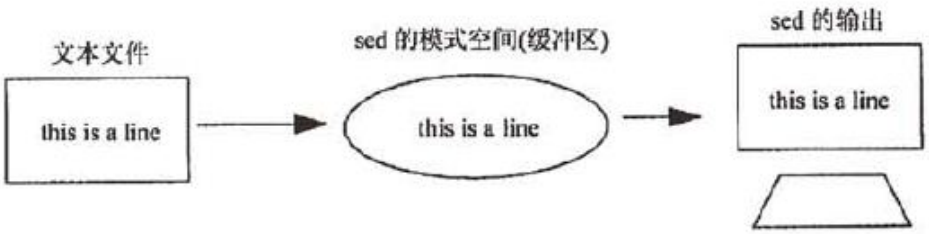
第4天-Shell 流编辑器 sed

一、sed 流编辑器介绍

- Linux 中，常使用流编辑器 sed 进行文本替换工作。与常使用的交互式编辑器（如vim）不同，sed 编辑器以批处理的方式来编辑文件，这比交互式编辑器快得多，可以快速完成对数据的编辑修改。sed 编辑器会执行以下操作：

1. 一次从输入中读取一行数据；
2. 根据所提供的编辑器命令匹配数据；
3. 按照命令修改流中的数据；
4. 将新的数据输出到 STDOUT。

- 在 sed 编辑器匹配完一行数据后，它会读取下一行数据并重复这个过程，直到处理完所有数据。使用 sed 命令打开一个 sed 编辑器。



- sed 是一种在线的、非交互式的编辑器，它一次处理一行内容。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”（pattern space），接着用sed命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有改变，除非你使用重定向存储输出。Sed主要用来自动编辑一个或多个文件；简化对文件的反复操作；编写转换程序等。

二、sed 命令格式

```
sed [options] edit_commands [file]       # [ ] 中的内容为可选可不选
```

注意：sed 和 grep 不一样，不管是否找到指定的模式，它的退出状态都是0，只有当命令存在语法错误时，sed的退出状态才是非0

三、sed 支持正则表达式

sed 在文件中查找模式时也可以使用正则表达式(RE)和各种元字符。正则表达式是括在斜杠间的模式，用于查找和替换，以下是sed支持的元字符。

元字符	功 能	示 例	示例的匹配对象
^	行首定位符	/^love/	匹配所有以 love 开头的行
$	行尾定位符	/love$/	匹配所有以 love 结尾的行
.	匹配除换行外的单个字符	/l..e/	匹配包含字符 l、后跟两个任意 字符、再跟字母 e 的行
*	匹配零个或多个前导字符	/*love/	匹配在零个或多个空格紧跟着模式 love 的行
[]	匹配指定字符组内任一字符	/[Ll]ove/	匹配包含 love 和 Love 的行
[^]	匹配不在指定字符组内任一字符	[^A-KM-Z]ove	匹配包含 ove，但 ove 之前的那 个字符不在 A 至 K 或 M 至 Z 间的行
..	保存已匹配的字符		
&	保存查找串以便在替换串中引用	s/love/&/	符号&代表查找串。字符串 love 将替换前后各加了两个的引 用，即 love 变成love
<	词首定位符	/<love/	匹配包含以 love 开头的单词的 行
>	词尾定位符	/love>/	匹配包含以 love 结尾的单词的 行
x{m}	连续 m 个 x	/o{5}/	分别匹配出现连续 5 个字母 o、 至少 5 个连续的 o、或 5~10 个 连续的 o 的行
x{m,}	至少 m 个 x	/o{5,}/	
x{m,n}	至少 m 个 x，但不超过 n 个 x	/o{5,10}/	

四、sed 常用选项

命令	功能描述
-n	使用安静（silent）模式。在一般sed的用法中，所有来自stdin的资料一般都会被列出到屏幕，但如果加上-n参数后，则只有经过sed特殊处理的那一行（或者command）才会被列出来。
-e	允许多点编辑。
-f	直接将sed的动作写在一个档案内，-f filename 则可以执行filename内的sed动作。
-r	sed 的动作支援的是延伸型正规表示法的语法。(预设是基础正规表示法语法)
-i	直接修改读取的档案内容，而不是由屏幕输出。

五、sed 常用命令选项

命令	功能描述
a\	新增， a 的后面可以接字串，而这些字串会在新的一行出现(目前的下一行)；
c\	取代， c 的后面可以接字串，这些字串可以取代 n1,n2 之间的行；
d	删除，因为是删除啊，所以 d 后面通常不接任何咚咚；
i\	插入， i 的后面可以接字串，而这些字串会在新的一行出现(目前的上一行)；
p	打印，亦即将某个选择的资料印出。通常 p 会与参数 sed -n 一起运作；
s	取代，可以直接进行取代的工作哩！通常这个 s 的动作可以搭配正规表示法！例如 1,20s/old/new/g 就是啦；

六、sed 高级命令选项

命令	功能描述
h	拷贝pattern space的内容到holding buffer(特殊缓冲区)。
H	追加pattern space的内容到holding buffer。
g	获得holding buffer中的内容，并替代当前pattern space中的文本。
G	获得holding buffer中的内容，并追加到当前pattern space的后面。
n	读取下一个输入行，用下一个命令处理新的行而不是用第一个命令。
P	打印pattern space中的第一行。 //大写
q	退出sed。
w file	写并追加pattern space到file的末尾。
!	表示后面的命令对所有没有被选定的行发生作用。
s/re/string	用string替换正则表达式re。
=	打印当前行号码。

七、sed 替换标志

命令	功能描述
g	在行内进行全局替换
w	将行写入文件
x	交换暂存缓冲区与模式空间的内容
y	将字符转换为另一字符（不能对正则表达式使用 y 命令）

八、sed 命令使用实例

1、sed 命令进行文本替换

1、sed 使用 s 命令来进行文本替换操作

```
[root@qfedu.com ~]# sed 's/srcStr/dstStr/' file
```

- 其中，srcStr 为想要替换的文本，dstStr 为将要替换成的文本。使用 s 命令时，sed 编辑器会在一行一行地读取文件 file，并在每行查找文本 srcStr，如果找到了，则将该处的 srcStr 替换为 dstStr。
- / 字符为界定符，用于分隔字符串（sed 编辑器允许使用其他字符作为替换命令中的字符串分隔符）：

```
[root@qfedu.com ~]# sed 's!/bin/bash!/BIN/BASH!' /etc/passwd    # 使用 ! 作为字符串分隔符
[root@qfedu.com ~]# sed 's#3#88#g' datafile
```

- 默认情况下，替换命令只会替换掉目标文本在每行中第一次出现的地方。若想要替换掉每行中所有匹配的地方，可以使用替换标记 g。替换标记放在编辑命令的末尾。除了 g 外，还有几种替换标记：

1、数字指明替换掉第几次匹配到的文本，

- 没有设置这个标记时，默认是替换第一次匹配的文本

```
[root@qfedu.com ~]# sed 's/root/ROOT/2' /etc/passwd
```

- 这行命令将 /etc/passwd 文件中每行的第 2 个 root 替换为 ROOT

2、g 替换所有匹配到的文本

```
[root@qfedu.com ~]# sed 's/root/ROOT/g' /etc/passwd
```

- 这行命令将 /etc/passwd 文件中的 root，全部替换为 ROOT；

3、p 打印与替换命令中指定模式（srcStr）相匹配的行

```
[root@qfedu.com ~]# sed 's/root/ROOT/p' /etc/passwd
```

- 执行这命令，会在 STDOUT 上看到包含有 root 的行被输出了两次，一次是 sed 编辑器自动输出的；另一次则是 p 标记打印出来的匹配行
- 单独地使用 p 标记没什么用处，通常将 p 标记和 -n 选项结合起来使用，这样就可以只输出被匹配替换过的行了

```
[root@qfedu.com ~]# sed -n 's/root/ROOT/gp' /etc/passwd       # 将 /etc/passwd 中所有的 root 都替换成 ROOT，并输出被修改的行
```

注：可以使用 " = " 命令来打印行号，用法与 p 一样。

4、w file ：将替换的结果写到文件中

- 只保存被修改的行，与 -n + p 的功能类似

```
[root@qfedu.com ~]# sed -n 's/root/ROOT/g w change.txt' /etc/passwd # 将/etc/passwd 中所有的 root 都替换成 ROOT,并将被修改的行保存到文change.txt 中去
```

5、正则匹配

```
[root@qfedu.com ~]# sed -r 's/^west/north/' datafile
[root@qfedu.com ~]# sed -r 's/[0-9][0-9]$/&.5/' datafile    # &代表在查找串中匹配到的内容
```

2、sed 使用行寻址对特定行进行编辑

默认情况下，sed 编辑器会对文件中的所有行进行编辑。当然，也可以只指定特定的某些行号，或者行范围来进行流编辑，这需要用到行寻址。所指定的行地址放在编辑命令之前：

```
[address] commands
```

1、使用数字方式进行行寻址

- sed 编辑器将文本流中的每一行都进行编号，第一行的编号为 1 ，后面的按顺序分配行号。通过指定特定的行号，可以选择编辑特定的行。举几个例子：

```
[root@qfedu.com ~]# sed '3 s/bin/BIN/g' /etc/passwd       # 将第3行中所有的 bin 替换成 BIN
[root@qfedu.com ~]# sed '2,5 s/bin/BIN/g' /etc/passwd     # 将第2到5行中所有的 bin 替换成 BIN

[root@qfedu.com ~]# sed '10,$ s/bin/BIN/g' /etc/passwd    # 将第10行到最后一行中所有的 bin 替换成 BIN
```

注：行寻址不止对替换命令有效，对其他命令也都是有效的，后面也会用到。

2、使用文本模式过滤器过滤行

- sed 编辑器允许指定文本模式来过滤出命令要作用的行，格式如下：

```
/pattern/command
```

- 必须使用斜杠符 " / " 将要指定的文本模式 pattern 包含起来。sed 编辑器会寻找匹配文本模式的行，然后对这些行执行编辑命令：

```
[root@qfedu.com ~]# sed -n '/root/s/bin/BIN/p' /etc/passwd    # 寻找包含有字符串 root 的行，并将匹配行的 bin 替换为 BIN
```

- 与数字寻址一样，也可以使用文本过滤区间来过滤行：

```
[root@qfedu.com ~]# sed '/pattern1/,/pattern2/ edit_command' file
```

- 这行命令会在文件 file 中先寻找匹配 pattern1 的行，然后从该行开始，执行编辑命令，直到找到匹配 pattern2 的行。但是需要注意的是，使用文本区间过滤文本时，只要匹配到了开始模式（ pattern1 ），编辑命令就会开始执行，直到匹配到结束模式（ pattern2 ），这会导致一种情况：一个文本中，先匹配到了一对 pattern1、pattern2，对该文本区间中的文本执行了编辑命

令；然后，在 pattern2 之后又匹配到了 pattern1，这时就会再次开始执行编辑命令，因此，在使用文本区间过滤时要注意这一点

```
[root@qfedu.com ~]# sed -n '/root/,/nologin/ s/bin/BIN/p' /etc/passwd
root:x:0:0:root:/root:/BIN/bash
BIN:x:1:1:bin:/bin:/sbin/nologin
operator:x:11:0:operator:/root:/sBIN/nologin
games:x:12:100:games:/usr/games:/sBIN/nologin
```

3、sed 命令删除行

- sed 编辑器使用 d 命令来删除文本流中的特定行。使用 d 命令时，一般需要带上位寻址，以删除指定的行，否则默认会删除所有文本行。

```
[root@qfedu.com ~]# sed '/root/d' /etc/passwd    # 删除匹配 root 的行

[root@qfedu.com ~]# sed '2,$d' /etc/passwd       # 删除第2到最后一行

[root@qfedu.com ~]# sed '3d' /etc/passwd
[root@qfedu.com ~]# sed '3{d;}' /etc/passwd
[root@qfedu.com ~]# sed '3{d}' /etc/passwd
[root@qfedu.com ~]# sed '$d' /etc/passwd
[root@qfedu.com ~]# sed '/north/d' /etc/passwd
[root@qfedu.com ~]# sed '/sout/d' /etc/passwd
```

4、sed 命令插入和附加文本

- sed 编辑器使用 i 命令来向数据流中插入文本行，使用 a 命令来向数据流中附加文本行。其中：i 命令会在指定行前增加一个新行；a 命令会在指定行后增加一个新行。
- 需要注意的是，这两个命令都不能在单个命令行上使用（即不是用来在一行中插入或附加一段文本的），只能指定插入还是附加到另一行。

```
[root@qfedu.com ~]# sed '[address][i | a]\newline' file
```

- newline 中的文本即为将要插入或附加在一行前面或后面的文本。常常使用这两个命令结合行寻址在特定的行前面或后面增加一个新行。

```
[root@qfedu.com ~]# sed 'i\Insert a line behind every line'
/etc/passwd          # 向数据流的每一行前面增加一个新行，新行的内容为 \ 后面的内容
[root@qfedu.com ~]# sed '1i\Insert a line behind the first line'
/etc/passwd     # 在数据流的第一行前面增加一个新行
[root@qfedu.com ~]# sed '3a\Append a line after the third line' /etc/passwd
# 在数据流的第三行后面增加一个新行
[root@qfedu.com ~]# sed '$a\Append a line in the last line'
/etc/passwd          # 在数据流的最后一行后面增加一个新行
```

5、sed 命令修改行

- 使用命令 c 可以将数据流中的整行文本修改为新的行，与插入、附加操作一样，这要求在 sed 命令中指定新的行

```
[root@qfedu.com ~]# sed '[address][c]\newtext' file
```

- newtext 中的文本为匹配行将要被修改成的文本。

```
[root@qfedu.com ~]# sed '3 c\New text' /etc/passwd          # 将数据流中第三行的内容修改为 \ 后面的内容
[root@qfedu.com ~]# sed '/root/ c\New text' /etc/passwd    # 将匹配到 root 的行的内容修改为 \ 后面的内容
[root@qfedu.com ~]# sed '2,4c\New text' /etc/passwd         # 将第2到4行的内容修改为 \ 后面的内容，但是不是逐行修改，而是会将这之间的 3 行用一行文本来替代
```

- 注意这里对地址区间使用 c 命令进行修改时，不会逐行修改，而是会将整个区间用一行修改文本替代。

6、sed 命令逐字符转换

- 使用 y 参数可以按要求对文本进行逐字符转换。格式如下：

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

- 转换命令会对 inchars 和 outchars 的值进行一对一的映射。inchars 中的第一个字符会被转换成 outchars 中的第一个字符；inchars 中的第二个字符会被转换成 outchars 中的第二个字符；... 直到处理完一行。如果 inchars 和 outchars 的长度不同，则 sed 编辑器会产生一个错误消息。举个例子：

```
[root@qfedu.com ~]# echo abcdefggfedcba | sed 'y/acg/ACG/'
 AbCdefGGfedCbA
```

7、sed 命令处理文件

1、向文件中写入数据

- 前面已经提到过，可以使用 w 命令向文件写入行。格式如下：

```
[address]w filename
```

- 举个例子

```
[root@qfedu.com ~]# sed '1,2w test.txt' /etc/passwd
```

- 该语句将数据流的第 1、2 行写入文件 test.txt 中去。

2、从文件中读取数据

- 可以使用 r 命令来将一个文本中的数据插入到数据流中去，与普通的插入命令 i 类似，这也是对行进行操作的，命令格式如下：

```
[address]r filename
```

- filename 为要插入的文件。r 命令常结合行寻址使用，以将文本插入到指定的行后面。举个例子：

```
[root@qfedu.com ~]# sed '3 r test.txt' /etc/passwd
```

- 这句话将文件 test.txt 中的内容插入到数据流第三行后面去。

九、sed 模式空间和保持空间

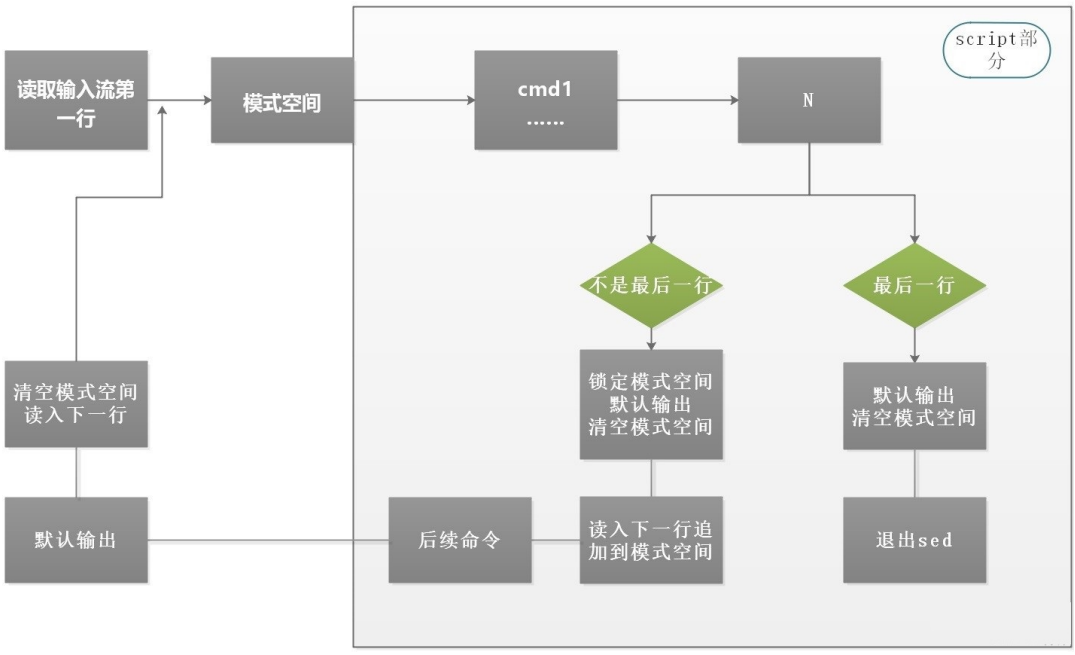
1、模式空间和保持空间介绍

- 模式空间和保持空间是两个独立的缓冲区，可以进行交互，命令可以寻址模式空间但是不能寻址保持空间。
- 模式空间：容纳当前输入行的缓冲区，通过模式匹配到的行被读入模式空间中。用来进行进一步的操作；在多行模式中，'\n'可以用来和模式空间(N命令的结果)的任意换行符匹配，单模式空间底部的换行符除外。^匹配多行的首，$匹配多行的尾，不是每行的行首和行尾
- 保持空间：作为辅助的一个缓冲区，可以和模式空间进行交互，但是命令不能直接作用于保持空间。可以通过h,H,g,G与模式空间进行交互。

2、模式空间和保持空间命令

sed命令都是一行一行的进行处理文本的，不过有些时候单行处理可能并不能满足我们的需要，所以sed还提供了多行模式，多行模式的命令主要有 NPD 这三个，

- N：读取匹配到的行的下一行追加至模式空间
- P：打印模式空间开端至\n内容，并追加到默认输出之前
- D：如果模式空间包含换行符，则删除直到第一个换行符的模式空间中的文本， 并不会读取新的输入行，而使用合成的模式空间重新启动循环。如果模式空间 不包含换行符，则会像发出d命令那样启动正常的新循环
- N命令的大致流程图，P 和 D 命令在懂了N命令之后也容易理解。



- 需要注意的是，之所以叫多行模式是因为可以存放不止两行，如下示例：

```
[root@qfedu.com ~]# cat /etc/passwd |sed -n '2{N;p}'
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
# 读取第二行的下一行，然后输出模式空间中的内容，此时模式空间中有两行

[root@qfedu.com ~]# cat /etc/passwd |sed -n '2{N;N;N;p}'
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
# 使用多个N命令可以读取多行进模式空间
```

- sed在处理文本的时候都是在模式空间中进行，但有时候有些复杂的操作单一的模式空间可能无法满足需求，于是就有了保持空间，这个空间通常是空闲的，并不处理数据，只在有需要的时候和模式空间进行一些必要的数据交换。下面是模式空间中的常用命令。
  - h: 把模式空间中的内容覆盖至保持空间中
  - H：把模式空间中的内容追加至保持空间中
  - g: 从保持空间取出数据覆盖至模式空间
  - G：从保持空间取出内容追加至模式空间
  - x: 把模式空间中的内容与保持空间中的内容进行互换
  - 假如你有两个杯子，
    - h 就相当于吧二个杯子中的东西替换为第一个杯子中的，
    - H就相当于把第一个杯子中的东西放到第二个杯子中，并且第二个杯子中的东西还在，
    - g和h相似，不过是反了过来吧第一个杯子中的东西替换为第二个杯子中的，
    - G也是和H类似，x就相当于把两个杯子中的东西进行了交换。
- 实例

```
[root@qfedu.com ~]# seq 1 6| sed -n '1,2H;4p;5{x;p}'
4

1
2
```

- -n 是不显示默认输出内容，1,2H是将前两行追加至保持空间，4p显示第四行，5{x;p}是在第五行的时候交换保持空间和模式空间中的内容并且输出。注意输出中的空行，这是因为H命令追加的时候是添加换行符，由于保持空间默认是空的，所以添加换行符之后就多了一个空行。以用下面的命令先往保持空间覆盖一行然后追加。

```
[root@qfedu.com ~]# seq 1 6| sed -n '1h;2H;4p;5{x;p}'
4
1
2
```

- 第一个循环结束之后：模式空间为空，保持空间为第一行内容
- 第二个循环，将第二行追加到模式空间，此时模式空间为两行内容
- 第三个循环，没有匹配内容，不执行操作，模式空间和保持空间内容不变
- 第四个循环，读取第四行并输出，保持空间内容不变
- 第五个循环，读入第五行，然后和保持空间中的内容交换，之后输出。
- 实例

```
# 暂存和取用命令：h H g G
[root@qfedu.com ~]# sed -r '1h;$G' /etc/hosts
[root@qfedu.com ~]# sed -r '1{h;d};$G' /etc/hosts
[root@qfedu.com ~]# sed -r '1h; 2,$g' /etc/hosts
[root@qfedu.com ~]# sed -r '1h; 2,3H; $G' /etc/hosts

# 暂存空间和模式空间互换命令：x
[root@qfedu.com ~]# sed -r '4h; 5x; 6G' /etc/hosts
```

十、sed 常见操作

```
```

```
# 删除配置文件中#号注释行
[root@qfedu.com ~]# sed -ri '/^#/d' file.conf
[root@qfedu.com ~]# sed -ri '/^[ \t]*#/d' file.conf

# 删除配置文件中//号注释行
[root@qfedu.com ~]# sed -ri '\Y^[ \t]*//Yd' file.conf

# 删除无内容空行
[root@qfedu.com ~]# sed -ri '/^[ \t]*$/d' file.conf

# 删除注释行及空行：
[root@qfedu.com ~]# sed -ri '/^[ \t]*#/d; /^[ \t]*$/d' /etc/vsftpd/vsftpd.conf
[root@qfedu.com ~]# sed -ri '/^[ \t]*#|^[ \t]*$/d' /etc/vsftpd/vsftpd.conf
[root@qfedu.com ~]# sed -ri '/^[ \t]*($|#)/d' /etc/vsftpd/vsftpd.conf

# 修改文件：
[root@qfedu.com ~]# sed -ri '$a\chroot_local_user=YES' /etc/vsftpd/vsftpd.conf
[root@qfedu.com ~]# sed -ri '/^SELINUX=/cSELINUX=disabled' /etc/selinux/config
[root@qfedu.com ~]# sed -ri '/UseDNS/cUseDNS no' /etc/ssh/sshd_config
[root@qfedu.com ~]# sed -ri '/GSSAPIAuthentication/cGSSAPIAuthentication no'
/etc/ssh/sshd_config

# 给文件行添加注释：
[root@qfedu.com ~]# sed -r '2,6s/^/#/' a.txt
[root@qfedu.com ~]# sed -r '2,6s/(.*)/#\1/' a.txt
[root@qfedu.com ~]# sed -r '2,6s/.*/#&/' a.txt
          # &匹配前面查找的内容
[root@qfedu.com ~]# sed -r '3,$ s/^#*/#/' a.txt          将行首零个或多个#换成一个#
[root@qfedu.com ~]# sed -r '30,50s/^[ \t]*#*/#/' /etc/nginx.conf
[root@qfedu.com ~]# sed -r '2,8s/^[ \t#]*/#/' /etc/nginx.conf

# sed中使用外部变量
[root@qfedu.com ~]# var1=11111
[root@qfedu.com ~]# sed -ri '3a$var1' /etc/hosts
[root@qfedu.com ~]# sed -ri "3a$var1" /etc/hosts
[root@qfedu.com ~]# sed -ri 3a$var1 /etc/hosts

[root@qfedu.com ~]# sed -ri "$a$var1" /etc/hosts
[root@qfedu.com ~]# sed -ri '$a'"$var1" /etc/hosts
[root@qfedu.com ~]# sed -ri "\$a$var1" /etc/hosts
```

十一、Shell 脚本 sed 实战

1、Shell 脚本 sed 实现网络配置

```
#!/bin/bash
cd /etc/sysconfig/network-scripts/
cp ifcfg-eth0 ifcfg-eth0.bak

cat ifcfg-eth0 > ifcfg-eth1.txt
sed -i 's/eth0/eth1/' ifcfg-eth1.txt
sed -i '/UUID/d' ifcfg-eth1.txt
sed -i '/IPADDR/s/192.168.0.11/192.168.0.33/' ifcfg-eth1.txt
cat ifcfg-eth1.txt > ifcfg-eth1

systemctl stop NetworkManager
systemctl disable NetworkManager
systemctl restart network
```

2、Shell 脚本 sed 实现 sshd 服务自动配置

```
#!/bin/bash
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak
read -p "Port（22）:" port
sed -i "13c Port ${port}" /etc/ssh/sshd_config
read -p "LoginGraceTime(s/m/h):" LoginGraceTime
sed -i "41c LoginGraceTime ${LoginGraceTime}" /etc/ssh/sshd_config
read -p "PermitRootLogin（yes/no）:" PermitRootLogin
sed -i "42c PermitRootLogin ${PermitRootLogin}" /etc/ssh/sshd_config
read -p "UseDNS（yes/no）:" UseDNS
sed -i "122c UseDNS ${UseDNS}" /etc/ssh/sshd_config
read -p "PrintMotd（yes/no）: " PrintMotd
case $PrintMotd in
yes)
sed -i "112c PrintMotd ${PrintMotd}" /etc/ssh/sshd_config
read -p "Prompt message:" message
echo "${message}" > /etc/motd
;;
no)
sed -i "112c PrintMotd ${PrintMotd}" /etc/ssh/sshd_config
;;
esac
/etc/init.d/sshd restart &>>/dev/null
```

3、Shell 脚本 sed 实现批量 nginx 配置文件增加配置

```
#!/bin/bash
confs=`ls  /etc/nginx/conf.d/cms_vhosts/`
cd /etc/nginx/conf.d/cms_vhosts
for log in $confs;do
  name=`echo ${log} |awk -F '.' '{print $1}'`
  sed -i "/root/a\error_log   /zz_data/nginx_log/${name}.error.log;" $log
  sed -i "/error_log/a\access_log  /zz_data/nginx_log/${name}.access.log
access;" $log
done

# confs 表示 vhost 里面的每一个配置文件名称
# name 表示 配置文件名称单个字符串截取，为域名
# sed 为增加访问和错误日志
```

4、Shell 脚本实现 sed 命令关闭 selinux

```
#!/bin/bash
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

5、Shell 脚本实现 sed 命令配置 sudo

1、shell 脚本实现 sed 命令在 /etc/sudoers 配置文件中添加内容

```
#!/bin/bash
sed -i '$a\Defaults logfile=/var/log/sudo.log' /etc/sudoers
```

2、shell 脚本实现 sed 命令在 /etc/sudoers 全局替换

```
#!/bin/bash
sed -i 's/GSSAPIAuthentication yes/GSSAPIAuthentication no/g'
/etc/ssh/sshd_config
```

6、Shell 脚本实现 sed 配置 MySQL

- 处理一个 MySQL 的配置文件 my.cnf 文本，输出配置文件有几个段（以 [ ] 为段），每个段有几个配置，

```
#!/bin/bash
FILE_NAME=./my.cnf

function get_all_segments
{
    echo "`sed -n '/\[.*\]/p' $FILE_NAME  | sed -e 's/\[//g' -e 's/\]//g'`"
}

function count_items_in_segment
{
    items=`sed -n '/\['$1'\]/,/\[.*\]/p' $FILE_NAME | grep -v "^#" | grep -v
"^$" | grep -v "\[.*\]"`

    index=0
    for item in $items
    do
        index=`expr $index + 1`
    done
    echo $index
}
number=0

for segment in `get_all_segments`
do
    number=`expr $number + 1`
    items_count=`count_items_in_segment $segment`
    echo "$number: $segment  $items_count"
done
```

7、shell 脚本实现 sed 修改 zabbix_agentd.conf 配置文件

```
#!/bin/bash
zabbixserverip=$1
sed -i "s/Server=127.0.0.1/Server=${zabbixserverip}/g"
/usr/local/zabbix/etc/zabbix_agentd.conf
sed -i "s/Server=127.0.0.1/Server=${zabbixserverip}/g"
/usr/local/zabbix/etc/zabbix_agentd.conf
sed -i '262s@#
Include=/usr/local/etc/zabbix_agentd.conf.d/@Include=/usr/local/zabbix/etc/zabbi
x_agentd.conf.d/@g' /usr/local/zabbix/etc/zabbix_agentd.conf
sed -i 's/# UnsafeUserParameters=0/UnsafeUserParameters=1/g'
/usr/local/zabbix/etc/zabbix_agentd.conf
```