[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY05/CASE/01/index.html" \l "page_top_case)

# NSD SHELL DAY05

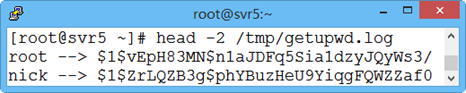
1. [案例1：sed综合脚本应用](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY05/CASE/01/index.html" \l "case1)
2. [案例2：使用awk提取文本](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY05/CASE/01/index.html" \l "case2)
3. [案例3：awk处理条件](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY05/CASE/01/index.html" \l "case3)
4. [案例4：awk综合脚本应用](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SHELL/DAY05/CASE/01/index.html" \l "case4)

## 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、密码字串
4. 按照指定格式追加到/tmp/getuupwd.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

1. [root@svr5 ~]# vim ./getupwd.sh
2. #/bin/bash
3. A=$(sed -n '/bash$/s/:.\*//p' /etc/passwd)             ## 提取符合条件的账号记录
4. for i in $A                                     ##遍历账号记录
5. do
6. pass1=$(grep $i /etc/shadow)
7. pass2=${pass1#\*:}
8. pass=${pass2%%:\*}
9. echo "$i --> $pass"
10. done
11. [root@svr5 ~]# chmod +x ./getupwd.sh

步骤二：测试、验证执行结果

1. [root@svr5 ~]# ./getupwd.sh
2. 用户分析完毕，请查阅文件 /tmp/getupwd.log
3. [root@svr5 ~]# less /tmp/getupwd.log
4. root --> $6$IWgMYmRACwdbfwBo$dr8Yn983nswiJVw0dTMjzbDvSLeCd1GMYjbvsDiFEkL8jnXOLcocBQypOCr4C6BRxNowIxjh6U2qeFU0u1LST/
5. zengye --> $6$Qb37LOdzRl5995PI$L0zTOgnhGz8ihWkW81J.5XhPp/l7x2./Me2ag0S8tRndCBL9nIjHIKkUKulHxJ6TXyHYmffbVgUT6pbSwf8O71
6. clamav --> !!
7. mysql --> !!
8. abc --> !!
9. .. ..

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

## 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：

1. [root@svr5 ~]# cat test.txt
2. hello the world
3. welcome to beijing
4. [root@svr5 ~]# awk '{print $1,$3}' test.txt        //打印文档第1列和第3列
5. hello world
6. welcome beijing

结合管道过滤命令输出：

1. [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：

1. [root@svr5 ~]# awk -F: '{print $1,$7}' /etc/passwd
2. root /bin/bash
3. bin /sbin/nologin
4. daemon /sbin/nologin
5. adm /sbin/nologin
6. lp /sbin/nologin
7. … …

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：

1. [root@svr5 ~]# awk -F [:/] '{print $1,$10}' /etc/passwd
2. root bash
3. bin nologin
4. daemon nologin
5. adm sbin
6. … …

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）：

1. [root@svr5 ~]# awk -F: '{print NR,NF}' passwd.txt
2. 1 7
3. 2 7
4. 3 7
5. .. ..

2）awk的print指令不仅可以打印变量，还可以打印常量

1. [root@svr5 ~]# awk -F: '{print $1,"的解释器:",$7}' /etc/passwd
2. root 的解释器: /bin/bash
3. bin 的解释器: /sbin/nologin
4. … …

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：

1. [root@svr5 ~]# ifconfig eth0
2. eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
3. inet 192.168.4.21 netmask 255.255.255.0 broadcast 192.168.4.255
4. inet6 fe80::fa64:c143:ad6a:5159 prefixlen 64 scopeid 0x20<link>
5. ether 52:54:00:b3:11:11 txqueuelen 1000 (Ethernet)
6. RX packets 313982 bytes 319665556 (304.8 MiB)
7. RX errors 0 dropped 0 overruns 0 frame 0
8. TX packets 51809 bytes 40788621 (38.8 MiB)
9. 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：

1. [root@svr5 ~]# ifconfig eth0 | awk '/RX p/{print $5}' //过滤接收数据的流量
2. 319663094
3. [root@svr5 ~]# ifconfig eth0 | awk '/TX p/{print $5}'     //过滤发送数据的流量
4. 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：

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

输出上述结果中最后一行的第4列：

1. [root@svr5 ~]# df -h / | tail -1 | awk '{print $6}'
2. 11G

或者直接在awk中使用正则：

1. [root@svr5 ~]# df -h | awk '/\/$/{print $4}'
2. 11G

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

1. [root@svr5 ~]# awk '/Failed/{print $11}' /var/log/secure
2. 192.168.2.254
3. 192.168.2.100
4. ... ...

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：

1. awk [选项] '[条件]{指令}' 文件
2. 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：

1. [root@svr5 ~]# awk 'BEGIN{A=24;print A\*2}'
2. [root@svr5 ~]# awk 'BEGIN{print x+1}' #x可以不定义，直接用，默认值位0
3. [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：

1. [root@svr5 ~]# awk 'BEGIN{x=0}/bash$/{x++} END{print x}' /etc/passwd
2. 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、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。

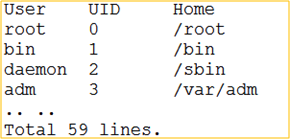


图-1

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：

1. [root@svr5 ~]# awk -F: 'BEGIN{print "User\tUID\tHome"} \
2. {print $1 "\t" $3 "\t" $6} \
3. END{print "Total",NR,"lines."}' /etc/passwd
4. User UID Home
5. root 0 /root
6. bin 1 /bin
7. daemon 2 /sbin
8. adm 3 /var/adm
9. 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：

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

输出包含root的行数据：

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

输出root或adm账户的用户名和UID信息：

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

输出账户名称包含root的基本信息（第1列包含root）：

1. [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：

1. [root@svr5 ~]# awk -F: '$7!~/nologin$/{print $1,$7}' /etc/passwd
2. root /bin/bash
3. sync /bin/sync
4. 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：

1. [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：

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

输出账户UID小于10的账户名称和UID信息：

1. [root@svr5 ~]# awk -F: '$3<10{print $1,$3}' /etc/passwd
2. root 0
3. bin 1
4. daemon 2
5. adm 3
6. lp 4
7. sync 5
8. shutdown 6
9. halt 7
10. mail 8

输出用户名为“root”的行：

1. [root@svr5 ~]# awk -F: '$1=="root"' /etc/passwd
2. 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：

1. [root@svr5 ~]# awk -F: '$3>10 && $3<20' /etc/passwd
2. operator:x:11:0:operator:/root:/sbin/nologin
3. games:x:12:100:games:/usr/games:/sbin/nologin
4. 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：

1. [root@svr5 ~]# awk -F: '$3>1000 || $3<10' /etc/passwd
2. root:x:0:0:root:/root:/bin/bash
3. bin:x:1:1:bin:/bin:/sbin/nologin
4. daemon:x:2:2:daemon:/sbin:/sbin/nologin
5. adm:x:3:4:adm:/var/adm:/sbin/nologin
6. lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
7. sync:x:5:0:sync:/sbin:/bin/sync
8. shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
9. 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

1. [root@svr5 ~]# awk 'BEGIN{x++;print x}'
2. 1
3. [root@svr5 ~]# awk 'BEGIN{x=8;print x+=2}'
4. 10
5. [root@svr5 ~]# awk 'BEGIN{x=8;x--;print x}'
6. 7
7. [root@svr5 ~]# awk 'BEGIN{print 2+3}'
8. 5
9. [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：

1. [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：

1. [root@svr5 ~]# awk '/^(127|192)/' /etc/hosts
2. 127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
3. 192.168.4.5 svr5.tarena.com svr5

3）列出100以内整数中7的倍数或是含7的数：

1. [root@svr5 ~]# seq 100 | awk '$1%7==0||$1~/7/'
2. 7
3. 14
4. 17
5. 21
6. 27
7. 28
8. 35
9. 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

IMG_258

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

1. [root@svr5 ~]# cat getupwd-awk.sh
2. #/bin/bash
3. A=$(awk -F: '/bash$/{print $1}' /etc/passwd)        ## 提取符合条件的账号记录
4. for i in $A
5. do
6. grep $i /etc/shadow | awk -F: '{print $1,"-->",$2}'
7. done

步骤三：验证、测试脚本

1. [root@svr5 ~]# ./getupwd-awk.sh
2. root --> $6$IWgMYmRACwdbfwBo$dr8Yn983nswiJVw0dTMjzbDvSLeCd1GMYjbvsDiFEkL8jnXOLcocBQypOCr4C6BRxNowIxjh6U2qeFU0u1LST/
3. zengye --> $6$Qb37LOdzRl5995PI$L0zTOgnhGz8ihWkW81J.5XhPp/l7x2./Me2ag0S8tRndCBL9nIjHIKkUKulHxJ6TXyHYmffbVgUT6pbSwf8O71
4. clamav --> !!
5. mysql --> !!
6. .. ..