课程目标

- 掌握sed的基本用法
- 掌握awk的基本用法

一、sed介绍

1. sed的工作流程

sed

- 首先sed把当前正在处理的行保存在一个临时缓存区中（也称为模式空间），然后处理临时缓冲区中的行，完成后把该行发送到屏幕上。
- sed把每一行都存在临时缓冲区中，对这个**副本**进行编辑，所以不会修改原文件。
- 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。

2. sed使用方法

sed常见的语法格式有两种，一种叫命令行模式，另一种叫脚本模式。

2.1 命令行格式

- 格式

```
sed [option]  'commands[地址定位]' filename
  说明：引用shell script中的变量应使用双引号，而非通常使用的单引号
option:
-e  进行多项编辑，即对输入行应用多条sed命令时使用
-n  取消默认的输出
-f  指定sed脚本的文件名
-r  使用扩展正则表达式       + ? {} () |
-i inplace，原地编辑（修改源文件）
```

- 常用命令和选项

```
命令1:
p   打印行
d   删除行

i\  在当前行之前插入文本。多行时除最后一行外，每行末尾需用"\"续行         vim  O
a\  在当前行后添加一行或多行。多行时除最后一行外，每行末尾需用“\”续行   vim o
c\  用此符号后的新文本替换当前行中的文本。多行时除最后一行外，每行末尾需用"\"续行   整行替换

 1059  sed -n 'p'  a.txt
 1060  sed -n '1p'  a.txt
 1061  sed -n '2p'  a.txt
 1062  sed -n '1,5p'  a.txt
 1063  cat -n a.txt
 1064  sed -n '5,10p' a.txt

 1065  sed -n '$p' a.txt
```

```
 1066  sed '5p' a.txt
 1067  sed -n '5p' a.txt
 1068  sed '1d' a.txt
 1069  sed -n '1d' a.txt
 1070  sed '1d' a.txt
 1071  sed '1,5d' a.txt
 1072  sed '$d' a.txt
 1081  sed '$a99999' a.txt
 1082  sed 'a99999' a.txt
 1083  cat -n a.txt
 1084  sed '5chello world' a.txt
 1085  sed 'chello world' a.txt
 1086  cat -n a.txt
 1087  sed '1,5chello world' a.txt

[root@MissHou shell05]# sed 'i\
aaaaa\
bbbbb\
88888' 1.txt

# sed '$a\
yyyyy\
8888' 1.txt

[root@MissHou shell05]# sed '/^user01/c888888' 1.txt
[root@MissHou shell05]# sed '18chello world' 1.txt
```

选项:

```
-e -r
[root@server shell06]# grep '^[^a-z]' 1.txt
[root@server shell06]# sed -n '/^[^a-z]/p' 1.txt
[root@server shell06]# grep -E '([0-9]{1,3}\.){3}[0-9]{1,3}' 1.txt
192.168.0.254
[root@server shell06]# sed -nr '/([0-9]{1,3}\.){3}[0-9]{1,3}/p' 1.txt
192.168.0.254

[root@jumper shell07]# grep -o -E '([0-9]{1,3}\.){3}[0-9]{1,3}' 2.txt
10.1.1.1
10.1.1.255
255.255.255.0
[root@jumper shell07]# sed -n '/([0-9]{1,3}\.){3}[0-9]{1,3}/p' 2.txt
[root@jumper shell07]# sed -nr '/([0-9]{1,3}\.){3}[0-9]{1,3}/p' 2.txt
10.1.1.1
10.1.1.255
255.255.255.0

[root@jumper shell07]# sed -n 's@/sbin/nologin@itcastheima@p' 2.txt
```

注意：搜索替换中的分隔符可以自己指定，@ #

命令2:

r　从文件中读取输入行

w　将所选的行写入文件

```
 1062  sed '5r /etc/hosts' 1.txt
 1063  sed 'r /etc/hosts' 1.txt
 1064  sed '$r /etc/hosts' 1.txt
 1065  sed '5w ./a.txt' 1.txt
 1067  cat a.txt
 1068  sed '1,5w ./a.txt' 1.txt

[root@jumper shell07]# sed  -r '/([0-9]{1,3}\.){3}[0-9]{1,3}/w b.txt' 2.txt


!   对所选行以外的所有行应用命令，放到行数之后
# sed -n '1!p' 1.txt
 1072  sed -n '4p' 1.txt
 1073  sed -n '4!p' 1.txt
 1074  cat -n 1.txt
 1075  sed -n '1,17p' 1.txt
 1076  sed -n '1,17!p' 1.txt

s   用一个字符串替换另一个
g   在行内进行全局替换


 1079  sed -n 's/root/ROOT/p' 1.txt
 1080  sed -n 's/root/ROOT/gp' 1.txt
 1081  vim 1.txt
 1082  cat -n 1.txt
 1083  sed -n 's/^#//gp' 1.txt

[root@server shell06]# sed -n 's@/sbin/nologin@itcast@gp' a.txt
[root@server shell06]# sed -n 's/\/sbin\/nologin/itcast/gp' a.txt
[root@server shell06]# sed -n '10s#/sbin/nologin#itcast#p' a.txt
uucp:x:10:14:uucp:/var/spool/uucp:itcast

[root@server shell06]# sed -n '1,5s/^/#/p' a.txt        注释掉文件的1-5行内容
#root:x:0:0:root:/root:/bin/bash
#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


&   保存查找串以便在替换串中引用    \(xxxxx\)
=   打印行号

[root@server shell06]# sed -n '/root/p' a.txt
root:x:0:0:root:/root:/bin/bash
[root@server shell06]# sed -n 's/root/#&/p' a.txt
#root:x:0:0:root:/root:/bin/bash
[root@server shell06]#
[root@server shell06]# sed -n 's/^root/#&/p' a.txt
#root:x:0:0:root:/root:/bin/bash
[root@server shell06]# sed -n 's/\(root\)/#\1/p' a.txt

#root:x:0:0:root:/root:/bin/bash
```

```
[root@server shell06]# sed -n 's/\(^root\)/#\1/p' a.txt
#root:x:0:0:root:/root:/bin/bash
[root@server shell06]#
[root@server shell06]# sed -n 's/^root/&#/p' a.txt
root#:x:0:0:root:/root:/bin/bash
[root@server shell06]# sed -n 's/\(^root\)/\1#/p' a.txt
root#:x:0:0:root:/root:/bin/bash


[root@MissHou shell05]# sed -n '1,5s/^/#&/p' 1.txt
#root:x:0:0:root:/root:/bin/bash
#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
[root@MissHou shell05]# sed -n '1,5s/\(^\)/#\1/p' 1.txt
#root:x:0:0:root:/root:/bin/bash
#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


[root@MissHou shell05]# sed -n '/root/p;/root/=' 1.txt
root:x:0:0:root:/root:/bin/bash
1
[root@MissHou shell05]# sed -ne '/root/p' 1.txt -ne '/root/='
root:x:0:0:root:/root:/bin/bash
1
[root@MissHou shell05]# sed -ne '/root/=' -ne '/root/p' 1.txt
1
root:x:0:0:root:/root:/bin/bash

在1.txt文件中的第5行的前面插入“hello world”;在1.txt文件的第8行下面插入“哈哈哈哈”

[root@MissHou shell05]# sed -e '5ihello world' -e '8a哈哈哈哈哈' 1.txt  -e '5=;8='


 1104  grep -Ev '^#|^$' /etc/vsftpd/vsftpd.conf
 1105  sed -e '/^#/d' -e '/^$/d' /etc/vsftpd/vsftpd.conf
 1106  sed '/^#/d;/^$/d' /etc/vsftpd/vsftpd.conf
 1107  sed -r '/^#|^$/d' /etc/vsftpd/vsftpd.conf


选项：-i   慎用！！！
[root@server shell06]# sed -i '1,5s/^/#&/' a.txt
注意：
-ni  不要一起使用
p命令 不要再使用-i时使用
```

- 定址

定址用于决定对哪些行进行编辑。地址的形式可以是数字、正则表达式、或二者的结合。如果没有指定地址，sed将处理输入文件的所有行。

```
x               指定x行号                          sed -n '5p' 1.txt
x,y             指定x到y行号                     sed -n '1,5p' 1.txt
/key/           查询包含关键字的行          sed -n '/root/p' 1.txt
/key1/,/key2/    匹配包含两个关键字之间的行   sed -n '/^adm/,/^mysql/p' 1.txt
/key/,x         从匹配关键字的行开始到文件第x行之间的行（包含关键字所在行）sed -n '/^lp/,7p'
x,/key/         从第x行开始到与关键字的匹配行之间的行
x,y!            不包含x到y行

[root@MissHou shell05]# sed -n '/bash$/!p' 1.txt

注意：sed使用的正则表达式是括在斜杠线"/"之间的模式。

//以下命令是找出以lp开头或者以mail开头的行


说明：
-e 多次编辑
-r 扩展正则
```

- 其他命令讲解

```
y命令
该命令与UNIX/Linux中的tr命令类似，字符按照一对一的方式从左到右进行转换。
正则表达式元字符对y命令不起作用。与s命令的分隔符一样，斜线可以被替换成其它的字符。
# sed '39,41y/stu/STU/' /etc/passwd
# sed '39,41y/stu:x/STU@%/' /etc/passwd


-e 选项 多项编辑
-e是编辑命令，用于sed执行多个编辑任务的情况下。在下一行开始编辑前，所有的编辑动作将应用到模式缓冲区中的行上。

-i 选项  直接修改原文件
# sed -i 's/root/ROOT/;s/stu/STU/' 11.txt
[root@MissHou shell05]# sed -i '17{s/YUNWEI/yunwei/;s#/bin/bash#/sbin/nologin#}' 1.txt

& 符号 保留查找字符串
# sed -n 's/^root/#&/p' passwd   注释掉以root开头的行
# sed -n -r 's/^root|^stu/#&/p' /etc/passwd 注释掉以root开头或者以stu开头的行
# sed -n '1,5s/^[a-z].*/#&/p' passwd  注释掉1~5行中以任意小写字母开头的行
# sed -n '1,5s/^/#/p' /etc/passwd  注释1~5行
或者
sed -n '1,5s/^/#/p' passwd 以空开头的加上#
sed -n '1,5s/^#//p' passwd 以#开头的替换成空
vim —>ctrl+v—>I(行头插入)—>ESC

 1140  sed -n '/^root/p' 1.txt
 1141  sed -n 's/^root/#&/p' 1.txt
 1142  sed -n 's/\(^root\)/#\1/p' 1.txt
```

```
 1143  sed -nr '/^root|^stu/p' 1.txt
 1144  sed -nr 's/^root|^stu/#&/p' 1.txt


= 打印行号
# sed -n '/bash$/=' passwd    打印以bash结尾的行的行号
# sed -ne '/root/=' -ne '/root/p' passwd
# sed -n '/nologin$/=;/nologin$/p' 1.txt
# sed -ne '/nologin$/=' -ne '/nologin$/p' 1.txt


q   退出
# sed '5q' 1.txt
# sed '/mail/q' 1.txt
# sed -r '/^yunwei|^mail/q' 1.txt
[root@MissHou shell05]# sed -n '/bash$/p;10q' 1.txt
ROOT:x:0:0:root:/root:/bin/bash

 1173  sed '/^#/d;/^$/d' /etc/vsftpd/vsftpd.conf
 1174  sed -r '/^#|^$/d' /etc/vsftpd/vsftpd.conf
 1175  sed -e '/^#/d' -e '/^$/d' /etc/vsftpd/vsftpd.conf



引申扩展:
h               把模式空间里的内容复制到暂存缓冲区    <覆盖>
H               把模式空间里的内容追加到暂存缓冲区
g               取出暂存缓冲区的内容，将其复制到模式空间 覆盖
G               取出暂存缓冲区的内容，将其复制到模式空间，追加在原有内容后面
x               交换暂存缓冲区与模式空间的内容


暂存和取用命令: h H g G

# sed '1H;$G' /etc/hosts
# sed '1h;$G' /etc/hosts
# sed '1h;$g' /etc/hosts

# sed '1{h;d};$G' /etc/hosts
# sed '1h;1d;$G'
# sed '1h;2,$g' /etc/hosts
# sed '1h;2,3H;$G' /etc/hosts

# sed '1!G;h;$!d' /etc/hosts        将文件内容从最后一行到第一行输出

暂存空间和模式空间互换: x
[root@server shell06]# sed '3h;4x;5G' 11.txt
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin
```

```
[root@server shell06]# cat -n 11.txt
     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
     4  adm:x:3:4:adm:/var/adm:/sbin/nologin
     5  lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
```

2.2 脚本格式

- 用法

```
# sed -f scripts.sed  file      //使用脚本处理文件
建议使用   ./sed.sh   file
脚本的第一行写上
#!/bin/sed -f
xxxx
3i777
5i888
a
p
```

- 注意事项

```
1）  脚本文件是一个sed的命令行清单。'commands'
2）  在每行的末尾不能有任何空格、制表符（tab）或其它文本。
3）  如果在一行中有多个命令，应该用分号分隔。
4）  不需要且不可用引号保护命令
5）  #号开头的行为注释
```

- 示例

```
# cat passwd
stu3:x:509:512::/home/user3:/bin/bash
stu4:x:510:513::/home/user4:/bin/bash
stu5:x:511:514::/home/user5:/bin/bash

# cat sed.sh
#!/bin/sed -f
2a\
******************
2,$s/stu/user/
$a\
we inster new line
s/^[a-z].*/#&/

[root@server shell06]# cat 1.sed
#!/bin/sed -f
3a**********************
```

```
$chelloworld
1,3s/^/#&/

[root@server shell06]# sed -f 1.sed -i 11.txt
[root@server shell06]# cat 11.txt
#root:x:0:0:root:/root:/bin/bash
#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
helloworld
```

3. sed和正则的综合运用

```
1、正则表达式必须以”/“前后规范间隔
例如: sed '/root/d' file
例如: sed '/^root/d' file

2、如果匹配的是扩展正则表达式，需要使用-r选来扩展sed
grep -E
sed -r
+ ? () {} | \d

注意:
在正则表达式中如果出现特殊字符(^$.*/[]),需要以前导 "\" 号做转义
eg: sed '/\$foo/p' file

3、逗号分隔符
例如: sed '5,7d' file              删除5到7行
例如: sed '/root/,/ftp/d' file
删除第一个匹配字符串"root"到第一个匹配字符串"ftp"的所有行本行不找 循环执行

4、组合方式
例如: sed '1,/foo/d' file          删除第一行到第一个匹配字符串"foo"的所有行
例如: sed '/foo/,+4d' file         删除从匹配字符串”foo“开始到其后四行为止的行
例如: sed '/foo/,~3d' file         删除从匹配字符串”foo“开始删除到3的倍数行（文件中）
例如: sed '1~5d' file              从第一行开始删每五行删除一行
例如: sed -nr '/foo|bar/p' file    显示配置字符串"foo"或"bar"的行
例如: sed -n '/foo/,/bar/p' file   显示匹配从foo到bar的行
例如: sed '1~2d'  file             删除奇数行
例如: sed '0-2d'   file            删除偶数行 sed '1~2!d'  file

5、特殊情况
例如: sed '$d' file                删除最后一行
例如: sed '1d' file                删除第一行

6、其他:
sed 's/.//' a.txt                  删除每一行中的第一个字符
sed 's/.//2' a.txt                 删除每一行中的第二个字符
sed 's/.//N' a.txt                 从文件中第N行开始，删除每行中第N个字符（N>2）

sed 's/.$//' a.txt                 删除每一行中的最后一个字符
```

```
[root@MissHou shell05]# cat 2.txt
1 a
2 b
3 c
4 d
5 e
6 f
7 u
8 k
9 o
[root@MissHou shell05]# sed '/c/,~2d' 2.txt
1 a
2 b
5 e
6 f
7 u
8 k
9 o
```

4. 课堂练习

1. 将任意数字替换成空或者制表符
2. 去掉文件1-5行中的数字、冒号、斜杠
3. 匹配root关键字替换成hello itcast，并保存到test.txt文件中
4. 删除vsftpd.conf、smb.conf、main.cf配置文件里所有注释的行及空行（不要直接修改原文件）
5. 使用sed命令截取自己的ip地址
6. 使用sed命令一次性截取ip地址、广播地址、子网掩码
7. 注释掉文件的2-3行和匹配到以root开头或者以ftp开头的行

```
1、将文件中任意数字替换成空或者制表符
# sed -n 's/[0-9]/\t/gp' a.txt
# sed -n 's/[0-9]//gp' a.txt
2、去掉文件1-5行中的数字、冒号、斜杠
# sed -n '1,5s/[:0-9/]//gp' a.txt
# sed -n '1,5{s/://g;s/[0-9]//g;s/[/]//gp}' a.txt
# sed -n '1,5s#[0-9/: ]##gp' 1.txt
3、匹配root关键字的行替换成hello itcast，并保存到test.txt文件中
# sed '/root/chello itcast' a.txt |tee bb.txt
# sed -n 's/root/hello itcast/gp w file2' 22.txt
hello itcast:x:0:0:hello itcast:/hello itcast:/bin/bash
# cat file2
hello itcast:x:0:0:hello itcast:/hello itcast:/bin/bash

4、删除vsftpd.conf、smb.conf、main.cf配置文件里所有注释的行及空行（不要直接修改原文件）
# sed '/^#/d;/^$/d' vsftpd.conf
# sed -e '/^#/d' -e '/^$/d' vsftpd.conf
# sed -r '/^#|^$/d' vsftpd.conf
```

```
# sed -r '/^#|^;|^\t$|^$/d' smb.conf
# sed -r '/^#|^\t$|^$|^\ /d' main.cf

5、使用sed命令截取自己的ip地址
# ifconfig eth0|sed -n '2p'|sed -n 's/.*addr://pg'|sed -n 's/Bcast.*//gp'
10.1.1.1
# ifconfig eth0|sed -n '2p'|sed 's/.*addr://g'|sed 's/ Bcast:.*//g'
6、使用sed命令一次性截取ip地址、广播地址、子网掩码
# ifconfig eth0|sed -n '2p'|sed -n 's#.*addr:\(.*\) Bcast:\(.*\) Mask:\(.*\)#\1\n\2\n\3#p'
10.1.1.1
10.1.1.255
255.255.255.0

7、注释掉文件的2-3行和匹配到以root开头或者以ftp开头的行
# sed -nr '2,3s/^/#&/p;s/^ROOT|^ftp/#&/p' 1.txt
#ROOT:x:0:0:root:/root:/bin/bash
#bin:x:1:1:bin:/bin:/sbin/nologin
#3daemon:x:2:2:daemon:/sbin:/sbin/nologin

# sed -ne '1,2s/^/#&/gp' a.txt -nre 's/^lp|^mail/#&/gp'
# sed -nr '1,2s/^/#&/gp;s/^lp|^mail/#&/gp' a.txt
```

补充:

```
&和\(\)      保留匹配的字符串
[root@MissHou shell05]# sed -n 's/^root/#&/p'  passwd
#root:x:0:0:root:/root:/bin/bash
[root@MissHou shell05]# sed -n 's/^root/&#/p'  passwd
root#:x:0:0:root:/root:/bin/bash
[root@MissHou shell05]# sed -n 's/\(root\)/#\1/p'  passwd
#root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/#root:/sbin/nologin
[root@MissHou shell05]# sed -n 's/\(root\)/\1#/p'  passwd
root#:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root#:/sbin/nologin
[root@MissHou shell05]# sed -n 's/^\(root\)/\1#/p'  passwd
root#:x:0:0:root:/root:/bin/bash


[root@MissHou shell05]# sed -n '/^mail/,/^ftp/s@^@#&@p' /etc/passwd
#mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
#uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
#operator:x:11:0:operator:/root:/sbin/nologin
#games:x:12:100:games:/usr/games:/sbin/nologin
#gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
#ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
[root@MissHou shell05]# sed -n '9,14s@^@#&@gp' /etc/passwd
#mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
#uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
#operator:x:11:0:operator:/root:/sbin/nologin
#games:x:12:100:games:/usr/games:/sbin/nologin

#gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
```

```
#ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

课后作业1

1、写一个初始化系统的脚本 1）自动修改主机名（如：ip是192.168.0.88，则主机名改为server88.itcast.cc)

a. 更改文件非交互式 sed

/etc/sysconfig/network

b.将本主机的IP截取出来赋值给一个变量ip;再然后将ip变量里以.分割的最后一位赋值给另一个变量ip1

2）自动配置可用的yum源

3）自动关闭防火墙和selinux

2、写一个搭建ftp服务的脚本，要求如下： 1）不支持本地用户登录 2） 匿名用户可以上传 新建 删除 3） 匿名用户限速500KBps

```

```

二、awk介绍

- awk是一种编程语言，主要用于在linux/unix下对文本和数据进行处理，是linux/unix下的一个工具。数据可以来自标准输入、一个或多个文件，或其它命令的输出。
- awk的处理文本和数据的方式：**逐行扫描文件**，默认从第一行到最后一行，寻找匹配的特定模式的行，并在这些行上进行你想要的操作。
- awk分别代表其作者姓氏的第一个字母。因为它的作者是三个人，分别是Alfred Aho、Brian Kernighan、Peter Weinberger。
- gawk是awk的GNU版本，它提供了Bell实验室和GNU的一些扩展。
- 下面介绍的awk是以GNU的gawk为例的，在linux系统中已把awk链接到gawk，所以下面全部以awk进行介绍。

1. awk使用方法

- 命令模式

```
awk [options] 'commands' filename

    option 部分
        -F 定义字段分割符号，默认的分隔符是空格
        -v 定义变量并赋值

    command 部分：
    '正则表达式，行号定位；BEGIN...END...{awk语句...}'

    '范围说明或正则表达式或者{awk命令语句1;awk命令语句2;...}'
```

```
    1.txt   10
    'BEGIN{awk语句}; {处理中};END{awk语句}'

    'BEGIN{}        {}          END{}'
     行处理前       行处理   行处理后

    1、范围说明部分可以是BEGIN、END、逻辑表达式或者为空
    2、awk命令语句间用分号间隔
    3、引用shell变量需用双引号引起

举例:
awk '/root/' /etc/passwd
awk 'NR==1' /etc/passwd
awk 'NR==1{print $1}' /etc/passwd
```

- 脚本模式

```
awk [options] -f scriptfile file(s)
特点:
1、awk脚本是awk命令的清单          '命令清单'
2、命令需要用分号间隔
3、#号开头的是注释行
4、#!/bin/awk -f

./awk.sh   filename
或者
awk -f awk.sh filename
```

2. awk工作原理

```
# awk -F: '{print $1,$3}' /etc/passwd
(1)awk使用一行作为输入，并将这一行赋给内部变量$0，每一行也可称为一个记录，以换行符(RS)结束

(2)然后，行被:（默认为空格或制表符）分解成字段（或域），每个字段存储在已编号的变量中，从$1开始。

(3)awk如何知道用空格来分隔字段的呢？ 因为有一个内部变量FS来确定字段分隔符。初始时，FS赋为空格

(4)awk打印字段时，将以设置的方法使用print函数打印，awk在打印的字段间加上空格，因为$1,$3之间
有一个逗号。逗号比较特殊，它映射为另一个内部变量，称为输出字段分隔符OFS，OFS默认为空格

(5)awk输出之后，将从文件中获取另一行，并将其存储在$0中，覆盖原来的内容，然后将新的字符串分隔
成字段并进行处理。该过程将持续到所有行处理完毕
```

3. awk基本应用

- **记录与字段相关内部变量**

```
$0  :表示当前所有记录
```

```
$1,$2,$3...$n: awk中用该顺序形式表示文件中中每行以间隔符号分割的各列的不同字段


注:
awk默认以空格符为间隔符号将每行分割为单独的字段，也可以使用awk内置变量FS定义间隔符号
# awk -F: '{print $1,$7}' a.txt
# awk 'BEGIN{FS=":"} {print $1,$7}' a.txt

NF          表示当前记录的字段数（列数）
$NF     最后一列
$(NF-1) 倒数第二列
FNR/NR  行号
FILENAME 文件名
"\t"        制表符
"\n"        换行符
""          打印字符串
FS      定义间隔符       awk 'BEGIN{FS=":"}{print $1,$3}' a.txt
OFS         定义输出字段分隔符，默认空格    awk -F: 'BEGIN{OFS="\t"}{print $1,$3}'  a.txt
RS          输入记录分割符，默认换行     awk -F: 'BEGIN{RS="\t"}{print $0}' a.txt
ORS     输出记录分割符，默认换行     awk -F: 'BEGIN{ORS="\n\n"}{print $1,$3}' a.txt
print   打印函数

awk -F: 'NR==2{print $1,$(NF-1)}'


格式化输出:
print函数     类似echo
# date |awk '{print "Month: "$2 "\nYear: "$NF}'
# awk -F: '{print "username is: " $1 "\t uid is: "$3}' /etc/passwd


printf函数        类似echo -n
# awk -F: '{printf "%-15s %-10s %-15s\n", $1,$2,$3}'  /etc/passwd
# awk -F: '{printf "|%15s| %10s| %15s|\n", $1,$2,$3}' /etc/passwd
# awk -F: '{printf "|%-15s| %-10s| %-15s|\n", $1,$2,$3}' /etc/passwd

awk 'BEGIN{FS=":"};{printf "%-15s %-15s %-15s\n",$1","$6","$NF}' a.txt

%s 字符类型  strings           %-20s
%d 数值类型
占15字符
- 表示左对齐，默认是右对齐
printf默认不会在行尾自动换行，加\n
```

- **定义变量**

```
# awk -v NUM=3 -F: '{ print $NUM }' /etc/passwd
# awk -v NUM=3 -F: '{ print NUM }' /etc/passwd
# awk -v num=1 'BEGIN{print num}'
1
# awk -v num=1 'BEGIN{print $num}'
注意:
awk中调用定义的变量不需要加$
```

- **定址**

```
关键字:
BEGIN: 表示在程序开始前执行
END : 表示所有文件处理完后执行

'BEGIN{开始处理之前};{处理中};END{处理结束后}'

示例:
1. 打印最后一列和倒数第二列(登录shell和家目录)   passwd
$NF $(NF-1)   -F:   FS=":"

awk 'BEGIN{FS=":";print "Login_shell\t\tLogin_home\n**********"};{print
$NF"\t\t"$(NF-1)}'  a.txt

# awk 'BEGIN{FS=":";print "Login_shell\t\tLogin_home\n**********"};{print
$NF"\t\t"$(NF-1)};END{print "**********"}'  a.txt
Login_shell     Login_home
**********
/bin/bash       /root
/sbin/nologin       /bin
/sbin/nologin       /sbin
/sbin/nologin       /var/adm
/sbin/nologin       /var/spool/lpd
/bin/bash       /home/redhat
/bin/bash       /home/user01
/sbin/nologin       /var/named
/bin/bash       /home/u01
/bin/bash       /home/YUNWEI
**********


2. 打印/etc/passwd里的用户名、家目录及登录shell
u_name      h_dir       shell
**********

**********




awk -F: 'BEGIN{print "u_name\t\th_dir\t\tshell" RS "**********"}  {printf "%-15s %-20s
```

```
%-20s\n",$1,$(NF-1),$NF}END{print "***************************"}'  /etc/passwd
```

格式化输出:

```
 {printf "%-15s %-20s %-20s\n",$1,$(NF-1),$NF}
```

```
awk 'BEGIN{}   {}   END{}'

awk 'BEGIN{FS=":";print "u_name\th_dir\tshell\n*************************************" }  {print
$1"\t"$(NF-1)"\t"$NF}  END{print "*****************************************"}'
```

3. 计算1~10的和

```
# for ((i=1;i<=10;i++));do echo $i;done|awk -v sum=0 '{sum=sum+$0};END{print sum}'
55
# for ((i=1;i<=10;i++));do echo $i;done|awk -v sum=0 '{sum+=$0};END{print sum}'
55
```

4. 打印1-10的奇数和

```
for ((i=1;i<=10;i+=2));do echo $i;done|awk -v sum=0 '{sum+=$0};END{print sum}'
```

5. awk打印1-5

```
awk 'BEGIN{for(i=1;i<=5;i++) print i}'

[root@MissHou shell05]# awk 'BEGIN{for(i=1;i<=10;i++) print i}'
1
2
3
4
5
6
7
8
9
10
[root@MissHou shell05]# awk 'BEGIN{for(i=1;i<=10;i+=2) print i}'
1
3
5
7
9
[root@MissHou shell05]# awk 'BEGIN{for(i=1;i<=10;i+=2) print i}'|awk '{sum+=$i}END{print sum}'
25

[root@MissHou shell05]# awk 'BEGIN{ for(i=1;i<=5;i++) print i   }'|awk -v sum=0 '{sum +=
$i};END{print sum}'
15


[root@MissHou shell05]# awk 'BEGIN{ for(i=1;i<=5;i++) print i   }'|awk -v sum=0 '{sum +=
```

```
$a};END{print sum}'
15
[root@MissHou shell05]# awk 'BEGIN{ for(i=1;i<=5;i++) print i   }'|awk -v sum=0 '{sum +=
$b};END{print sum}'
15

# awk -v sum=0 'BEGIN{for (i=1;i<=5;i++) (sum+=i);print sum}'
15
# awk -v sum=0 'BEGIN{for (i=1;i<=10;i+=2) (sum+=i);print sum}'
25
```

4. awk和正则的综合运用

- 逻辑运算符

```
==              (等于)
!=              (不等于)        !=
>               (大于)
<               (小于)
>=              (大于等于)
<=              (小于等于)
~               (匹配于)
!~              (不匹配)
!               (非)
&&              (与)
||              (或)
```

- 示例

```
awk '[xxx]{}'

# awk '/root/{print $1}' passwd         使用普通字符定位
# awk '$0 ~ /^root/ {print $1}'  passwd     使用正则表达式定位

注意：正则需要用"/xx/"包住

从第一行开始匹配到以lp开头行
awk -F: 'NR==1,/^lp/{print $0 }' passwd
从第一行到第5行
awk -F: 'NR==1,NR==5{print $0 }' passwd
从以lp开头的行匹配到第10行
awk -F: '/^lp/,NR==10 {print $0 }' passwd
从以root开头的行匹配到以lp开头的行
awk -F: '/^root/,/^lp/{print $0}' passwd
打印以root开头或者以lp开头的行
awk -F: '/^root/ || /^lp/{print $0}' passwd
awk -F: '/^root/;/^lp/{print $0}' passwd
显示5-10行
awk -F':' 'NR>=5 && NR<=10 {print $0}' /etc/passwd
awk -F: 'NR<10 && NR>5 {print $0}' passwd
```

```
//打印30-39行以bash结尾的内容:
[root@MissHou shell05]# awk 'NR>=30 && NR<=39 && $0 ~ /bash$/{print $0}' passwd
stu1:x:500:500::/home/stu1:/bin/bash
yunwei:x:501:501::/home/yunwei:/bin/bash
user01:x:502:502::/home/user01:/bin/bash
user02:x:503:503::/home/user02:/bin/bash
user03:x:504:504::/home/user03:/bin/bash


打印IP地址
# ifconfig eth0|awk 'NR>1 {print $2}'|awk -F':' 'NR<2 {print $2}'
# ifconfig eth0|grep Bcast|awk -F':' '{print $2}'|awk '{print $1}'
# ifconfig eth0|grep Bcast|awk '{print $2}'|awk -F: '{print $2}'
# ifconfig eth0|awk NR==2|awk -F '[ :]+' '{print $4RS$6RS$8}'
# ifconfig eth0|awk "-F[ :]+" '/inet addr:/{print $4}'


[root@MissHou shell05]# ifconfig eth0|awk 'NR==2{print $0}'| awk -F'[ :]+' '{print $4 ORS $6 ORS
$8}'
10.1.1.1
10.1.1.255
255.255.255.0
[root@MissHou shell05]# ifconfig eth0|awk 'NR==2{print $0}'
          inet addr:10.1.1.1  Bcast:10.1.1.255  Mask:255.255.255.0
[root@MissHou shell05]# ifconfig eth0|awk -F'[ :]+' 'NR==2{print $4 ORS $6 ORS $8}'
10.1.1.1
10.1.1.255
255.255.255.0
```

5. 课堂练习

```
1、显示可以登录操作系统的用户所有信息      //从第7列匹配以bash结尾，输出整行（当前行所有的列）
 awk '/bash$/{print $0}'    /etc/passwd
 1042  awk '/bash$/{print $0}' /etc/passwd
 1043  awk '/bash$/' /etc/passwd
 1044  awk -F: '$7 ~ /bash/' /etc/passwd
 1045  awk -F: '$NF ~ /bash/' /etc/passwd
 1046  awk -F: '$0 ~ /bash/' /etc/passwd
 1047  awk -F: '$0 ~ /\/bin\/bash/' /etc/passwd

2、显示可以登录系统的用户名
# awk -F: '$0 ~ /\/bin\/bash/{print $1}' /etc/passwd
3、打印出系统中普通用户的UID和用户名如下显示:
# awk -F: 'BEGIN{print "UID\tUSERNAME"} {if($3>=500 && $3 !=65534 ) {print $3"\t"$1} }'
/etc/passwdUID  USERNAME
500 stu1
501 yunwei
502 user01
503 user02
504 user03
```

```
# awk -F: '{if($3 >= 500 && $3 != 65534) print $1,$3}' a.txt
redhat 508
user01 509
u01 510
YUNWEI 511

4、以数字开头
awk '/^[0-9]/{print $0}'   1.txt
5、以任意大小写字母开头
awk '/^[a-Z]/{print $0}' 1.txt
```

6. awk的脚本编程

5.1 流程控制语句

```
if语句:

if [ xxx ];then
xxx
fi

格式:
{ if(表达式) {语句;语句; ...}  }

[root@MissHou shell05]# awk -F: '{if($3==0) {print $1"是管理员"} }' passwd
root是管理员

# awk 'BEGIN{if($(id -u)==0) {print "admin"} }'
admin


if...else语句:
if [ xxx ];then
    xxxxx
else
    xxx
fi


格式:
{   if(表达式) {语句;语句;...} else {语句;语句;...}   }

# awk -F: '{ if($3>=500 && $3 != 65534) {print $1"是普通用户"} else {print $1"不是普通用户"}}'
passwd

awk 'BEGIN{if( $(id -u)>=500 && $(id -u) !=65534 ) {print "是普通用户"} else {print "不是普通用
户"}}'



if [xxxx];then
    xxxx
elif [xxx];then
```

```
    xxx
....
else
...
fi
```

if...else if...else语句:

格式:

```
{ if(表达式1) {语句;语句; ...} else if(表达式2) {语句;语句; ...} else if(表达式3) {语句;语句; ...}
else {语句;语句; ...} }
```

```
# awk -F: '{if($3==0) {print $1,"is admin"} else if($3>=1 && $3<=499 || $3==65534) {print $1,"is
sys users"} else {print $1,"is general user"} }' a.txt
root is admin
bin is sys users
daemon is sys users
adm is sys users
lp is sys users
redhat is general user
user01 is general user
named is sys users
u01 is general user
YUNWEI is general user

awk -F: '{  if($3==0) {print $1":管理员"} else if($3>=1 && $3<500 || $3==65534 ) {print $1":是系
统用户"} else {print $1":是普通用户"}}'   /etc/passwd


awk -F: '{if($3==0) {i++} else if($3>=1 && $3<500 || $3==65534){j++} else {k++}};END{print "管理
员个数为:" i RS "系统用户个数为:"j RS "普通用户的个数为:"k }' /etc/passwd
管理员个数为:1
系统用户个数为:28
普通用户的个数为:27


# awk -F: '{ if($3==0) {print $1":是管理员"} else if($3>=500 && $3!=65534) {print $1":是普通用户"}
else {print $1":是系统用户"}}' passwd

awk -F: '{if($3==0){i++} else if($3>=500){k++} else{j++}} END{print i; print k; print j}'
/etc/passwd

awk -F: '{if($3==0){i++} else if($3>999){k++} else{j++}} END{print "管理员个数: "i; print "普通用
个数: "k; print "系统用户: "j}' /etc/passwd

如果是普通用户打印默认shell，如果是系统用户打印用户名
# awk -F: '{if($3>=1 && $3<500 || $3 == 65534) {print $1} else if($3>=500 && $3<=60000 ) {print
$NF} }' /etc/passwd
```

5.2 循环语句

```
while:
# i=1;while (( $i <=10 ));do echo $i;let i++;done
# awk 'BEGIN{i=1; while(i<=10){print i; i++}}'                    //打印1-10

# awk -F: '{i=1; while(i<=10) {print $0;i++}}' /etc/passwd       //将每行打印10次

for:
# awk 'BEGIN{for(i=1;i<=5;i++){print i} }'                                  //C风格for
1
2
3
4
5
# awk -F: '{for(i=1;i<=10;i++) print $0}' /etc/passwd                //将每行打印10次


for
打印1~5
# awk 'BEGIN{for(i=1;i<=5;i++) print i}'
1
2
3
4
5
打印1~10中的奇数
# awk 'BEGIN{for(i=1;i<=10;i=i+2) print i}'
1
3
5
7
9

while
打印1~5
# awk 'BEGIN {i=1;while(i<=5) {print i;i++}}'
打印1~10中的奇数
# awk 'BEGIN {i=1;while(i<=10) {print i;i+=2}}'
1
3
5
7
9


# awk 'BEGIN{for(y=1;y<=5;y++){for(x=1;x<=y;x++)printf x;print}}'
1
12
123
1234
12345
```

```
# awk 'BEGIN{y=1;while(y<=5) { for(x=1;x<=y;x++) {printf x} y++ ;print } }'
1
12
123
1234
12345

尝试用三种方法打印99口诀表:
# awk 'BEGIN{for(y=1;y<=9;y++) { for(x=1;x<=y;x++) printf x"*"y"="x*y"\t";print} }'
# awk 'BEGIN{i=1;while(i<=9){for(j=1;j<=i;j++){printf j"*"i"="j*i"\t"}print;i++ }}'
# awk 'BEGIN{for(i=1;i<=9;i++){j=1;while(j<=i){printf j"*"i"="i*j"\t";j++};print}}'

循环的控制:
break       条件满足的时候中断循环
continue    条件满足的时候跳过循环
# awk 'BEGIN{for(i=1;i<=5;i++){if(i==3) break;print i}}'
1
2
# awk 'BEGIN{for(i=1;i<=5;i++){if(i==3) continue;print i}}'
1
2
4
5

# awk 'BEGIN{i=1;while(i<=5){if(i==3) break;print i;i++}}'
1
2

# awk 'BEGIN{i=0;while(i<5){i++;if(i==3) continue;print i}}'
1
2
4
5
```

5.3 算数运算

```
+ - * / %(模) ^(幂2^3)
可以在模式中执行计算，awk都将按浮点数方式执行算术运算
# awk 'BEGIN{print 1+1}'
# awk 'BEGIN{print 1**1}'
# awk 'BEGIN{print 2**3}'
# awk 'BEGIN{print 2/3}'
```

7. 统计案例

```
1. 统计/etc/passwd 中各种类型shell的数量
# awk -F: '{shells[$NF]++};  END{for (i in shells) {print i,shells[i]}}' /etc/passwd

/bin/bash 5

/sbin/nologin 6
```

```
shells[/bin/bash]++         a
shells[/sbin/nologin]++     b
shells[/sbin/shutdown]++    c

books[linux]++
books[php]++
```

2. 网站访问状态统计 <当前时实状态 netstat>

```
# ss -antp|grep 80|awk '{states[$1]++};END{for(i in states){print i,states[i]}}'
TIME_WAIT 578
ESTABLISHED 1
LISTEN 1

# ss -an |grep :80 |awk '{states[$2]++};END{for(i in states){print i,states[i]}}'
LISTEN 1
ESTAB 5
TIME-WAIT 25

# ss -an |grep :80 |awk '{states[$2]++};END{for(i in states){print i,states[i]}}' |sort -k2 -rn
TIME-WAIT 18
ESTAB 8
LISTEN 1
```

3. 统计当前访问的每个IP的数量 <当前时实状态 netstat,ss>

```
# netstat -ant |grep :80 |awk -F: '{ip_count[$8]++};END{for(i in ip_count){print i,ip_count[i]}
}' |sort


# ss -an |grep :80 |awk -F":" '!/LISTEN/{ip_count[$(NF-1)]++} END{for(i in ip_count){print
i,ip_count[i]}}' |sort -k2 -rn |head
```

4. 统计Apache/Nginx日志中某一天的PV量 <统计日志>

```
# grep '27/Jul/2017' mysqladmin.cc-access_log |wc -l
14519
```

5. 统计Apache/Nginx日志中某一天不同IP的访问量 <统计日志>

```
# grep '27/Jul/2017' mysqladmin.cc-access_log |awk '{ips[$1]++} END{for(i in ips){print
i,ips[i]} }' |sort -k2 -rn |head

# grep '07/Aug/2017' access.log |awk '{ips[$1]++} END{for(i in ips){print i,ips[i]} }' |awk
'$2>100' |sort -k2 -rn
```

名词引入：

名词：VV = Visit View（访问次数）
说明：从访客来到您网站到最终关闭网站的所有页面离开，计为1次访问。若访客连续30分钟没有新开和刷新页面，或者访客关闭了浏览器，则被计算为本次访问结束。

独立访客（UV）
名词：UV= Unique Visitor（独立访客数）

说明：1天内相同的访客多次访问您的网站只计算1个UV。

网站浏览量（PV）
名词：PV=PageView（网站浏览量）
说明：指页面的浏览次数，用以衡量网站用户访问的网页数量。多次打开同一页面则浏览量累计。用户每打开一个页面便记录1次PV。

独立IP（IP）
名词：IP=独立IP数
说明：指1天内使用不同IP地址的用户访问网站的数量。同一IP无论访问了几个页面，独立IP数均为1

总结：

1. 使用shell脚本判断一个局域网中哪些ip通哪些不通

分析：
条件判断
循环判断

```
#!/bin/bash
#注释说明

#定义变量
ip1=10.1.1
#定义函数
fun_name(){


}

#根据需求或者思路展开脚本编写
for i in {1..254}
do
{
ping $ip1.$i &>/dev/null
test $? -eq 0 && echo "$ip1.$i" >> ip_up.txt || echo "$ip1.$i" >> ip_down.txt
}&
done
wait
```

2. 跳板机上的yunwei用户将自己的公钥推送到100台线上服务器上（业务用户pos1）

```
#!/bin/bash

#判断yunwei用户是否有公钥

#将生成的公钥远程推送到100台服务器上（交互）
##如何使用非交互形式来推送（expect）
#判断是否安装expect
while read ip pass

do
```

```
/usr/bin/expect<<-EOF
spawn ssh-copy-id -i pos1@$ip
expect {
"yes/no" { send "yes\r";exp_continue}
"password:" { send "$pass\r"}
}
expect eof
EOF
done <ip_up.txt
```

3. 简单的用shell脚本实现一个jumper的功能

```
case语句菜单选择
```

课后作业2

作业1： 1、写一个自动检测磁盘使用率的脚本，当磁盘使用空间达到90%以上时，需要发送邮件给相关人员 2、写一个脚本监控系统内存和交换分区使用情况

```

```

作业2： 输入一个IP地址，使用脚本判断其合法性： 必须符合ip地址规范，第1、4位不能以0开头，不能大于255 不能小于0

```

```

三、课后实战

1. 任务/背景

web服务器集群中总共有9台机器，上面部署的是Apache服务。由于业务不断增长，每天每台机器上都会产生大量的访问日志，现需要将每台web服务器上的apache访问日志**保留最近3天**的，3天以前的日志**转储**到一台专门的日志服务器上，已做后续分析。如何实现每台服务器上只保留3天以内的日志?

2. 具体要求

1. 每台web服务器的日志对应日志服务器相应的目录里。如：web1——>web1.log（在日志服务器上）
2. 每台web服务器上保留最近3天的访问日志，3天以前的日志每天凌晨5:03分转储到日志服务器
3. 如果脚本转储失败，运维人员需要通过跳板机的菜单选择手动清理日志

3. 涉及知识点

1. shell的基本语法结构
2. 文件同步rsync
3. 文件查找命令find
4. 计划任务crontab
5. apache日志切割
6. 其他