sed 命令

作者：张亚

归档：学习笔记

2021/4/14

快捷键：

Ctrl + 1　　标题 1
Ctrl + 2　　标题 2
Ctrl + 3　　标题 3
Ctrl + 4　　实例
Ctrl + 5　　程序代码
Ctrl + 6　　正文

格式说明：

蓝色字体：注释
黄色背景：重要
绿色背景：注意

老男孩linux运维实战培训

老男孩教育教学核心思想 6 重：重目标、重思路、重方法、重实践、重习惯、重总结

学无止境，老男孩教育成就你人生的起点！

联系方式：

网站运维 QQ 交流群：

Linux 385168604
架构师 390642196
Python 29215534
大数据 421358633

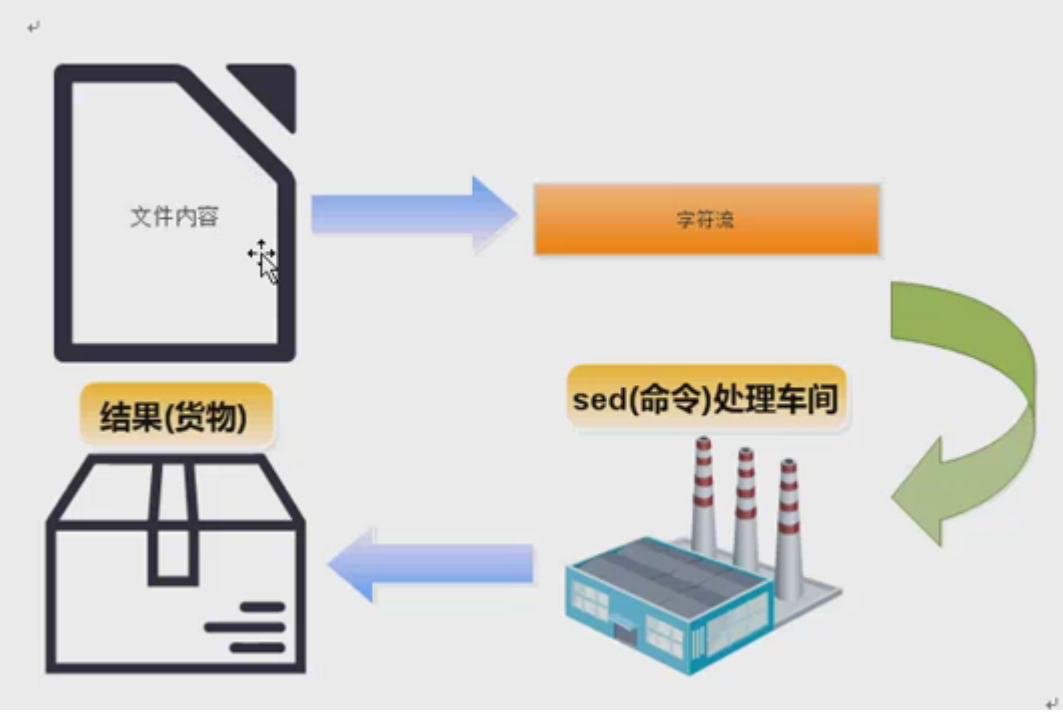
官方网站:

http://www.oldboyedu.com

目 录

前言

sed 是 Stream Editor(字符流编辑器)的缩写,简称流编辑器.什么意思呢?就像流水线,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,最后从流水线上下来就变成货物了.



当然上图中的文件内容可以使来自文件,也可以直接直接来自键盘或管道等标准输入,最后的结果默认情况下是先是到终端的屏幕,但也可以输出到文件中.

sed 软件功能于版本

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)其中查询的功能最常用的 2 大功能是过滤(过滤指定字符串),取行(去除指定行).

我们现在准备学习的 sed 版本是 GNU 开源版本的,试验环境如下:

```
[root@Alaska141 shell]# cat /etc/redhat-release
CentOS release 6.8 (Final)
[root@Alaska141 shell]# uname -r
2.6.32-642.13.1.el6.x86_64
[root@Alaska141 shell]# sed --version
GNU sed version 4.2.1
.....省略部分内容
```

语法格式

```
sed [options] [sed-commands] [input-file]
sed [选项]    [sed 命令]     [输入文件]
说明:
1. 注意 sed 软件及后面选项, 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, 文本称呼 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 还能够从标准输入或管道获取输入
```

命令执行过程

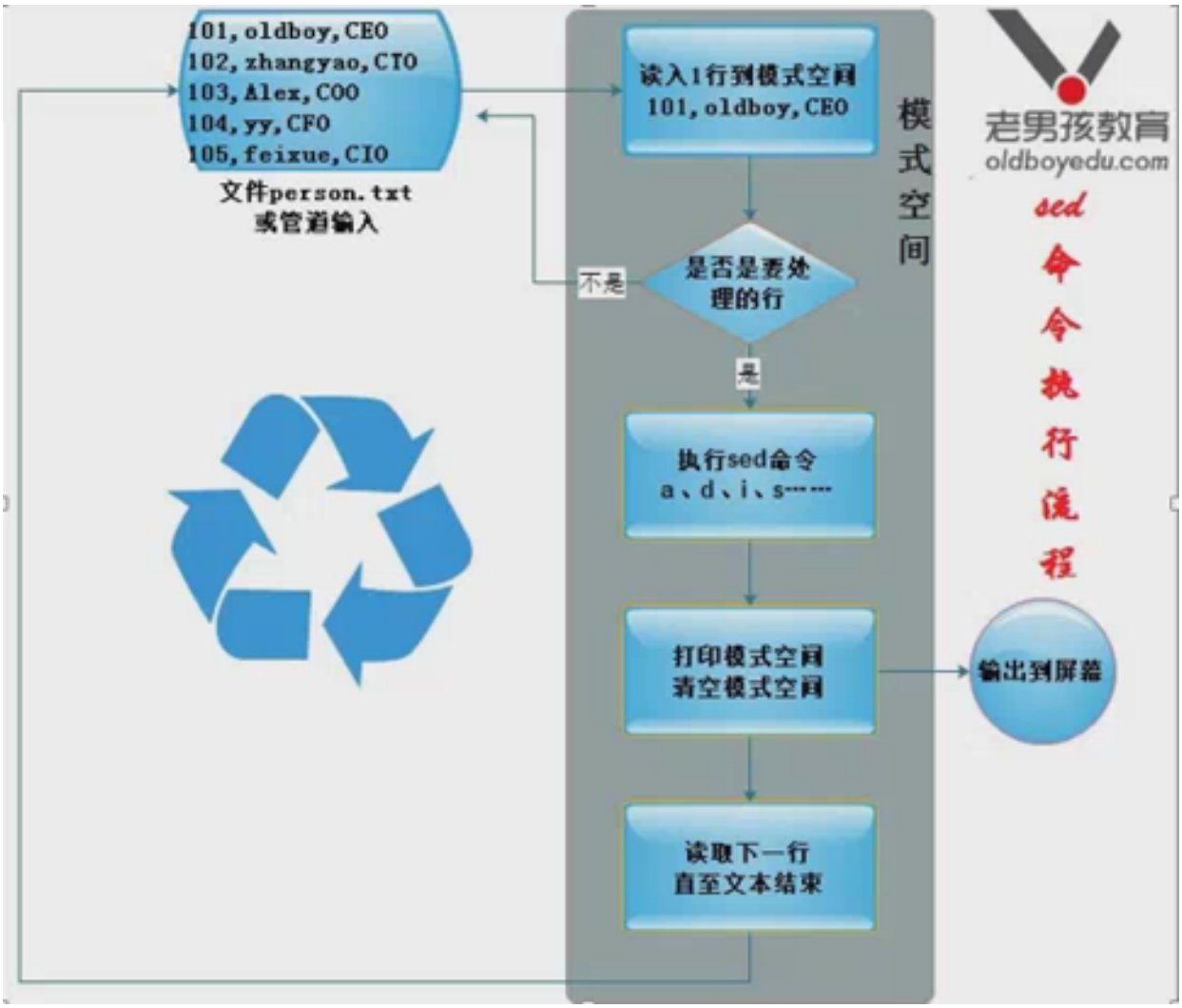
概括流程:sed 软件从文件或管道读取一行,处理一行,输出一行;再读取一行.再处理一行.在输出一行....

小知识:一次一行的设计是的 sed 软件性能很高,sed 在读取非常庞大的文件是不会出现卡顿现象.大家都用过 vi 命令,用 vi 命令打开几十 M 或更大的文件,会发现有卡顿现象,这是因为 vi 命令打开文件是一次性将文件加载到内容,然后再打开.因此卡短时间长短就取决于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,oldboy,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,比喻为工程流水线的传送带)

文件 person.txt 在模式空间的完整处理流程:
1.判断第 1 行是否是需要处理的行,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,指导文件结束.

上面的流程图概括如下所示



sed 软件有两个内置的存储空间:

模式空间(pattern space):是 sed 软件从文件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 软件另一个缓冲区,用来存放临时数据,也是在内存中的,但是模式空间和保持空间的用途不一样.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,也就说我们可以在保持空间存储数据.

初始情况,模式空间和保持空间都是没有内容的,每次循环读取数据的过程中,模式空间的元内容都会被清空写入新的内容,但保持空间的内容不变,不会再循环中被删除,除非使用 sed 命令操作保持空间.

选项说明

options[选项]	解释说明
-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	使用正则拓展表达式,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,并不影响磁盘上的文件

sed-commands[sed 命令]	解释说明
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,在指定的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,并读取下一行数据并追加到模式空间
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	打印模式空间的内容,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\你结束操作
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
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
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
!	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
&	代表被替代的内容
;	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

使用范例

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

```
[root@Alaska141 sed]# cat >person.txt<<EOF
> 101,oldboy,CEO
> 102,zhangyao,CTO
> 103,Alex,COO
> 104,yy,CFO
> 105,feixue,CIO
> EOF
[root@Alaska141 sed]# cat person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

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.

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,操作有些繁琐但还是能用,但是当我们学会了 shell 脚本后,我们就会发现脚本中不能正常使用 vi 或 vim,为什么呐?读者们可以在脚本中写个 vi 命令体验一下

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,他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,记忆方法:a 的全拼是 apend,,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

2.1.1 单行增加

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,读者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.

```
[root@Alaska141 sed]# sed '2a 106,dandan,CSO' person.txt
101,oldboy,CEO
102,zhangyao,CTO
106,dandan,CSO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]#
```

命令行详解:

我们可以看到在第二行之后新增加了一行我们输入的内容

前面讲了 sed 软件 a 命令追加的功能,我们在简单的看一下命令 i 插入的用法,请看下面的具体案例

```
[root@Alaska141 sed]# sed '2i 106,dandan,CSO' person.txt
101,oldboy,CEO
106,dandan,CSO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令行解释:

在第二行前面插入了一行我们自定义的内容

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写上'2i 106,dandan,CSO'

- 2 代表指定对第 2 行操作,其他行忽略
- i 代表插入的意思,2i 即第 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

2.1.2 引号的区别总结

在教学中,有同学问 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("") ---把双引号内的内容输出出来;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).对于这些字符使用\可以去除它们的特殊含义.

2.单引号(' ') ---所见即所得,将单引号内的内容原样输出,组织所有字符的转义.

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,然后再输出最终结果来,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.反引号(``)---进行命令的替换,在反引号内部的 shell 命令将会被执行,其结果输出代替用反引号括起来的文本.

结论一:

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,那就是用单引号

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

2.1.3 多行增加

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.

```
[root@Alaska141 sed]# cat >test.txt<<EOF
> welcome to my blog aaaaa
> if you like my blog \'s aaaaa
>
>
> bye!boys and girls
> EOF
[root@Alaska141 sed]# cat test.txt
welcome to my blog aaaaa
if you like my blog \'s aaaaa



bye!boys and girls
```

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,只能往文本末添加内容,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下一个特殊符号"\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

```
[root@Alaska141 sed]# echo "oldboy";echo "oldboy"
oldboy
oldboy
[root@Alaska141 sed]# echo -e "oldboy\noldboy"
oldboy
oldboy
命令说明: -e 参数别是字符串如果出现以下特殊字符\n\t 等, 则加以特殊处理, 而不会将它当成一般文字处理.
```

学习完换行符,我们来看一下 sed 命令如何使用

```
[root@Alaska141 sed]# sed '2a 106,dandan,CSO\n107,bingbing,CCO' person.txt
101,oldboy,CEO
102,zhangyao,CTO
106,dandan,CSO
107,bingbing,CCO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令行解析:

```
2a 在第二行后追加内容
\n  换行
这样最终的结果为在第二行后面追加了两行内容,
106,dandan,CSO
107,bingbing,CCO
```

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"\".他也有换行的意思.如果大家在执行很长的命令的时候,如果都写在一行,即难看也难以理解.就利用到这个符号可以将一条完成的命令分成多行,举个例子:

```
[root@Alaska141 sed]# sed '2a 106,dandan,CSO \
> 107,bingbing,CCO' person.txt
101,oldboy,CEO
102,zhangyao,CTO
106,dandan,CSO
107,bingbing,CCO
103,Alex,COO
104,yy,CFO
105,feixue,CIO.
###注意最后接\的时候要有空格,不然会出错,而且这种方法比较麻烦,不推荐使用
```

2.1.4 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

```
Port 52113
PermitRootLogin no
PerminEmptyPasswords no
UseDNS no
GSSAPIAuthentication no
```

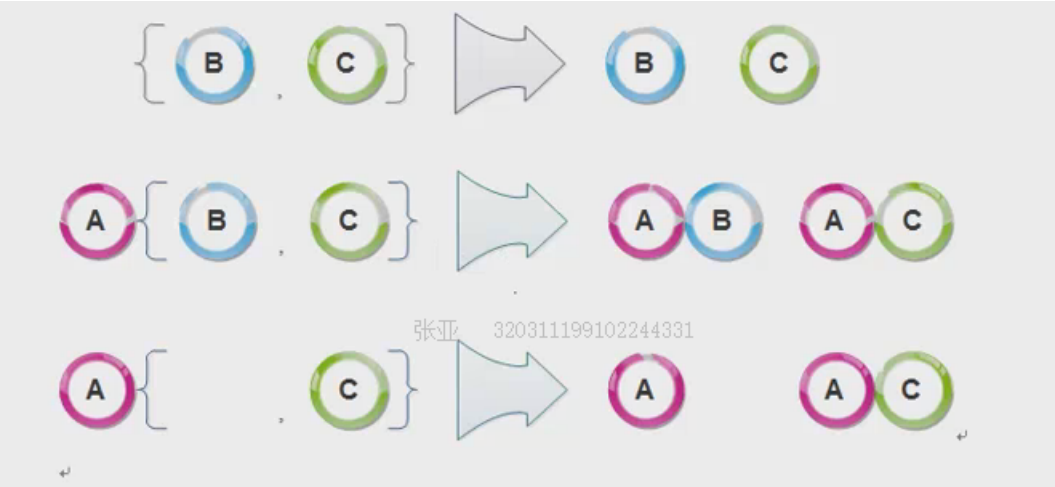
当然我们可以使用 vi/vim 命令编辑这个文本.但这样就比较麻烦,现在想用一条命令增加 5 行文本到第 13 行前?

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

为了分隔答案,这里插入一个小知识点,大家能看懂上面的备份命令吗?

```
[root@Alaska141 ~]# echo {B,C}
B C
[root@Alaska141 ~]# echo A{B,C}
AB AC
[root@Alaska141 ~]# echo A{,C}
A AC
[root@Alaska141 ~]# echo A{,.C}
A A.C
```

图形参考:



```
[root@Alaska141 sed]# sed '3i Port 52113\nPermitRootLogin no\nPerminEmptyPasswords no\nUseDNS
no\nGSSAPIAuthentication no' person.txt
101,oldboy,CEO
102,zhangyao,CTO
Port 52113
PermitRootLogin no
PerminEmptyPasswords no
UseDNS no
GSSAPIAuthentication no
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

2.2 删

前面介绍了第一招,下面学习第 2 招===>删除制定文本行

这个功能也非常有用,比如我们想删除文件中的某行,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),加下来我们看一下如何对多行文本为目标操作.

2.2.1 指定执行的地址范围

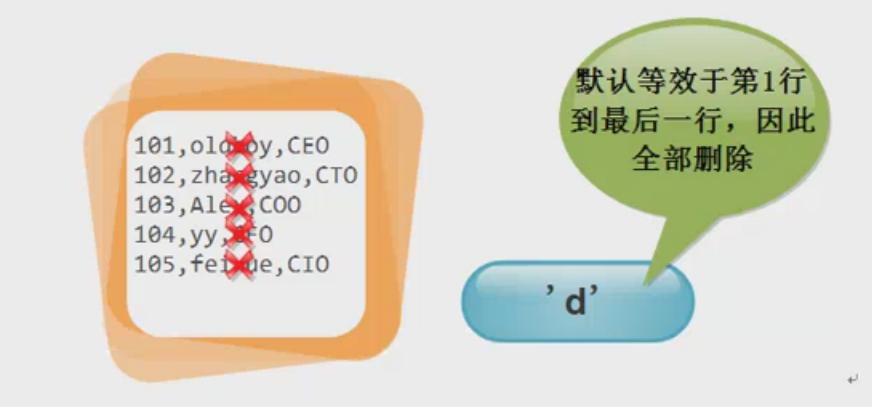
```
sed 软件可以对单行或多行文本进行处理, 如果在 sed 命令前面不指定地址范围, 那么默认会匹配所有行.

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
/oldboy/{sed-commands}	对匹配 oldboy 的行操作
/oldboy/,/Alex/{sed-commands}	对匹配 oldboy 的行到匹配 Alex 的行操作
/oldboy/,${sed-commands}	对匹配 oldboy 的行到最后一行操作
/oldboy/,10{sed-commands}	对匹配 oldboy 的行到第 10 行操作.注意:如果前 10 行没有匹配到 oldboy,sed 软件会显示 10 行以后的匹配 oldboy 的行,如果有
1,/Alex/{sed-commands}	对第一行匹配到 Alex 的行操作
/oldboy/,+2{sed-commands}	对匹配 oldboy 的行到其后的 2 行操作

下面用具体例子演示一下,测试文件还是 person.txt

```
[root@Alaska141 sed]# sed 'd' person.txt
[root@Alaska141 sed]#
```

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文件的所有内容.




```
[root@Alaska141 sed]# sed '2d' person.txt
101,oldboy,CEO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明:单行删除想必都能理解,制定删除第 2 行的文本 102,zhangyao,CTO

```
[root@Alaska141 sed]# sed '2,5d' person.txt
101,oldboy,CEO
```

命令说明:这个是 2 个数字地址的组合,用逗号作为分隔,删除第二行到第五行(删除多行),包括第 2 行和第 5 行,因,因此只剩下第 1 行


```
[root@Alaska141 sed]# sed '2d;5d' person.txt
101,oldboy,CEO
103,Alex,COO
104,yy,CFO
```

命令说明:删除指定的几行可以分别写,用分号隔开 2d;5d 即删除第 2 行和第 5 行.

正则地址范围匹配

上面实验完数字地址范围,接下来我们试验一下正则表达式地址范围,虽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.

```
[root@Alaska141 sed]# sed '/zhangyao/d' person.txt
101,oldboy,CEO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明:

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/".

这里我们给出具体的匹配字符串"zhangyao",这样可以精确的删除第 2 行,当然也可以使用正则表达式.如下所示

```
[root@Alaska141 sed]# sed '/A.*x/d' person.txt
101,oldboy,CEO
102,zhangyao,CTO
104,yy,CFO
105,feixue,CIO
```

```
[root@Alaska141 sed]# sed '/101/,/104/d' person.txt
105,feixue,CIO
```

命令说明:

和删除数字范围的行到行一样,sed 也可以使用正则匹配行到行来删除,即删除从 101 的行到 104 的行

```
[root@Alaska141 sed]# sed '/101/d;/104/d' person.txt
102,zhangyao,CTO
103,Alex,COO
105,feixue,CIO
```

命令说明:

和删除指定的数字行一样,sed 也可以使用正则表达式来删除匹配到的指定行

这里是删除 101 的行和 104 的行

```
[root@Alaska141 sed]# sed '/101/,/10*/d' person.txt
103,Alex,COO
104,yy,CFO
105,feixue,CIO
命令说明:
这样如果用了正则的匹配的话就会最短删除,只会删除最近匹配的行
```

比如下面的例子:

```
[root@Alaska141 sed]# echo -e "aaa\naab\naac\n\aaad"
aaa
aab
aac
aad
[root@Alaska141 sed]# echo -e "aaa\naab\naac\n\aaad\n"|sed '/aaa/,/aa*/d'
aac
aad
命令说明:
如果使用了这样的正则匹配删除,就只会删除了就近匹配的行
```

提示:工作中常用数字匹配行来进行删除!

```
[root@Alaska141 sed]# sed '1,/103/d' person.txt
104,yy,CFO
105,feixue,CIO
命令说明:
删除了第一行到匹配到 103 的行
```

```
[root@Alaska141 sed]# echo -e "aaa\naab\naac\n\aaad"
aaa
aab
aac
aad
[root@Alaska141 sed]# echo -e "aaa\naab\naac\n\aaad"|sed '1,/aa*/d'
aac
aad
命令说明:删除数字第一行到最短匹配到 aa*的行,注意第一行已经被数字 1 定义了,所以最短匹配到的是第 2 行
```

下面来看一些特殊情况:

```
[root@Alaska141 sed]# sed '/oldboy/,3d' person.txt
104,yy,CFO
105,feixue,CIO
```

命令说明:

这个例子是删除包含"oldboy"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"oldboy"字眼,sed 软件还是会找他"麻烦",请看下面的例子

```
[root@Alaska141 sed]# sed '$a 106,oldboy,CMO' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
106,oldboy,CMO
```

命令说明:

为了不造成实验文本改来改去导致不统一,因此这里的$a 追加语句只是临时修改内存数据,然后通过管道符传递给 sed 软件.

```
[root@Alaska141 sed]# sed '$a 106,oldboy,CMO' person.txt |sed '/oldboy/,3d'
104,yy,CFO
105,feixue,CIO
```

命令说明:

从命令结果我们可以看到,不仅是第 1 行(101,oldboy,CEO)到第 3 行(103,Alex,COO)被删除了,而且最后一行(106,oldboy,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时会找出所有匹配的行,即使是有数字地址限制.

下面再看一个特殊的例子

```
[root@Alaska141 sed]# sed '2,/O/d' person.txt
101,oldboy,CEO
104,yy,CFO
105,feixue,CIO
```

命令说明:

从第 2 行开始删除到含字母 O(大写 O)的行结束,但是我们发现第 3,4,5 行都有字母 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?

还是可以从上面的命令执行流程图来理解,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.

```
[root@Alaska141 sed]# sed '2,/o/d' person.txt
101,oldboy,CEO
```

命令说明:

从第 2 行开始删除,但是后面文本没有字幕 o,因此一直循环下去,直到文本结束,sed 软件自动终止.

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.

2.2.2 特殊符号~(步长)解析

格式: "First~Step" 表示从 First 开始,以步长 Step 递增.这个在数学中叫等差数列.

举例

```
1~2 匹配 1,3,5,7,....    #用于只输出奇数行,大伙仔细观察下每个数字的差值
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,...
```

```
 [root@Alaska141 sed]# seq 1 10
1
2
.....省略
[root@Alaska141 sed]# seq 1 10|sed -n '1~2p'
1
3
5
7
9
```

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.

上面的例子测试了"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.

```
[root@Alaska141 sed]# seq 1 10|sed -n "2~2p"
2
4
6
8
10
[root@Alaska141 sed]# seq 1 10|sed -n "2~3p"
2
5
8
```

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.

```
[root@Alaska141 sed]# seq 1 2 10
1
3
5
7
9
命令说明:seq 命令格式 seq 起始值 公差 结束值
```

再来一列:

```
[root@Alaska141 sed]# sed '1~2d' person.txt
102,zhangyao,CTO
104,yy,CFO
命令说明:"1~2"这是指定行数的另一种形式,从第一行开始以步长 2 递增的行(1,3,5).因此删掉第 1,3,5 行,即所有的奇数行.
```

2.2.3 特殊符号+解析

```
[root@Alaska141 sed]# sed '1,+2d' person.txt
104,yy,CFO
105,feixue,CIO
命令说明:这其实是做加法运算, '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.

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.

```
[root@Alaska141 sed]# sed '2,3!d' person.txt
102,zhangyao,CTO
103,Alex,COO
命令说明:在地址范围 "2,3"后面加上"!",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.
```

2.3 改

现在大家已经学完了 sed 软件 2 大招式,现在来学习更厉害的大招--改.这是一个杀招,杀手锏,一出手就定乾坤,在学习 Linux 时,我们最常见的操作就是修改配置文件,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现实现

2.3.1 按行替换

首先说明一下按行替换,这个功能用地的很少,所以大家了解一点即可,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.

举例:

```
[root@Alaska141 sed]# sed '2c 106,dandan,CSO' person.txt
101,oldboy,CEO
106,dandan,CSO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
命令说明: 使用 sed 命令 c 将原来第 2 行 "102,zhangyao,CTO" 替换成 "106,dandan,CSO" ,整行替换.
```

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 区分.

sed 软件替换模型

sed -i 's/AAA/BBB#g'

sed -i 's#AAA#BBB#g'

观察特点

1.两边是引导,引号里面的两边分别是 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.定界符可以使任意字符如:或|等,但当替换内容包含定界符时,需要转义即\;或\|经过长期实践,建议大家使用#作为定界符.

举例:

你能瞬间理解's\//etc/sysconfig/network-scripts/\//etc/hosts\/g'吗?

那这样写's#/etc/sysconfig/network-scripts/#/etc/hosts#g' 感觉如何?

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#@@@#$$$#g 中@@@能用正则表达式,但$$$不能用,必须是具体的,因为$$$使用正则的话会让 sed 软件无所适从,他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.

举例:

```
[root@Alaska141 sed]# sed 's#zhangyao#XXXXX#g' person.txt
101,oldboy,CEO
102,XXXXX,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO

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.
```

企业案例 3: 指定行修改配置文件

前面我们学过了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了很多地方.

```
[root@Alaska141 sed]# sed '3s#0#9#' person.txt
101,oldboy,CEO
102,zhangyao,CTO
193,Alex,COO
104,yy,CFO
105,feixue,CIO

命令说明: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.
而这个案例要求只将第 3 行的 0 替换成 9,这里就用到我们前面学过的地址范围只是,在 sed 命令"s"前加上"3"就代表对第 3 行进行替换.
```

2.3.3 变量替换

变量替换其实和前面的文本替换一样,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,请往前翻页到[引号的区别总结]再复习一下.

单引号的情况

```
[root@Alaska141 sed]# x=1
[root@Alaska141 sed]# y=2
[root@Alaska141 sed]# echo $x $y
1 2
命令说明:定义两个变量 x y 并赋值 x=1 , y=2
[root@Alaska141 sed]# sed 's#$x#$y#g' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
命令说明:单引号内所见即所得,因为没有"$x"这个字符串,所以就没有替换,结果原样输出
```

单引号的特殊用法;

```
[root@Alaska141 sed]# sed 's#'$x'#'$y'#g' person.txt
202,oldboy,CEO
202,zhangyao,CTO
203,Alex,COO
204,yy,CFO
205,feixue,CIO
命令说明:sed 的单引号是成对识别的,所以$x $y 其实是相当于没有引号,没有引号的情况是可以识别变量的,所以$x 和$y 就被正常解析了.
注意!如果 sed 要替换的内容里含有'' 就需要注意一下单引号封闭的问题了.
[root@Alaska141 sed]# x=XXXX
[root@Alaska141 sed]# echo $x
XXXX
[root@Alaska141 sed]# echo 'aa'$x'bb'
aaXXXXbb
命令说明:第一对单引号把 aa 括起来了,第二对单引号把 bb 括起来了,所以$x 相当于没有加引号.即解析出变量 x
```

双引号的情况

```
[root@Alaska141 sed]# sed "s#$x#$y#g" person.txt
202,oldboy,CEO
202,zhangyao,CTO
203,Alex,COO
204,yy,CFO
205,feixue,CIO
[root@Alaska141 sed]#
```

命令说明:双引号内如果有变量就会被解析,所以把文本里所有的 1 都替换成了 2 并打印输出.

```
[root@Alaska141 sed]# sed "s#$x#$y#" person.txt
201,oldboy,CEO
202,zhangyao,CTO
203,Alex,COO
204,yy,CFO
205,feixue,CIO
```

命令说明:如果去掉 g ， 那么每行只会匹配第一次 1,后面的不会被匹配到. 不用引号也可以,区别是如果字符串有空格的话单引号会认为是两个字符串而不是一个整体.

不使用引号

```
[root@Alaska141 sed]# sed s#$x#$y#g person.txt
202,oldboy,CEO
202,zhangyao,CTO
203,Alex,COO
204,yy,CFO
205,feixue,CIO
```

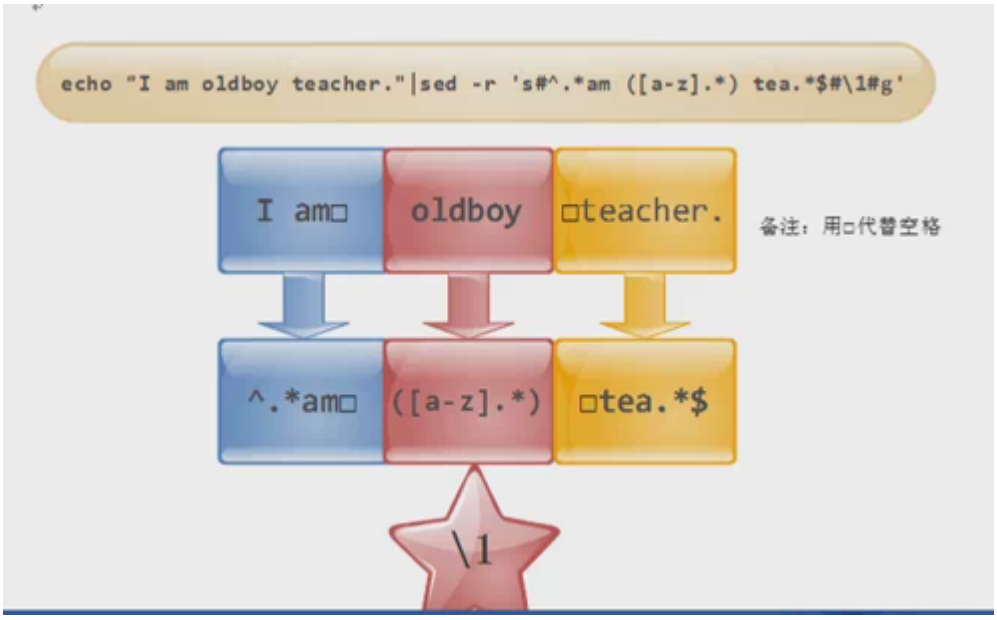
命令说明:不使用引号的话

2.3.4 分组替换\(\)和\1 的使用说明

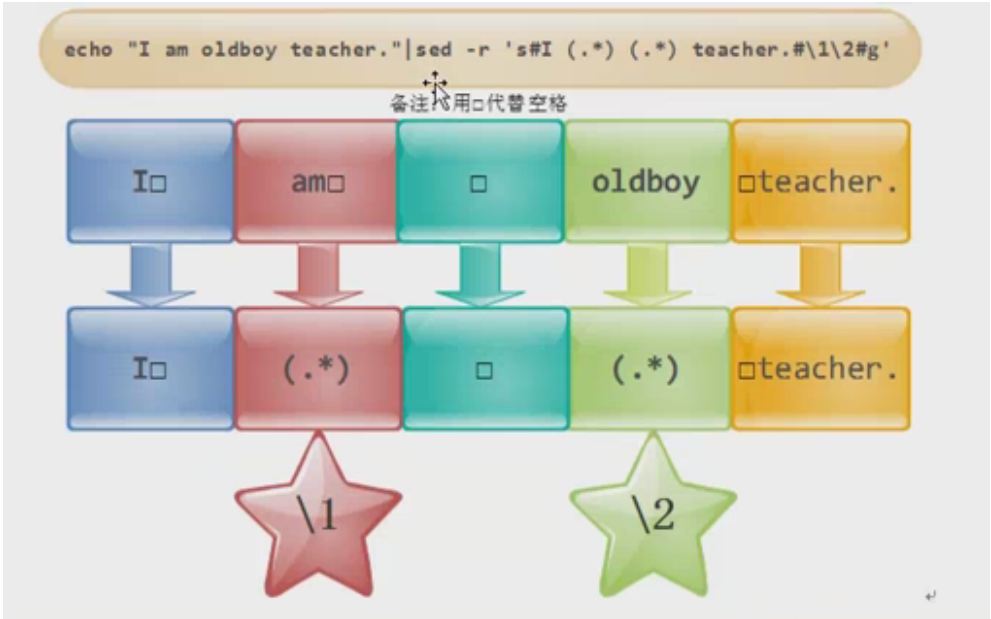
sed 软件的\(\)的功能可以记住正则表达式的一部分,其中 ,\1 为第一个记住的模式即第一个小括号中的匹配内容,\2 表示记住第二记住的模式,即第二个小括号中的匹配内容,sed 最多可以记住 9 个.

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

```
[root@Alaska141 ~]# echo "I am oldboy teacher"|sed 's#^.*am \([a-z].*\) tea.*$#\1#g'
 oldboy
```




```
 [root@Alaska141 ~]# echo "I am oldboy teacher"|sed -r 's#I (.*) (.*) teacher#\1\2#g'
amoldboy
```



命令说明

思路:用 oldboy 字符替换 I am oldboy teacher.

下面解释用口代替空格

1. ^.am 口 ===>这句的意思是以任意字符开头到 am 口为止,匹配文件中的 I am 口字符串

2. \([a-z].*\)口 ===>这句话的外壳就是括号\(\) .里面的[a-z]表示匹配 26 个字母的任何一个,[a-z].*合起来就是匹配任意多个字符,本题来说就是匹配 oldboy 字符串,由于 oldboy 字符串是需要保留的,因此用括号括起来匹配,后面通过\1 来取 oldboy 字符串.

3.口 tea.*$ ===>表示空格 tea 起始,任意字符结尾,实际就是匹配 oldboy 字符串后,紧接着的字符串口 teacher

4.后面被替换的内容中的\1 就是去前面的括号里的内容了,也就是我们要的 oldboy 字符串.

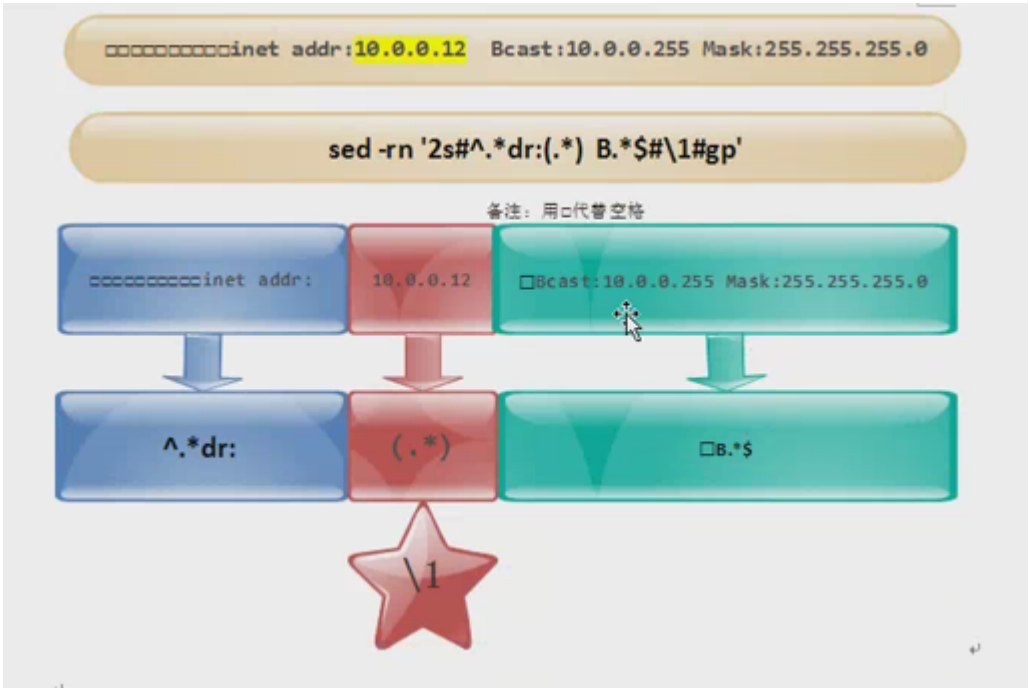
5.( )是拓展正则表达式的元字符,sed 软件默认只识别基本正则表达式,想要使用拓展正则表达式则需要使用\转义,即\(\). sed 使用-r 选项则可以识别拓展正则表达式,此时使用\(\)反而会出错.

两种格式总结

```
sed 's#\(.\)#\1#g'
sed -r 's#(.*)#\1#g'
命令说明:固定搭配,弄反了就错了.不使用"-r"选项就需要使用转义符号"\",使用"-r"选项就可以直接使用“（）”，推荐使用这种写法
```

再来看个题目，执行命令取出 linux 中 eth0 的 IP 地址

```
[root@Alaska141 ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:5E:4A:99
          inet addr:10.1.1.2  Bcast:10.1.1.255  Mask:255.255.255.0
...省略....
[root@Alaska141 ~]# ifconfig eth0|sed -rn '2s#^.*r:(.*) Bc.*$#\1#gp'
10.1.1.2
```



企业案例 4 系统开机启动项优化

方法一：egrep+awk+sed|bash

```
[root@Alaska141 /]# chkconfig |egrep "3:on"|egrep -v "sshd|crond|network|rsyslog|sysstat"|awk
'{print $1}'|sed -r 's#^(.*)#chkconfig \1 off#g'|bash
[root@Alaska141 /]# chkconfig --list|egrep "3:on"
crond           0:off   1:off   2:on    3:on    4:on    5:on    6:off
network         0:off   1:off   2:on    3:on    4:on    5:on    6:off
rsyslog         0:off   1:off   2:on    3:on    4:on    5:on    6:off
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
sysstat         0:off   1:on    2:on    3:on    4:on    5:on    6:off
```

方法二：egrep+awk+sed 's###e'

```
[root@Alaska141 /]# chkconfig |egrep "3:on"|egrep -v "sshd|crond|network|rsyslog|sysstat"|awk
'{print $1}'|sed -r 's#^(.*)#chkconfig \1 off#e'
```

命令说明：这个方法只是没用 bash 命令，而是利用 sed 软件的命令 s 的执行标志 e

方法三：egrep+awk

```
[root@Alaska141 /]# chkconfig --list|egrep "3:on"|egrep -v "crond|sshd|network|rsyslog|sysstat"|awk
'{print "chkconfig",$1,"off"}'|bash
```

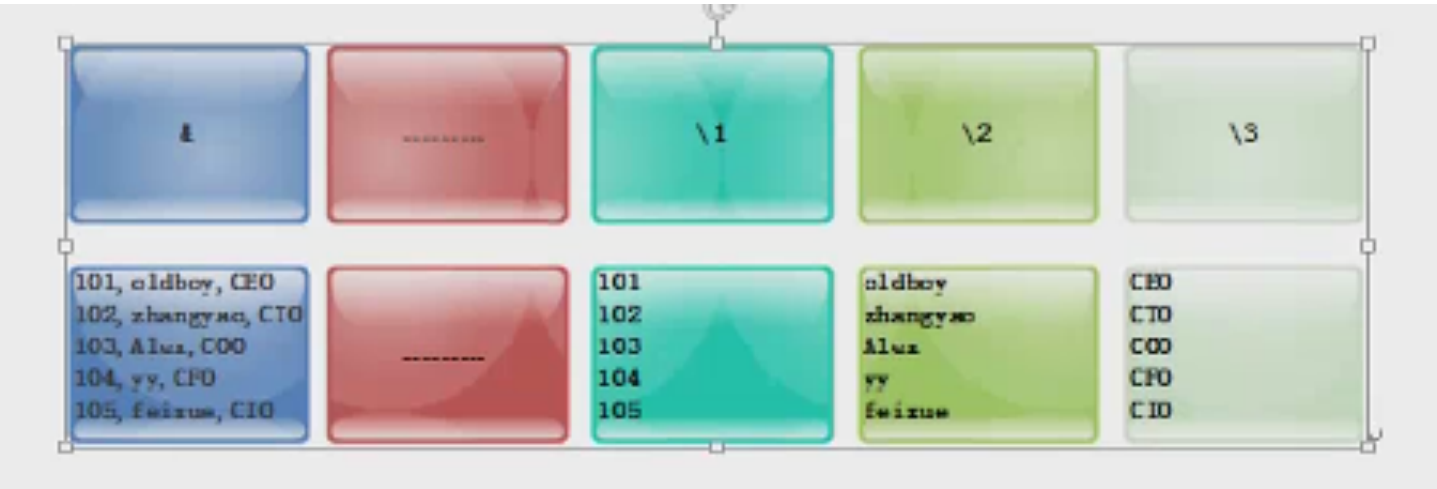
2.3.5 特殊符号&代表被替换的内容

这是一个特殊计较，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

```
[root@Alaska141 sed]# sed -rn 's#(.*),(.*),(.*)#& -------- \1 \2 \3#gp' person.txt
101,oldboy,CEO -------- 101 oldboy CEO
102,zhangyao,CTO -------- 102 zhangyao CTO
103,Alex,COO -------- 103 Alex COO
104,yy,CFO -------- 104 yy CFO
105,feixue,CIO -------- 105 feixue CIO
```

命令说明：

1. 这里将分组替换和&符号放在一起对比
2. 命令行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。
3. 上面命令的&符号代表每一行，即模型中‘s#口#▷#g’的口



1.1.1.1 eval

企业案例 5：批量重命名文件 删除文件名中的 finished

```
[root@Alaska141 filename]# touch stu_102999_{1..5}_finished.jpg
[root@Alaska141 filename]# ll
-rw-r--r-- 1 root root 0 Feb 18 05:11 stu_102999_1_finished.jpg
.....省略
[root@Alaska141 filename]# ls|sed -r 's#(.*)_(.*)_(.*)_(.*)(\..*)#\1_\2_\3\5#g'
stu_102999_1.jpg
.....省略
[root@Alaska141 filename]# ls|sed -r 's#(.*)_(.*)_(.*)_(.*)(\..*)#mv & \1_\2_\3\5#g'
mv stu_102999_1_finished.jpg stu_102999_1.jpg
......省略
[root@Alaska141 filename]# ls|sed -r 's#(.*)_(.*)_(.*)_(.*)(\..*)#mv & \1_\2_\3\5#g'|bash
[root@Alaska141 filename]# ll
-rw-r--r-- 1 root root 0 Feb 18 05:11 stu_102999_1.jpg
.....省略
```

或者：

```
[root@Alaska141 filename]# ls *.jpg|sed -r 's#(^.*)_finished.*$#mv & \1.jpg#'
mv stu_102999_1_finished.jpg stu_102999_1.jpg
.....省略
```

或者

```
[root@LSD filename]# ls stu*|xargs -n1|sed -r "s#(^.*[1-9]).*(.jpg)#mv & \1\2#g"
```

解题思路：因为这是文件名，不能直接用 sed 命令替换，所以要借助 mv 命令重命名

格式为：mv stu_102999_1_finished.jpg stu_102999_1.jpg

当然 awk 也可以实现：

```
[root@Alaska141 filename]# ls|awk -F "_finished" '{print "mv",$0,$1$2}'
mv stu_102999_1_finished.jpg stu_102999_1.jpg
.....省略
```

3.在 linux 中有专门重命名的命令 rename:

```
rename "_finished"  ""
```

rename 被替换的内容 替换的内容（此处为空） 替换的文件（可以使用通配符）

```
[root@Alaska141 filename]# rename "_finished" "" *.jpg
[root@Alaska141 filename]# ll
-rw-r--r-- 1 root root 0 Feb 18 05:52 stu_102999_1.jpg
```

rename 命令使用案例

```
注意！使用 rename 命令他只会替换每一个文件名第一次匹配的字符串，如下所示。
[root@Alaska141 filename]# ll
-rw-r--r-- 1 root root 0 Feb 18 05:52 stu_102999_1.jpg
[root@Alaska141 filename]# rename ″9″ ″X″ *
[root@Alaska141 filename]# ll
-rw-r--r-- 1 root root 0 Feb 18 05:52 stu_102X99_1.jpg
[root@Alaska141 filename]# rename ″9″ ″X″ *
[root@Alaska141 filename]# ll
-rw-r--r-- 1 root root 0 Feb 18 05:52 stu_102XX9_1.jpg

###rename 还支持反引号
[root@Alaska141 filename]# rename .jpg -`date +%F`.jpg *
[root@Alaska141 filename]# ll
-rw-r--r-- 1 root root 0 Feb 18 05:52 stu_102XX9_1-2017-02-18.jpg
```

2.4 查

学到这里，大家可以稍微松口气了，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

这个功能也是非常的有用，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 命令就有这个功能了，而且我们前面也说过使用 sed 比其他 vim 命令读取速度更快些

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。

2.4.1 按行查询

```
[root@Alaska141 sed]# sed '2p' person.txt
101,oldboy,CEO
102,zhangyao,CTO     #默认输出行
102,zhangyao,CTO     #p 命令输出的行
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]# sed -n '2p' person.txt
102,zhangyao,CTO
命令说明：选项-n 取消默认输出，指数初匹配的文本，大家只需要记住使用命令 p 必用选项-n

[root@Alaska141 sed]# sed -n '2,3p' person.txt
102,zhangyao,CTO
103,Alex,COO
命令说明：查看第 2 行到第 3 行，使用地址范围“2，3”取行就用 sed 最简单

[root@Alaska141 sed]# sed -n '1~2p' person.txt
101,oldboy,CEO
103,Alex,COO
105,feixue,CIO
命令说明：打印文件 1，3，5 行
```

2.4.2 按字符串查询

```
[root@Alaska141 sed]# sed -n '/CTO/p' person.txt
102,zhangyao,CTO
命令说明：打印 CTO 的行
[root@Alaska141 sed]# sed -n '/CTO/,/CFO/p' person.txt
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
命令说明：打印包含 CTO 的行到 CFO 的行
```

2.4.3 混合查询

```
[root@Alaska141 sed]# sed -n '2,/CFO/p' person.txt
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
```

命令说明：打印第 2 行到 CFO 的行

```
[root@Alaska141 sed]# sed -n '/feixue/,2p' person.txt
105,feixue,CIO
```

命令说明：特殊情况，前两行没有匹配 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。

2.4.4 过滤多个字符

```
[root@Alaska141 sed]# sed -rn '/oldboy|feixue/p' person.txt
101,oldboy,CEO
105,feixue,CIO
```

命令说明：使用拓展正则“|”，为了不使用转义字符“\”，因此使用-r 选项开启拓展正则表达式模式。

可以同时查找 2 个，3 个等更多字符

‘/oldboy|feixue/p’

‘/oldboy|feixue|yy/p’

.....

3.修改文件

大家也知道前面的增删改查操作并没有实际修改文件，只是改变模式空间的内容，但有时我们需要真正的改变文件，因此用到了-i 选项。

友情提示：修改文件前要备份，一般可以用 cp 备份，但 sed 软件的选项自带备份功能。

选项说明：

-i[SUFFIX]，--in-place[=SUFFIX]

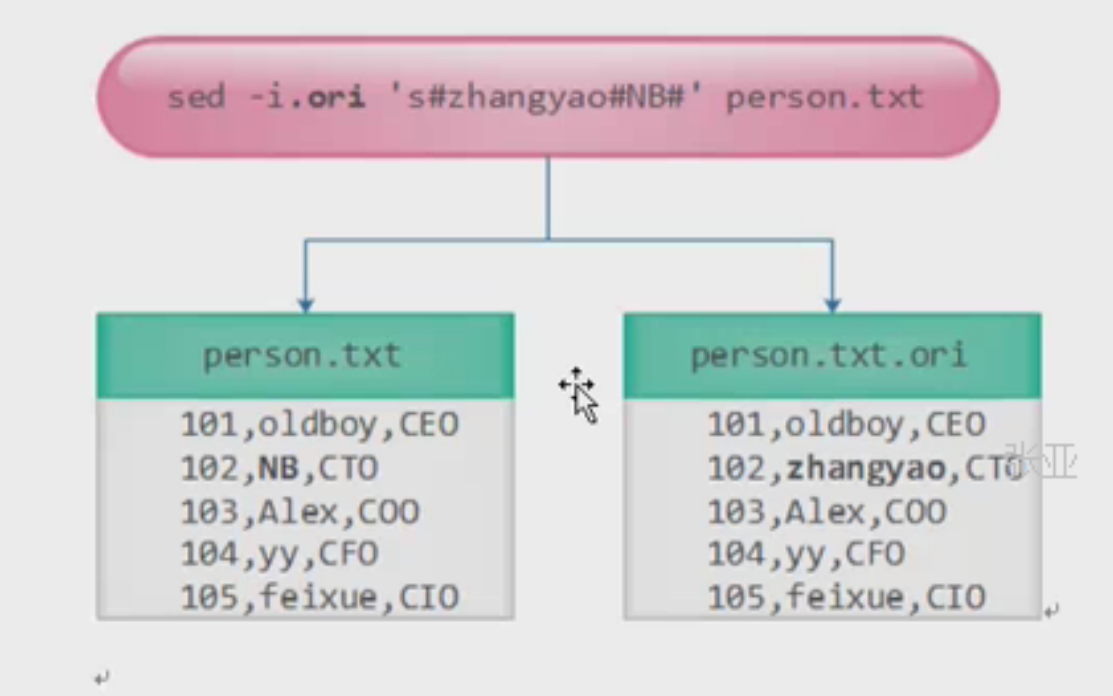
-i[备份文件后缀]，--in-place[=备份文件后缀]

```
[root@Alaska141 sed]# sed -i.ori 's#zhangyao#NB#g' person.txt
[root@Alaska141 sed]# cat person.txt
101,oldboy,CEO
102,NB,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]# cat person.txt.ori
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明：直接在-i 选项后接上你想要的备份文件后缀.ori（可以是其他任意字符），命令执行成功会自动备份生成文件 person.txt.ori。我们还发现上面的 sed 语句没有任何输出，因为所有内容已经写入 person.txt 文件中

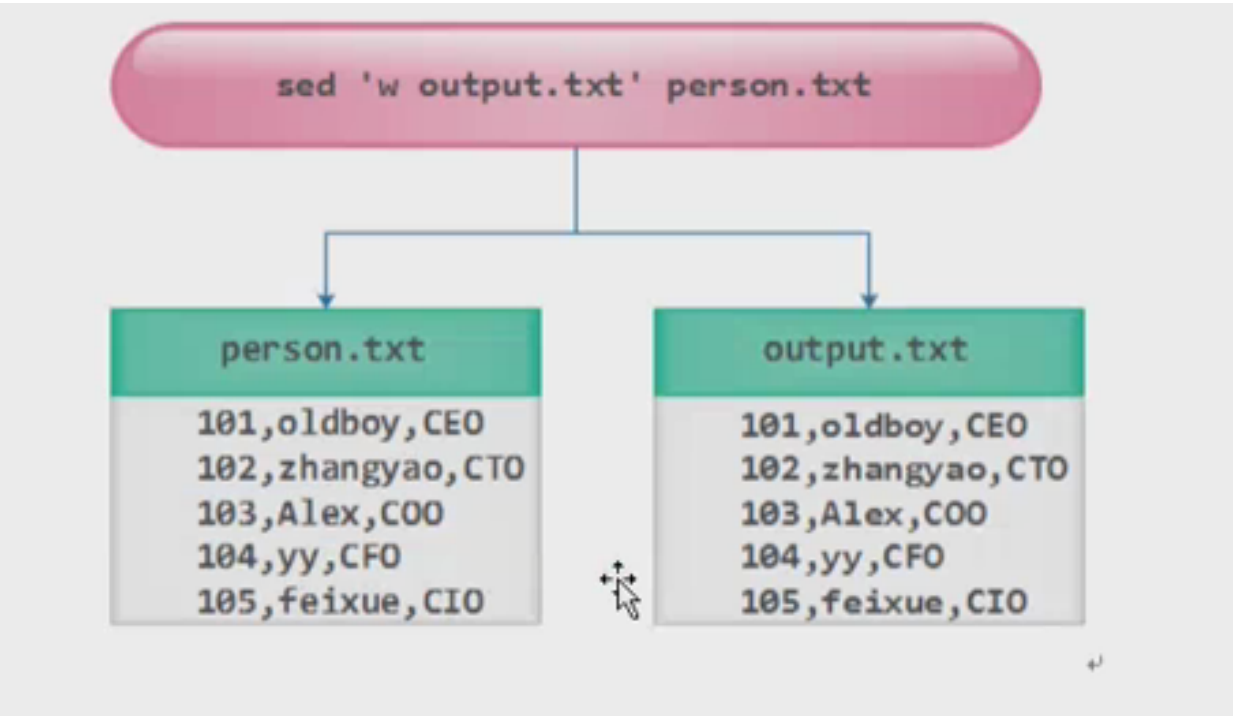
```
[root@Alaska141 sed]# cat person.txt.ori > person.txt
[root@Alaska141 sed]# cat person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明：使用 person.txt.ori 还原文件，也算是 cat 的一种用法吧。



4.另存文件

命令 w 可以把当前模式空间的内容保存到文件中，选项-i 是直接修改源文件默认情况下模式空间的内容每次都会打印到标准输出（屏幕），如果要把输出内容保存到文件同时不显示到屏幕上，还需要使用-n 选项取消默认输出。



```
[root@Alaska141 sed]# sed 'w output.txt' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明：sed 命令 w 后面直接接另存的文件名 output.txt 即可，但是屏幕上依然有命令结果输出。

```
[root@Alaska141 sed]# cat output.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明：查看保存的文件 output.txt

```
[root@Alaska141 sed]# sed -n 'w output.txt' person.txt
```

命令说明：使用 -n 选项取消默认输出，这样就可以不输出到屏幕上。

```
[root@Alaska141 sed]# cat output.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO

[root@Alaska141 sed]# sed -n '2w output.txt' person.txt
[root@Alaska141 sed]# cat output.txt
102,zhangyao,CTO
```

命令说明：这里也可以利用地址范围的功能，比如只另存为第 2 行到 output.txt

其实这个功能有点鸡肋，因为我们还会重定向符号，同样能够达到另存为的目的

```
[root@Alaska141 sed]# sed -n '2p' person.txt > output.txt
[root@Alaska141 sed]# cat output.txt
102,zhangyao,CTO
```

5.sed 软件替换命令详解

下面讲的是替换命令的最常用用法 sed -i 's#口#▷#g' oldboy.log
下面会完整的介绍 sed 软件替换命令。

5.1 替换命令语法

```
sed '[address-range|pattern-range] s#original-string#replacement-string#[substitute-flags]' [input file]

sed '[地址范围|模式范围] s#[被替换的字符串]#[替换后的字符串]#[替换标志]' [输入文件]
```

命令说明：

1.[地址范围|模式范围]:可选选项，如果没有指定，sed 软件将在所有行上进行替换。

2. “s”即执行替换命令 substitute

3. 被替换的字符串，可以是一个正则表达式

4. 替换后的字符串，只能是一个具体的内容

5. 替换标志：可选选项，包括前面详述的全局标志 g，还有数字标志（1，2，3...），打印标志 p，写标志 w，忽略大小写标志 i，执行命令标志 e，根据需要可以把一个或多个替代标志组合起来使用。

5.2 Ms# # #Ng 的使用

语法说明：

Ms ==》对第 M 行处理，无 g 替换标志，只处理第一行匹配，有 g 替换标志则对第 M 行全部替换

Ng ==》对每一行，从第 N 出开始替换

Ms,Ng 组合表示只对第 M 行从第 N 处匹配开始替换

案例演示：

使用下面的命令语句创建新的测试文件，通过下面的矩阵我们可以清晰的看到替换的结果。

```
[root@Alaska141 sed]# sed 's#1#0#2g' num.txt
1 0 0 0 0
1 0 0 0 0
1 0 0 0 0
1 0 0 0 0
```

命令说明 s#1#0#2g 表示全局替换每一行从第二处开始匹配的内容

```
[root@Alaska141 sed]# sed ' 2s#1#0#g ' num.txt
```

```
1 1 1 1 1
0 0 0 0 0
1 1 1 1 1
1 1 1 1 1
```

命令说明：2s#1#0#g 表示只针对第 2 行进行全局替换。

```
[root@Alaska141 sed]# sed '2s#1#0#2g' num.txt
1 1 1 1 1
1 0 0 0 0
1 1 1 1 1
1 1 1 1 1
```

命令说明：2s###2g 只针对第二行从第二处匹配开始替换。

5.3 数字标志

数字标志，s# # #N，注意 N 后面没有 g，这种用法表示替换每行中第 N 次出现的匹配项，属于一种精确匹配，N 后面的取值范围：1<N<512

```
[root@Alaska141 sed]# cat person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]# sed '1s#o#XXX#2i' person.txt
101,oldbXXXy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明：替换第一行的第二个匹配的 o，并且忽略大小写

```
[root@Alaska141 sed]# sed '1s#o#XXX#3i' person.txt
```

```
101,oldboy,CEXXX
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
命令说明：只处理第一行的第三个匹配的 o，并且忽略大小写
[root@Alaska141 sed]# sed '1s#o#XXX#3' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
命令说明：只处理第一行的第三个匹配的小写的 o，不忽略大小写，因为没有第三个匹配小写的 o，所以没有任何修改
```

5.4 打印标志

打印标志 p：当替换操作完成后，打印替换后的行，与选项-n 连用。大家学到这里估计又懵了。这里的 p 和前面我学过的 sed 命令 p 有什么关系？其实他们都是打印 print 英文缩写，但是这里的 p 是附属与 sed 命令 s 的，是一个标志，就这么点区别，功能一样，角色不同。

```
[root@Alaska141 sed]# sed -n 's#zhangyao#NB#p' person.txt
102,NB,CTO
命令说明：
首先将 zhangyao 替换成 NB，然后使用 p 标志打印到屏幕，同时使用-n 选项取消默认输出，
这个标志可以实现修改完文本直接显示修改的结果
```

5.5 写标志

写标志 w，当替换操作执行成功后，它把替换后的结果保存到文件中，这个也类似于前面学过的 sed 命令 w。

```
[root@Alaska141 sed]# sed -n 's#zhangyao#NB#w output.txt' person.txt
命令说明：首先将 zhangyao 替换成 NB，然后使用 w 标志接一个新的文件名，这样就可以将替换后的结果保存到这个新文件中，使用 w 标志的好处是不会影响文件。

[root@Alaska141 sed]# cat output.txt
102,NB,CTO
```

5.6 忽略大小写标志

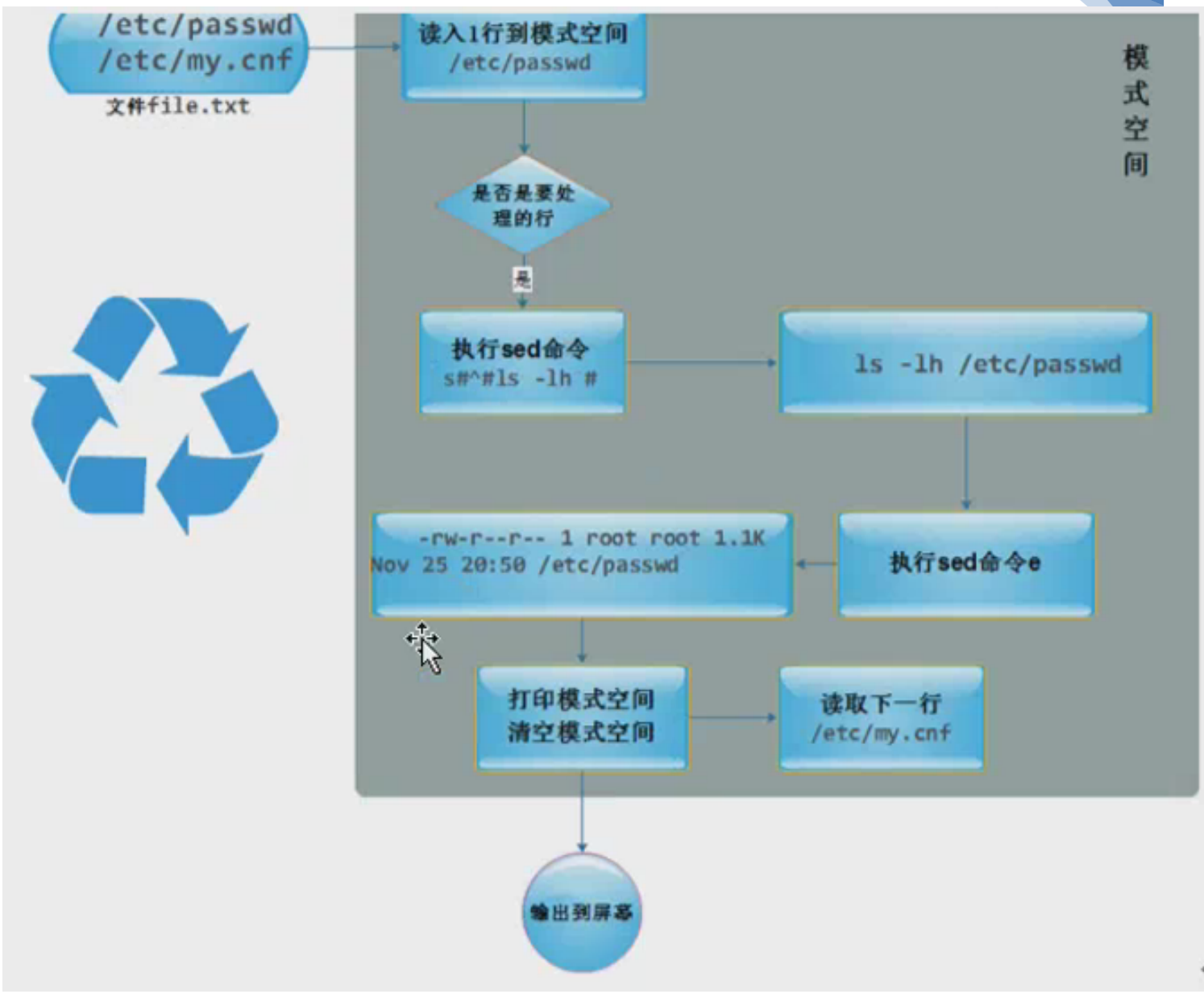
前面我们学习过的匹配功能都是要将匹配字符原样写上，是区分大小写。这也是默认的功能，但是我们可以使用忽略大小写标志 i 来忽略大小写。

```
[root@Alaska141 sed]# cat person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]# sed 's#alex#NB#i' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,NB,COO
104,yy,CFO
105,feixue,CIO
```

5.7 执行命令标志

执行命令标志 e，可以将模式空间中的任何内容当作 shell 命令执行，并把命令执行的结果返回模式空间。

```
[root@Alaska141 sed]# cat file.txt
/etc/passwd
/etc/my.cnf
命令说明：新建一个测试文本

[root@Alaska141 sed]# sed 's#^#ls -lh #e' file.txt
-rw-r--r-- 1 root root 2.1K Feb 17 21:33 /etc/passwd
-rw-r--r-- 1 root root 251 May 11  2016 /etc/my.cnf
命令说明：注意，“^”尖角号代表每一行最头头的地方，后面的命令后有一个空格要注意！
我们可以使用 p 打印出来看一下：
[root@Alaska141 sed]# sed 's#^#ls -lh #' file.txt    ###有空格，会显示正确
ls -lh /etc/passwd
ls -lh /etc/my.cnf
[root@Alaska141 sed]# sed 's#^#ls -lh#' file.txt     ###无空格会报错
ls -lh/etc/passwd
ls -lh/etc/my.cnf
命令说明：使用 p 标志将替换后的模式空间的内容打印到屏幕上，我们就可以看到上面的结果，上面的结果是一个个可以被 bash 命令执行后的字符串，因此前面的例子使用 e 标志把这些字符串交付给命令行执行，然后将执行后的结果输出到屏幕上。
```


/etc/passwd
/etc/my.cnf
文件file.txt
读入1行到模式空间
/etc/passwd
模式空间
是否是要处理的行
是
执行sed命令
s#^#ls -lh #
ls -lh /etc/passwd
-rw-r--r-- 1 root root 1.1K Nov 25 20:50 /etc/passwd
执行sed命令e
打印模式空间
清空模式空间
读取下一行
/etc/my.cnf
输出到屏幕

5.8 其他替换标志

替换标志	含义
\l	在替换字符中 shiyong\l 标志时，他会把紧跟其后面的 1 个字符当做小写字符来处理
\L	在替换字符中 shiyong\L 标志时，他会把后面所有的字符都当作小写字符来处理
\u	在替换字符中使用\u 标志时，他会把紧其后面的 1 个字符当作大写字符来处理
\U	在替换字符中使用\U 标志时，他会把后面所有的字符都当作大写字符来处理。
\E	需要\U 或\L 一起使用，他将关闭\U 或\L 的功能

举例；

```
[root@Alaska141 sed]# sed -n 's#zhangyao#ZH\lANGYAO#p' person.txt
102,ZHaNGYAO,CTO
[root@Alaska141 sed]# sed -n 's#zhangyao#ZH\LANGYAO#p' person.txt
102,ZHangyao,CTO
[root@Alaska141 sed]# sed -n 's#zhangyao#zh\uangyao#p' person.txt
102,zhAngyao,CTO
[root@Alaska141 sed]# sed -n 's#zhangyao#zh\Uangyao#p' person.txt
102,zhANGYAO,CTO
[root@Alaska141 sed]# sed -n 's#zhangyao#zh\Uangy\Eao#p' person.txt
102,zhANGYao,CTO
```

命令提示：大家看完上面的例子，肯定是吐槽慢慢，这些功能有什么用？其实上面的例子只是为了实验参数，没什么实际意义，因为我们已经知道要替换的字符串。重点：这些参数的意义是针对不确定的字符串，比如前面讲过的分组替换。

```
[root@Alaska141 sed]# sed -r 's#(.*),(.*),(.*)#\L\3,\E\1,\U\2#' person.txtceo,101,OLDBOY
cto,102,ZHANGYAO
coo,103,ALEX
cfo,104,YY
cio,105,FEIXUE
```

命令说明：将源文件的第 3 列字符串变成小写，然后移动到第 1 列；源文件的第 1 列移动到第 2 列；源文件的第 2 列移动到第 3 列，并将字符串变成大写，怎么实现？

6.特殊符号=获取行号

面试题：

如何获取一个文件的行号

```
[root@Alaska141 sed]# sed "=" person.txt
1
101,oldboy,CEO
2
102,zhangyao,CTO
3
103,Alex,COO
4
104,yy,CFO
5
105,feixue,CIO
```

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，行号和行不再一行。

```
[root@Alaska141 sed]# sed "1,3=" person.txt
1
101,oldboy,CEO
2
102,zhangyao,CTO
3
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明：只打印 1，2，3 行的行号，同时打印输入文件中的内容

```
[root@Alaska141 sed]# sed "/zhangyao/=" person.txt
101,oldboy,CEO
2
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明：打印包含关键字“zhangyao”的行的行号，同时打印输入文件中的内容

```
[root@Alaska141 sed]# sed -n "/zhangyao/=" person.txt
2
```

命令说明：只显示行号但不显示行的内容即取消默认输出

```
[root@Alaska141 sed]# sed -n "$=" person.txt
5
```

命令说明：“$=”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。

改进方法：

```
[root@Alaska141 sed]# sed "=" person.txt|sed 'N;s#\n# #'
1 101,oldboy,CEO
2 102,zhangyao,CTO
3 103,Alex,COO
4 104,y y,CFO
5 105,feixue,CIO
```

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 来弥补这个缺点。sed 命令 N 读取下一行数据并附加到模式空间，在第 13.1.2 节会具体分析这个命令。

使用 sedsed 调试工具可以看的更明显一些：

```
[root@Alaska141 sed]# sed "=" person.txt|sed-fuck 'N;s#\n# #'
PATT:1$
COMM:N
PATT:1\n101,oldboy,CEO$
COMM:s#\n# #
PATT:1 101,oldboy,CEO$
1 101,oldboy,CEO
```

7.一条 sed 语句执行多条命令

面试题：用一条 sed 语句实现删除文件的第三行到末尾的数据，并把剩余的数字 10 替换为 01

7.1 选项-e

第 1 中方法使用-e 选项，每个-e 选项后可接一个命令

```
[root@Alaska141 sed]# sed -e '3,$d' -e 's#10#01#' person.txt
011,oldboy,CEO
012,zhangyao,CTO
命令说明：第一个-e 选项后接‘3，$d’表示删除文件的第三行到末尾的数据，第二个-e 选项后接‘s#10#01#’
表示数字 10 替换为 01
```

7.2 分号；的使用

第 2 种方法使用；（分号）也可以执行多条命令。这个分号其实我们在学龄命令的时候遇到过。

```
[root@Alaska141 sed]# sed '3,$d;s#10#01#' person.txt
011,oldboy,CEO
012,zhangyao,CTO
命令说明：上面 sed 语句的核心是 3，$d；s#10#01#,使用分号隔开了 2 个操作。这个方法也是推荐大家使用的。
```

7.3 选项 -f

第 3 种方法使用-f 选项接 sed 脚本。把多个 sed 命令合并到一个文件中，这个文件被称为 sed 脚本，然后使用-f 选项调用他。

```
[root@Alaska141 sed]# echo -e '3,$d\ns#10#01#'>>person.sed
[root@Alaska141 sed]# cat person.sed
3,$d
s#10#01#
[root@Alaska141 sed]# sed -f person.sed person.txt
011,oldboy,CEO
012,zhangyao,CTO
命令说明：使用 sed 选项-f 接 sed 脚本 person.sed。这个方法仅供大家参考。
```

企业案例 6：一个文件 100 行，把 5，35，70 行单独拿出来

前面我们取行都是连续的行，从第 5 行到第 10 行，那么如何取不连续的行？

```
[root@Alaska141 sed]# sed -n '5p;35p;70p' /etc/services
# IANA services version: last updated 2009-11-10
qotd            17/tcp          quote
whois++         63/udp
```

命令说明：上面讲的 3 种方法都可以实现，这里以最简单的方法即分号“；”来使每一个操作都隔开

其他方法：awk 也可以指定显示行号

```
[root@Alaska141 sed]# awk '{if (NR==5||NR==35||NR==70)print $0}' /etc/services
# IANA services version: last updated 2009-11-10
qotd            17/tcp          quote
whois++         63/udp
```

8.特殊符号｛｝的使用

｛command｝可以把多个命令括起来，用来处理单个地址或者地址范围。

要求：打印文件的第 2 行到第 4 行，并只显示这 3 行的行号

```
[root@Alaska141 sed]# sed -n '2,4p' person.txt
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
```

命令说明：打印第 2 到 4 行

```
[root@Alaska141 sed]# sed -n '2,4p;=' person.txt
1
102,zhangyao,CTO
2
103,Alex,COO
3
104,yy,CFO
4
5
```

命令说明：

使用我们前面学习过的分号，将显示第 2 行到 4 行“2，4p”和下士行号“=”结合起来。但是我们发现一个问题：虽然正确的将第 2 行到第 4 行显示出来，但是行号显示不对，没有只显示第 2 到 4 行，为什么呢？

sed 软件使用-n 选项取消默认输出，sed 软件读入第 1 行，不符合“2，4p”命令，跳过；但符合“=”，因此输出第 1 行行号。sed 软件读入第 2 行，符合 2，4p 命令，输出第 2 行内容；又符合“=”，输出第 2 行行号，最后结果就和上面显示的一样。

那怎么样才能达到题目要求呢？这里就需要借助一个特殊符号｛｝

```
[root@Alaska141 sed]# sed -n '2,4{p;=}' person.txt
102,zhangyao,CTO
2
103,Alex,COO
3
104,yy,CFO
4
```

命令说明：使用"{}"则表示命令“p”和“=”是一个整体，对第 2 到 4 行执行“｛｝”的内容

```
[root@Alaska141 sed]# sed -n '2,4{=;p}' person.txt
2
102,zhangyao,CTO
3
103,Alex,COO
4
104,yy,CFO
```

命令说明：花括号的内容换个顺序也是可以的

9.打印不可见字符 l

```
[root@Alaska141 sed]# sed -n 'l' person.txt
101,oldboy,CEO$
102,zhangyao,CTO$
103,Alex,COO$
104,yy,CFO$
105,feixue,CIO$
```

命令说明：这个功能有点类似 cat 命令的-A 选项。这个命令有助于我们查看文件的特殊字符，进行一些学习辅助或排错等。比如学习正则表达式的时候，为什么“$”代表文件的结尾！

```
[root@Alaska141 sed]# sed -n 'l 12' person.txt
101,oldboy,\
CEO$
102,zhangya\
o,CTO$
103,Alex,CO\
O$
104,yy,CFO$
105,feixue,\
CIO$
```

命令说明：如果在 l 后面指定了数字 y，那么会在第 y 个字符处自动折行

10.转换字符 y

命令 y 根据对应位置转换字符。大写字符转换为小写，反之亦然。

```
[root@Alaska141 sed]#  sed 'y#abc#ABC#' person.txt
101,oldBoy,CEO
102,zhAngyAo,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
命令说明：把 a 换为 A，b 换为 B，c 换为 C。是一一对应，并不是将 abc 换为 ABC，这个事 s 替换命令的功能，

[root@Alaska141 sed]#  sed 'y#abc#123#' person.txt
101,old2oy,CEO
102,zh1ngy1o,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
命令说明：只要一一对应即可，可以是数字也可以是字母
```

11.退出 sed

命令 q 终止正在执行的命令并退出 sed。正常的 sed 执行流程是：读取数据、执行命令、打印结果、重复循环。当 sed 遇到 q 命令，便立刻退出，当前循环中的后续命令不会被执行，也不会继续循环。

```
[root@Alaska141 sed]# sed '' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]# sed 'q' person.txt
101,oldboy,CEO
命令说明：sed 软件读取第 1 行就遇到命令 q，因此打印第 1 行后退出。
```

```
[root@Alaska141 sed]# sed '3q' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
```

命令说明：打印第 3 行后退出，即显示前 3 行。

```
[root@Alaska141 sed]# sed '/Alex/q' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
```

命令说明：打印所有行，直到遇到包含关键字 Alex 的行停止退出

注意！q 命令不能指定地址范围（或模式范围），只能用于单个地址（或单个模式）
大家需要小心，在想要编辑文件的前一部分并保留剩余部分不改变的情况下，不要使用 q 命令，在执行 q 命令之后，就不会在产生输出，在这种情况下使用 q 是初学者常犯的错误。

```
[root@Alaska141 sed]# cat person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]# sed -i 'q' person.txt
[root@Alaska141 sed]# cat person.txt
101,oldboy,CEO
```

命令说明：使用-i 选项将修改写入到文件中，查看修改后的文件只剩下一行了，很危险的操作。

12 从文件读取数据

在处理输入文件时，命令 r 会从另一个文件读取内容，并在指定的位置打印出来

```
[root@Alaska141 sed]# sed 'r num.txt' person.txt
101,oldboy,CEO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
102,zhangyao,CTO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
103,Alex,COO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
104,yy,CFO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
105,feixue,CIO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

命令说明：

命令 r 接另一个文件名 num.txt,并从中读取内容。执行流程是从 person.txt 文件读取 1 行，然后追加 num.txt 文件的内容。接着再读取 person.txt 文件的第 2 行，再追加 num.txt 文件的内容。。。。

上面的 sed 语句因为没有指定合并的位置，因此就在 person.txt 文件每行后面都追加了 num.txt 文件的内容。

```
[root@Alaska141 sed]# sed '$r num.txt' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

命令说明：指定在 person.txt 的最后一行追加 num.txt 文件的内容。

```
[root@Alaska141 sed]# sed '/Alex/r num.txt' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
104,yy,CFO
105,feixue,CIO
```

命令说明：指定在 person.txt 匹配 Alex 的行后追加 num.txt 文件的内容

13.保持空间和模式空间

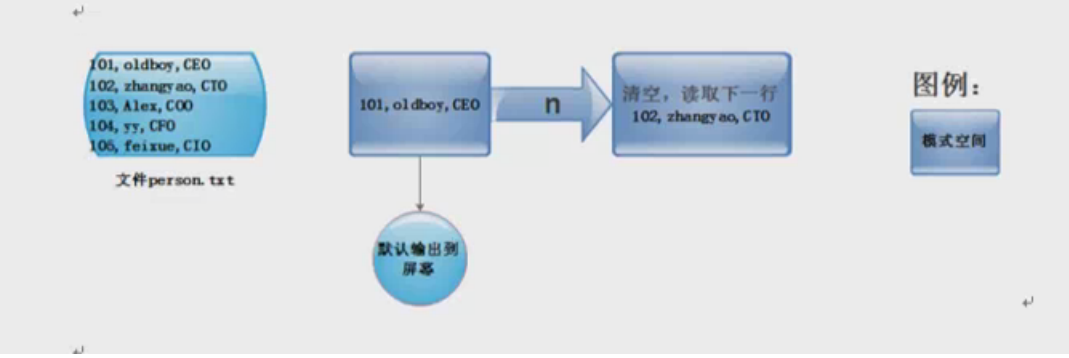
13.1 模式空间操作命令

13.1.1 命令 n

命令 n 的作用：清空当前模式空间的内容，然后从输入文件中读取下一行。如果在命令执行过程中遇到 n，那么改变他会改变正常的执行流程（读取数据，执行命令，打印输出，重复循环）。为什么这么说呢，请看下面例子：

```
sed-command-1
sed-command-2
n
sed-command-3
sed-command-4
```

这种情况下：sed-command-1 和 sed-command-2 会在当前模式空间中执行，然后遇到 n，他就会打印当前模式空间的内容。并清空模式空间，读取下一行，然后把 sed-command-3 和 sed-command-4 应用于新的模式空间的内容。




```
[root@Alaska141 sed]# sed 'n' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

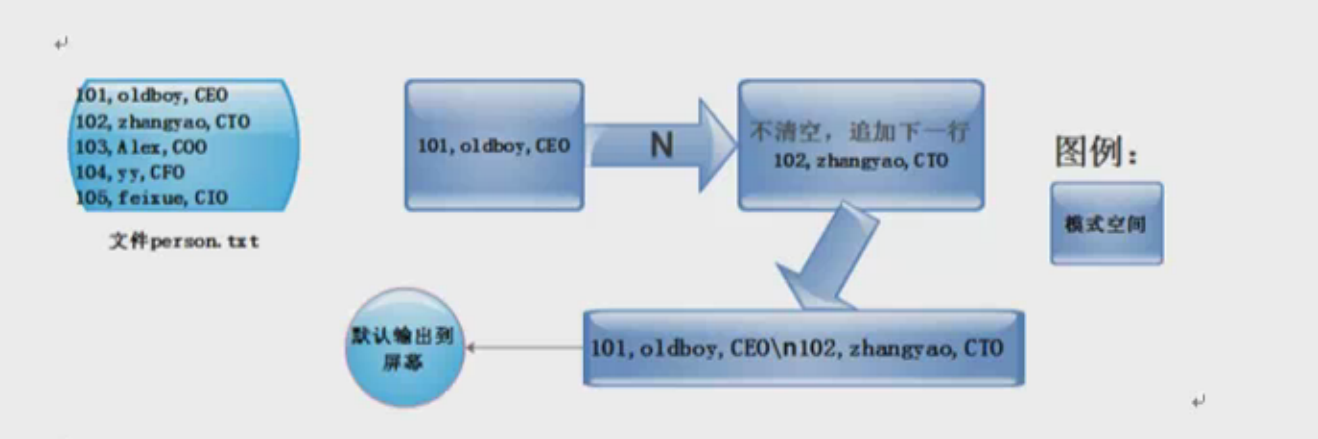
命令说明：默认打印模式空间的内容。

```
[root@Alaska141 sed]# sed -n 'n' person.txt
[root@Alaska141 sed]#
```

命令说明：读入第 1 行“101，oldboy，CEO”，模式空间被“n”清空。然后读入第 2 行“102，zhangyao，CTO”被"p"打印，此时完成一个循环：读入第 3 行，清空，读入第 4 行，打印第 4 行：读入第 5 行，清空，文件读取完毕，结束。

13.1.2 命令 N

命令 N 的作用。不会清空模式空间内，并且从输入文件中读取下一行数据，追加到模式空间中，两行数据以换行符\n 连接。



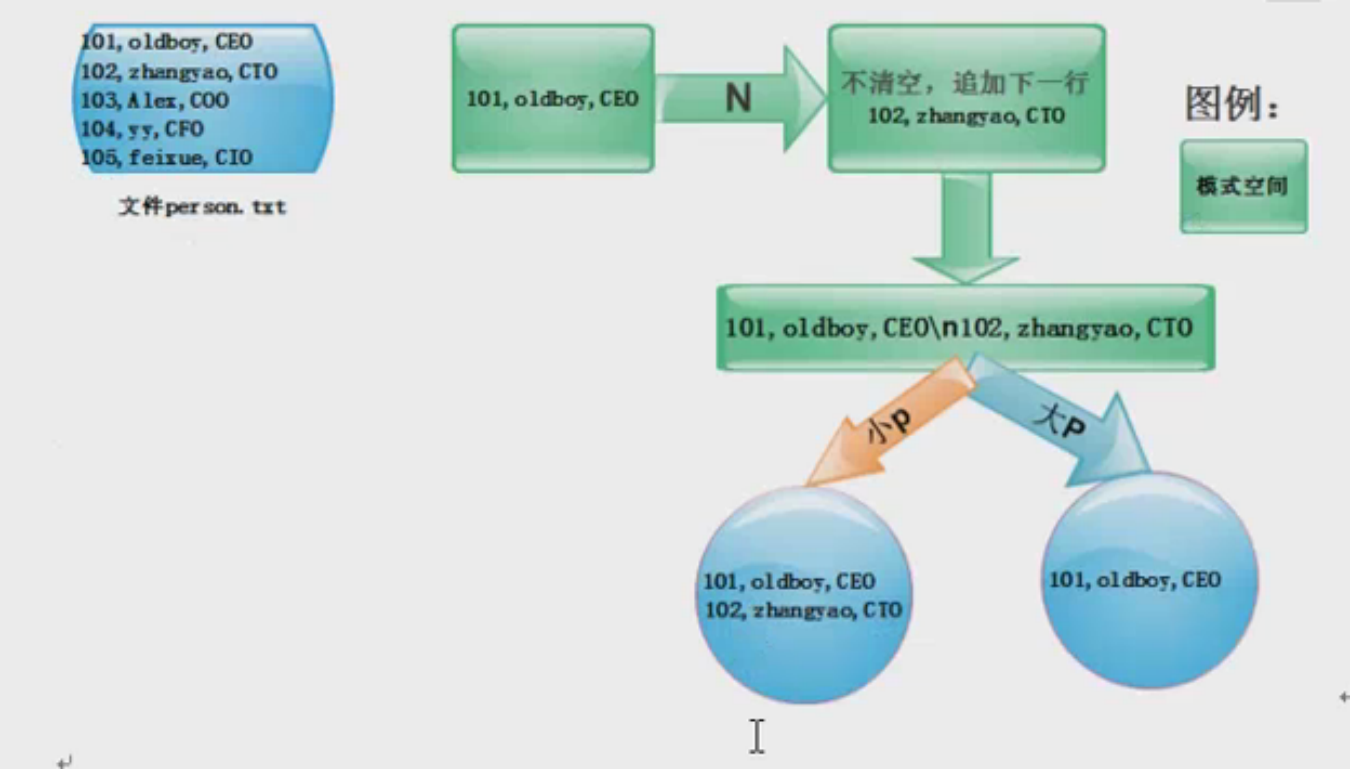

```
[root@Alaska141 sed]# sed "=" person.txt
1
101,oldboy,CEO
2
102,zhangyao,CTO
3
103,Alex,COO
4
104,yy,CFO
5
105,feixue,CIO
[root@Alaska141 sed]# sed "=" person.txt|sed 'N;s#\n# #'
1 101,oldboy,CEO
2 102,zhangyao,CTO
3 103,Alex,COO
4 104,yy,CFO
5 105,feixue,CIO
```

命令说明：首先注意的是，我们处理的文本是第 1 个例子的结果（管道前命令执行的结果）。因此，第一行是“1”存入模式空间，碰到命令“N”，读取第二行，“101，oldboy,CEO”,此时模式空间内容为“1\n101,oldboy,CEO”然后执行“s#\n# #g”将“\n”替换为“1 101，oldboy,CEO”,输出到屏幕上。依次循环

13.1.3 打印多行模式中的第一行 p

前面的命令 N 可以在模式空间形成多行文本，因此针对这种多行模式，sed 软件有一些特定的命令。

小写的命令 p 会打印模式空间的所有内容，大写的 P 也打印模式空间内容，但遇到换行符\n 结束操作。




```
[root@Alaska141 sed]# sed -n 'N;p' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
```

命令说明：

因为取消了默认输出，“101，oldboy,CEO\n102,zhangyao,CTO”是模式空间的第 1 次内容，使用“p”输出“101，oldboy,CEO”和“102，zhangyao,CTO”这 2 行内容。

第 2 次内容“103，Alex,COO\n104,yy,CFO”,使用“p”输出"103,Alex,COO"和“104,yy,CFO”这 2 行内容

第 3 次只剩下“105，feixue,CIO”,提前结束，不会执行后面的命令“p”，默认的输出“105，feixue,CIO”被“-n”屏蔽了，结果如上所示。

```
[root@Alaska141 sed]# sed -n 'N;P' person.txt
101,oldboy,CEO
103,Alex,COO
```

命令说明：

因为取消了默认输出，“101，oldboy,CEO\n102,zhangyao,CTO”是模式空间的第 1 次内容，使用“P”输出模式空间内容，但是遇到"\n"字符就结束操作，因此只输出“101,oldboy,CEO”

第 2 次内容“103，Alex,COO\n104,yy,CFO”,使用“P”输出。“103,Alex,COO”

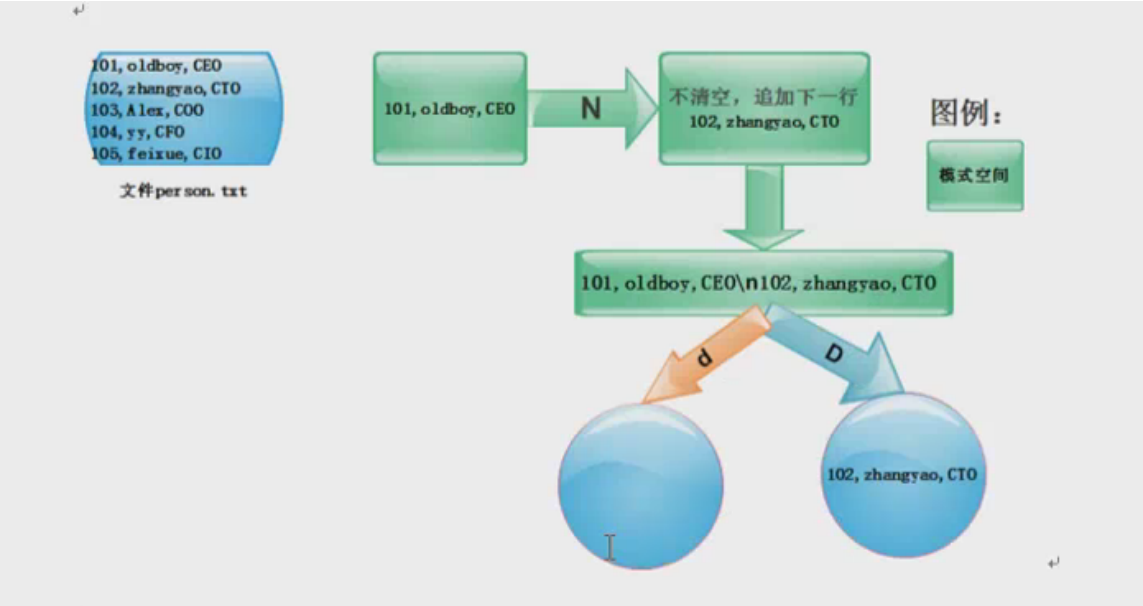
第 3 次只剩下“105,feixue,CIO”,提前结束，不会执行后面的命令"P"

默认的输出“105,feixue,CIO”被“-n”屏蔽了，结果如上所示。

13.1.4 删除多行模式中的第一行 D

sed 命令 d 会删除模式空间内容，关于命令 d 的用法，在常用功能==》删除已经讲解的很详细了，这里就不再多说了

sed 命令 D，即不会读取下一条记录，也不会完全清空模式空间（除非模式空间内只有一行）。他只会删除模式空间的部分内容，直接遇到换行符\n；忽略后续命令，在当前模式空间中从头开始执行命令。




```
[root@Alaska141 sed]# sed 'N;D' person.txt
105,feixue,CIO
命令说明：
模式空间存入第 1 行“101,oldboy,CEO”,执行命令 N 读取下一行，此时模式空间变为
“101,oldboy,CEO/n102,zhangyao,CTO”：
接着再执行命令 D，遇到"\n"结束，此时模式空间剩“102，zhangyao,CTO”：
然后从头开始执行 N，模式空间变“102，zhangyao,CTO\n103,Alex,COO”
执行命令 D 模式空间 "103,Alex,COO"
执行命令 N 模式空间 "103,Alex,COO\n104,yy,CFO"
执行命令 D 模式空间 "104,yy,CFO"
执行命令 N 模式空间 "104,yy,CFO\n105,feixue,CIO"
执行命令 D 模式空间 "105,feixue,CIO"
执行命令 N 文件结束 sed 循环终止
最后,默认打印模式空间内容"105,feixue,CIO",如果使用-n 选项就可以不打印
```

企业案例 7:SVN 帐号密码案例

很久以前,由于一些未知原因,我们的 SVN 服务器的生成帐号密码的脚本生成的帐号密码如下:

stu100

5ece1c4e

stu101

b851544d

stu102

d4d09288

stu103

3eb4bf73

.....省略

但是 SVN 服务器识别的帐号密码格式为:

stu100=997f42a5

stu101=0c4ab328

stu102=346edd70

stu103=ec231043

....省略

聪明的你来帮老师决绝这个麻烦吧,方法不限,手段不限

(提示:将奇数行和偶数行用"="等号连接)

测试文件:echo 生成序列

```
[root@Alaska141 sed]# echo -e stu{100..110}"\n"$(date +%N|md5sum|cut -c 1-8)"\n"|xargs -n1 >>
    svn.txt
[root@Alaska141 sed]# cat svn.txt
stu100
997f42a5
stu101
0c4ab328
stu102
346edd70
stu103
ec231043
...省略
```

解法一:使用 xargs 和 sed

```
[root@Alaska141 sed]# cat svn.txt |xargs -n2|sed -n 's# #=#gp'
[root@Alaska141 sed]# cat svn.txt |xargs -n2|sed -r 's#(.*) (.*)#\1=\2#g'
```

解法二:使用 sed

```
[root@Alaska141 sed]# sed 'N;s#\n#=#g' svn.txt
```

解法三:AWK

```
[root@Alaska141 sed]# awk '{printf "%s=",$0;getline;print}' svn.txt
stu100=997f42a5
.....省略
```

命令说明:printf 与 print 的区别:默认情况 printf 不会自动添加换行符,需要手动添加.

%s 格式替代符,对应后面的$0,\n 是换行符

```
[root@Alaska141 sed]# awk '{print $0}' svn.txt
stu100
997f42a5
[root@Alaska141 sed]# awk '{printf $0}' svn.txt
stu100997f42a5stu1010c4ab328stu102346edd70stu103ec231043stu104c485f890stu105388b888estu10671888b03s
    tu107296cd7f6stu1086a4b2ebcstu109ccc8b689stu110ab0fc520[root@Alaska141 sed]#

### %s 是格式替代符,对应后面的$0,\n 是换行符
[root@Alaska141 sed]# awk '{printf "%s\n",$0}' svn.txt
stu100
997f42a5
stu101
0c4ab328
....省略

###为了符合题意,把\n 替换为=
[root@Alaska141 sed]# awk '{printf "%s=",$0}' svn.txt
stu100=997f42a5=stu101=0c4ab328=stu102=346edd70=stu103=ec231043=stu104=c485f890=stu105=388b888e=stu
106=71888b03=stu107=296cd7f6=stu108=6a4b2ebc=stu109=ccc8b689=stu110=ab0fc520=[root@Alaska141 sed]#
```

2.getline 的功能:读入下一行的数据赋予$0

===>printf "%s=",$0

结果:stu100=(不换行)

===>getline;print $0

结果:$0 为 997f42a5

拼起来

stu100=997f42a5(print 自动换行)

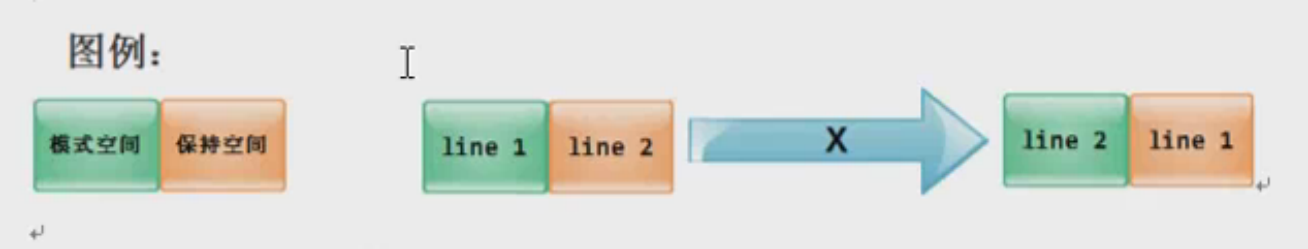
```
最终结果输出:
[root@Alaska141 sed]# awk '{printf "%s=",$0;getline;print}' svn.txt
stu100=997f42a5
stu101=0c4ab328

最后 print 后面加上$0 效果一样
[root@Alaska141 sed]# awk '{printf "%s=",$0;getline;print $0}' svn.txt
stu100=997f42a5
stu101=0c4ab328
```

13.2 保持空间操作命令

13.2.1 用保持空间替换模式空间

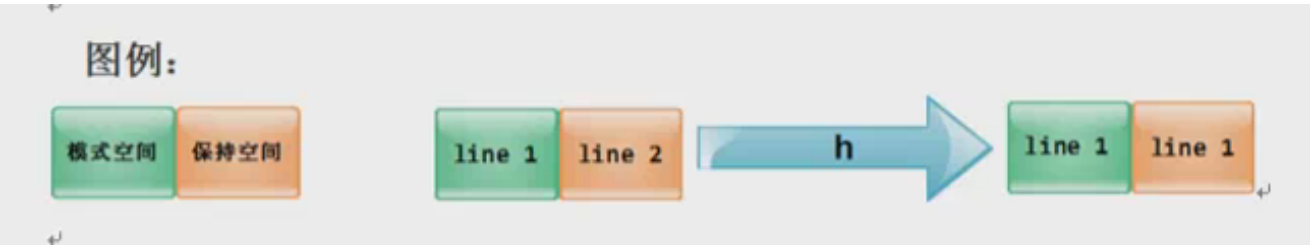
交换模式空间和保持空间的内容.该命令本身没有多大用处,但如果和其他命令配合使用,则非常强大了.

假设目前模式空间内容为 line 1,保持空间内容为 line 2.那么执行命令 x 后,模式空间的内容变为 line 2 , 保持空间的内容为 line 1



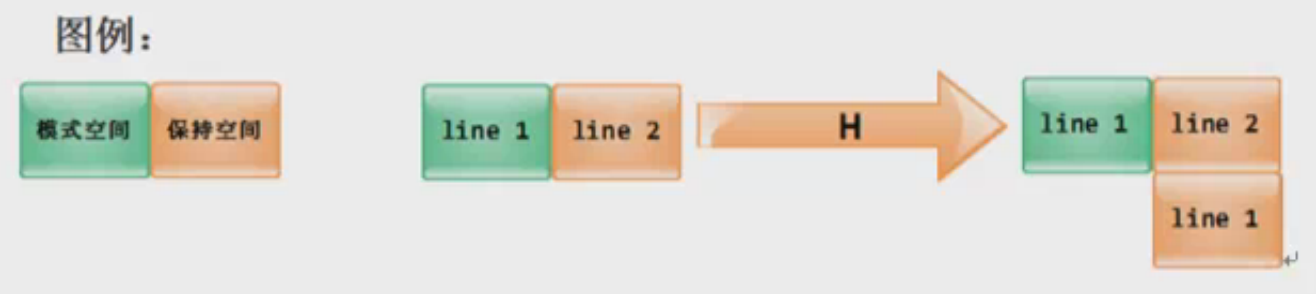
13.2.2 把模式空间的内容复制到保持空间 h

假定目前模式空间内容为 line 1,保持空间内容为 line 2.那么执行命令 h 后,模式空间的内容仍为 line 1,保持空间的内容则变为 line 1



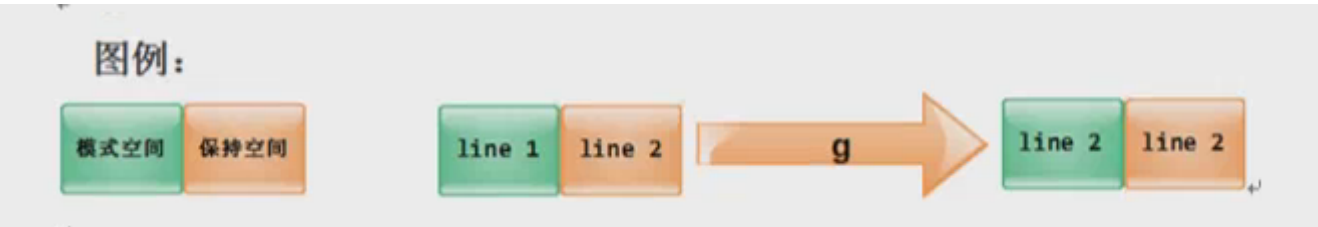
13.2.3 把模式空间内容追加到保持空间 H

假定目前模式空间内容为 line 1,保持空间内容为 line 2.那么执行命令 H 后,模式空间的内容没有改变,仍为 line 1,保持空间的内容则变为 line2\line 1



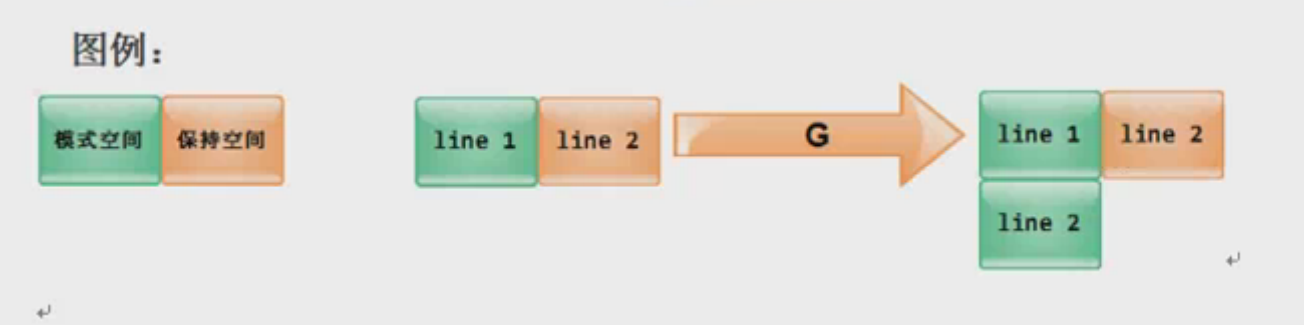
13.2.4 把保持空间内容复制到模式空间 g

假定当前模式空间内容为 line 1,保持空间内容为 line 2;执行命令 g 之后,模式空间内容变为 line 2,保持空间内容仍为 line 2



13.2.5 把保持空间追加到模式空间 G

假定当前模式空间为 line 1,保持空间内容为 line 2;命令 G 执行后,模式空间内容为 line 1\nline 2,同时保持空间内容不变,仍为 line 2.



13.2.6 命令综合运用

举例 1:给指定行前加一个空行

```
[root@Alaska141 sed]# sed '/zhang/{x;p;x}' person.txt
101,oldboy,CEO

102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
```

命令说明:

上面命令的结果是在匹配 zhang 的所有行前面插入一个空行.我们先看其他 4 行没有匹配"zhang"的执行情况,因为没有匹配上,所以后面的"{x;p;x}"都不会执行,因此最后默认打印模式空间,输出这个的内容.

接下来,我们再看一下第 2 行的执行流程,因为匹配上正则表达式"/zhang/",因此执行后面的命令"{x;p;x}"

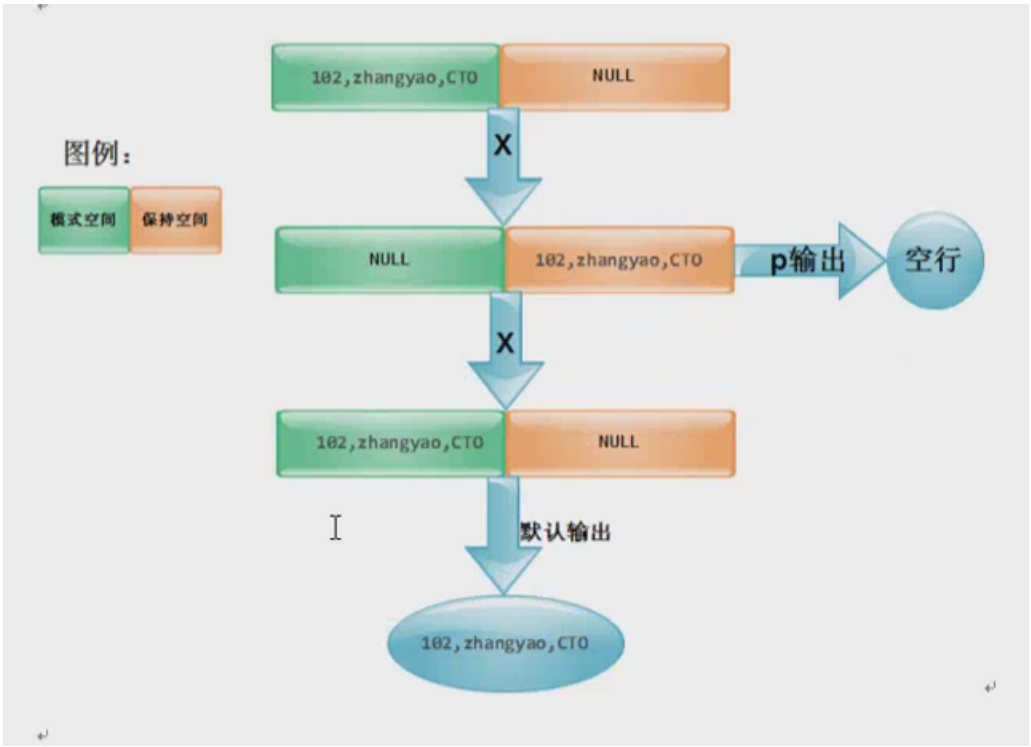
最开始模式空间内容为"102.zhangyao,CTO",保持空间内容为空;

第一步执行命令 x : 交换模式空间和保持空间的内容,此时模式空间内容为空,保持空间爱你内容为"102,zahngyao,CTO";

第二步执行命令 p : 打印模式空间内容,因此输出内容为空行;

第三步执行命令 x : 交换模式空间和保持空间的内容,此时模式空间内容为"102,zhangyao,CTO",保持空间内容 空

最后,默认输出模式空间的内容 102,zhangyao,CTO



例 2:给文件的每一行后都追加一个空行

```
[root@Alaska141 sed]# sed 'G' person.txt
101,oldboy,CEO

102,zhangyao,CTO

103,Alex,COO

104,yy,CFO

105,feixue,CIO
```

命令说明:上面命令的结果是给文件的每一行后都追加一个空行.

命令执行流程:

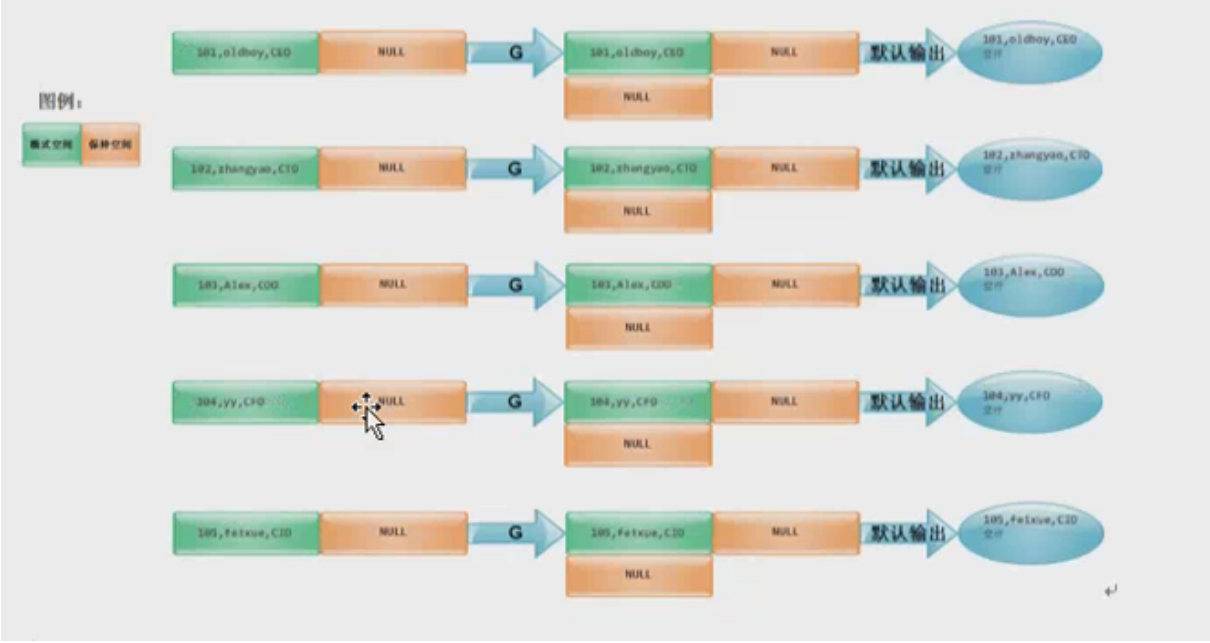
读取第一行"101,oldboy,CEO" 到模式空间,同时保持空间的内容为空,此时遇到命令 G,将保持空间的内容追加到模式空间,因此模式空间的内容为"101,oldboy,CEO\n '空行' ",最后默认输出如下:

101,oldboy,CEO\n '空行' ,最后默认输出如下:

101,oldboy,CEO

这是一个空行

后面的 4 行的执行流程和第 1 行一致



举例 3:实现倒叙打印文本内容

现有文本内容如下所示,如何将文件反向输出?

```
[root@Alaska141 sed]# cat person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]# tac person.txt
105,feixue,CIO
104,yy,CFO
103,Alex,COO
102,zhangyao,CTO
101,oldboy,CEO
```

命令说明:使用 sed 命令实现上面的效果,tac 命令用于反向输出文件内容.

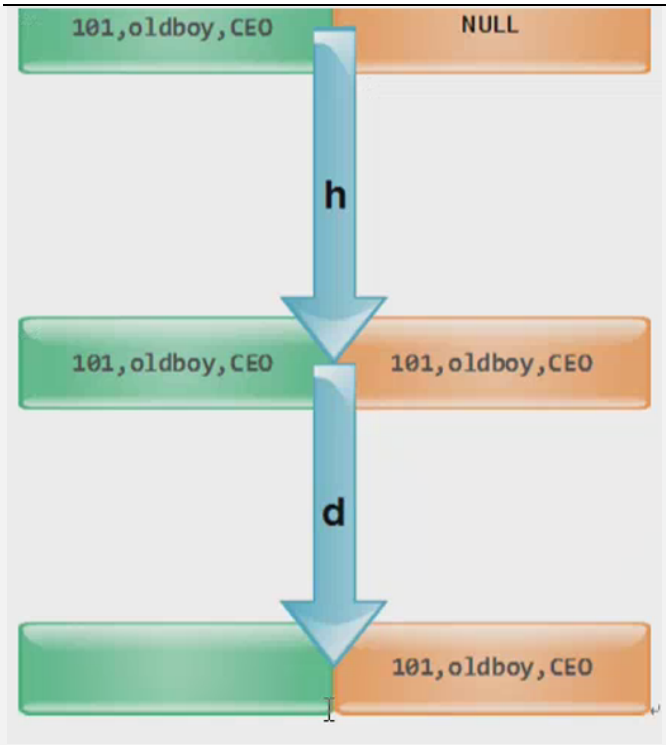
使用 sed 实现反向输出:

```
[root@Alaska141 sed]# sed '1!G;h;$!d' person.txt
105,feixue,CIO
104,yy,CFO
103,Alex,COO
102,zhangyao,CTO
101,oldboy,CEO
```

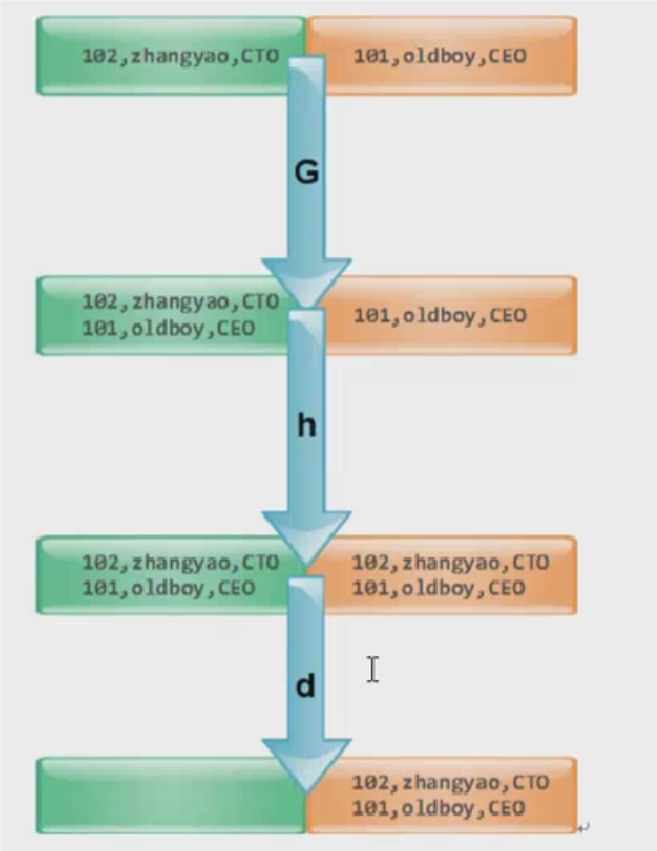
命令说明:

最开始我们看一下命令组合'1!G;h;$!d',执行命令组合'1!G;h;$!d'

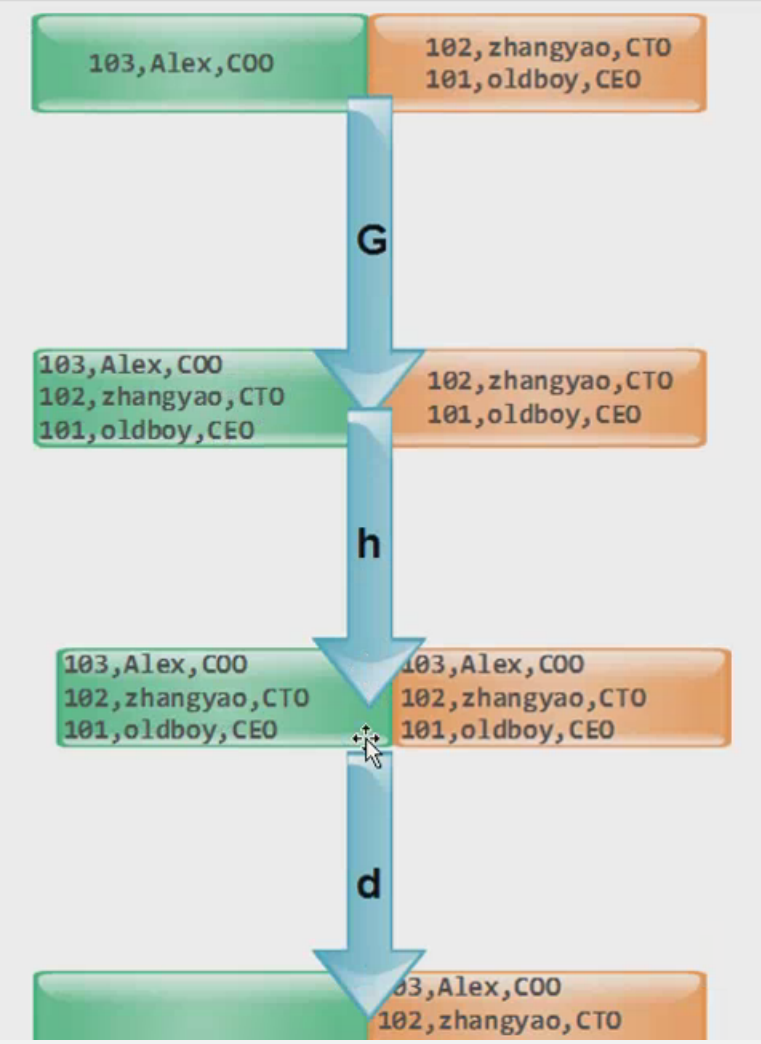
因为是第一行,所以"1!G"不执行.执行命令 h,将模式空间内容复制到保持空间.接着删除模式空间内容.



读入第 2 行"102, zhangyao, CTO" , 第 1 次执行命令 G, 将保持空间的内容追加到模式空间, 此时模式空间内容为"102, zhangyao, CTO\n101, oldboy, CEO";第 2 次执行命令 h, 将模式空间内容复制到保持空间, 最后删除模式空间容



读入第 3 行"103,Alex,COO",第 1 次执行命令 G,将保持空间的内容追加到模式空间,此时模式空间内容为"103,Alex,COO\n102,zhangyao,CTO\n101,oldboy,CEO";第 2 次执行命令 h,将模式空间内容复制到保持空间:最后删除模式空间内容.



读入第 4 行"104,yy,CFO" 第一次执行命令 G,将保持空间的内容追加到模式空间,此时模式空间内容为"104,yy,CFO\n103,Alex,COO\n102,zhangyao,CTO\n101,oldboy,CEO";第 2 次执行命令 h,将模式空间内容复制到保持空间,最后删掉模式空间内容.

14．循环和分支

一般情况下,sed 是将编辑命令从上到下,从左到右依次应用到读入模式空间的行上,但是像前面学过的 d/D /n /N 等命令能够在一定程度上改变默认的执行流程,甚至利用 N/D/P 三个命令可以形成一个强大的循环处理流.除此之外,其实 sed 还提供了分支命令(b)和测试命令(t)这两个命令来控制流程,这两个命令可以跳转到指定的标签(label)位置继续执行命令.标签是以冒号开头的标记,如":oldboy",":oldgirl"等.

分支命令 b 和测试命令 t 的作用与差异

- 分支命令 b 格式为"b label",sed 遇到这个分支命令会跳转到该标签 label,然后执行 label 后面的命令.命令 b 后面可以不跟任何标签,这种情况下,他会直接跳到 sed 脚本的结尾.
- 测试命令 t:格式为 "t label" ， 如果"t label"前面的替换(s)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.
- 测试命令 T:与 t 相反,如果"T label"前面的替换(s)命令执行不成功,那么久跳转到 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.

15. 操作多个文件

```
[root@Alaska141 sed]# sed -n ’/root/ p’ /etc/passwd /etc/group
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
root:x:0:alaska141,catcat
```

16.模拟其他命令

16.1.1 查看文件内容

实现原理,sed 命令默认会输出文件所有内容,只要命令不修改文件内容即可.

```
[root@Alaska141 sed]# sed -n 'p' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO

[root@Alaska141 sed]# sed 'n' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO

[root@Alaska141 sed]# sed 'N' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO

[root@Alaska141 sed]# sed 's# # #' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO

命令说明:上面列出了几种常见方法,当然还有更多的方法.
```

16.1.2 合并 2 个文件

```
[root@Alaska141 sed]# cat person.txt num.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1

[root@Alaska141 sed]# sed '$r num.txt' person.txt
101,oldboy,CEO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
```

16.1.3 模拟 grep 命令

```
[root@Alaska141 sed]# sed -n '/zhangyao/p' person.txt
102,zhangyao,CTO
[root@Alaska141 sed]# sed -n 's#zhangyao#&#p' person.txt
102,zhangyao,CTO
命令说明:查询含有 zhangyao 的行.grep 'zhangyao' person.txt

[root@Alaska141 sed]# sed -n '/oldboy/ !p' person.txt
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
命令说明:排除含有 oldboy 的行.grep -v 'oldboy' person.txt
```

16.1.4 模拟 head 命令

```
[root@Alaska141 sed]# sed '3,$d' person.txt
101,oldboy,CEO
102,zhangyao,CTO
[root@Alaska141 sed]# sed '1,2p' person.txt
101,oldboy,CEO
101,oldboy,CEO
102,zhangyao,CTO
102,zhangyao,CTO
103,Alex,COO
104,yy,CFO
105,feixue,CIO
[root@Alaska141 sed]# sed '2q' person.txt
101,oldboy,CEO
102,zhangyao,CTO
命令说明:获取文件前 2 行. head -2 person.txt
```

16.1.5 模拟 wc 命令

```
[root@Alaska141 sed]# sed -n "$=" person.txt
5
命令说明:打印文件的总行数. wc -l person.txt
```

附录 ：sed 调试工具 sedsed

1.2 eval 特殊用法

```
hostname -I

命令说明：使用单引号显示 hostname -I 是因为所见即所得，使用双引号还是 hostname -I 是因为引号里面的不是变量，而是一个可以执行的命令。

[root@oldboy ~]# echo `hostname -I`
192.168.20.131

命令说明：想要在引号里面执行命令，需要用到反引号，也叫倒引号。

[root@oldboy ~]# echo ``hostname -I`'
`hostname -I`

命令说明：倒引号外层套单引号，所以结果是`hostname -I`。

[root@oldboy ~]# eval echo ``hostname -I`'
192.168.20.131

命令说明：这里就用到了 eval 命令了，它可以将单引号内的变量或命令优先解析或执行。
```