

NSD SHELL DAY06

1. [案例1：使用awk提取文本](#)
2. [案例2：awk处理条件](#)
3. [案例3：awk综合脚本应用](#)
4. [案例4：awk流程控制](#)
5. [案例5：awk扩展应用](#)

1 案例1：使用awk提取文本

1.1 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

1.2 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
```

[Top](#)

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. ... ..
```

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

[Top](#)

```

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：

```

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 MB)
07.      RX errors 0 dropped 0 overruns 0 frame 0
08.      TX packets 51809 bytes 40788621 (38.8 MB)
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%    /

```

[Top](#)

输出上述结果中最后一行的第4列：

```
01. [ root@svr5 ~] # df -h | tail -1 | awk '{print $6}'
02. 11G
```

或者直接在awk中使用正则：

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

```
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}'
```

[Top](#)

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、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.
```

2 案例2：awk处理条件

2.1 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

[Top](#)

- 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

2.2 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
```

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
```

[Top](#)

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
```

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
```

[Top](#)

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
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. ... ..
```

[Top](#)

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
```

3) 列出100以内整数中7的倍数或是含7的数：

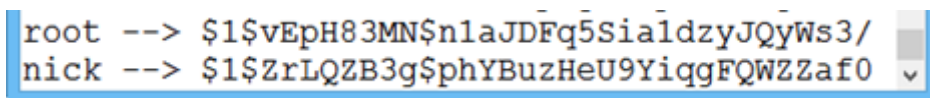
```
01. [ root@svr5 ~] # seq 100 | awk '$1%7==0 || $1~/7/'
02. 7
03. 14
04. 17
05. 21
06. 27
07. 28
08. 35
09. 37
10. 42
11. 47
12. ...
```

3 案例3：awk综合脚本应用

3.1 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$n1aJDfQ5Sia1dzyJQyWs3/
nick --> $1$ZrLQZB3g$phYBuzHeU9YiqgFQWZZaf0
```

图 - 2

3.2 步骤

[Top](#)

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}'
08. done
```

步骤三：验证、测试脚本

```
01. [ root@svr5 ~] # ./getupwd-awk.sh
02. root --> $6$IWgMYmRAOwdbfwBo$dr8Yn983nswiVwOdTMjzbDvSLeCd1GMYjbv sDiFEkL8jnXOLcocB
03. zengye --> $6$Qb37LOdzRI5995PI$LOzTOgnhGz8ihWkW81J.5XhPp/I7x2./Me2ag0S8tRndCBL9nljHlK
04. clamav --> !!
05. mysql --> !!
06. ....
```

4 案例4：awk流程控制

4.1 问题

本案例要求了解awk的流程控制操作，可自行设置awk语句来验证以下操作：

- if分支结构（单分支、双分支、多分支）
- 练习awk数组的使用

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：awk过滤中的if分支结构

1) 单分支

统计/etc/passwd文件中UID小于或等于1000的用户个数：

[Top](#)

```
01. [ root@svr5 ~] # awk - F: '{ if ( $3<=1000) { i++ } } END{ print i } ' /etc/passwd
02. 39
```

统计/etc/passwd文件中UID大于1000的用户个数：

```
01. [ root@svr5 ~] # awk - F: '{ if ( $3>1000) { i++ } } END{ print i } ' /etc/passwd
02. 8
```

统计/etc/passwd文件中登录Shell是 “/bin/bash” 的用户个数：

```
01. [ root@svr5 ~] # awk - F: '{ if ( $7~/bash$/) { i++ } } END{ print i } ' /etc/passwd
02. 29
```

2) 双分支

分别统计/etc/passwd文件中UID小于或等于1000、UID大于1000的用户个数：

```
01. [ root@svr5 ~] # awk - F: '{ if ( $3<=1000) { i++ } else{ j++ } } END{ print i,j } ' /etc/passwd
02. 39 8
```

分别统计/etc/passwd文件中登录Shell是 “/bin/bash” 、 登录Shell不是 “/bin/bash” 的用户个数：

```
01. [ root@svr5 ~] # awk - F: '{ if ( $7~/bash$/) { i++ } else{ j++ } } END{ print i,j } ' /etc/passwd
02. 29 38
```

步骤二：awk数组

1) 数组的语法格式

数组是一个可以存储多个值的变量，具体使用的格式如下：

定义数组的格式：数组名[下标]=元素值

调用数组的格式：数组名[下标]

遍历数组的用法：for(变量 in 数组名){print 数组名[变量]}。

```
01. [ root@svr5 ~] # awk 'BEGIN{ a[ 0 ] =11; a[ 1 ] =88; print a[ 1 ],a[ 0 ] } '
02. 88 11
```

[Top](#)

```

03. [ root@svr5 ~] # awk 'BEGIN{ a++; print a }'
04. 1
05. [ root@svr5 ~] # awk 'BEGIN{ a0++; print a0 }'
06. 1
07. [ root@svr5 ~] # awk 'BEGIN{ a[ 0] ++; print a[ 0] }'
08. 1
09. [ root@svr5 ~] # awk 'BEGIN{ a[ 0] =0; a[ 1] =11; a[ 2] =22; for( i in a){ print i,a[ i] } }'
10. 0 0
11. 1 11
12. 2 22

```

注意，awk数组的下标除了可以使用数字，也可以使用字符串，字符串需要使用双引号：

```

01. [ root@svr5 ~] # awk 'BEGIN{ a[ "hehe" ] =11; print a[ "hehe" ] }'
02. 11

```

5 案例5：awk扩展应用

5.1 问题

本案例要求使用awk工具完成下列两个任务：

- 分析Web日志的访问量排名，要求获得客户机的地址、访问次数，并且按照访问次数排名

5.2 方案

1) awk统计Web访问排名

在分析Web日志文件时，每条访问记录的第一列就是客户机的IP地址，其中会有很多重复的IP地址。因此只用awk提取出这一列是不够的，还需要统计重复记录的数量并且进行排序。

通过awk提取信息时，利用IP地址作为数组下标，每遇到一个重复值就将此数组元素递增1，最终就获得了这个IP地址出现的次数。

针对文本排序输出可以采用sort命令，相关的常见选项为-r、-n、-k。其中-n表示按数字顺序升序排列，而-r表示反序，-k可以指定按第几个字段来排序。

5.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：统计Web访问量排名

分步测试、验证效果如下所述。

1) 提取IP地址及访问量

```

01. [ root@svr5 ~] # awk '{ ip[ $1] ++ } \

```

[Top](#)

```
02. > END{ for(i in ip) { print ip[i],i } } ' /var/log/httpd/access_log
03. 4 127.0.0.1
04. 17 192.168.4.5
05. 13 192.168.4.110
06. ... ..
```

2) 对第1) 步的结果根据访问量排名

```
01. [ root@svr5 ~] # awk '{ ip[ $1 ] ++} END{ for( i in ip) { print i,ip[ i ] } } ' /var/log/httpd/access_log | s
02. 17 192.168.4.5
03. 13 192.168.4.110
04. 4 127.0.0.1
05. ... ..
```



[Top](#)