一、扩展正则表达式

二、文本流编辑器-sed

1.概念：

2.格式：

3.选项：

4.编辑命令格式

5.操作

6.案例：

1.输出行　p

2.删除行　d

3.替换　　s

4.整行替换

5. r读取指定文件

6.a 追加内容

7.w 另存为----用的不多

8.H复制　G粘贴到指定的行

7.分组调用----后向引用

三、AWK

1.格式

2.awk工作原理

3.编辑指令的分隔符

4.命令区块的构成

一、扩展正则表达式

\? :匹配其前面的字符0或1次

\+:匹配其前面的字符至少1次 (非贪婪模式)

\(\) ： 分组将一个或多个字符捆绑在一起，当作一个整体进行处理，如：\(root\)\+

\ | ： 或者

```
[root@localhost ~]# grep -E "go+d" test1.txt
god
good
goood
gooood
[root@localhost ~]# grep -E "go?d" test1.txt
gd
god
[root@localhost ~]# grep -E "go*d" test1.txt
gd
god
good
goood
gooood
```

```
[root@localhost ~]# egrep "go+d" test1.txt
god
good
goood
gooood
[root@localhost ~]# grep -E "go+d" test1.txt
god
good
goood
gooood
[root@localhost ~]# egrep "g(oo)+d" test1.txt
good
gooood
[root@localhost ~]# ifconfig | grep -Eo "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}"
[root@localhost ~]# ifconfig | grep -Eo "([0-9]{1,3}\.){3}[0-9]{1,3}"
192.168.200.110
255.255.255.0
192.168.200.255
127.0.0.1
255.0.0.0
192.168.122.1
255.255.255.0
192.168.122.255
[root@localhost ~]# df -Th | awk '/\/$/{print $6}' | awk -F% '{print $1}'
[root@localhost ~]# df -Th | awk -F'[ %]+' '/\/$/{print $6}'
[root@localhost ~]# df -Th | awk -F'[ %]+' 'NR==2{print $6}'
9
```

二、文本流编辑器-sed

思路--表达意思　很重要

1.概念：

非交互 ，与vim相比

流----流水线　　流文本编辑器

注意匹配行，正则，行号，否则全文　　！！！

2.格式：

sed［选项］地址 ’编辑命令’ 输入文件

sed［选项］sed［选项］’地址 编辑命令’

3.选项：

-e：指定处理动作

-n　:取消sed默认输出；通常都加

-i ： 改错了也会加到文件里，一定要反复确认，没问题再加。直接编辑原文本，不会在屏幕上显示结果。 不加-i。不改变源文件

4.编辑命令格式

地址：行数，正则表达式，$,没有地址代表全文

5.操作

p ： 输出

d ： 删除，整行

s ： 替换

c ： 替换（整行）

r ： read 读取

a： append 追加

i ： insert插入

w : 另存为

H ： 复制

G ： 粘贴

> H,d 剪切

```
[root@localhost ~]# sed '10p' test1.txt          # 输出10行
gd
god
good
goood
gooood
gold
glad
gaad
abcDfg
food
food
601151272
HELLO
010-6666888
0666-5666888
IP 192.168.200.108
IP 173.16.16.1
pay $180
```

[root@localhost ~]# sed -n '10p' test1.txt
food

6.案例：

1.输出行　p

1.输出11,15行

[root@localhost ~]# head -15 /etc/passwd | tail -5
[root@localhost ~]# sed -n '11,15p' /etc/passwd
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin

2.输出所有奇数行

[root@localhost ~]# sed -n 'p;n' test1.txt
gd
good
gooood
glad
abcDfg
601151272
010-6666888
IP 192.168.200.108
pay $180

3.输出所有偶数行

[root@localhost ~]# sed -n 'n;p' test1.txt
[root@localhost ~]# sed -n '1,$n;p' test1.txt

4.输出大写字母的行

[root@localhost ~]# sed -n '/[A-Z]/p' test1.txt
abcDfg
HELLO
IP 192.168.200.108
IP 173.16.16.1

5.输出以g开头的行

[root@localhost ~]# sed -n '/^g/p' test1.txt

6.输出第一行

[root@localhost ~]# sed -n '1p' test1.txt

7. 输出最后一行

[root@localhost ~]# sed -n '$p' test1.txt

8. 输出指定行数

[root@localhost ~]# sed -n '3p;5p' test1.txt

2.删除行　d

1. 删除第五行

[root@localhost ~]# cat -n test1.txt | sed '5d' # 删除一般不加-n

2. 删除空行

[root@localhost ~]# sed -i '/^$/d' test1.txt
[root@localhost ~]# cat test1.txt

3. 删除第1行和第3行

[root@localhost ~]# sed '1d;3d' test1.txt

3.替换 s

[root@localhost ~]# sed 's/0/o/g' test1.txt
[root@localhost ~]# sed '13,$s/o/0/g' test1.txt
[root@localhost ~]# sed '1,5s/^/#/g' test1.txt
[root@localhost ~]# sed 's/^/ /g' test1.txt
[root@localhost ~]# sed 's/^//g' test1.txt
[root@localhost ~]# sed '/^IP/ s/^/#/g' test1.txt
[root@localhost ~]# sed 's/$/EOF/g' test1.txt
[root@localhost ~]# sed '20,25 s#/sbin/nologin#/bin/bash#;=' /etc/passwd

替换IP

[root@localhost ~]# ifconfig ens33 | awk '/inet /{print $2}'
[root@localhost ~]# ifconfig ens33 | awk 'NR==2{print $2}'
[root@localhost ~]# ifconfig ens33 | sed -n '2 s/.*et //gp' | sed 's/n.*//g' #贪婪匹配
[root@localhost ~]# ifconfig ens33 | sed -rn '2 s/.*et (.*) n.*/\1/gp' # 分组调用

4.整行替换

[root@localhost ~]# sed '2c1111111111111' test1.txt

把最后一行替换为两行:

[root@localhost ~]# sed '$cAAAAAAAAAAAAAAA\nBBBBBBBBBBBBBBBBBBB' test1.txt

\n 为换行

[root@localhost ~]# sed '5,$cEEEEEEEEEEEE\nRRRRRRRRRRRRRRR' test1.txt

#替换5到最后一行

5. r读取指定文件

将 /etc/hosts读到当前文件的结尾

[root@localhost ~]# sed '$r /etc/hosts' test1.txt

将 /etc/hosts读到当前文件的第5行后

root@localhost ~]# sed '5r /etc/hosts' test1.txt

6.a 追加内容

追加到指定内容到行后

在第二行后添加NNNNN
[root@localhost ~]# sed '2aNNNNNNNNN' test1.txt

在第3行后追加　　append
[root@localhost ~]# sed '3a1111111111\n222222222222222' test1.txt

在第3行处插入　insert
[root@localhost ~]# sed '3i1111111111\n222222222222222' test1.txt

在数字下加 ==========
[root@localhost ~]# sed '/[0-9]/a==========' test1.txt
7.w 另存为----用的不多
[root@localhost ~]# sed -n '15,16w ip.txt' test1.txt

8.H复制 G粘贴到指定的行
把1,5行复制到文档末
[root@localhost ~]# sed '1,5H;$G' test1.txt

7.分组调用----后向引用

1. 过滤IP

[root@localhost ~]# cat ip.txt
IP is 192.168.200.108
IP is 173.16.16.1
[root@localhost ~]# sed -n '1s/IP is//gp' ip.txt
[root@localhost ~]# sed -n '1s/IP is 192.168.200.108/192.168.200.108/gp' ip.txt
[root@localhost ~]# sed -nr '1s/IP is (192.168.200.108)/\1/gp' ip.txt
[root@localhost ~]# sed -nr '1s/IP is (.*)/\1/gp' ip.txt
[root@localhost ~]# sed -nr '1s/(.*) (.*) (.*)/\3/gp' ip.txt
[root@localhost ~]# sed -nr '1s/(.*)s (.*)/\2/gp' ip.txt
[root@localhost ~]# sed -nr '1s/.*s (.*)/\1/gp' ip.txt

2. 过滤ifconfig

[root@localhost ~]# ifconfig ens33 | sed -nr 's/.*et (.*) ne.*/\1/gp'
[root@localhost ~]# ifconfig ens33 | sed -nr '2s/.*et (.*) n.*/\1/gp'
192.168.200.110
3. 过滤df -Th

```
[root@localhost ~]# df -Th | sed -nr '/\/$/p'
[root@localhost ~]# df -Th | sed -nr '2s/.*([0-9]{1,3}%).*/\1/gp'
                          #查找不一定是第二行，正则该用还是要用
9%
[root@localhost ~]# df -Th | sed -nr '2s/.*([0-9]{1,3})%.*/\1/gp'
9
[root@localhost ~]# ifconfig ens33 | sed -nr '2 s/.*et(.*)n.*k(.*)br.*/\1\2/gp'
192.168.200.110   255.255.255.0
[root@localhost ~]# ifconfig ens33 | sed -nr '2 s/.*et (.*) n.*k (.*) br.*/\1 \2/gp'
# 相比上一种方法，有空格没那么紧凑，好看一点
192.168.200.110  255.255.255.0
```

4. 使用sed获取/etc/passwd文件第1列和第7列

有以下几种方式：

1. 替换

```
[root@localhost ~]# sed -nr '1,5s/:.*:/ \t/gp' /etc/passwd        #贪婪匹配
```

2. 找规律/etc/passwd 文件第二个占位符是密码

```
[root@localhost ~]# sed -nr '1,5s/(.*):x.*:(.*)/\1\t\2/gp' /etc/passwd
```

3. 分组找规律：:.* 中间重复五次

```
[root@localhost ~]# sed -nr '1,5s/(.*)(:.*)(:.*)(:.*)(:.*)(:.*):(.*)/\1\t \7/gp' /etc/passwd
[root@localhost ~]# sed -nr '1,5s/(.*)(:.*){5}:(.*)/\1\t \3/gp' /etc/passwd
                          # {n}重复n次前面的字符
```

4. 抽取第2列和第七列

```
[root@localhost ~]# sed -nr '1,5s/.*:(.*)(:.*){4}:(.*)/\1\t \3/gp' /etc/passwd
```

5. awk

```
[root@localhost ~]# head -5 /etc/passwd | awk 'BEGIN{FS=":"}{print $1"\t"$7}'
[root@localhost ~]# head -5 /etc/passwd | awk -F: '{print $1"\t"$7}'
#用-F多好，非要内置变量
```

5. 使用sed获取/etc/passwd文件前5行的最后一个字符

1. 打散每一个字符，打印最后一个字符

```
[root@localhost ~]# sed -nr '1,5 s/(.)/\1 /gp' /etc/passwd | awk '{print $NF}'

[root@localhost ~]# sed -nr '1,5 s/.*(.)/\1 /gp' /etc/passwd
```

2. head截取字符，通过这个来取最后一个字符，让最后一个放开头，截取第一个

```
[root@localhost ~]# echo "123456" | head -c 3
123[root@localhost ~]# echo "123456" | head -c 1
```

1[root@localhost ~]#

h[root@localhost ~]# head -5 /etc/passwd |rev | head -c 1
原因：前五行没有循环转动起来，传给head来执行
[root@localhost ~]# for i in $(head -5 /etc/passwd |rev)
> do
> echo $i | head -c 1
> echo   #换行
> done

三、AWK

1.格式

awk 选项 ’地址（编辑指令）输入指令’   文件

2.awk工作原理

- 格式统一
- linux中大部分是格式化文件
- 提取信息
- $位置变量，$0表示整行
- 默认分隔符是空格

3.编辑指令的分隔符

- 不一定是一条；多条用 “；” 隔开，
- 多个区域用{ } 隔开

原始：[root@localhost ~]# vim mem.sh
```
 1 #!/bin/bash
 2 mt=$(free | awk '/^Mem:/{print $2}')
 3 mu=$(free | awk '/^Mem:/{print $3}')
 4 mused=$(expr $mu \* 100 / $mt)
 5 echo "内存使用百分比：$mused%"
```
改进：[root@localhost ~]# vim mem.sh
```
 1 #!/bin/bash
 2 mt=$(free | awk '/^Mem:/{print $2}')
 3 mu=$(free | awk '/^Mem:/{print $3}')
 4 mused=$(expr $mu \* 100 / $mt)
 5 if [ $mused -gt 10 ]
 6 then
 7     #echo "内存使用百分比：$mused%"
 8     echo "内存不够啦"
 9 fi
```

```
[root@localhost ~]# free | awk 'NR==2 {num=int($3/$2*100); print num"%"}'
#加判断如下，如果是if(num>20)就不会输出
[root@localhost ~]# free | awk 'NR==2 {num=int($3/$2*100); if(num>10)print num"%"}'12%
```

awk一条命令搞定！

```
[root@localhost ~]# free | awk '/^Mem:/{num=int($3/$2*100);print "内存百分比：  "num"%"}'
内存百分比：  12%
#  int取整的意思
```

4.命令区块的构成

做Excel表格

打印第1, 7列

```
[root@localhost ~]# awk -F: 'NR<=5 {print $1,$7}' /etc/passwd
```

打印对齐

```
[root@localhost ~]# awk -F: 'NR<=5 {print $1"\t"$7}' /etc/passwd
```

转行

```
[root@localhost ~]# awk -F: 'NR<=5 {print $1"\n"$7}' /etc/passwd
```

加表头\t

```
[root@localhost ~]# awk -F: 'BEGIN{print "name\tshell"}NR<=5 {print $1"\t"$7}' /etc/passwd
```

换行加分割线

```
[root@localhost ~]# awk -F: 'BEGIN{print "name\tshell\n==========
==========="}NR<=5 {print $1"\t"$7}' /etc/passwd
```

结尾追加结束语，或者统计总行数

```
[root@localhost ~]# awk -F: 'BEGIN{print "name\tshell\n==========
==========="}NR<=5 {print $1"\t"$7} END{print "by:lireuifang"}' /etc/passwd
```

5.awk的执行流程

练习文本

```
[root@localhost ~]# cat grade.txt
M.Tansley 05/99 48311 Green   8  40 44
J.Lulu    06/99 48317 green   9  24 26
P.Bunny   02/99 48    Yellow  12 35 26
```

J.Troll 07/99 4842 Brown-3 12 26 26
L.Tansley 05/99 4712 Brown-2 12 30 28

7个域：

名字

升级日期

学生序号

腰带级别

年龄

目前比赛积分

比赛最高分

打印全文

```
[root@localhost ~]# awk '{print}' grade.txt
[root@localhost ~]# awk '{print $0}' grade.txt
```

打印1, 4行

```
[root@localhost ~]# awk '{print $1,$4}' grade.txt
[root@localhost ~]# awk '{print $1"\t"$4}' grade.txt
[root@localhost ~]# awk '{print $1"\t      "$4}' grade.txt
```

格式化输出

```
[root@localhost ~]# awk 'BEGIN {print "date age\n------------"} {print $2,$5}' grade.txt
[root@localhost ~]# awk 'BEGIN {print "date\t age\n------------"} {print $2"\t "$5}' grade.txt
```

终极改进：

```
[root@localhost ~]# awk 'BEGIN {print "date\t age\n---------------"} {print $2"\t "$5"\n"} END{print "---------------\nby:liruifang\n"}' grade.txt
```

再次改进：^_^

```
[root@localhost ~]# awk 'BEGIN {print "date\t age\n---------------"} {print $2"\t "$5"\n"} END{print NR} END{print "---------------\nby:liruifang\n"}' grade.txt
```

再来：

```
[root@localhost ~]# awk 'BEGIN {print "---------------\ndate\t age\n---------------"} {print $2"\t "$5"\n"} END{print "行数:" NR} END{print "---------------\nby:liruifang\n"}' grade.txt
```

案例

```
[root@localhost ~]# awk -F: 'BEGIN{print "name\tshell\n-------------------------"} {print $1"\t"$7} END{print "\ncount:" NR"\n"}' /etc/passwd
```

匹配以root开头的行

```
[root@localhost ~]# awk -F: 'BEGIN{print "name\tshell\n----------------------
"}/^root/{print $1"\t"$7} END{print "\ncount:" NR"\n"}' /etc/passwd
```

匹配地址范围

```
[root@localhost ~]# awk -F: 'BEGIN{print "  name\t  shell\n----------------------
"}NR<=20{print NR,$1"\t  "$7} END{print "\ncount:" NR"\n"}' /etc/passwd
```

6.awk条件操作符

赋值

= += -= /= *=

逻辑与 逻辑或 逻辑非

&& || !

匹配正则或不匹配,正则需要用 /正则/ 包围住

~ !~

关系 比较字符串时要把字符串用双引号引起来

< <= > >= != ==

字段引用

$ 字段引用需要加$,而变量引用直接用变量名取

运算符

+ - * / % ++ --

转义序列

\\ \自身
\$ 转义$
\t 制表符
\b 退格符
\r 回车符
\n 换行符
\c 取消换行

操作案例：

1. 放到文件里：

```
[root@localhost ~]# ifconfig > ip.txt
[root@localhost ~]# cat ip.txt
```

2. 在放到文件里的同时显示在屏幕上：

```
[root@localhost ~]# ifconfig | tee i.txt
```

3.判断整行是否匹配xxx,匹配则输出整行

相当于过滤

```
[root@localhost ~]# awk '/Brown/' grade.txt
J.Troll   07/99 4842  Brown-3 12 26 26
L.Tansley 05/99 4712  Brown-2 12 30 28
[root@localhost ~]# awk '$0~/Brown/' grade.txt
```

4.判断第三列是否等于48，匹配则输出整行

```
[root@localhost ~]# awk '$3=="48"{print $0}' grade.txt
P.Bunny   02/99 48    Yellow  12 35 26
```

5.判断整行是否不匹配Brown，不匹配则输出整行

```
[root@localhost ~]# awk '$0 ! /Brown/' grade.txt
ley 05/99 48311 Green   8  40 44
J.Lulu    06/99 48317 green   9  24 26
P.Bunny   02/99 48    Yellow  12 35 26
J.Troll   07/99 4842  Brown-3 12 26 26
L.Tansley 05/99 4712  Brown-2 12 30 28
```

6.判断第四列是否等于Brown-2，不等于则输出整行

```
[root@localhost ~]# awk '$4 !="Brown-2" {print $0}' grade.txt
ley 05/99 48311 Green   8  40 44
J.Lulu    06/99 48317 green   9  24 26
P.Bunny   02/99 48    Yellow  12 35 26
J.Troll   07/99 4842  Brown-3 12 26 26
```

7.过滤文件中的Green或green

```
[root@localhost ~]# awk '/[Gg]reen/' grade.txt
[root@localhost ~]# awk '$0 !~ /Brown/' grade.txt
ley 05/99 48311 Green   8  40 44
J.Lulu    06/99 48317 green   9  24 26
```

8.判断第一列是前面三个任意字母a开头

```
[root@localhost ~]# awk '$1~/^...a/' grade.txt
L.Tansley 05/99 4712  Brown-2 12 30 28
```

9.NR后面跟内容$1,$2，或$0，否则只显示行号

```
[root@localhost ~]# awk '{print NR ,$0}' grade.txt
```

10.判断整行匹配Yellow或者Brown

```
[root@localhost ~]# awk '$0~/(Yellow|Brown)/' grade.txt
P.Bunny   02/99 48    Yellow  12 35 26
J.Troll   07/99 4842  Brown-3 12 26 26
L.Tansley 05/99 4712  Brown-2 12 30 28
M.Tansley 05/99 4712  Brown-2 12 30 28
[root@localhost ~]# awk '/(Yellow|Brown)/{print $0}' grade.txt
```

P.Bunny 02/99 48 Yellow 12 35 26
J.Troll 07/99 4842 Brown-3 12 26 26
L.Tansley 05/99 4712 Brown-2 12 30 28
M.Tansley 05/99 4712 Brown-2 12 30 28

11. 过滤P开头的行

[root@localhost ~]# awk '/^P/' grade.txt
P.Bunny 02/99 48 Yellow 12 35 26

12. 输出密码为空的用户行

[root@localhost ~]# awk -F: '$2==""{print}' /etc/shadow
或者定义变量[root@localhost ~]# awk 'BEGIN{FS=":"};$2==""{print}' /etc/shadow
例如:
[root@localhost ~]# useradd zhangsan
[root@localhost ~]# grep "zhangsan" /etc/shadow
zhangsan:!!:18346:0:99999:7:::
[root@localhost ~]# passwd -d zhangsan
清除用户的密码 zhangsan。
passwd: 操作成功
[root@localhost ~]# grep "zhangsan" /etc/shadow
zhangsan::18346:0:99999:7:::
[root@localhost ~]# awk -F: '$2==""{print $0}' /etc/shadow
zhangsan::18346:0:99999:7:::

7.awk的内置变量

属性	说明
$0	当前记录（作为单个变量）
$1~$n	当前记录的第n个字段，字段间由FS分隔
FS	输入字段分隔符 默认是空格
NF	当前记录中的字段个数，就是有多少列
NR	已经读出的记录数，就是行号，从1开始
RS	输入的记录他隔符默 认为换行符
OFS	输出字段分隔符 默认也是空格
ORS	输出的记录分隔符，默认为换行符
ARGC	命令行参数个数

ARGV	命令行参数数组
FILENAME	当前输入文件的名字
IGNORECASE	如果为真，则进行忽略大小写的匹配
ARGIND	当前被处理文件的ARGV标志符
CONVFMT	数字转换格式 %.6g
ENVIRON	UNIX环境变量
ERRNO	UNIX系统错误消息
FIELDWIDTHS	输入字段宽度的空白分隔字符串
FNR	当前记录数
OFMT	数字的输出格式 %.6g
RSTART	被匹配函数匹配的字符串首
RLENGTH	被匹配函数匹配的字符串长度
SUBSEP	\034

1.统计行数和列数

```
[root@localhost ~]# cat grade.txt
ley       05/99 48311 Green   8  40 44
J.Lulu    06/99 48317 green   9  24 26
P.Bunny   02/99 48    Yellow  12 35 26
J.Troll   07/99 4842  Brown-3 12 26 26
L.Tansley 05/99 4712  Brown-2 12 30 28
M.Tansley 05/99 4712  Brown-2 12 30 28
[root@localhost ~]# awk '{print NF,NR}' grade.txt
#NF,NR后面不加内容，就会纯输出数字
7 1
7 2
7 3
7 4
7 5
7 6
[root@localhost ~]# awk '{print NR,NF}' grade.txt
1 7
2 7
3 7
4 7
5 7
```

```
6 7
[root@localhost ~]# awk '{print $NF}' grade.txt
#NF或NR前面加上$,就表示位置处所代表的具体内容
44
26
26
26
28
28
[root@localhost ~]# awk '{print NF}' grade.txt
#单纯输出最后一列的位置数字
7
7
7
7
7
7
[root@localhost ~]# awk '{print $(NF-1)}' grade.txt
#输出倒数第二列
40
24
35
26
30
30
[root@localhost ~]# awk '{print $NR}' grade.txt
#取对角线，$1,NR1;$2,NR2;$3,NR3;.....
ley
06/99
48
Brown-3
12
30
[root@localhost ~]# awk '{print $(NR+1)}' grade.txt
#取第一行，则列去1+1行；取第二行，则列去2+1行.....
05/99
48317
Yellow
12
30
28
[root@localhost ~]# wc -l grade.txt
6 grade.txt
[root@localhost ~]# awk 'END{print NR}' grade.txt
6
[root@localhost ~]# awk 'END{print NF}' grade.txt
```

输出行号、列数和文件名

原版：[root@localhost ~]# awk '{print NR,$0}END{print FILENAME}' grade.txt
改进版：[root@localhost ~]# awk '{print NR,$0}END{print "----------\n"FILENAME}' grade.txt
再改：[root@localhost ~]# awk '{print NF,NR,$0}END{print "----------\n"FILENAME}' grade.txt

当路径都特别深的时候，输出打印更简单些：

```
[root@localhost ~]# cd /usr/local/etc/
[root@localhost etc]# echo $PWD
/usr/local/etc
[root@localhost etc]# echo $PWD | awk -F/ '{print $NF}'
etc
[root@localhost etc]# cd /etc/sysconfig/network-scripts/
[root@localhost network-scripts]# pwd
/etc/sysconfig/network-scripts

[root@localhost network-scripts]# pwd | awk -F/ '{print $NF}'
network-scripts

[root@localhost network-scripts]# basename $(pwd)  #basename获取具体路径
network-scripts

[root@localhost network-scripts]# dirname $(pwd)     #dirname获取引导路径
/etc/sysconfig
```

案例：

```
[root@localhost ~]# awk '{print NR,$0}' /etc/passwd
#输出全文

[root@localhost ~]# awk 'NR==1,NR==3{print NR,$0}' /etc/passwd
或[root@localhost ~]# awk 'NR<=3{print NR,$0}' /etc/passwd
#输出前三行
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin

获取3~7行，逻辑与
[root@localhost ~]# awk 'NR>=3&&NR<=7{print NR,$0}' /etc/passwd

获取奇数行
[root@localhost ~]# awk 'NR%2==1{print NR,$0}' /etc/passwd

获取偶数行
[root@localhost ~]# awk 'NR%2==0{print NR,$0}' /etc/passwd
```

统计当前所有系统用户的用户名、UID、GID、登录的shell，制成windows系统中的Excel表格（工作中常用）

```
[root@localhost ~]# awk -F: '{print $1","$3","$4","$7}' /etc/passwd
#列与列之间用逗号分开
[root@localhost ~]# awk -F: 'BEGIN{print "name,uid,gid,shell"}{print
$1","$3","$4","$7}' /etc/passwd
#添加表头，BEGIN
[root@localhost ~]# awk -F: 'BEGIN{print "name,uid,gid,shell"}{print
$1","$3","$4","$7}' /etc/passwd > user.csv
#重定向到user.csv
[root@localhost ~]# sz user.csv
#这是用来做报告用的
```

8.awk中引用shell变量

单引号引用，值不能包含空格

```
[root@localhost ~]# name=san
[root@localhost ~]# awk 'BEGIN{print "'$name'"}'
san
[root@localhost ~]# name="zhang san"
[root@localhost ~]# awk 'BEGIN{print "'"$name"'"}'   #再套一层双引号
zhang san
#上面的方法比较麻烦
```

如果是全局变量

```
[root@localhost ~]# export name
[root@localhost ~]# awk 'BEGIN{print ENVIRON["name"]}'
zhang san
#这个方法也挺麻烦
```

！！！！！！！！！！！！！！记这个方法就行啦

```
[root@localhost ~]# awk -v nm="$name" 'BEGIN{print nm}'
zhang san
#先定义一个变量，再引用变量
```

案例：

文件批量改名

```
[root@localhost ~]# mkdir dir
[root@localhost ~]# touch dir/{1..10}.txt
[root@localhost ~]# ls dir/
10.txt  1.txt  2.txt  3.txt  4.txt  5.txt  6.txt  7.txt  8.txt  9.txt
```

1.将txt修改为jpg

```
[root@localhost ~]# ls dir/ | sed -r 's/(.*)\.txt/\1.jpg/g'
10.jpg
```

```
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
6.jpg
7.jpg
8.jpg
9.jpg
[root@localhost ~]# ls dir/ | sed -r 's/(.*)\.txt/mv & \1.jpg/g'
mv 10.txt 10.jpg
mv 1.txt 1.jpg
mv 2.txt 2.jpg
mv 3.txt 3.jpg
mv 4.txt 4.jpg
mv 5.txt 5.jpg
mv 6.txt 6.jpg
mv 7.txt 7.jpg
mv 8.txt 8.jpg
mv 9.txt 9.jpg
[root@localhost ~]# cd dir
[root@localhost dir]# ls dir/ | sed -r 's/(.*)\.txt/mv & \1.jpg/g' | bash
#管道交给bash执行就OK
[root@localhost dir]# ls
10.jpg  1.jpg  2.jpg  3.jpg  4.jpg  5.jpg  6.jpg  7.jpg  8.jpg  9.jpg


awk的做法：笔试题 ----还有更简单的方法------
[root@localhost dir]# ls | awk -F. '{print "mv " $0,$1".txt"}'
mv 10.jpg 10.txt
mv 1.jpg 1.txt
mv 2.jpg 2.txt
mv 3.jpg 3.txt
mv 4.jpg 4.txt
mv 5.jpg 5.txt
mv 6.jpg 6.txt
mv 7.jpg 7.txt
mv 8.jpg 8.txt
mv 9.jpg 9.txt
[root@localhost dir]# ls | awk -F. '{print "mv " $0,$1".txt"}' | bash
[root@localhost dir]# ls
10.txt  1.txt  2.txt  3.txt  4.txt  5.txt  6.txt  7.txt  8.txt  9.txt

来了，简单方法：rename from to files...
[root@localhost dir]# rename txt pdf *
[root@localhost dir]# ls
10.pdf  1.pdf  2.pdf  3.pdf  4.pdf  5.pdf  6.pdf  7.pdf  8.pdf  9.pdf
```

```
[root@localhost dir]# rename pdf jpg *
[root@localhost dir]# ls
10.jpg  1.jpg  2.jpg  3.jpg  4.jpg  5.jpg  6.jpg  7.jpg  8.jpg  9.jpg
```

作业：

SHELL-70道笔试题..案.txt
8.14KB

linux分析apache日...P.txt
1.55KB

双色球彩票shell脚本