

NSD SHELL DAY05

1. [案例1：sed综合脚本应用](#)
2. [案例2：使用awk提取文本](#)
3. [案例3：awk处理条件](#)
4. [案例4：awk综合脚本应用](#)

1 案例1：sed综合脚本应用

1.1 问题

本案例要求编写脚本getupwd.sh，实现以下需求：

- 找到使用bash作登录Shell的本地用户
- 列出这些用户的shadow密码记录
- 按每行“用户名 --> 密码记录”保存到getupwd.log，如图-1所示

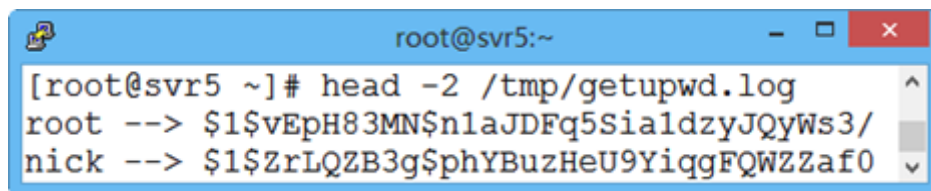


图 - 1

1.2 方案

基本思路如下：

1. 先用sed工具取出登录Shell为/bin/bash的用户记录，保存为临时文件/tmp/urec.tmp，并计算记录数量
2. 再结合while循环遍历取得的账号记录，逐行进行处理
3. 针对每一行用户记录，采用掐头去尾的方式获得用户名、密码字符串
4. 按照指定格式追加到/tmp/getupwd.log文件
5. 结束循环后删除临时文件，报告分析结果

1.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写getupwd.sh脚本

```
01. [root@svr5 ~]# vim ./getupwd.sh
02. #/bin/bash
03. A=$(sed -n '/bash$/s/:.*/p' /etc/passwd)      ## 提取符合条件的账号记录
04. for i in $A                                     ##遍历账号记录
05. do
06.     pass1=$(grep $i /etc/shadow)
07.     pass2=${pass1#*:}
08.     pass=${pass2%*:}
```

[Top](#)

```
09.      echo "$i --> $pass"
10.      done
11.
12.      [root@svr5 ~]# chmod +x ./getupwd.sh
```

步骤二：测试、验证执行结果

```
01.      [root@svr5 ~]# ./getupwd.sh
02.      用户分析完毕，请查阅文件 /tmp/getupwd.log
03.
04.      [root@svr5 ~]# less /tmp/getupwd.log
05.      root --> $6$IWgMYmRACwdbfwBo$dr8Yn983nswiJVwOdTMjzbDvSLeCd1GMYjbvsDiFEkl
06.      zengye --> $6$Qb37LOdzRI5995PI$L0zTOgnhGz8ihWkW81J.5XhPp/I7x2./Me2agOS8tf
07.      clamav --> !!
08.      mysql --> !!
09.      abc --> !!
10.      .. ..
```

从上述参考脚本可以发现，使用sed来实现字段提取会比较复杂。下一章课程将会学到awk命令，届时可以通过更简单的方法来改进此脚本内容。

总结知识点：

#sed [选项] '条件指令' 文件

选项：

-n 屏蔽默认输出

-r 支持扩展正则

-i 修改源文件

条件：

行号 4 4,5 4~2 4,+10

/正则/

指令：

p 打印

d 删除

s 替换s/旧/新/g

a 追加

i 插入

c 替换行

[Top](#)

2 案例2：使用awk提取文本

2.1 问题

本案例要求使用awk工具完成下列过滤任务：

- 练习awk工具的基本用法
- 提取本机的网卡流量、根分区剩余容量、获取SSH远程失败的IP地址
- 格式化输出/etc/passwd文件中的用户名、UID、宿主目录信息

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：awk文本过滤的基本用法

1) 基本操作方法

格式：awk [选项] '[条件]'{指令}' 文件

其中，print 是最常用的编辑指令；若有多条编辑指令，可用分号分隔。

Awk过滤数据时支持仅打印某一列，如第2列、第5列等。

处理文本时，若未指定分隔符，则默认将空格、制表符等作为分隔符。

直接过滤文件内容：

```
01. [root@svr5 ~]# cat test.txt
02. hello the world
03. welcome to beijing
04. [root@svr5 ~]# awk '{print $1,$3}' test.txt //打印文档第1列和第3列
05. hello world
06. welcome beijing
```

结合管道过滤命令输出：

```
01. [root@svr5 ~]# df -h | awk '{print $4}' //打印磁盘的剩余空间
```

2) 选项 -F 可指定分隔符

输出passwd文件中以分号分隔的第1、7个字段，显示的不同字段之间以逗号隔开，操作如下：

```
01. [root@svr5 ~]# awk -F: '{print $1,$7}' /etc/passwd
02. root /bin/bash
03. bin /sbin/nologin
04. daemon /sbin/nologin
05. adm /sbin/nologin
06. lp /sbin/nologin
07. ... ..
```

[Top](#)

awk还识别多种单个的字符，比如以“:”或“/”分隔，输出第1、10个字段：

```
01. [root@svr5 ~]# awk -F [:/] '{print $1,$10}' /etc/passwd
02. root bash
03. bin nologin
04. daemon nologin
05. adm sbin
06. ... ..
```

awk常用内置变量：

\$0 文本当前行的全部内容

\$1 文本的第1列

\$2 文件的第2列

\$3 文件的第3列，依此类推

NR 文件当前行的行号

NF 文件当前行的列数（有几列）

输出每次处理行的行号，以及当前行以“:”分隔的字段个数（有几列）：

```
01. [root@svr5 ~]# awk -F: '{print NR,NF}' passwd.txt
02. 1 7
03. 2 7
04. 3 7
05. .. ..
```

2) awk的print指令不仅可以打印变量，还可以打印常量

```
01. [root@svr5 ~]# awk -F: '{print $1,"的解释器:",$7}' /etc/passwd
02. root 的解释器: /bin/bash
03. bin 的解释器: /sbin/nologin
04. ... ..
```

步骤二：利用awk提取本机的网络流量、根分区剩余容量、获取远程失败的IP地址

1) 提取IP地址

分步实现的思路及操作参考如下——

通过ifconfig eth0查看网卡信息，其中包括网卡流量：

[Top](#)

```

01. [root@svr5 ~]# ifconfig eth0
02. eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
03.      inet 192.168.4.21 netmask 255.255.255.0 broadcast 192.168.4.255
04.      inet6 fe80::fa64:c143:ad6a:5159 prefixlen 64 scopeid 0x20<link>
05.      ether 52:54:00:b3:11:11 txqueuelen 1000 (Ethernet)
06.      RX packets 313982 bytes 319665556 (304.8 MiB)
07.      RX errors 0 dropped 0 overruns 0 frame 0
08.      TX packets 51809 bytes 40788621 (38.8 MiB)
09.      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

RX为接收的数据量，TX为发送的数据量。packets以数据包的数量为单位，bytes以字节为单位：

```

01. [root@svr5 ~]# ifconfig eth0 | awk '/RX p/{print $5}' //过滤接收数据的流量
02. 319663094
03. [root@svr5 ~]# ifconfig eth0 | awk '/TX p/{print $5}' //过滤发送数据的流量
04. 40791683

```

2) 提取根分区剩余容量

分步实现的思路及操作参考如下——

通过df命令查看根分区的使用情况，其中包括剩余容量：

```

01. [root@svr5 ~]# df -h /
02. 文件系统      容量  已用   可用   已用%   挂载点
03. /dev/sda2      19G   7.2G   11G    40%    /

```

输出上述结果中最后一行的第4列：

```

01. [root@svr5 ~]# df -h / | tail -1 | awk '{print $4}'
02. 11G

```

或者直接在awk中使用正则：

```

01. [root@svr5 ~]# df -h | awk '/\V$/{print $4}'
02. 11G

```

[Top](#)

3) 根据/var/log/secure日志文件，过滤远程连接密码失败的IP地址

```
01. [root@svr5 ~]# awk '/Failed/{print $11}' /var/log/secure
02. 192.168.2.254
03. 192.168.2.100
04. ... ..
```

步骤三：格式化输出/etc/passwd文件

1) awk处理的时机

awk会逐行处理文本，支持在处理第一行之前做一些准备工作，以及在处理完最后一行之后做一些总结性质的工作。在命令格式上分别体现如下：

```
01. awk [选项] '[条件][指令]' 文件
02. awk [选项] ' BEGIN{指令} {指令} END{指令}' 文件
```

- BEGIN{ } 行前处理，读取文件内容前执行，指令执行1次
- { } 逐行处理，读取文件过程中执行，指令执行n次
- END{ } 行后处理，读取文件结束后执行，指令执行1次

只做预处理的时候，可以没有操作文件，比如：

```
01. [root@svr5 ~]# awk 'BEGIN{A=24;print A*2}'
02. [root@svr5 ~]# awk 'BEGIN{print x+1}'          #x可以不定义，直接用，默认值位0
03. [root@svr5 ~]# awk 'BEGIN{print 3.2+3.5}'
```

举个例子（统计系统中使用bash作为登录Shell的用户总个数）：

a.预处理时赋值变量x=0

b.然后逐行读入/etc/passwd文件，如果发现登录Shell是/bin/bash则x加1

c.全部处理完毕后，输出x的值即可。相关操作及结果如下：

```
01. [root@svr5 ~]# awk 'BEGIN{x=0}/bash$/{x++} END{print x}' /etc/passwd
02. 29
```

2) 格式化输出/etc/passwd文件

要求: 格式化输出passwd文件内容时，要求第一行为列表标题，中间打印用户的名称、[Top](#)UID、家目录信息，最后一行提示一共已处理文本的总行数，如图-1所示。

```

User      UID      Home
root      0        /root
bin        1        /bin
daemon    2        /sbin
adm        3        /var/adm
.. ..
Total 59 lines.

```

图-1

3) 根据实现思路编写、验证awk过滤语句

输出信息时，可以使用“\t”显示Tab制表位：

```

01. [root@svr5 ~]# awk -F: 'BEGIN{print "User\tUID\tHome"} \
02.                               {print $1 "\t" $3 "\t" $6} \
03.                               END{print "Total",NR,"lines."}' /etc/passwd
04. User  UID  Home
05. root  0    /root
06. bin   1    /bin
07. daemon 2    /sbin
08. adm   3    /var/adm
09. lp    4    /var/spool/lpd
10. sync  5    /sbin
11. .. ..
12. Total 67 lines.

```

3 案例3：awk处理条件

3.1 问题

本案例要求使用awk工具完成下列过滤任务，注意awk处理条件的设置：

- 列出UID间于1~1000的用户详细信息
- 输出/etc/hosts文件内以127或192开头的记录
- 列出100以内整数中7的倍数或是含7的数

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：认识awk处理条件的设置

1) 使用正则设置条件

输出其中以bash结尾的完整记录：

```

01. [root@svr5 ~]# awk -F: '/bash$/ {print}' /etc/passwd
02. root:x:0:0:root:/root:/bin/bash

```

[Top](#)

输出包含root的行数据：

01. [root@svr5 ~]# awk -F: '/root/' /etc/passwd

输出root或adm账户的用户名和UID信息：

```
01. [root@svr5 ~]# awk -F: '/^(root|adm)/{print $1,$3}' /etc/passwd
02. root 0
03. adm 3
```

输出账户名称包含root的基本信息（第1列包含root）：

01. [root@svr5 ~]# awk -F: '\$1~/root/' /etc/passwd

输出其中登录Shell不以nologin结尾（对第7个字段做!~反向匹配）的用户名、登录Shell信息：

```
01. [root@svr5 ~]# awk -F: '$7!~/nologin${print $1,$7}' /etc/passwd
02. root /bin/bash
03. sync /bin/sync
04. shutdown /sbin/shutdown
```

2) 使用数值/字符串比较设置条件

比较符号：==(等于) != (不等于) > (大于)

>= (大于等于) < (小于) <= (小于等于)

输出第3行（行号NR等于3）的用户记录：

01. [root@svr5 ~]# awk -F: 'NR==3{print}' /etc/passwd

输出账户UID大于等于1000的账户名称和UID信息：

```
01. [root@svr5 ~]# awk -F: '$3>=1000{print $1,$3}' /etc/passwd
02. tom 1000
03. jerry 1001
```

[Top](#)

输出账户UID小于10的账户名称和UID信息：

```
01. [root@svr5 ~]# awk -F: '$3<10{print $1,$3}' /etc/passwd
02. root 0
03. bin 1
04. daemon 2
05. adm 3
06. lp 4
07. sync 5
08. shutdown 6
09. halt 7
10. mail 8
```

输出用户名为“root”的行：

```
01. [root@svr5 ~]# awk -F: '$1=="root"' /etc/passwd
02. root:x:0:0:root:/root:/bin/bash
```

3) 逻辑测试条件

输出账户UID大于10并且小于20的账户信息：

```
01. [root@svr5 ~]# awk -F: '$3>10 && $3<20' /etc/passwd
02. operator:x:11:0:operator:/root:/sbin/nologin
03. games:x:12:100:games:/usr/games:/sbin/nologin
04. ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

输出账户UID大于1000或者账户UID小于10的账户信息：

```
01. [root@svr5 ~]# awk -F: '$3>1000 || $3<10' /etc/passwd
02. root:x:0:0:root:/root:/bin/bash
03. bin:x:1:1:bin:/bin:/sbin/nologin
04. daemon:x:2:2:daemon:/sbin:/sbin/nologin
05. adm:x:3:4:adm:/var/adm:/sbin/nologin
06. lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
07. sync:x:5:0:sync:/sbin:/bin/sync
08. shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
```

[Top](#)

09. halt:x:7:0:halt:/sbin:/sbin/halt
10. mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
11. varnish:x:1001:1001::/home/varnish:/sbin/nologin
12. nginx:x:1002:1002::/home/nginx:/sbin/nologin

4) 数学运算

01. [root@svr5 ~]# awk 'BEGIN{x++;print x}'
02. 1
03. [root@svr5 ~]# awk 'BEGIN{x=8;print x+=2}'
04. 10
05. [root@svr5 ~]# awk 'BEGIN{x=8;x--;print x}'
06. 7
07. [root@svr5 ~]# awk 'BEGIN{print 2+3}'
08. 5
09. [root@svr5 ~]# awk 'BEGIN{print 2*3}'
10. 6
11. [root@svr5 ~]# awk 'BEGIN{print 2*3}'
12. 6
13. [root@svr5 ~]# awk 'BEGIN{ print 23%8}'
14. 7
15. [root@svr5 ~]# seq 200 | awk '\$1%3==0' //找200以内3的倍数
16.

步骤二：完成任务要求的awk过滤操作

1) 列出UID间于1~1000的用户详细信息：

01. [root@svr5 ~]# awk -F: '\$3>=1 && \$3<=1000' /etc/passwd

2) 输出/etc/hosts映射文件内以127或者192开头的记录：

01. [root@svr5 ~]# awk '/^(127|192)/' /etc/hosts
02. 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
03. 192.168.4.5 svr5.tarena.com svr5

[Top](#)

3) 列出100以内整数中7的倍数或是含7的数：

```

01. [root@svr5 ~]# seq 100 | awk '$1%7==0||$1~/7/'
02. 7
03. 14
04. 17
05. 21
06. 27
07. 28
08. 35
09. 37
10. 42
11. 47
12. ...

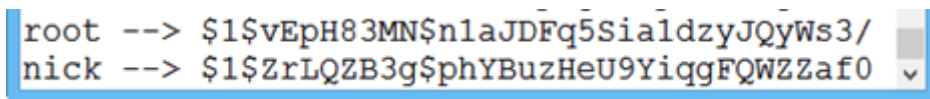
```

4 案例4：awk综合脚本应用

4.1 问题

本案例要求编写脚本，实现以下需求：

- 找到使用bash作登录Shell的本地用户
- 列出这些用户的shadow密码记录，如图-2所示



```

root --> $1$vEpH83MN$nlajDFq5SialdzyJQyWs3/
nick --> $1$ZrLQZB3g$phYBuzHeU9YiqgFQWZZaf0

```

图 - 2

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：任务需求及思路分析

编写脚本的任务要求如下：

- 分析出使用bash作登录Shell的本地用户
- 列出这些用户的shadow密码记录
- 按每行“用户名 -- 密码记录”保存结果

步骤二：根据实现思路编写脚本

```

01. [root@svr5 ~]# cat getupwd-awk.sh
02. #/bin/bash
03. A=$(awk -F: '/bash${print $1}' /etc/passwd)    ## 提取符合条件的账号记录
04.
05. for i in $A
06. do
07.     grep $i /etc/shadow | awk -F: '{print $1,"-->",$2}'

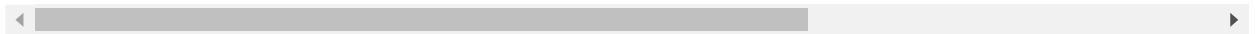
```

[Top](#)

08. done

步骤三：验证、测试脚本

01. [root@svr5 ~]# ./getupwd-awk.sh
02. root --> \$6\$IWgMYmRACwdbfwBo\$dr8Yn983nswiJVw0dTMjzbDvSLeCd1GMYjbvsDiFEkl
03. zengye --> \$6\$Qb37LOdzRI5995PI\$L0zTOgnhGz8ihWkW81J.5XhPp/I7x2./Me2ag0S8tf
04. clamav --> !!
05. mysql --> !!
06.



[Top](#)