讲师(老司机)

Linux高级使用篇

第1章 高级指令

1.1 Linux指令回顾

1.1.1 命令的基本格式

centos7命令提示符格式如下：

```
[root@localhost ~]#
```

- []：这是提示符的分隔符号，没有特殊含义
- root：显示的是当前的登录用户，老司机现在使用的是 root 用户登录
- @：分隔符号，没有特殊含义
- localhost：当前系统的简写主机名（完整主机名是 localhost.localdomain）
- ~：代表用户当前所在的目录，此例中用户当前所在的目录是家目录
- #：命令提示符。超级用户是#，普通用户是$

centos7命令格式如下：

```
[root@localhost ~]# 命令  [选项] [参数]
```

1.1.2 帮助命令

man命令

man是 Linux 最主要、最常见的帮助命令，其基本信息如下

```
[root@localhost ~]# man [选项] 命令
```

常用参数：

- -f：查看命令拥有哪个级别的帮助
- -k：查看和命令相关的所有帮助

1.1.3 正则表达式

正则表达式(regular expression)描述了一种字符串匹配的模式（pattern），可以用来检查一个串是否含有某种子串、将匹配的子串替换或者从某个串中取出符合某个条件的子串等。

例如：

- **runoo+b**，可以匹配 runoob、runooob、runooooooob 等，+ 号代表前面的字符必须至少出现一次（1次或多次）。
- **runoo*b**，可以匹配 runob、runoob、runooooooob 等，* 号代表前面的字符可以不出现，也可以出现一次或者多次（0次、或1次、或多次）。
- **colou?r** 可以匹配 color 或者 colour，? 问号代表前面的字符最多只可以出现一次（0次、或1次）。

构造正则表达式的方法和创建数学表达式的方法一样。也就是用多种元字符与运算符可以将小的表达式结合在一起来创建更大的表达式。正则表达式的组件可以是单个的字符、字符集合、字符范围、字符间的选择或者所有这些组件的任意组合。

正则表达式是由普通字符（例如字符 a 到 z）以及特殊字符（称为"元字符"）组成的文字模式。模式描述在搜索文本时要匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。

1.非打印字符

非打印字符也可以是正则表达式的组成部分。下表列出了表示非打印字符的转义序列：

字符	描述
\cx	匹配由x指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。注意 Unicode 正则表达式会匹配全角空格符。
\S	匹配任何非空白字符。等价于[^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。

2.特殊字符

所谓特殊字符，就是一些有特殊含义的字符，如上面说的 **runoo*b** 中的 *，简单的说就是表示任何字符串的意思。如果要查找字符串中的 * 符号，则需要对 * 进行转义，即在其前加一个 **\:** runo*ob，匹配 runo*ob。

许多元字符要求在试图匹配它们时特别对待。若要匹配这些特殊字符，必须首先使字符"转义"，即，将反斜杠字符\放在它们前面。下表列出了正则表达式中的特殊字符：

特别字符	描述
$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 \$。
( )	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 (和)。
*	匹配前面的子表达式零次或多次。要匹配 * 字符，请使用 *。
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用\+。
.	匹配除换行符 \n 之外的任何单字符。要匹配 . ，请使用\ . 。
[	标记一个中括号表达式的开始。要匹配 [，请使用 \[。
?	匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 '\(' 则匹配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，当该符号在方括号表达式中使用时，表示不接受该方括号表达式中的字符集合。要匹配 ^ 字符本身，请使用 \^。
{	标记限定符表达式的开始。要匹配 {，请使用 \{。
|	指明两项之间的一个选择。要匹配 |，请使用 \

3.限定符

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。有 ***** 或 **+** 或 **?** 或 **{n}** 或 **{n,}** 或 **{n,m}** 共6种。

正则表达式的限定符有：

字符	描述
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。* 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 、 "does" 中的 "does" 、 "doxy" 中的 "do" 。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

1.1.4正则表达式在线生成工具

本工具提供了常用正则表达式的在线生成功能，可实现诸如字符、网址、邮编、日期、中文等的正则表达式生成功能，并且提供各类常见语言如：javascript、php、Go语言、java、ruby、Python等的正则表达式测试语句供大家参考使用。

```
http://tools.jb51.net/regex/create_reg/
```

1.1.5常见正则表达式汇总

```
一、校验数字的表达式

1. 数字：^[0-9]*$

2. n位的数字：^d{n}$

3. 至少n位的数字：^d{n,}$

4. m-n位的数字：^d{m,n}$

5. 零和非零开头的数字：^(0|[1-9][0-9]*)$

6. 非零开头的最多带两位小数的数字：^([1-9][0-9]*)+(.[0-9]{1,2})?$

7. 带1-2位小数的正数或负数：^(-)?d+(.d{1,2})?$

8. 正数、负数、和小数：^(-|+)?d+(.d+)?$

9. 有两位小数的正实数：^[0-9]+(.[0-9]{2})?$

10. 有1~3位小数的正实数：^[0-9]+(.[0-9]{1,3})?$
```

11. 非零的正整数：^[1-9]d*$ 或 ^([1-9][0-9]*){1,3}$ 或 ^+?[1-9][0-9]*$

12. 非零的负整数：^-[1-9][]0-9"*$ 或 ^-[1-9]d*$

13. 非负整数：^d+$ 或 ^[1-9]d*|0$

14. 非正整数：^-[1-9]d*|0$ 或 ^((-d+)|(0+))$

15. 非负浮点数：^d+(.d+)?$ 或 ^[1-9]d*.d*|0.d*[1-9]d*|0?.0+|0$

16. 非正浮点数：^((-d+(.d+)?)|(0+(.0+)?))$ 或 ^(-([1-9]d*.d*|0.d*[1-9]d*))|0?.0+|0$

17. 正浮点数：^[1-9]d*.d*|0.d*[1-9]d*$ 或 ^(([0-9]+.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*.[0-9]+)|([0-9]*[1-9][0-9]*))$

18. 负浮点数：^-([1-9]d*.d*|0.d*[1-9]d*)$ 或 ^(-(([0-9]+.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*.[0-9]+)|([0-9]*[1-9][0-9]*)))$

19. 浮点数：^(-?d+)(.d+)?$ 或 ^-?([1-9]d*.d*|0.d*[1-9]d*|0?.0+|0)$

二、校验字符的表达式

1. 汉字：^[u4e00-u9fa5]{0,}$

2. 英文和数字：^[A-Za-z0-9]+$ 或 ^[A-Za-z0-9]{4,40}$

3. 长度为3-20的所有字符：^.{3,20}$

4. 由26个英文字母组成的字符串：^[A-Za-z]+$

5. 由26个大写英文字母组成的字符串：^[A-Z]+$

6. 由26个小写英文字母组成的字符串：^[a-z]+$

7. 由数字和26个英文字母组成的字符串：^[A-Za-z0-9]+$

8. 由数字、26个英文字母或者下划线组成的字符串：^w+$ 或 ^w{3,20}$

9. 中文、英文、数字包括下划线：^[u4E00-u9FA5A-Za-z0-9_]+$

10. 中文、英文、数字但不包括下划线等符号：^[u4E00-u9FA5A-Za-z0-9]+$ 或 ^[u4E00-u9FA5A-Za-z0-9]{2,20}$

11. 可以输入含有^%&',;=?$"等字符：[^%&',;=?$x22]+ 12 禁止输入含有~的字符：[^~x22]+

三、特殊需求表达式

1. Email地址：^w+([-+.]w+)*@w+([-.]w+)*.w+([-.]w+)*$

2. 域名：[a-zA-Z0-9][-a-zA-Z0-9]{0,62}(/.[a-zA-Z0-9][-a-zA-Z0-9]{0,62})+/.?

3. InternetURL：[a-zA-z]+://[^s]* 或 ^http://([w-]+.)+[w-]+(/[w-./?%&=]*)?$

4. 手机号码：^(13[0-9]|14[5|7]|15[0|1|2|3|5|6|7|8|9]|18[0|1|2|3|5|6|7|8|9])d{8}$

5. 电话号码("XXX-XXXXXXX"、"XXXX-XXXXXXXX"、"XXX-XXXXXXX"、"XXX-XXXXXXXX"、"XXXXXXX"和"XXXXXXXX)：^((d{3,4}-)|d{3.4}-)?d{7,8}$

6. 国内电话号码(0511-4405222、021-87888822)：d{3}-d{8}|d{4}-d{7}

7. 身份证号(15位、18位数字)：^d{15}|d{18}$

8. 短身份证号码(数字、字母x结尾)：^([0-9]){7,18}(x|X)?$ 或 ^d{8,18}|[0-9x]{8,18}|[0-9X]{8,18}?$

9. 帐号是否合法(字母开头，允许5-16字节，允许字母数字下划线)：^[a-zA-Z][a-zA-Z0-9_]{4,15}$

10. 密码(以字母开头，长度在6~18之间，只能包含字母、数字和下划线)：^[a-zA-Z]w{5,17}$

11. 强密码(必须包含大小写字母和数字的组合，不能使用特殊字符，长度在8-10之间)：^(?=.*d)(?=.*[a-z])(?=.*[A-Z]).{8,10}$

12. 日期格式：^d{4}-d{1,2}-d{1,2}

13. 一年的12个月(01～09和1～12)：^(0?[1-9]|1[0-2])$

14. 一个月的31天(01～09和1～31)：^((0?[1-9])|((1|2)[0-9])|30|31)$

15. 钱的输入格式：

16. 1.有四种钱的表示形式我们可以接受:"10000.00" 和 "10,000.00", 和没有 "分" 的 "10000" 和 "10,000"：^[1-9][0-9]*$

17. 2.这表示任意一个不以0开头的数字,但是,这也意味着一个字符"0"不通过,所以我们采用下面的形式：^(0|[1-9][0-9]*)$

18. 3.一个0或者一个不以0开头的数字.我们还可以允许开头有一个负号：^(0|-?[1-9][0-9]*)$

19. 4.这表示一个0或者一个可能为负的开头不为0的数字.让用户以0开头好了.把负号的也去掉,因为钱总不能是负的吧.下面我们要加的是说明可能的小数部分：^[0-9]+(.[0-9]+)?$

20. 5.必须说明的是,小数点后面至少应该有1位数,所以"10."是不通过的,但是 "10" 和 "10.2" 是通过的：^[0-9]+(.[0-9]{2})?$

21. 6.这样我们规定小数点后面必须有两位,如果你认为太苛刻了,可以这样：^[0-9]+(.[0-9]{1,2})?$

22. 7.这样就允许用户只写一位小数.下面我们该考虑数字中的逗号了,我们可以这样：^[0-9]{1,3}(,[0-9]{3})*(.[0-9]{1,2})?$

23 8.1到3个数字,后面跟着任意个 逗号+3个数字,逗号成为可选,而不是必须：^([0-9]+|[0-9]{1,3}(,[0-9]{3})*)(.[0-9]{1,2})?$

24. 备注：这就是最终结果了,别忘了"+"可以用"*"替代如果你觉得空字符串也可以接受的话(奇怪,为什么?)最后,别忘了在用函数时去掉去掉那个反斜杠,一般的错误都在这里

25. xml文件：^([a-zA-Z]+-?)+[a-zA-Z0-9]+.[x|X][m|M][l|L]$

26. 中文字符的正则表达式：[u4e00-u9fa5]

27. 双字节字符：[^x00-xff] (包括汉字在内，可以用来计算字符串的长度(一个双字节字符长度计2，ASCII字符计1))

28. 空白行的正则表达式：ns*r (可以用来删除空白行)

29. HTML标记的正则表达式：<(S*?)[^>]*>.*?</1>|<.*? /> (网上流传的版本太糟糕，上面这个也仅仅能部分，对于复杂的嵌套标记依旧无能为力)

30. 首尾空白字符的正则表达式：^s*|s*$或(^s*)|(s*$) (可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等等)，非常有用的表达式)

31. 腾讯QQ号：[1-9][0-9]{4,} (腾讯QQ号从10000开始)

32. 中国邮政编码：[1-9]d{5}(?!d) (中国邮政编码为6位数字)

33. IP地址：d+.d+.d+.d+ (提取IP地址时有用)

34. IP地址：((?:(?:25[0-5]|2[0-4]d|[01]?d?d).){3}(?:25[0-5]|2[0-4]d|[01]?d?d))

1.2 Linux三剑客

awk、grep、sed是linux操作文本的三大利器，合称文本三剑客，也是必须掌握的linux命令之一。三者的功能都是处理文本，但侧重点各不相同，其中属awk功能最强大，但也最复杂。grep更适合单纯的查找或匹配文本，sed更适合编辑匹配到的文本，awk更适合格式化文本，对文本进行较复杂格式处理。

1.2.1grep

Linux系统中grep命令是一种强大的文本搜索工具，它能使用**正则表达式**搜索文本，并把匹配的行打印出来（匹配到的标红）。grep全称是Global Regular Expression Print，表示全局正则表达式版本，它的使用权限是所有用户。

grep的工作方式是在一个或多个文件中搜索字符串模板。如果模板包括空格，则必须被引用，模板后的所有字符串被看作文件名。搜索的结果被送到标准输出，不影响原文件内容。

grep可用于shell脚本，因为grep通过返回一个状态值来说明搜索的状态，如果模板搜索成功，则返回0，如果搜索不成功，则返回1，如果搜索的文件不存在，则返回2。我们利用这些返回值就可进行一些自动化的文本处理工作。

1.find和grep命令区别

- find命令是根据**文件的属性**进行查找，如文件名，文件大小，所有者，所属组，是否为空，访问时间，修改时间等。
- grep是根据**文件的内容进行**查找，会对文件的每一行按照给定的模式(pattern)进行匹配查找。
- find 命令用于在系统中搜索符合条件的文件名，如果需要模糊查询，则使用通配符进行匹配。搜索时文件名是完全匹配的。
- grep 命令用于在文件中搜索符合条件的字符串，如果需要模糊查询，则使用正则表达式进行匹配。搜索时字符串是包含匹配的。

这两个都是 "搜索" "查找"的相关命令, 所以还是容易混淆的, 但是记住:grep是用来查找文字内容的, 而find是用来查找文件的. 这样就清晰了!

2.命令格式

用于过滤/搜索的特定字符。可使用正则表达式能多种命令配合使用，使用上十分灵活。

```
grep [option] pattern file
```

3.参数

常用参数已加粗：

- **-A<显示行数>：除了显示符合范本样式的那一列之外，并显示该行之后的内容。**
- **-B<显示行数>：除了显示符合样式的那一行之外，并显示该行之前的内容。**
- **-C<显示行数>：除了显示符合样式的那一行之外，并显示该行之前后的内容。**
- -c：统计匹配的行数
- -e ： 实现多个选项间的逻辑or 关系
- -E：扩展的正则表达式
- -f FILE：从FILE获取PATTERN匹配
- -F ： 相当于fgrep
- -i --ignore-case #忽略字符大小写的差别。
- **-n：显示匹配的行号**
- -o：仅显示匹配到的字符串
- -q： 静默模式，不输出任何信息
- -s：不显示错误信息。
- **-v：显示不被pattern 匹配到的行，相当于[^] 反向匹配**
- -w ： 匹配 整个单词

4.实战案例

mysqlrbac.yml

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: nfs-client-provisioner
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: nfs-client-provisioner-runner
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create", "update", "patch"]
```

案例一

基本使用方法

```
grep "rbac" mysqlrbac.yml
```

案例二

显示行号

```
grep -n "rbac" mysqlrbac.yml
```

案例三

查找不包含"rbac"的行

```
grep -v "rbac" mysqlrbac.yml
```

案例四

查找包含"rbac"的行，并显示后3行内容

```
grep -A3 "rbac" mysqlrbac.yml
```

案例五

查找包含"rbac"的行，并显示前3行内容

```
grep -B3 "rbac" mysqlrbac.yml
```

案例六

查找包含"rbac"的行，并显示前后3行内容

```
grep -C3 "rbac" mysqlrbac.yml
```

5.通配符与正则表达式的区别

5.1通配符

用于匹配文件名，完全匹配

通 配 符	作 用
?	匹配一个任意字符
*	匹配 0 个或任意多个任意字符，也就是可以匹配任何内容
[]	匹配中括号中任意一个字符。例如，[abc]代表一定匹配一个字符，或者是 a，或者是 b，或者是c
[-]	匹配中括号中任意一个字符，-代表一个范围。例如，[a-z]代表匹配一个小写字母
[^]	逻辑非，表示匹配不是中括号内的一个字符。例如，[^0-9]代表匹配一个不是数字的字符

```
touch abc
touch acc
touch ab
touch ac
ls
```

实战案例

```
find . -name "a*c"

find . -name "a?c"

find . -name "a[bc]c"

find . -name "a[a-b]c"

find . -name "a[^b]c"
```

5.2正则表达式

用于匹配字符串，包含匹配

正则符	作用
*	匹配前一个字符重复 0 次，或任意多次
?	匹配前一个字符重复 0 次，或 1 次（?是扩展正则，需要使用 egrep 命令）
[]	匹配中括号中任意一个字符。例如，[abc]代表一定匹配一个字符，或者是 a，或者是 b，或者是c
[-]	匹配中括号中任意一个字符，-代表一个范围。例如，[a-z]代表匹配一个小写字母
[^]	逻辑非，表示匹配不是中括号内的一个字符。例如，[^0-9]代表匹配一个不是数字的字符
^	匹配行首
$	匹配行尾

测试文档

```
vi 123.txt
```

文件内容

```
bbbbbbbbbbbbbbbbb
11111111111111111
123abc
123456abcdefghijk123
123
abc
abc 123
123aaaaaaaaaaaaaabc
123bc
123aabb

abc123abc
ccccccccccccccc
```

```
cat 123.txt
```

实战案例

```
无意义操作
grep "a*" 123.txt

包含字母a的行
grep "aa*" 123.txt

到底哪个a起作用？
grep "ab*" 123.txt
grep "ba*" 123.txt
grep "abc8*" 123.txt

匹配行首为字母a的行
grep "^aa*" 123.txt
grep "^a" 123.txt


匹配多个连续的a
grep "aaa*" 123.txt
grep "aaaa*" 123.txt

?错误的匹配
grep "aaaa?" 123.txt
正确的匹配方式
egrep "aaaa?" 123.txt


限位符匹配
egrep "3aaa?b" 123.txt
匹配数字结尾的行
grep "[0-9]$" 123.txt
匹配数字开头的行
grep "^[0-9]" 123.txt
```

1.2.2sed

sed 是一种流编辑器(Strem Editor)，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，过滤，取行。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**”（patternspace ），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。然后读入下行，执行下一个循环。如果没有使诸如‘d’ 的特殊命令，那会在两个循环之间清空模式空间，但不会清空**保留空间**。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**重定向存储输出或-i**。主要功能用来自动编辑一个或多个文件, 简化对文件的反复操作。

1.命令格式

```
sed [options] '[地址定界] command' file(s)
```

2.参数

- **-n**：不输出模式空间内容到屏幕，即不自动打印，只打印匹配到的行
- **-e**：多点编辑，对每行处理时，可以有多个Script
- **-f**：把Script写到文件当中，在执行sed时-f 指定文件路径，如果是多个Script，换行写
- **-r**：支持**扩展的正则**表达式
- **-i**：直接将处理的结果写入文件
- **-i.bak**：在将处理的结果写入文件之前备份一份

3.动作说明

- a： 新增， a 的后面可以接字串。添加多行时，除最后一行外，每行末尾需要用“\”代表数据未完结
- c： 替换， c 的后面可以接字串。替换多行时，除最后一行外，每行末尾需用“\”代表数据未完结。
- d： 删除，因为是删除啊，所以 d 后面通常不接任何命令。
- i： 插入， i 的后面可以接字串。插入多行时，除最后一行外，每行末尾需要用“\”代表数据未完结。
- p： 打印，亦即将某个选择的数据印出。通常 p 会与参数 sed -n 一起运行。
- 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 1,20 s/old/new/g 。

4.实战案例

注意事项

请先删除mysqlrbac.yml文件。重新导入一份新的mysqlrbac.yml文件。

案例一

只打印输出第二行内容

```
sed -n "2p" mysqlrbac.yml
```

案例二

删除第二行到第四行的数据。只是在屏幕输出中删除数据。但是文件本身并没有修改任何内容。

```
sed "2,4d" mysqlrbac.yml
```

删除文件数据。删除数据要慎重！！！！

```
sed -i "2,4d" mysqlrbac.yml
```

案例三

追加行数据。动作"a"是在指定行后追加一行

```
sed -i "2a lagouedu" mysqlrbac.yml

cat mysqlrbac.yml
```

插入行数据。动作"i"是在指定行前插入一行

```
sed -i "2i lagou" mysqlrbac.yml

cat mysqlrbac.yml
```

如果是想追加或插入多行数据，除最后一行外，每行的末尾都要加入“\”代表数据未完结。

```
sed -i "2i 1111111111111111111 \
> 22222222222222222222" mysqlrbac.yml

cat mysqlrbac.yml
```

案例四

替换行数据。

```
sed -i "2c 33333333333333 4444444444" mysqlrbac.yml

cat mysqlrbac.yml
```

案例五

替换指定字符串。语法规则："s/old/new/g "

```
sed -i "s/get/set/g" mysqlrbac.yml

cat mysqlrbac.yml
```

指定行范围替换

```
sed -i "1,17 s/set/get/g" mysqlrbac.yml

cat mysqlrbac.yml
```

1.2.3awk

awk是一种编程语言，用于在linux/unix下对文本和数据进行处理。数据可以来自标准输入(stdin)、一个或多个文件，或其它命令的输出。它**支持用户自定义函数**和动态正则表达式等先进功能，是linux/unix下的一个强大编程工具。它在命令行中使用，但更多是作为脚本来使用。**awk有很多内建的功能**，比如数组、函数等，这是它和C语言的相同之处，灵活性是awk最大的优势。awk其实不仅仅是工具软件，还是一种编程语言。不过，本章节只介绍它的命令行用法，对于大多数场合，应该足够用了。

1.命令格式

```
awk [options] 'program' var=value file…
awk [options] -f programfile var=value file…
awk [options] 'BEGIN{ action;… } pattern{ action;… } END{ action;… }' file ...
```

2.参数

- -F fs：fs指定输入分隔符，fs可以是字符串或正则表达式，如-F:
- -v var=value：赋值一个用户定义变量，将外部变量传递给awk
- -f scripfile：从脚本文件中读取awk命令

3.awk变量

变量：内置和自定义变量，每个变量前加 -v 命令选项

3.1内置变量

格式

- **FS**： **输入字段分隔符**，**默认为空白字符**
- **OFS**： **输出字段分隔符**，默认为空白字符
- RS： **输入记录分隔符**，指定输入时的换行符，原换行符仍有效
- ORS： **输出记录分隔符**，输出时用指定符号代替换行符
- **NF**： 字段数量，**共有**多少字段， **$NF引用最后一列，$(NF-1)引用倒数第2列**
- **NR**： **行号**，后可跟多个文件，第二个文件行号继续从第一个文件最后行号开始
- FNR： 各文件分别计数, 行号，后跟一个文件和NR一样，跟多个文件，第二个文件**行号从1开始**
- FILENAME： **当前文件名**
- ARGC： **命令行参数**的个数
- ARGV： 数组，保存的是命令行所给定的各参数

3.2自定义变量

自定义变量(区分字符大小写)

1. -v var=value
2. 在program 中直接定义

4. awk的条件

条件的类型	条 件	说 明
保留字	BEGIN	在 awk 程序一开始时，尚未读取任何数据之前执行。BEGIN 后的动作只在程序开始时执行一次
保留字	END	在 awk 程序处理完所有数据，即将结束时执行。END 后的动作只在程序结束时执行一次
关系运算符	>	大于
关系运算符	<	小于
关系运算符	>=	大于等于
关系运算符	<=	小于等于
关系运算符	==	等于。用于判断两个值是否相等，如果是给变量赋值，请使用“=”号
关系运算符	!=	不等于
关系运算符	A~B	判断字符串 A 中是否包含能匹配 B 表达式的子字符串
关系运算符	A!~B	判断字符串 A 中是否不包含能匹配 B 表达式的子字符串
正则表达式	/正则/	如果在“//”中可以写入字符，也可以支持正则表达式

5.实战案例

注意事项

新建users.txt文件。

```
vi users.txt
```

```
usrid username password age
10080 lagouedu 1234     18
10081 laosiji  5678     19
10082 zhangsan 1234     20
10083 lisi     1234     21
```

案例一

打印第一列和第三列数据，注意语法规则，awk需要用单引号

```
awk '{print $1 "\t" $3}' users.txt
```

打印输出前增加一行输出。

```
awk 'BEGIN{print "1111"} {print $1 "\t" $3}' users.txt
```

打印输出后增加一行输出。

```
awk 'END{print "22222222"} {print $1 "\t" $3}' users.txt
```

案例二

关系运算符操作。

```
输出年龄为20的用户名
awk '$4==20 {print $2}' users.txt

输出年龄不是20的用户名
awk '$4!=20 {print $2}' users.txt

输出年龄大于20的用户名
awk '$4>20 {print $2}' users.txt

输出年龄小于20的用户名
awk '$4<20 {print $2}' users.txt

输出年龄大于等于20的用户名
awk '$4>=20 {print $2}' users.txt

输出年龄小于等于20的用户名
awk '$4<=20 {print $2}' users.txt
```

案例三

正则表达式操作。如果要想让 awk 识别字符串，必须使用“//”包含

```
在某一列里查找包含lagouedu的数据
awk '$2~ /lagouedu/ {print}' users.txt
在某一列里查找不包含lagouedu的数据
awk '$2!~ /lagouedu/ {print}' users.txt

在整行里查找包含lagouedu的数据
awk '/lagouedu/ {print}' users.txt
```

1.3 grep awk sed对比

grep 主要用于搜索某些字符串，sed，awk 用于处理文本。

grep基本是以行为单位处理文本的； 而awk可以做更细分的处理，通过指定分隔符将一行（一条记录）划分为多个字段，以字段为单位处理文本。awk中支持C语法，可以有分支条件判断、循环语句等，相当于一个小型编程语言。awk功能比较多是一个编程语言了。 grep功能简单，就是一个简单的正则表达式的匹配。 awk的功能依赖于grep。

grep可以理解为主要作用是在一个文件中查找过滤需要的内容。awk不是过滤查找，而是文本处理工具，是把一个文件处理成你想要的格式。

awk是一种样式扫描与处理工具。但其功能却大大强于sed和grep。awk提供了极其强大的功能：它几乎可以完成grep和sed所能完成的全部工作，同时，它还可以可以进行**样式装入、流控制、数学运算符、进程控制语句甚至于内置的变量和函数。**它具备了一个完整的语言所应具有的几乎所有精美特性。实际上，awk的确拥有自己的语言：awk程序设计语言，awk的三位创建者已将它正式定义为：样式扫描和处理语言。 使用awk的第一个理由是基于文本的样式扫描和处理是我们经常做的工作，awk所做的工作有些象数据库，但与数据库不同的是，它处理的是文本文件，这些文件没有专门的存储格式，普通的人们就能编辑、阅读、理解和处理它们。而数据库文件往往具有特殊的存储格式，这使得它们必须用数据库处理程序来处理它们。既然这种类似于数据库的处理工作我们经常会遇到，我们就应当找到处理它们的简便易行的方法，UNIX有很多这方面的工具，例如sed 、grep、sort以及find等等，awk是其中十分优秀的一种。

sed是一个非交互性文本流编辑器。它编辑文件或标准输入导出的文本拷贝。sed编辑器按照一次处理一行的方式来处理文件（或者输入）并把输出送到屏幕上。你可以在vi和ex/ed编辑器里识别他的命令。sed把当前正在处理的行保存在一个临时缓存里，这个缓存叫做模式空间。一但sed完成了对模式空间里的行的处理（即对该行执行sed命令），就把模式空间的行送到屏幕上（除非该命令要删除该行活禁止打印）。处理完该行之后，从模式空间里删除它，然后把下一行读入模式空间，进行处理，并显示。当输入文件的最后一行处理完后，sed终止。通过把每一行存在一个临时缓存里并编辑该行，初始文件不会被修改或被破坏。

1.4 高级指令

1.4.1cut命令

列提取命令

1.语法

```
cut [选项] 文件名
```

2.参数

- -f 列号： 提取第几列
- -d 分隔符： 按照指定分隔符分割列
- -c 字符范围： 不依赖分隔符来区分列，而是通过字符范围（行首为 0）来进行字段提取。“n-”表示从第 n 个字符到行尾；“n-m”从第 n 个字符到第 m 个字符；“-m”表示从第 1 个字符到第 m 个字符。

3.实例

```
rm -rf users.txt
vi users.txt
```

users.txt文档内容不能使用空格分隔，必须要使用tab键进行处理。而且只能使用一次tab键，使用多次后，提取数据会出现错误。

```
usrid   username    password    age
10080   lagouedu    1234    18
10081   laosiji 5678    19
10082   zhangsan    1234    20
10083   lisi    1234    21
```

提取第二列数据

```
cut -f 2 users.txt
```

去掉表头行后，提取第二列数据

```
grep -v "username" users.txt | cut -f 2
```

去掉表头行后，提取第二列、第三列数据

```
grep -v "username" users.txt | cut -f 2,3
```

按照字符位置提取数据

```
提取第8个字符后边的数据
cut -c 8- users.txt

提取第8个字符前边的数据
cut -c -8 users.txt
```

1.4.2printf命令

格式化输出

1.语法

```
printf ‘输出类型输出格式’ 输出内容
```

2.输出类型

- %ns： 输出字符串。n 是数字指代输出几个字符
- %ni： 输出整数。n 是数字指代输出几个数字
- %m.nf： 输出浮点数。m 和 n 是数字，指代输出的整数位数和小数位数。如%8.2f
- 代表共输出 8 位数，其中 2 位是小数，6 位是整数。

3.输出格式

- \a: 输出警告声音
- \b: 输出退格键，也就是 Backspace 键
- \f: 清除屏幕
- \n: 换行
- \r: 回车，也就是 Enter 键
- \t: 水平输出退格键，也就是 Tab 键
- \v: 垂直输出退格键，也就是 Tab 键

4.实例

输出格式非常乱，不理想

```
printf '%s' $(cat users.txt)
```

格式化输出，非常麻烦

```
printf '%s\t %s\t %s\t %s\t \n' $(cat users.txt)
```

1.4.3alias命令

Linux alias命令用于设置指令的别名。用户可利用alias，自定指令的别名。若仅输入alias，则可列出目前所有的别名设置。alias的效力仅及于该次登入的操作。若要每次登入是即自动设好别名，可在.profile或. bashrc 中设定指令的别名。

1.语法

```
alias[别名]='[指令名称]'

取消别名:
unalias [别名]
```

注意：在定义别名的时候等号两边不能够有空格。

2.参数说明

- -p： 打印出现有的别名（唯一的参数）若不加任何参数，则列出目前所有的别名设置

3.实例

给命令设置别名

```
alias c='cut'
c -f 2 users.txt
cut -f 2 users.txt

unalias c
c -f 2 users.txt
```

1.4.4curl命令

在Linux中curl是一个利用URL规则在命令行下工作的文件传输工具，可以说是一款很强大的http命令行工具。它支持文件的上传和下载，是综合传输工具，但按传统，习惯称url为下载工具。

1.语法

```
curl [option] [url]
```

2.参数

- -A/--user-agent 设置用户代理发送给服务器
- -b/--cookie <name=string/file> cookie字符串或文件读取位置
- -c/--cookie-jar 操作结束后把cookie写入到这个文件中
- **-C/--continue-at 断点续转**
- -D/--dump-header 把header信息写入到该文件中
- -e/--referer 来源网址
- -f/--fail 连接失败时不显示http错误
- **-o/--output 把输出写到该文件中**
- **-O/--remote-name 把输出写到该文件中，保留远程文件的文件名**
- -r/--range 检索来自HTTP/1.1或FTP服务器字节范围
- -s/--silent 静音模式。不输出任何东西
- -T/--upload-file 上传文件
- -u/--user user[:password] 设置服务器的用户和密码
- -w/--write-out [format] 什么输出完成后
- -x/--proxy host[:port] 在给定的端口上使用HTTP代理

3.实例

```
curl https://www.linux.com

使用linux的重定向功能保存网页
curl https://www.linux.com >> linux.html

可以使用curl的内置option:-o(小写)保存网页
curl -o linux1.html https://www.linux.com

利用curl下载文件
curl -o dodo1.jpg https://www.linux.com/dodo1.JPG

断点续传
curl -C -O https://www.linux.com/dodo1.JPG

上传文件
curl -T dodo1.JPG -u 用户名:密码 ftp://www.linux.com/img/

显示抓取错误
curl -f https://www.linux.com/error
```

1.4.5scp命令

Linux scp 命令用于 Linux 之间复制文件和目录。scp 是 secure copy 的缩写, scp 是 linux 系统下基于 ssh 登陆进行安全的远程文件拷贝命令。

1.语法

```
scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
[-l limit] [-o ssh_option] [-P port] [-S program]
[[user@]host1:]file1 [...] [[user@]host2:]file2
```

简易写法:

```
scp [可选参数] file_source file_target
```

2.参数

- -1： 强制scp命令使用协议ssh1
- -2： 强制scp命令使用协议ssh2
- -4： 强制scp命令只使用IPv4寻址
- -6： 强制scp命令只使用IPv6寻址
- -B： 使用批处理模式（传输过程中不询问传输口令或短语）
- -C： 允许压缩。（将-C标志传递给ssh，从而打开压缩功能）
- -p：保留原文件的修改时间，访问时间和访问权限。
- -q： 不显示传输进度条。
- -r： 递归复制整个目录。
- -v：详细方式显示输出。scp和ssh(1)会显示出整个过程的调试信息。这些信息用于调试连接，验证和配置问题。
- -c cipher： 以cipher将数据传输进行加密，这个选项将直接传递给ssh。
- -F ssh_config： 指定一个替代的ssh配置文件，此参数直接传递给ssh。
- -i identity_file： 从指定文件中读取传输时使用的密钥文件，此参数直接传递给ssh。
- -l limit： 限定用户所能使用的带宽，以Kbit/s为单位。
- -o ssh_option： 如果习惯于使用ssh_config(5)中的参数传递方式，
- -P port：注意是大写的P, port是指定数据传输用到的端口号
- -S program： 指定加密传输时所使用的程序。此程序必须能够理解ssh(1)的选项。