

课程目标

- 熟悉awk的**命令行模式**基本语法结构
- ==熟悉awk的相关内部变量==
- 熟悉awk常用的打印==函数print==
- 能够在awk中匹配正则表达式打印相关的行

一、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使用方式

- 命令模式语法

```
awk 选项 'commands' 文件名
```

- 常用选项

- -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

- =='==命名部分(commands)=='==

- 正则表达式，地址定位

'/root/{awk语句}'	sed中: '/root/p'
'NR==1,NR==5{awk语句}'	sed中: '1,5p'
'/^root/,/^ftp/{awk语句}'	sed中: '/^root/,/^ftp/p'

- {awk语句1==;==awk语句2==;==...}

'{print \$0;print \$1}'	sed中: 'p'
'NR==5{print \$0}'	sed中: '5p'

注: awk命令语句间用分号间隔

- BEGIN...END....

```
'BEGIN{awk语句};{处理中};END{awk语句}'  
'BEGIN{awk语句};{处理中}'  
'{处理中};END{awk语句}'
```

- 引用shell变量需用双引号引起

- 脚本模式

- 脚本执行

方法1:

awk 选项 -f awk的脚本文件 要处理的文本文件

awk -f awk.sh filename

sed -f sed.sh -i filename

方法2:

./awk的脚本文件(或者绝对路径) 要处理的文本文件

./awk.sh filename

./sed.sh filename

- 脚本编写

#!/bin/awk -f 定义魔法字符

以下是awk引号里的命令清单，不要用引号保护命令，多个命令用分号间隔

BEGIN{FS=":"}

NR==1,NR==3{print \$1"\t"\$NF}

...

2. awk内部相关变量

变量	变量说明	备注
\$0	当前处理行的所有记录	
\$1,\$2,\$3...\$n	文件中每行以间隔符号分割的不同字段	awk -F: '{print \$1,\$3}'
NF	当前记录的字段数（列数）	awk -F: '{print NF}'
\$NF	最后一列	\$(NF-1)表示倒数第二列
FNR/NR	行号	
FS	定义间隔符	'BEGIN{FS=":"};{print \$1,\$3}'
OFS	定义输出字段分隔符，默认空格	'BEGIN{OFS="\t"};print \$1,\$3}'
RS	输入记录分割符，默认换行	'BEGIN{RS="\n"};{print \$0}'
ORS	输出记录分割符，默认换行	'BEGIN{ORS="\n\n"};{print \$1,\$3}'
FILENAME	当前输入的文件名	

- 示例1

```
[root@server shell107]# awk -F: '{print $1,$(NF-1)}' 1.txt
```

```
[root@server shell07]# awk -F: '{print $1,$(NF-1),$NF,NF}' 1.txt
[root@server shell07]# awk '/root/{print $0}' 1.txt
[root@server shell07]# awk '/root/' 1.txt
[root@server shell07]# awk -F: '/root/{print $1,$NF}' 1.txt
root /bin/bash
[root@server shell07]# awk -F: '/root/{print $0}' 1.txt
root:x:0:0:root:/root:/bin/bash
[root@server shell07]# awk 'NR==1,NR==5' 1.txt
[root@server shell07]# awk 'NR==1,NR==5{print $0}' 1.txt
[root@server shell07]# awk 'NR==1,NR==5;/^root/{print $0}' 1.txt
root:x:0:0:root:/root:/bin/bash
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
```

- 示例2

FS和OFS:

```
[root@server shell07]# awk 'BEGIN{FS=":"};/^root/,/^lp/{print $1,$NF}' 1.txt
[root@server shell07]# awk -F: 'BEGIN{OFS="\t\t"};/^root/,/^lp/{print $1,$NF}' 1.txt
root          /bin/bash
bin           /sbin/nologin
daemon        /sbin/nologin
adm           /sbin/nologin
lp            /sbin/nologin
[root@server shell07]# awk -F: 'BEGIN{OFS="@@"};/^root/,/^lp/{print $1,$NF}' 1.txt
root@@@/bin/bash
bin@@@/sbin/nologin
daemon@@@/sbin/nologin
adm@@@/sbin/nologin
lp@@@/sbin/nologin
[root@server shell07]#
```

RS和ORS:

修改源文件前2行增加制表符和内容:

```
vim 1.txt
root:x:0:0:root:/root:/bin/bash hello    world
bin:x:1:1:bin:/bin:/sbin/nologin        test1   test2
```

```
[root@server shell07]# awk 'BEGIN{RS="\t"};{print $0}' 1.txt
[root@server shell07]# awk 'BEGIN{ORS="\t"};{print $0}' 1.txt
```

- 格式化输出 print 和 printf

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
```

3. 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. 每行被间隔符===(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
问：awk如何知道用空格来分隔字段的呢？
答：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
3. awk使用print函数打印字段，打印出来的字段会以==空格分隔==，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
4. awk处理完一行后，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

4. awk变量定义

```
# 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 $num}'
注意：
awk中调用定义的变量不需要加$
```

5. awk中BEGIN...END使用

- ①==BEGIN==：表示在程序开始前执行
- ②==END==：表示所有文件处理完后执行
- ③用法：'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）

```
awk -F: 'BEGIN{ print "Login_shell\t\tLogin_home\n*****"};{print $NF"\t\t"$(NF-1)};END{print "*****"}' 1.txt

awk 'BEGIN{ FS=":";print "Login_shell\tLogin_home\n*****"};{print $NF"\t"$(NF-1)};END{print "*****"}' 1.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{OFS="\t\t";print"u_name\t\tth_dir\t\tshell\n*****"}
{printf "%-20s %-20s %-20s\n",$1,$(NF-1),$NF};END{print
"*****"}'

# awk -F: 'BEGIN{print "u_name\t\tth_dir\t\tshell" RS "*****"}
{printf "%-15s %-20s %-20s\n",$1,$(NF-1),$NF}END{print
"*****"}' /etc/passwd

格式化输出:
echo      print
echo -n printf

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

```

6. awk和正则的综合运用

运算符	说明
==	等于
!=	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于
~	匹配
!~	不匹配
!	逻辑非
&&	逻辑与
	逻辑或

- 示例

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 shell105]# 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
```

```
[root@server shell107]# awk 'NR>=3 && NR<=8 && /bash$/' 1.txt
```

```
stu7:x:1007:1007::/rhome/stu7:/bin/bash
```

```
stu8:x:1008:1008::/rhome/stu8:/bin/bash
```

```
stu9:x:1009:1009::/rhome/stu9:/bin/bash
```

理解;号和||的含义:

```
[root@server shell107]# awk 'NR>=3 && NR<=8 || /bash$/' 1.txt
```

```
[root@server shell107]# awk 'NR>=3 && NR<=8;/bash$/' 1.txt
```

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}'
```

7. 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）

```
[root@Misshou ~] awk '/bash$/ {print $0}' /etc/passwd
[root@Misshou ~] awk '/bash$/ {print $0}' /etc/passwd
[root@Misshou ~] awk '/bash$/' /etc/passwd
[root@Misshou ~] awk -F: '$7 ~ /bash/' /etc/passwd
[root@Misshou ~] awk -F: '$NF ~ /bash/' /etc/passwd
[root@Misshou ~] awk -F: '$0 ~ /bash/' /etc/passwd
[root@Misshou ~] 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
```

8. awk的脚本编程

8.1 流程控制语句

if语句：

```
if [ xxx ];then
xxx
fi
```

格式：

```
awk 选项 '正则，地址定位{awk语句}' 文件名
{ if(表达式) {语句1;语句2;...} }
```

```
awk -F: '{if($3>=500 && $3<=60000) {print $1,$3} }' passwd
```

```
[root@Misshou shell105]# 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,"是管理员"} else if($3>=1 && $3<=499 || $3==65534 ) {print $1,"是系统用户"} else {print $1,"是普通用户"}}'
```

```
awk -F: '{ if($3==0) {i++} else if($3>=1 && $3<=499 || $3==65534 ) {j++} else {k++}};END{print "管理员个数为:"i "\n系统用户个数为:"j"\n普通用户的个数为:"k }'
```

```
# 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 "普通用户个数: "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
```

8.2 循环语句

```
while:
# awk 'BEGIN { i=1; while(i<=10) {print i;i++} }'
```

文件里的每一行循环打印10次:

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

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

文件里的每一行循环打印10次:

```
# awk -F: '{ for(i=1;i<=10;i++) {print $0}}' /etc/passwd
```

打印1~5

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

```
# 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} }'
```

```
awk 选项 '{awk语句1 语句2 }'
```

计算1-5的和:

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

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

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

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

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}'
```

嵌套循环:

```
#!/bin/bash
```

```
for ((y=1;y<=5;y++))
```

```
do
```

```
    for ((x=1;x<=$y;x++))
```

```
    do
```

```
        echo -n $x
```

```
    done
```

```
echo
```

```
done
```

```
# 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{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

```

8.3 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}'

```

###二. awk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

books[linux]++
books[linux]=1
shells[/bin/bash]++
shells[/sbin/nologin]++

/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

###三、课后作业

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

四、企业实战案例

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