**[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201801/SHELL/DAY06/CASE/01/index.html" \l "page_top_case)**

# NSD SHELL DAY06

1. [案例1：使用awk提取文本](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201801/SHELL/DAY06/CASE/01/index.html" \l "case1)
2. [案例2：awk处理条件](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201801/SHELL/DAY06/CASE/01/index.html" \l "case2)
3. [案例3：awk综合脚本应用](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201801/SHELL/DAY06/CASE/01/index.html" \l "case3)
4. [案例4：awk流程控制](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201801/SHELL/DAY06/CASE/01/index.html" \l "case4)
5. [案例5：awk扩展应用](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201801/SHELL/DAY06/CASE/01/index.html" \l "case5)

## **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。

制表符：tab键

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. **...** **...**
5. **awk '/Accepted/{print $1,$2,$3,$11}' /var/log/secure**
6. [**root@svr5 ~]#**  **df | awk '/ \/$ /{print $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 (初始化)
* { } 逐行处理，读取文件过程中执行，指令执行n次 （数据处理）
* END{ } 行后处理，读取文件结束后执行，指令执行1次 （总结）

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。

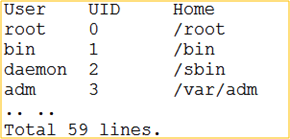


图-1

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：

awk -F: 'BEGIN{"User Uid Home"}{print $1,$3,$6}END{print "总:"NR}' /etc/passwd | column -t column -t(对齐)

1. **[**root@svr5 **~]**# awk **-**F**:** 'BEGIN{print "User**\t**UID**\t**Home"} \
2. {princolumn -t t $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**.**

## **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
* 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：

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）：

~正则匹配 1～只对第一列模糊匹配正则

若含有rootrx 则也会显示

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 (精确匹配)

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
13. 数学运算

X默认为0 字符为空

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. **..** **..**

## **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

image002

图－2

### **3.2 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. **..** **..**

## **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语句来有验证以下操作：

* 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：

1. **[**root@svr5 **~]**# awk **-**F**:** '{if($3<=1000){i++}}END{print i}' /etc/passwd
2. **39**

统计/etc/passwd文件中UID大于1000的用户个数：

1. **[**root@svr5 **~]**# awk **-**F**:** '{if($3>1000){i++}}END{print i}' /etc/passwd
2. **8**

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

1. **[**root@svr5 **~]**# awk **-**F**:** '{if($7~/bash$/){i++}}END{print i}' /etc/passwd
2. **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：

1. **[**root@svr5 **~]**# awk **-**F**:** '{if($3<=1000){i++}else{j++}}END{print i,j}' /etc/passwd
2. **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：

1. **[**root@svr5 **~]**# awk **-**F**:** '{if($7~/bash$/){i++}else{j++}} END{print i,j}' /etc/passwd
2. **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]}。

1. **[**root@svr5 **~]**# awk 'BEGIN{a[0]=11;a[1]=88;print a[1],a[0]}'
2. **88** **11**
3. **[**root@svr5 **~]**# awk 'BEGIN{a++;print a}'
4. **1**
5. **[**root@svr5 **~]**# awk 'BEGIN{a0++;print a0}'
6. **1**
7. **[**root@svr5 **~]**# awk 'BEGIN{a[0]++;print a[0]}'
8. **1**
9. **[**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：

1. **[**root@svr5 **~]**# awk 'BEGIN{a["hehe"]=11;print a["hehe"]}'
2. **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**

**虚拟构造访问用户 lili -c 100 -n 10000 http://172.25.0.11**

**-c 用户数量 -n 点击量**

分步测试、验证效果如下所述。

1）提取IP地址及访问量

1. **[**root@svr5 **~]**# awk '{ip[$1]++} \
2. > END{for(i in ip) {print ip[i],i }}' /var/log**/**httpd**/**access\_log
3. **4** **127.0.0.1**
4. **17** **192.168.4.5**
5. **13** **192.168.4.110**
6. **..** **..**

2）对第1）步的结果根据访问量排名

1. **[**root@svr5 **~]**# awk '{ip[$1]++} END{for(i in ip) {print i,ip[i]}}' /var/log**/**httpd**/**access\_log **|** sort **-**nr
2. **17** **192.168.4.5**
3. **13** **192.168.4.110**
4. **4** **127.0.0.1**
5. **..** **..**

who | awk '{ip[$1]++} END{for(i in ip) {print i,ip[i]}}'

who | awk '{ip[$5]++} END{for(i in ip) {print i,ip[i]}}'

**（无序）**

**统计公式：awk '{ip[$1]++} END{for(i in ip) {print i,ip[i]}}'**

**（排序）sort 按字符串比较 sort -n 按数字排序 r降序**

**统计公式：awk '{ip[$1]++} END{for(i in ip) {print i,ip[i]}}' | sort -n**