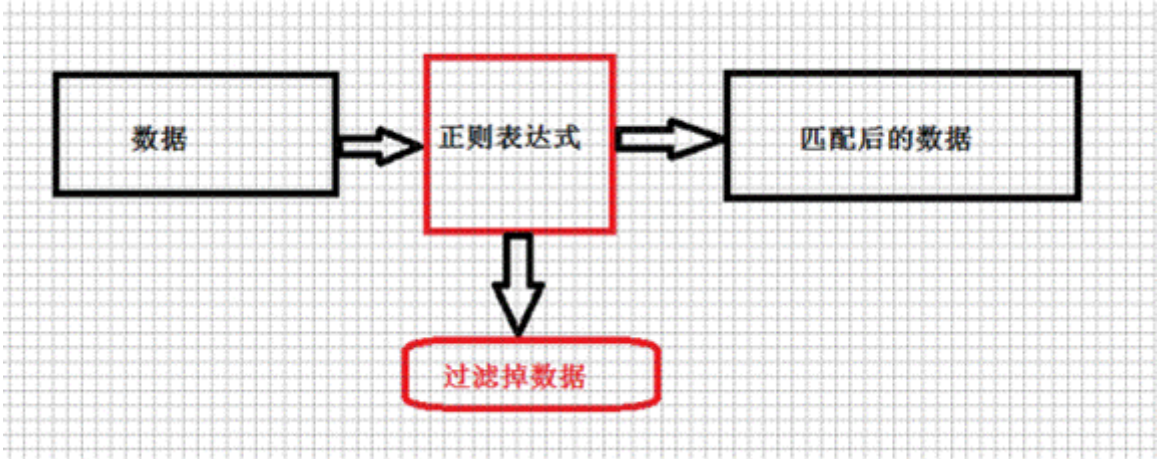
正则表达式的使用



正则表达式，又称规则表达式。（英语：Regular Expression[ˈregjulə]规则的[ikspreʃən]表达），在代码中常简写为regex、regexp或RE），计算机科学的一个概念。正则表达式通常被用来检索、替换那些符合某个模式(规则)的文本。

正则表达式不只有一种，而且LINUX中不同的程序可能会使用不同的正则表达式，如：

工具：grep sed awk(脚本三剑客)

LINUX中常用的有两种正则表达式引擎

- 基础正则表达式:BRE
- 扩展正则表达式:ERE

Shell正则表达式的组成

基础正则表达式

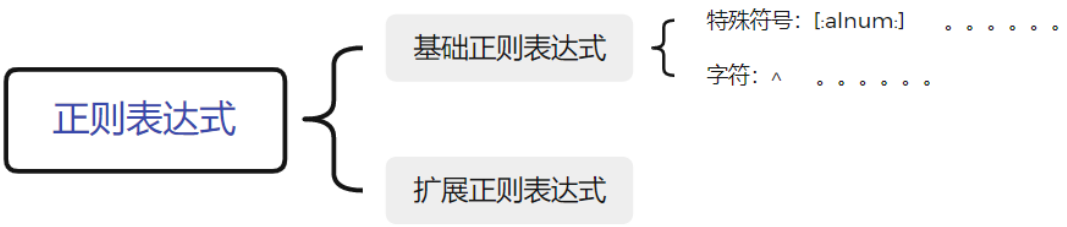
特殊符号	意义
[:alnum:]	代表英文大小写字符及数字，即0~9，a-z，A-Z
[:alpha:]	代表英文大小写字符，即a-z，A-Z
[:upper:]	代表大写字符，即A-Z
[:digit:]	代表数字，0-9
[:lower:]	代表小写字符，a-z
[:blank:]	代表空格键与tab键两者
[:cntrl:]	代表键盘上面的控制按键
[:graph:]	除了空格符外的其他所有按键
[:print:]	代表任何可以被打印出来的字符
[:punct:]	代表标点符号，即：；‘“
[:space:]	代表任何会产生空白的字符，包括空格键、tab键

基础正则字符	意义
^	待查找的字符串在首行
$	匹配输入字符串的结尾位置。要匹配$字符本身，请使用\$

- | 标记一个中括号表达式的开始。要匹配[，请使用*
  . | 匹配除换行符\n之外的任何单字符。要匹配. 请使用\.
  \ | 转义符，将特殊符号的特殊意义去除
  [list] | 匹配list中的任意单一字符 例：[0-9]
  [!list] | 匹配除list中的任意单一字符
  [^list] | 反向选择list中的任意单一字符

扩展正则表达式

```
+    | 匹配前面的字符一次或多次。要匹配+字符，请使用\\+
?    | 匹配前面的字符零次或一次。要匹配?字符，请使用\\?
\|   | 用或（or）的方式使用数个字符串
()   | 标记一个子表达式的开始和结束位置。要匹配这些字符，请使用\(和\)
()+  | 多个重复群组的判断
[    | 标记一个中括号表达式的开始。要匹配[，请使用\\[
```




```
例：
[root@exercise1 opt]# vim a1   #创建测试文件
abbbc
askhk
abbbcskhskjh
abbbchjkhjkjhabbbc
abcbcbcdsjh
hkjhkjhabchhhhhd
[root@exercise1 opt]# grep -E 'ab+c' /opt/a1   #匹配以a开头，c结尾，有一个或多个重复的b字符

[root@exercise1 opt]# grep -E 'a(bc)+d' /opt/a1   #匹配以a开头，d结尾，有一个或多个重复的bc字符串

[root@home ~]# echo caabbeab |grep -E  [ab]+
caabbeab
```

grep小总结

- grep不支持正则表达式，需要加个参数-E，grep加强版egrep不用加

- -B 数字：除了列出查找的那行外，前面的N行也列出来
- -A 数字：除了列出查找的那行外，后面的N行也列出来

```
例1：
[root@exercise1 opt]# vim a1.txt    #创建测试文件
shsh,keagdahd
sdjkgvjfgfdsa,djgsjf
lhadkkdhhbjf.dkugfhds
dskgfsjdgfsh
dsjfhsjdfgiuweriwnd
ksdhfsuh
[root@exercise1 opt]# grep -B3 f /opt/a1.txt    #查找f并把前3行列出来
[root@exercise1 opt]# grep -A3 d /opt/a1.txt    #查找d并把后3行列出来
```

例2：统计/etc/ssh/sshd_config文件中除去空行和#号开头的行的行数

[root@exercise1 opt]# grep -v "^$\|^#" /etc/ssh/sshd_config
[root@exercise1 opt]# grep -E -v "^$|^#" /etc/ssh/sshd_config #扩展正则表达式
[root@exercise1 opt]# egrep -v "^$|^#" /etc/ssh/sshd_config #扩展正则表达式

例3：点字符
[root@exercise1 opt]# grep .ot /etc/passwd #查找 passwd 文件包括.ot 的字符
root❌0:0:root:/root:/bin/bash
operator❌11:0:operator:/root:/sbin/nologin
[root@exercise1 opt]#

```
例4：
[root@exrcise1 opt]# num=10
[root@exrcise1 opt]# [[ $num =~ ^[0-9]+$ ]] && echo "成立" || echo "不成立"
成立
[root@exrcise1 opt]# num=10q
[root@exrcise1 opt]# [[ $num =~ ^[0-9]+$ ]] && echo "成立" || echo "不成立"
不成立
```

sed流编辑器

原理：sed编辑器是一行一行的处理文件内容的。正在处理的内容存放在内存(缓冲区)内，处理完成后按照选项的规定进行输出或文件的修改。接着处理下一行，这样不断重复，直到文件末尾。

sed的执行过程:

```
1、一次读取一行数据

2、根据我们提供的规则来匹配相关的数据，比如查找root。

3、按照命令修改数据流中的数据，比如替换

4、将结果进行输出

5、重复上面四步
```

如何使用

语法格式：

```
#老林自己总结的中文意思
#{}可以不加

sed  [-选项]   ' [范围条件] { [动作] / 模式匹配空间 / [动作补充] } '   文件
```

---

sed选项

-n 在打印处理时，不输出模式空间中的内容，常与动作补充里的p连用
-e 执行多个 sed 指令
-f 运行脚本
-i 编辑文件内容 ***
-i后接命令或自定义名称，编辑的同时创造备份
-r 使用扩展的正则表达式

范围条件： 处理文件的一部分，范围条件不写，默认读取文档的全部行

```
例如   121{p}                          从第121行开始处理文件
例如   ${p}                            表示最后一行处理文件
例如   /^root/{p}                      从开头是 root 的行开始处理文件
例如   1,7 {p}                         从第1行到第7行处理文件
例如   3,/^$/{p}                       从第3行到空行处理文件
例如   /^root/,/^mail/{p}              从开头是root到开头是mail的行处理文件
例如   2~2                             从第2行开始，每隔两行输出一次
例如   2,+4                            从第2行开始，打印接下来的4行
例如   /[0-9]/                         匹配数字（基础正则）

以什么开头，以什么结尾
[root@Superwei-Mk1 ~]# cat passwd
xxxxxxxxxxxx xxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxx
[root@Superwei-Mk1 ~]# sed '4s/\<xxxxxxxxxxxx\>/llllllllll/' passwd
xxxxxxxxxxxx xxxxxxxxxxxx
```

```
xxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx 1111111111
```

动作：

i 在当前行上面插入文本
a 在当前行下面插入文本
c 把选定的行改为新的指定的文本
s 查找替换
r/R 读取文件/一行
y 替换 （对应单个字符替换)
& 引用被 查找区间内 被 匹配到的内容。
\1：子串匹配标记，前面搜索可以用元字符集(..）可以分多组\2，\3....

动作补充：

数字：表明新文本将替换第几处模式匹配的地方
g：表示新文本将会替换所有匹配的文本(全局匹配)
w 另存
p 打印处理后输出结果
d 删除 ***

例

```
例1：s只替换第一个匹配到的字符，将passwd中的root用户替换成ufo
[root@exercise1 opt]# sed 's/root/ufo/' /etc/passwd
ufo:x:0:0:root:/root:/bin/bash                    #发现只替换了第一个匹配的 root，后面的没有替换
bin:x:1:1:bin:/bin:/sbin/nologin
```

例2：可以用数字指定第几处替换新文本
[root@exercise1 opt]# head -1 /etc/passwd | sed 's/0/123/2'
root❌0:123:root:/root:/bin/bash
[root@exercise1 opt]#

例3：全面替换标记g
[root@exercise1 opt]# head -1 /etc/passwd | sed 's/0/123/g'
root❌123:123:root:/root:/bin/bash
[root@exercise1 opt]#

```
例4：将sed中默认的/定界符改成#号，也就是说可以自定义分隔符
[root@exercise1 opt]# head -1 /etc/passwd | sed 's#/bin/bash#/sbin/nologin#'
root:x:0:0:root:/root:/sbin/nologin

以/来做定界符：
[root@exercise1 opt]# head -1 /etc/passwd | sed 's/\/bin\/bash/\/sbin\/nologin/'
root:x:0:0:root:/root:/sbin/nologin

以其它字符作定界符
[root@home opt]# grep root /etc/passwd |sed 's:/sbin/nologin:/sbin/login:'
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/login
[root@home opt]# grep root /etc/passwd |sed 's@/sbin/nologin@/sbin/login@'
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/login
[root@home opt]# grep root /etc/passwd |sed 's;/sbin/nologin;/sbin/login;'
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/login
[root@home opt]# grep root /etc/passwd |sed 's /sbin/nologin /sbin/login '
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/login
```

例5：将 a1.txt中的包括aa字样的行保存到c.txt中

```
[root@exercise1 opt]# sed '/aa/w c.txt' a1.txt
shsh,keagdahd
sdjkgvjfgfdsa,djgsjf
lhadkkdhhbjf.dkugfhds
dskgfsjdgfsh
dsjfhsjdfgiuweriwnd
ksdhfsuh
aahshsh
aa
aabb
[root@exercise1 opt]# cat c.txt
aahshsh
aa
aabb
```

例6：将 a1.txt中第2到4行保存到c.txt中

```
[root@exercise1 opt]# sed -n '2,4w c.txt' a1.txt
[root@exercise1 opt]# cat c.txt
sdjkgvjfgfdsa,djgsjf
lhadkkdhhbjf.dkugfhds
dskgfsjdgfsh
```

例7：将a1.txt中第3到最后一行保存到c.txt中

```
[root@exercise1 opt]# sed -n '3,$w c.txt' a1.txt
[root@exercise1 opt]# cat c.txt
lhadkkdhhbjf.dkugfhds
dskgfsjdgfsh
dsjfhsjdfgiuweriwnd
ksdhfsuh
aahshsh
aa
```

aabb

例8：将a1.txt中d开头的行到最后一行保存到c.txt中
[root@exercise1 opt]# sed -n '/^d/,$w c.txt' a1.txt
[root@exercise1 opt]# cat c.txt
dskgfsjdgfsh
dsjfhsjdfgiuweriwnd
ksdhfsuh
aahshsh
aa
aabb

打印，直接输入文件中的内容
例9：多打印第3行内容
[root@exercise1 opt]# sed '3p' a1.txt
shsh,keagdahd
sdjkgvjfgfdsa,djgsjf
lhadkkdhhbjf.dkugfhds
lhadkkdhhbjf.dkugfhds
dskgfsjdgfsh
dsjfhsjdfgiuweriwnd
ksdhfsuh
aahshsh
aa
aabb

例10：打印第3行到第5行的内容
[root@exercise1 opt]# sed '3,5p' a1.txt
shsh,keagdahd
sdjkgvjfgfdsa,djgsjf
lhadkkdhhbjf.dkugfhds
lhadkkdhhbjf.dkugfhds
dskgfsjdgfsh
dskgfsjdgfsh

dsjfhsjdfgiuweriwnd
dsjfhsjdfgiuweriwnd
ksdhfsuh
aahshsh
aa
aabb

例11：打印第3行和接下来的4行的内容

[root@exercise1 opt]# sed '3,+4p' a1.txt

[root@home opt]# seq 10 |sed -n '6,-2p'　　#不能反向打印
sed：-e 表达式 #1，字符 3：意外的“,”

例12：打印匹配关键字和接下来的3行的内容

[root@exercise1 opt]# sed '/aa/,+3p' a1.txt

例13：打印从第2行开始每隔两行的内容

[root@home opt]# seq 10 |sed -n '2~2p'
2
4
6
8
10

例14：-i对原文件修改，保存（必会）使用场景：替换或修改服务器配置文件，请注意务必先备份原文件!!!

[root@exercise1 opt]# sed -i 's/aa/oo/' a1.txt
[root@exercise1 opt]# cat a1.txt
shsh,keagdahd
sdjkgvjfgfdsa,djgsjf

lhadkkdhhbjf.dkugfhds
dskgfsjdgfsh
dsjfhsjdfgiuweriwnd
ksdhfsuh
oohshsh
oo
oobb
[root@exercise1 opt]#

例15：只打印第3行和第6行的内容
[root@exercise1 opt]# sed -ne '3p;6p' a1.txt
lhadkkdhhbjf.dkugfhds
ksdhfsuh
[root@exercise1 opt]# cat /etc/passwd | sed -ne '3p;/bash$/p' #打印第三行和以bash结尾的行
root❌0:0:root:/root:/bin/bash
daemon❌2:2:daemon:/sbin:/sbin/nologin
[root@exercise1 opt]#

例16：只将修改过的行内容输出打印
[root@exercise1 opt]# sed -n 's/root/123/p' /etc/passwd
123❌0:0:root:/root:/bin/bash
operator❌11:0:operator:/123:/sbin/nologin

例17：取反将修改过的行内容输出打印

```
[root@home opt]# seq 10 |sed -n '2,7!p'          #! 表示取反
1
8
9
10
```

例18：替换标记位参数gp连用，将全局修改过的行内容输出打印

[root@exercise1 opt]# sed -n 's/root/123/gp' /etc/passwd

123❌0:0:123:/123:/bin/bash

operator❌11:0:operator:/123:/sbin/nologin

[root@exercise1 opt]#

例19：-e 执行多个sed指令　　#sed默认用-e

[root@exercise1 opt]# sed -e 's/root/abc/;s/0/1234/' /etc/passwd | head -1

abc❌1234:0:root:/root:/bin/bash

[root@exercise1 opt]# sed 's/root/abc/;s/0/1234/' /etc/passwd | head -1

abc❌1234:0:root:/root:/bin/bash

例20：执行多个sed指令，可以不加\直接回车换行输入

[root@exercise1 opt]# sed 's/root/abc/

> s/0/1324/

> s/x/789/' /etc/passwd | head -1

abc:789:1324:0:root:/root:/bin/bash

例21：-f 运行存放sed命令的文件

[root@exercise1 opt]# vim /opt/e.sed

s/root/abc/

s/0/1324/

s/x/789/

[root@exercise1 opt]# sed -f e.sed /etc/passwd | head -1

abc:789:1324:0:root:/root:/bin/bash

[root@exercise1 opt]#

例22：使用-i.bak对源文件修改的同时进行备份

[root@exercise1 opt]# sed -i.bak 's/oo/aa/' /opt/a1.txt

[root@exercise1 opt]# cat a1.txt.bak

shsh,keagdahd

sdjkgvjfgfdsa,djgsjf

lhadkkdhhbjf.dkugfhds

dskgfsjdgfsh

dsjfhsjdfgiuweriwnd
ksdhfsuh
aahshsh
aa
aabb
[root@exercise1 opt]# cat a1.txt
shsh,keagdahd
sdjkgvjfgfdsa,djgsjf
lhadkkdhhbjf.dkugfhds
dskgfsjdgfsh
dsjfhsjdfgiuweriwnd
ksdhfsuh
aahshsh
aa
aabb
[root@exercise1 opt]#

-i后接命令或自定义名称
[root@exercise1 opt]# sed -i#bak 's/oo/aa/' /opt/a1.txt
[root@exercise1 opt]# sed -i.bak"`date +%H:%M:%S`" 's/oo/aa/' /opt/a1.txt
[root@exercise1 opt]# sed -i-"`date`" 's/oo/aa/' /opt/a1.txt

例23：使用-r支持扩展正则表达式
[root@exercise1 opt]# vim a2.txt
aaabb
aacbb
2165476897684
1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]# sed -r -i '$s/[0-9]+/(today)/' a2.txt
[root@exercise1 opt]# cat /opt/a2.txt
aaabb
aacbb
2165476897684

1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#

例24：y替换，一对一字符进行替换　#默认是全局替换，而且替换字符长度要与原来一样

[root@exercise1 opt]# sed 'y/wi/wo/' /opt/a2.txt
aaabb
aacbb
2165476897684
1d54545adbl
wooejroojdkl
dfhdsj(today)dlfwjoor

```
按行查找替换
写法如下：
用数字表示行范围；$表示行尾
用文本模式配置来过滤

[root@exercise1 opt]# sed '3,$y/wo/az/' /opt/a2.txt
aaabb
aacbb
2165476897684
1d54545adbl
aizejrzijdkl
dfhdsj(tzday)dlfajzir
[root@exercise1 opt]#
```

例25：使用文本模式过滤器
格式：/xxxx/command
[root@exercise1 opt]# sed '/abc/s/1000/123456/' /etc/passwd | grep abc
abc❌123456:1000::/home/abc:/bin/bash

例26：单行替换，将第2行中b替换成123456

```
[root@exercise1 opt]# sed '2s/b/123456/' /opt/a2.txt
aaabb
aac123456b
2165476897684
1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#
```

例27：多行替换，如果涉及到多行处理，用逗号表示行间隔。将第2行到最后一行中第一个b替换成123465

```
[root@exercise1 opt]# sed '2,$s/b/123456/' /opt/a2.txt
aaabb
aac123456b
2165476897684
1d54545ad123456l
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#
```

例28：d删除第2行到第4行的内容

```
[root@exercise1 opt]# sed '2,4d' /opt/a2.txt
aaabb
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#
```

例29：删除一样也适用文本模式

```
[root@exercise1 opt]# sed '/aa/d' /opt/a2.txt   #将包括aa的行删除
2165476897684
1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#
```

#添加行
#命令i(insert插入)，在当前行前面插入一行　i\
#命令a(append附加)，在当前行后面添加一行　a **(重点)**
例30：
[root@exercise1 opt]# echo "hello world" | sed 'i\ufo'
ufo
hello world
[root@exercise1 opt]# echo "hello world" | sed 'a\ufo'
hello world
ufo
[root@exercise1 opt]#

例31：在文件最后追加内容
[root@exercise1 opt]# sed '$a\test' /opt/a2.txt
aaabb
aacbb
2165476897684
1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
test
[root@exercise1 opt]#

例32：在文件中第2行之后，开始追加内容
[root@exercise1 opt]# sed '2a\test' /opt/a2.txt
aaabb
aacbb
test
2165476897684
1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#

例33：在文件第3行之后，开始追加多行内容　#在添加每行内容后面加\

```
[root@exercise1 opt]# sed '3a jkjkjkjk \
> lolita \
> bilibili' /opt/a2.txt
aaabb
aacbb
2165476897684
jkjkjkjk
lolita
bilibili
1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
```

例34：在文件中第2行到第4行之后分别追加内容

```
[root@exercise1 opt]# sed '2,4a\today' /opt/a2.txt
aaabb
aacbb
today
2165476897684
today
1d54545adbl
today
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#
```

修改行命令c(change)　c\　　#默认匹配整行

例35：将第4行内容改成today

```
[root@exercise1 opt]# sed '4c\today' /opt/a2.txt
aaabb
aacbb
2165476897684
today
```

wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#

例36：将第2行到最后全部修改成today
[root@exercise1 opt]# sed '2,$c\today' /opt/a2.txt
aaabb
today
[root@exercise1 opt]#

例37：只将第2行和第6行修改　　#相当于-e
[root@exercise1 opt]# sed '2c\ab
> 6c\ioe' /opt/a2.txt
aaabb
ab
2165476897684
1d54545adbl
wioejroijdkl
ioe
[root@exercise1 opt]#

例38：将k行的内容修改成today
[root@exercise1 opt]# sed '/k/c\tody' /opt/a2.txt　#修改行也可以用文本模式
aaabb
aacbb
2165476897684
1d54545adbl
tody
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#

命令r允许将一个文件的数据插入数据流中
格式： sed '3r b' a #往a文件里插入b文件的数据 #默认是在行后插入
例39：
[root@exercise1 opt]# head -1 /etc/passwd >> /opt/a3.txt
[root@exercise1 opt]# sed '3r a3.txt' a2.txt
aaabb
aacbb
2165476897684
root❌0:0:root:/root:/bin/bash
root❌0:0:root:/root:/bin/bash
1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#

例40：同样支持文本模式
[root@exercise1 opt]# sed '/aa/r a3.txt' a2.txt
aaabb
root❌0:0:root:/root:/bin/bash
root❌0:0:root:/root:/bin/bash
aacbb
root❌0:0:root:/root:/bin/bash
root❌0:0:root:/root:/bin/bash
2165476897684
1d54545adbl
wioejroijdkl
dfhdsj(today)dlfwjoir
[root@exercise1 opt]#

例41：&引用被查找区间内被匹配到的内容
[root@exercise1 opt]# sed 's/root/&ob/' /opt/a3.txt |head -1
rootob❌0:0:root:/root:/bin/bash
[root@exercise1 opt]#

例42：匹配的值分组
[root@exercise1 opt]# echo moding | sed 's#(mod)ing#\1en#'
#注意\1en是数字1不是字母l，\数字指的是第几组
moden
[root@exercise1 opt]#

例43：可以定义分组，再合成
[root@exercise1 opt]# echo a and b |sed 's#(a) and (b)# \2 and \1 # '
b and a
[root@exercise1 opt]#

```
例44：sed结合正则表达式
[root@exercise1 opt]# cat /etc/ssh/sshd_config |sed 's/#.*$//g'|sed '/^$/d'     #使用sed匹配出配置文件的配置项
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
SyslogFacility AUTHPRIV
AuthorizedKeysFile  .ssh/authorized_keys
PasswordAuthentication yes
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
UsePAM yes
X11Forwarding yes
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS
Subsystem   sftp    /usr/libexec/openssh/sftp-server
```

例45：如何显示一行（了解）

```
[root@exrcise1 ~]# seq 10
1
```

```
2
3
4
5
6
7
8
9
10
[root@exrcise1 ~]# seq 10 |sed 'N;s#\n# #g'          #N表示读取一行内容时，会连带读取下一行
1 2
3 4
5 6
7 8
9 10
[root@exrcise1 ~]# seq 10 |sed 'N;N;N;N;N;N;N;N;s#\n# #g'
1 2 3 4 5 6 7 8 9 10

N就是把下一行加入到当前的hold space模式空间中，使之进行后续处理，最后sed会默认打印除hold space模式空间里的内容。也就是说，sed是可以处理多行数据的。

:a和ta是配套使用的，实现跳转功能。t是test测试的意思。
另外，还有:a和ba的配套使用方法，也可以实现调转功能。b是branch分支的意思。

[root@exrcise1 ~]# seq 10 |sed ':label ;N;s#\n# #g;tlabel'          #循环写法
1 2 3 4 5 6 7 8 9 10
```

例46:

```
[root@home opt]# num=100
[root@home opt]# echo $num|sed -n "s/$num/111/p"
#当sed修改变量时，需把单引号改成双引号
111
```

扩展

bash脚本语法检查和查看详细的执行过程

检查语法是否有错：

bash　-n test.bash#未运行脚本检查语法错误

bash　-v　test.bash#查看bash是否存在语法错误

bash　-x　test.bash#查看bash详细的执行过程

```
[root@base ~]#cat    a.sh
# Script to show debug of shell #
tot=`expr $1 + $2`
secho $tot #这里故意写错

[root@base  ~]# bash -v a.sh
# Script to show debug of shell #
tot=`expr $1 + $2` expr: 语法错误 #语法哪错了？ 运行时没有给参数
secho $tot #这里故意写错
a.sh:行 4: secho: 未找到命令
[root@base  ~]# sed -i 's/secho/echo/' a.sh #修改正确后
[root@base  ~]# bash -x a.sh 2 3 #查看详细执行过程。 注：这个脚本是真正执行一遍，不是预执 行
++ expr 2 + 3
+ tot=5
+ echo 5

#是真的运行一遍脚本，若有错误会提示错哪里
```

注意

w、p、d可以直接接动作补充或者范围条件 ==> 2,4w or sp

-s 一定要接模式匹配，否则报错

不用s会直接打印全部

直接参数p会打印两次，一次是源文件的，一次是修改后的

通畅p参数与n参数连用

sed默认具有-e参数，不写也可

sed、awk自带正则

.* ==> 正则代表所有
