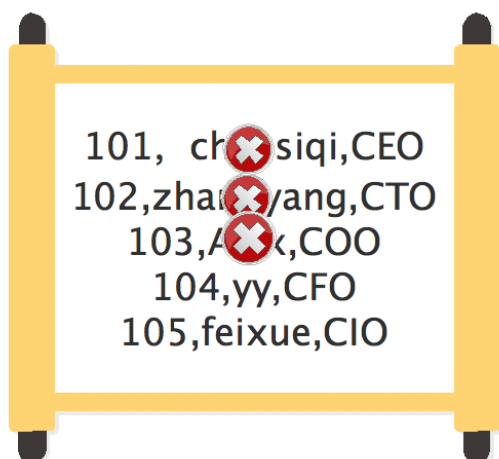
ok，上面我们实验完了数字地址范围，接下来我们实验一下正则表达式的地址范围，虽然说可以使用正则表达式，但是我们还是习惯写出完整的匹配字符串，达到精确匹配的目的。

```
1. [root@chensiqi1 ~]# sed '/zhangyang/d' person.txt
2. 101,chensiqi,CEO
3. 103,Alex,COO
4. 104,yy,CFO
5. 105,feixue,CIO
6.
7. 命令说明：
8. 在sed软件中，使用正则的格式和awk一样，使用2个"/"包含指定的正则表达式，即"/正则表达式/"。
```



当然也可以使用两个正则表达式，如下例所示。

```
1. [root@chensiqi1 ~]# sed '/chensiqi/,/Alex/d' person.txt
2. 104,yy,CFO
3. 105,feixue,CIO
4.
5. 命令说明：这是正则表达式形式的多行删除，也是以逗号分隔2个地址，最后结果是删除包含"chensiqi"的行到包含"Alex"的行
```

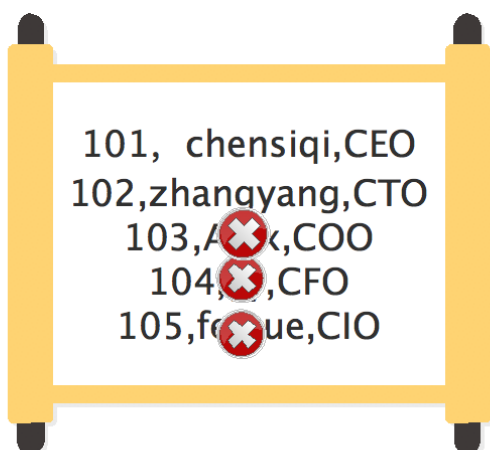


删除匹配chensiqi
字符串的行至匹配
Alex字符串的行

' / chensiqi / , / Alex / d '

```
1. [root@chensiqi1 ~]# sed '3,$d' person.txt
2. 101,chensiqi,CEO
3. 102,zhangyang,CTO
4.
```

5. 命令说明：学过正则表达式后我们知道“\$”代表行尾，但是在sed中就有一些变化了，“\$”在sed中代表文件的最后一行。因此本例子的含义是删除第3行到最后一行的文本，包含第3行和最后一行，因此剩下第1, 2行的内容。



从第3行开始删，
直到删除最后一
行。

'3, \$d'

接下来我们看一些特殊情况。

```
1. [root@chensiqi1 ~]# sed '/chensiqi/,3d' person.txt
2. 104,yy,CFO
3. 105,feixue,CIO
```

```

4.
5. 命令说明：这个例子是删除包含“chensiqi”的行到第3行的内容。但这种组合有一个比较特殊的情况，如果前3行之
   外还有这个“chensiqi”字眼，sed软件还是会找他“麻烦”，请看下面例子。
6.
7. [root@chensiqi1 ~]# sed '$a 106,chensiqi,CMO' person.txt
8. 101,chensiqi,CEO
9. 102,zhangyang,CTO
10. 103,Alex,COO
11. 104,yy,CFO
12. 105,feixue,CIO
13. 106,chensiqi,CMO
14.
15. 命令说明：为了不造成同学们实验文本改来改去导致不同意，因此我用上面的命令语句只是临时修改内存数据，然后
   通过管道符号传给sed软件。
16.
17. [root@chensiqi1 ~]# sed '$a 106,chensiqi,CMO' person.txt | sed '/chensiqi/,3d'
18. 104,yy,CFO
19. 105,feixue,CIO
20.
21. 命令说明：从命令结果我们可以看到，不仅是第1行（101, chensiqi, CEO）到第3行（103, ALex, COO）被删
   除了，而且最后一行（106, chensiqi, CMO）也被删除了。因此我们可以得出一个小结论，sed软件使用正则表
   达式会找出所有匹配的行，即使是有数字地址限制。

```

再来看一个特殊情况。

```

1. [root@chensiqi1 ~]# sed '2,/O/d' person.txt
2. 101,chensiqi,CEO
3. 104,yy,CFO
4. 105,feixue,CIO
5.
6. 命令说明：
7. 从第2行开始删除到含字母O的行结束，但是我们发现第3, 4, 5行都含有字母O，命令结果显示只删除了第2, 3
   行，属于最短删除。这个怎么理解？
8. 还是可以从上面命令执行流程图理解，从第2行开始循环，sed软件第一次遇到字母O（第三行）就认为循环结
   束了。
9. [root@chensiqi1 ~]# sed '2,/o/d' person.txt
10. 101,chensiqi,CEO
11.
12. 命令说明：
13. 从第2行开始删除，但是后面文本没有字母o，因此一直循环下去，直到文本结束，sed软件自动终止。

```

大家看完上面的例子是不是一脸懵逼，其实很简单。在工作中我们最常用的还是数字地址这种精确匹配方式，像上面的正则地址或混合地址这种模糊匹配用的比较少，了解即可。

6.2.2.2 特殊符号~（步长）解析

格式：“First~step”表示从开始，以步长step递增，这个在数学中叫做等差数列

例子：

- 1~2 匹配1, 3, 5, 7.....#-->用于只输出奇数行, 大伙仔细观察一下每个数字的差值。
- 2~2 匹配2, 4, 6, 8.....#-->用于只输出偶数行
- 1~3 匹配1, 4, 7, 10.....
- 2~3 匹配2, 5, 8, 11.....

```
1. [root@chensiqi1 ~]# seq 10
2. 1
3. 2
4. 3
5. 4
6. 5
7. 6
8. 7
9. 8
10. 9
11. 10
12.
```

命令说明: seq命令能够生成从1到10的数字序列。

```
13.
14.
15. [root@chensiqi1 ~]# seq 10 | sed -n '1~2p'
16. 1
17. 3
18. 5
19. 7
20. 9
21.
```

命令说明:

上面的命令主要验证特殊符号“~”的效果, 其他sed命令用法n和p请见后文详解, 大家只需要知道这个命令可以将“1~2”指定的行显示出来即可。

上面例子测试了“1~2”的效果, 大家也可以手动测试一下“2~2”, “1~3”, “2~3”, 看一下他们的结果是不是符合等差数列。

补充小知识:

如果大家想生成奇数数列, 其实上面的方法是为了举例, 并不是一个很好的方法, 因为seq命令自带这种功能。

```
1. [root@chensiqi1 ~]# seq 1 2 10
2. 1
3. 3
4. 5
5. 7
6. 9
7.
8. 命令说明: seq命令格式seq起始值 公差 结束值
```

再来一例:

```
1. [root@chensiqi1 ~]# sed '1~2d' person.txt
2. 102,zhangyang,CTO
3. 104,yy,CFO
4.
5. 命令说明：“1~2”这是指定行数的另一种格式，从第1行开始以步长2递增的行（1，3，5），因此删掉第1，3，5行，即所有的奇数行。
```

6.2.2.3 特殊符号+解析

```
1. [root@chensiqi1 ~]# sed '1,+2d' person.txt
2. 104,yy,CFO
3. 105,feixue,CIO
4.
5. 命令说明：这其实是做个加法运算，‘1, +2d’==>删除第1行到第3（1+2）行的文本。
```

上面是特殊符号“+”的用法，大家知道即可。

6.2.2.4 特殊符号！解析

感叹号“！”我们在很多命令里都接触过，大部分都是取反的意思，在sed中也不例外。

```
1. [root@chensiqi1 ~]# sed '2,3!d' person.txt
2. 102,zhangyang,CTO
3. 103,Alex,COO
4.
5. 命令说明：在地址范围“2, 3”后面加上“
6. ！”，如果不加“！”表示删除第2行和第3行，结果如下面的例子所示，然后加上“！”的结果就是除了第2行和第3行
   以外的内容都删除，这个方法可以作为显示文件的第2, 3行题目的补充方法。
7.
8. [root@chensiqi1 ~]# sed '2,3d' person.txt
9. 101,chensiqi,CEO
10. 104,yy,CFO
11. 105,feixue,CIO
```

企业案例2:打印文件内容但不包含chensiqi

这是一道企业面试题，很简单，就是把包含chensiqi字符串的行删除掉即可。我们可以用“grep -v”取反得到我们想要的结果，但是这里我们使用sed软件实现。

```
1. [root@chensiqi1 ~]# sed '/chensiqi/d' person.txt
2. 102,zhangyang,CTO
3. 103,Alex,COO
4. 104,yy,CFO
5. 105,feixue,CIO
6.
7. 命令说明：删除包含“chensiqi”的行，就直接用正则匹配字符串chensiqi即可
```

6.2.3 改

大家现在已经学完了sed软件2大招式，是不是很厉害！当然后面还有更厉害的--->改.在学习linux时，我们最常见的操作就是改配置文件，改参数等等，而且更妙的是前面我们学习过的增加和删除其实都可以用我们准备要学的修改变相实现。

6.2.3.1 按行替换

首先说一下按行替换，这个功能用的很少，所以大家了解即可。这里用到的sed命令是：

“c”：用新行取代旧行，记忆方法：c的全拼是change，意思是替换。

例子：

```
1. [root@chensiqi1 ~]# sed '2c 106,dandan,CSO' person.txt
2. 101,chensiqi,CEO
3. 106,dandan,CSO
4. 103,Alex,COO
5. 104,yy,CFO
6. 105,feixue,CIO
7.
8. 命令说明：使用sed命令c将原来第2行“102, zhangyang, CTO”替换成“106, dandan,CSO”，整行替换
```

6.2.3.2 文本替换

- 接下来说的这个功能，有工作经验的同学应该非常的熟悉，因为使用sed软件80%的场景就是使用替换功能。
- 这里用到的sed命令，选项：
 - “s”：单独使用-->将每一行中第一处匹配的字符串进行替换==>sed命令
 - “g”：每一行进行全部替换-->sed命令s的替换标志之一（全局替换），非sed命令。
 - “-i”：修改文件内容-->sed软件的选项，注意和sed命令i区别。

sed软件替换模型

```
sed -i 's/目标内容/替换内容/g' chensiqi.log
```

```
sed -i 's#目标内容#替换内容#g'
```

观察特点

1，两边是引号，引号里面的两边分别为s和g，中间是三个一样的字符/或#作为定界符。字符#能在替换内容包含字符/有助于区别。定界符可以是任意字符如：或|等，但当替换内容包含定界符时，需要转义\：或|.经过长期实践，建议大家使用#作为定界符。

2，定界符/或#，第一个和第二个之间的就是被替换的内容，第二个和第三个之间的就是替换后的内容。

3，s#目标内容#替换内容#g，“目标内容”能用正则表达式，但替换内容不能用，必须是具体的。因为替换内

容使用正则的话会让sed软件无所适从，它不知道你要替换什么内容。

4，默认sed软件是对模式空间（内存中的数据）操作，而-i选项会更改磁盘上的文件内容。

例子：

```
1. [root@chensiqi1 ~]# sed 's#zhangyang#dandan#g' person.txt
2. 101,chensiqi,CEO
3. 102,dandan,CTO
4. 103,Alex,COO
5. 104,yy,CFO
6. 105,feixue,CIO
7.
8. 命令说明：将需要替换的文本“zhangyang”放在第一个和第二个“#”之间，将替换后的文本“dandan”放在第二个核第三个“#”之间。结果为第二行的“zhangyang”替换为“dandan”。
```

修改文件：

```
1. [root@chensiqi1 ~]# cat person.txt
2. 101,chensiqi,CEO
3. 102,zhangyang,CTO
4. 103,Alex,COO
5. 104,yy,CFO
6. 105,feixue,CIO
7.
8. 命令说明：从上面命令的结果我们就知道sed命令默认不会修改文件的内容
9.
10. [root@chensiqi1 ~]# sed -i 's#zhangyang#dandan#g' person.txt
11.
12. 命令说明：如果想真正的修改文件内容，我们就需要使用选项“-i”，这个要和sed命令“-i”区分开来。同时我们可以发现命令执行后的结果是没有任何输出的。
13.
14. [root@chensiqi1 ~]# cat person.txt
15. 101,chensiqi,CEO
16. 102,dandan,CTO
17. 103,Alex,COO
18. 104,yy,CFO
19. 105,feixue,CIO
20.
21. 命令说明：当我们再次查看这个文件时，我们就发现这个文件已经被修改成功了。因此大家以后如果使用替换功能时，应该首先不用选项“-i”测试一下，确保操作无误，最后使用“-i”修改文件
```

还原测试文件：

```
1. [root@chensiqi1 ~]# sed -i 's#dandan#zhangyang#g' person.txt
2.
3. 命令说明：还原测试文件，这一步大家不要忘了执行，不然后面就跟不上步骤了
```

企业案例3：指定行修改配置文件

前面我们学过的模型能够将文件中所有满足条件的文本进行操作，但是我们也会碰到指定行精确修改配置文件的需求，因为这样可以避免修改多了地方。

```
1. [root@chensiqi1 ~]# sed '3s#0#9#' person.txt
2. 101,chensiqi,CEO
3. 102,zhangyang,CTO
4. 193,Alex,COO
5. 104,yy,CFO
6. 105,feixue,CIO
7.
8. 命令说明：
9.     前面学习的例子在sed命令"s"前没有指定地址范围，因此默认是对所有行进行操作。
10.    而这个案例要求只将第3行的0换成9，这里就用到了我们前面学过的地址范围知识，在sed命令"s"前加上"3"
    就代表对第3行进行替换
```

6.2.3.3 变量替换

变量替换其实和前面的文本替换是一样的，就是具体的文本变成了变量，同时要求大家对引号的用法要有清晰的理解，因此对双引号和单引号的区别不太理解的同学，请往前面翻页到【引号的区别总结】再次复习一下。

```
1. [root@chensiqi1 ~]# cat >test.txt<<KOF #再新建一个简单的测试文本
2. > a
3. > b
4. > a
5. > KOF
6. [root@chensiqi1 ~]# cat test.txt
7. a
8. b
9. a
10. [root@chensiqi1 ~]# x=a #->设置变量x并 赋值a
11. [root@chensiqi1 ~]# y=b #-> 设置变量y并赋值b
12. [root@chensiqi1 ~]# echo $x
13. a
14. [root@chensiqi1 ~]# echo $x $y
15. a b
16.
17. 命令说明：打印变量x, y验证一下，需要使用$符号引用变量
```

不使用引号：

```
1. [root@chensiqi1 ~]# sed s#$x#$y#g test.txt
2. b
3. b
4. b
5.
6. 命令说明：使用变量进行替换，从执行结果中我们可以发现替换成功了，test.txt文件中的所有a都替换成了b。同时我们可以发现s#$x#$y#g没有使用引号，当然这种写法并不是特别标准。
```

```

7.
8. [root@chensiqi1 ~]# sed 's#'$x'#$y'#g' test.txt
9. b
10. b
11. b
12.
13. 命令说明：表面看起来单引号是可以用的，但其实这里用了障眼法，在你们眼中分段 '$x' 和 '$y'，但其实分段是 's#
    '和 '#'和 '#g'，所以$x和$y并没有被引号扩起来，和上面的例子就一样了。

```

使用eval命令：

```

1. [root@chensiqi1 ~]# eval sed 's#$x#$y#g' test.txt
2. b
3. b
4. b
5.
6. 命令说明：这里给大家扩展一个Linux内置命令eval，这个命令能读入变量，并将他们组合成一个新的命令，然后
    执行。首先eval会解析变量$x和变量$y，最后达到的效果和双引号是一样的。

```

扩展：最快速的获取IP地址的方法

```

1. [root@chensiqi1 ~]# hostname -I
2. 192.168.197.133

```

6.2.3.4 分组替换()和\1的使用说明

sed软件的()的功能可以记住正则表达式的一部分，其中，\1为第一个记住的模式即第一个小括号中的匹配内容，\2第二个记住的模式，即第二个小括号中的匹配内容，sed最多可以记住9个。

例：echo "I am chensiqi teacher."如果想保留这一行的单词chensiqi，删除剩下部分，使用圆括号标记想保留的部分。

```

1. [root@chensiqi1 ~]# echo "I am chensiqi teacher." | sed 's#^.*am \([a-z]\+\) tea.*$#\1#g'
2. chensiqi
3.
4. [root@chensiqi1 ~]# echo "I am chensiqi teacher." | sed -r 's#^.*am ([a-z]+) tea.*$#\1#g'
5. chensiqi
6.
7. [root@chensiqi1 ~]# echo "I am chensiqi teacher." | sed -r 's#I (.*) (.*) teacher.#\1\2#g'
8. amchensiqi
9.
10. 命令说明：
11. sed如果不加-r后缀，那么默认不支持扩展正则表达式，需要\符号进行转义。小括号的作用是将括号里的匹配内容进行分组以便在第2和第3个#号之间进行sed的反向引用，\1代表引用第一组，\2代表引用第二组

```

再来看个题目：请执行命令取出linux中的eth0的IP地址？

```
1. [root@chensiqi1 ~]# ifconfig eth0 | sed -n '2p'
2.         inet addr:192.168.197.133   Bcast:192.168.197.255   Mask:255.255.255.0
3. [root@chensiqi1 ~]# ifconfig eth0 | sed -n '2p' | sed -r 's#^.*addr:(.*)
4. Bcast:.*$#\1#g'
5. 192.168.197.133
6. 也可以进行组合
7.
8. [root@chensiqi1 ~]# ifconfig eth0 | sed -rn '2s#^.*addr:(.*) Bcast:.*$#\1#gp'
9. 192.168.197.133
10.
11. 命令说明：
12. 这道题是需要把ifconfig eth0执行结果的第2行的IP地址取出来，上面答案的思路是用IP地址来替换第2行的内容。
```

企业案例4:系统开机启动项优化（利用sed）

```
1. [root@chensiqi1 ~]# chkconfig --list | egrep -v "sshd|crond|rsyslog|sysstat|network"
2. | awk '{print $1}' | sed -r 's#^(.*)#chkconfig \1 off#g' | bash
3. 这题也可以应用awk直接一步到位
4. [root@chensiqi1 ~]# chkconfig --list | egrep -v "sshd|crond|network|rsyslog|sysstat"
5. | awk '{print "chkconfig",$1,"off"}' | bash
```

6.2.3.5 特殊符号&代表被替换的内容

这是一个特殊技巧，在适合的场景使用特别方便。下面用特殊符号“&”与分组替换一起使用，进行对比。

```
1. [root@chensiqi1 ~]# sed -r 's#(.*) (.*) (.*)#& ----- \1 \2 \3#' person.txt
2. 101,chensiqi,CEO ----- 101 chensiqi CEO
3. 102,zhangyang,CTO ----- 102 zhangyang CTO
4. 103,Alex,COO ----- 103 Alex COO
5. 104,yy,CFO ----- 104 yy CFO
6. 105,feixue,CIO ----- 105 feixue CIO
7.
8. 命令说明：
9. 1, 这里将分组替换和&符号放在一起对比
10. 2, 命令中的分组替换使用了3个小括号，每个小括号分别代表每一行以逗号作为分隔符的每一列。
11. 3, 上面命令的&符号代表每一行，即模型中`s#目标内容#替换内容#g`的目标内容。
```

企业案例5:批量重命名文件

当前目录下有文件如下所示：

```
1. [root@chensiqi1 chen]# find ./ -name "*_finished.jpg"
2. ./stu_102999_1_finished.jpg
```

```
3. ./stu_102999_5_finished.jpg
4. ./stu_102999_3_finished.jpg
5. ./stu_102999_2_finished.jpg
6. ./stu_102999_4_finished.jpg
```

要求用sed命令重命名，效果为

stu_102999_1_finished.jpg==>stu_102999_1.jpg,即删除文件名的_finished

解题思路：因为这是文件名，不能直接用sed命令替换，因此还需要借助mv命令重命名，格式为：mv stu_102999_1_finished.jpg stu_102999_1.jpg.我们需要拼凑这样的格式，然后使用bash命令执行即可。

```
1. [root@chensiqi1 chen]# find ./ -name "*_finished.jpg" | sed -r
   's#^(.*)_finished(.*)#\1\2#g'
2. ./stu_102999_1.jpg
3. ./stu_102999_5.jpg
4. ./stu_102999_3.jpg
5. ./stu_102999_2.jpg
6. [root@chensiqi1 chen]# find ./ -name "*_finished.jpg" | sed -r
   's#^(.*)_finished(.*)#& \1\2#g'
7. ./stu_102999_1_finished.jpg ./stu_102999_1.jpg
8. ./stu_102999_5_finished.jpg ./stu_102999_5.jpg
9. ./stu_102999_3_finished.jpg ./stu_102999_3.jpg
10. ./stu_102999_2_finished.jpg ./stu_102999_2.jpg
11. ./stu_102999_4_finished.jpg ./stu_102999_4.jpg
12. [root@chensiqi1 chen]# find ./ -name "*_finished.jpg" | sed -r
   's#^(.*)_finished(.*)#mv & \1\2#g'
13. mv ./stu_102999_1_finished.jpg ./stu_102999_1.jpg
14. mv ./stu_102999_5_finished.jpg ./stu_102999_5.jpg
15. mv ./stu_102999_3_finished.jpg ./stu_102999_3.jpg
16. mv ./stu_102999_2_finished.jpg ./stu_102999_2.jpg
17. mv ./stu_102999_4_finished.jpg ./stu_102999_4.jpg
18. [root@chensiqi1 chen]# find ./ -name "*_finished.jpg" | sed -r
   's#^(.*)_finished(.*)#mv & \1\2#g' | bash
19. [root@chensiqi1 chen]# ls
20. stu_102999_1.jpg stu_102999_2.jpg stu_102999_3.jpg stu_102999_4.jpg stu_102999_5.
   jpg
21.
22. 命令说明：
23. 1. "\1"代表前面"^(.*)"匹配内容，"&"代表"s# #"里被替换的内容，这里匹配到的是完整的文件名。
24. 2. 使用bash命令执行，bash命令执行标准输入的语句，如同我们在命令行输入语句后敲回车。
```

6.2.4 查

- 学到这里，大家可以稍微喘口气了，因为sed里最常用最重要的我们已经学完了，接下来我们轻松的学完最后一招-->查看文本
- 这个功能也是非常得有用，比如我们想查看文件中的某些行，以前最常用的是cat或more或less命令等，但这些命令有些缺点，就是不能查看指定的行。而我们用了很久的sed命令就有了这个功能了。而且我们前面也说过使用sed比其他命令vim等读取速度更快！

- 这里我们需要用到1个sed命令
- “p”：输出指定内容，但默认会输出2次匹配的结果，因此使用-n选项取消默认输出，记忆方法：p的全拼是print，意思是打印。

6.2.4.1 按行查询

```
1. [root@chensiqi1 ~]# sed '2p' person.txt
2. 101,chensiqi,CEO
3. 102,zhangyang,CTO
4. 102,zhangyang,CTO
5. 103,Alex,COO
6. 104,yy,CFO
7. 105,feixue,CIO
8. [root@chensiqi1 ~]# sed -n '2p' person.txt
9. 102,zhangyang,CTO
10.
11. 命令说明：选项-n取消默认输出，只输出匹配的文本，大家只需要记住使用命令p必用选项-n。
12.
13. [root@chensiqi1 ~]# sed -n '2,3p' person.txt
14. 102,zhangyang,CTO
15. 103,Alex,COO
16.
17. 命令说明：查看文件的第2行到3行，使用地址范围“2, 3”。取行就用sed，最简单
18.
19. [root@chensiqi1 ~]# sed -n '1~2p' person.txt
20. 101,chensiqi,CEO
21. 103,Alex,COO
22. 105,feixue,CIO
23.
24. 命令说明：打印文件的1, 3, 5行。~代表步长
25.
26. [root@chensiqi1 ~]# sed -n 'p' person.txt
27. 101,chensiqi,CEO
28. 102,zhangyang,CTO
29. 103,Alex,COO
30. 104,yy,CFO
31. 105,feixue,CIO
32.
33. 命令说明：不指定地址范围，默认打印全部内容。
```

6.2.4.2 按字符串查询

```
1. [root@chensiqi1 ~]# sed -n '/CTO/p' person.txt
2. 102,zhangyang,CTO
3.
4. 命令说明：打印含CTO的行
5.
6. [root@chensiqi1 ~]# sed -n '/CTO/,/CFO/p' person.txt
7. 102,zhangyang,CTO
8. 103,Alex,COO
9. 104,yy,CFO
```

```
10.
11. 命令说明：打印含CTO的行到含CFO的行。
```

6.2.4.3 混合查询

```
1. [root@chensiqi1 ~]# sed -n '2,/CFO/p' person.txt
2. 102,zhangyang,CTO
3. 103,Alex,COO
4. 104,yy,CFO
5.
6. 命令说明：打印第2行到含CFO的行。
7.
8. [root@chensiqi1 ~]# sed -n '/feixue/,2p' person.txt
9. 105,feixue,CIO
10.
11. 命令说明：特殊情况，前两行没有匹配到feixue，就向后匹配，如果匹配到feixue就打印此行。所以这种混合地址不推荐使用。
```

6.2.4.4 过滤多个字符

```
1. [root@chensiqi1 ~]# sed -rn '/chensiqi|yy/p' person.txt
2. 101,chensiqi,CEO
3. 104,yy,CFO
4.
5. 命令说明：
6. 使用扩展正则"|", 为了不使用转义符号"\", 因此使用-r选项开启扩展正则表达式模式
```

7, sed命令应用知识扩展

7.1 sed修改文件的同时进行备份

```
1. [root@chensiqi1 ~]# cat person.txt
2. 101,chensiqi,CEO
3. 102,zhangyang,CTO
4. 103,Alex,COO
5. 104,yy,CFO
6. 105,feixue,CIO
7. [root@chensiqi1 ~]# sed -i.bak 's#zhangyang#NB#g' person.txt
8. [root@chensiqi1 ~]# cat person.txt
9. 101,chensiqi,CEO
10. 102,NB,CTO
11. 103,Alex,COO
12. 104,yy,CFO
13. 105,feixue,CIO
14. [root@chensiqi1 ~]# cat person.txt.bak
15. 101,chensiqi,CEO
16. 102,zhangyang,CTO
17. 103,Alex,COO
18. 104,yy,CFO
```

```
19. 105,feixue,CIO
20.
21. 命令说明：
22.     在-i参数的后边加上.bak（.任意字符），sed会对文件进行先备份后修改
```

7.2 特殊符号=获取行号

```
1. [root@chensiqi1 ~]# sed '=' person.txt
2. 1
3. 101,chensiqi,CEO
4. 2
5. 102,NB,CTO
6. 3
7. 103,Alex,COO
8. 4
9. 104,yy,CFO
10. 5
11. 105,feixue,CIO
12.
13. 命令说明：使用特殊符号“=”就可以获取文件的行号，这是特殊用法，记住即可。从上面的命令结果我们也发现了一个不好的地方：行号和行不在一行。

14.
15. [root@chensiqi1 ~]# sed '1,3=' person.txt
16. 1
17. 101,chensiqi,CEO
18. 2
19. 102,NB,CTO
20. 3
21. 103,Alex,COO
22. 104,yy,CFO
23. 105,feixue,CIO
24.
25. 命令说明：只打印1, 2, 3行的行号，同时打印输出文件中的内容
26.
27. [root@chensiqi1 ~]# sed '/yy/=' person.txt
28. 101,chensiqi,CEO
29. 102,NB,CTO
30. 103,Alex,COO
31. 4
32. 104,yy,CFO
33. 105,feixue,CIO
34.
35. 命令说明：
36.     只打印正则匹配行的行号，同时输出文件中的内容
37.
38. [root@chensiqi1 ~]# sed -n '/yy/=' person.txt
39. 4
40.
41. 命令说明：只显示行号但不显示行的内容即取消默认输出。
42.
43. [root@chensiqi1 ~]# sed -n '$=' person.txt
```

```
44. 5
45.
46. 命令说明：
47.     "$"代表最后一行，因此显示最后一行的行号，变相得出文件的总行数。
```

方法改进：

```
1. [root@chensiqi1 ~]# sed '=' person.txt | sed 'N;s#\n# #'
2. 1 101,chensiqi,CEO
3. 2 102,NB,CTO
4. 3 103,Alex,COO
5. 4 104,yy,CFO
6. 5 105,feixue,CIO
7.
8. 命令说明：前面sed获取文件的行号有一个缺点，我们这里使用Sed命令N来补偿这个缺点。Sed命令N读取下一行数据并附加到模式空间。
```

7.3 sed如何取不连续的行

```
1. [root@chensiqi1 ~]# sed -n '1p;3p;5p' person.txt
2. 101,chensiqi,CEO
3. 103,Alex,COO
4. 105,feixue,CIO
```

7.4 特殊符号{}的使用

```
1. [root@chensiqi1 ~]# sed -n '2,4p;=' person.txt
2. 1
3. 102,NB,CTO
4. 2
5. 103,Alex,COO
6. 3
7. 104,yy,CFO
8. 4
9. 5
10.
11. 命令说明：-n去掉默认输出，2,4p，输出2到4行内容，=输出全部的行的行号
12.
13. [root@chensiqi1 ~]# sed -n '2,4{p;=}' person.txt
14. 102,NB,CTO
15. 2
16. 103,Alex,COO
17. 3
18. 104,yy,CFO
19. 4
20.
21. 命令说明：
22.     '2, 4{p;=}'代表统一输出2, 4行的行号和内容
```