链滴

文本处理三剑客之 awk-9

作者：Carey

原文链接：https://ld246.com/article/1610184635863

来源网站：链滴

4 文本处理三剑客之awk

awk：Aho，Weinberger，Kernighan，报告生成器，格式化文本输出，GNU/Linux发布的AWK目由自由软件基金会（FSF）进行开发维护，通常也称为GNU AWK

版本：

- AWK：原先来源于AT&T实验室的AWK
- NAWK：AT&T实验室的AWK的升级
- GAWK：即GNU AWK。所有的GNU/Linux发布版都自带GAWK，他与AWK和NAWK完全兼容

gawk：模式扫描和处理语言，可以实现下面功能

- 文本处理
- 输出格式化的文本报表
- 执行算数运算
- 执行字符串操作

格式：

```
awk [options] 'program' var=value file...
awk [options] -f programfile var=value file...
```

说明：

program通常是被放在单引号中，并可以由三种部分组成

- **BEGIN语句块，文本没有读取时就执行**
- **模式匹配的通用块**

- **END语句块，文本内容执行完毕后执行**

常见选项：

- **-F “分隔符”指明输入时用到的字段分隔符，默认的分隔符是若干个连续空白符**
- **-v var=value 变量赋值**

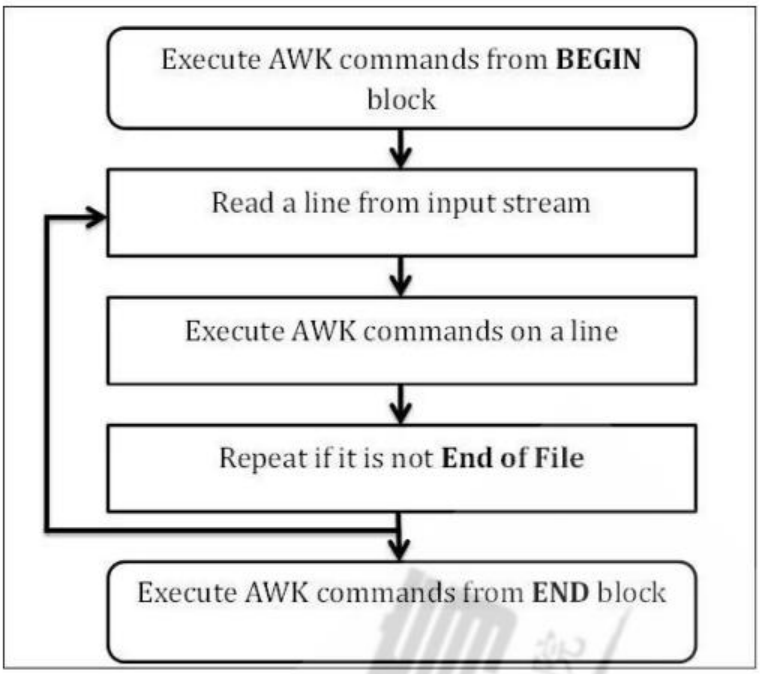
Program格式：

pattern{action statements;...}

pattern:决定动作语句何时触发即触发事件，比如：BEGIN，END，正则表达式等

action statements：对数据进行处理，放在{}内指明，常见：print，printf

awk工作过程：



第一步：执行BEGIN{action;...}语句块中的语句

第二步：从文件或标准输入（stdin）读取一行，然后执行pattern{ action;...}语句块，它逐行扫描文，从第一行到最后一行重复这个过程，直到文件全部被读取完毕。

第三步：当读至输入流末尾时执行END{action;...}语句块

BEGIN语句块在awk开始从输入流中读取行之前被执行，这是一个可选择的语句块，比如变量初始化打印输出表格的表头等语句通常可以写在BEGIN语句块中

END语句块在awk从输入流中读取完所有的行之后立即被执行，比如打印所有行的分析结果这类信息汇都是在END语句块中完成，它也是一个可选语句块

pattern语句块中的通用命令是最重要的部分，也是可选的。如果没有提供pattern语句块，则默认执行print}，即打印每一个读取到的行，awk读取的每一行都会执行该语句块

分割符、域和记录

- 由分隔符分割的字段（列column，域field）标记$1,$2.. n称为域标识，$0为所有域，注意：和shell中变量符含义不同

- 文件的每一行称为记录record
- 如果省略action，则默认执行print $0的操作

常用的action分类

- output statements：print，printf
- Expressions：算术，比较表达式等
- Compound statements：组合语句
- Control statements：if，while等
- input statements

awk控制语句

- {statements;...}组合语句
- if(condition){statements;...}
- if(condition){statements;...} else {statements;...}
- while(conditon){statements;...}
- do {statements;...} while(conditon)
- for(expr1;expr2;expr3){statements;...}
- break
- continue
- exit

4.1 动作print

格式：

print item1，item2，...

说明：

- 逗号分隔符
- 输出item可以字符串，也可是数值；当前记录的字段、变量或awk的表达式
- 如果省略item,相当于print $0
  - 固定字符需要用“”引起来，而变量和数字不需要

范例：

```
[18:01:30 root@centos8 boot]#awk '{print "hello,awk"}'
[19:02:55 root@centos8 boot]#seq 10 | awk '{print "hello,awk"}'
hello,awk
hello,awk
hello,awk
hello,awk
hello,awk
hello,awk
```

```
hello,awk
hello,awk
hello,awk
hello,awk
[19:03:50 root@centos8 boot]#seq 3 | awk '{print 2*3}'
6
6
6
[19:04:06 root@centos8 boot]#awk -F: '{print "zhang"}' /etc/passwd
[19:04:42 root@centos8 boot]#awk -F: '{print }' /etc/passwd
[19:05:00 root@centos8 boot]#awk -F: '{print $0}' /etc/passwd
[19:05:53 root@centos8 ~]#awk -F: '{print $1,$3}' /etc/passwd
[19:06:55 root@centos8 ~]#awk -F: '{print $1"\t"$3}' /etc/passwd

[19:07:12 root@centos8 ~]#grep "^UUID" /etc/fstab | awk '{print $2,$3}'
/boot ext4
```

面试题：取出网站访问量最大的前3个IP

```
[19:11:46 root@centos8 ~]#awk '{print $1}' nginx.access.log-20200428 |sort |uniq -c | sort -nr
head -3
5498 122.51.38.20
2161 117.157.173.214
953 211.159.177.120

[19:11:55 root@centos8 ~]#awk '{print $1}' nginx.access.log-20200428 |sort |uniq -c|sort -nr |
ead
5498 122.51.38.20
2161 117.157.173.214
953 211.159.177.120
219 58.87.87.99
100 222.218.17.189
100 218.201.62.71
100 122.139.5.237
100 120.195.144.116
100 118.121.41.14
100 1.177.191.161
```

面试题：取出分区利用率

```
[19:16:38 root@centos8 ~]#df | awk '{print $1,$5}'
Filesystem Use%
devtmpfs 0%
tmpfs 0%
tmpfs 3%
tmpfs 0%
/dev/mapper/cl-root 14%
/dev/sda1 14%
tmpfs 0%

#使用扩展正则表达式
[19:17:58 root@centos8 ~]#df | awk -F" +|%" '{print $1,$5}'
Filesystem Use
```

```
devtmpfs 0
tmpfs 0
tmpfs 3
tmpfs 0
/dev/mapper/cl-root 14
/dev/sda1 14
tmpfs 0
[19:19:51 root@centos8 ~]#df | grep "^/dev/" | awk -F" +|%" '{print $1,$5}'
/dev/mapper/cl-root 14
/dev/sda1 14
[19:20:35 root@centos8 ~]#df | awk -F" +|%" '/^\/dev\//{print $1,$5}'
/dev/mapper/cl-root 14
/dev/sda1 14
```

面试题：取nginx访问日志中IP和时间

```
[19:23:33 root@centos8 ~]#head -n 3 nginx.access.log-20200428
58.87.87.99 - - [27/Apr/2020:03:10:51 +0800] "POST /wp-cron.php?doing_wp_cron=1587928
51.003294944763183593750 0 HTTP/1.1" ""sendfileon
61.131.3.225 - - [27/Apr/2020:03:10:51 +0800] "GET / HTTP/1.1" ""sendfileon
157.245.106.153 - - [27/Apr/2020:03:10:52 +0800] "GET /wp-login.php HTTP/1.1" ""sendfileon
[19:23:56 root@centos8 ~]#awk -F"[[ ]" '{print $1,$5}' nginx.access.log-20200428 | head -3
58.87.87.99 27/Apr/2020:03:10:51
61.131.3.225 27/Apr/2020:03:10:51
157.245.106.153 27/Apr/2020:03:10:52
```

面试题：取ifconfig输出结果中的IP地址

```
[19:26:21 root@centos8 ~]#ifconfig eth0| awk '/netmask/{print $2}'
192.168.10.81
#这个使用变量的写法
[19:27:03 root@centos8 ~]#ifconfig eth0| awk 'NR==2{print $2}'
192.168.10.81
```

面试题：文件host_list.log如下格式，请提取“.magedu.com”前面的主机部分并写入到该文件中

```
[19:30:36 root@centos8 ~]#cat host_list.log
1 www.magedu.com
2 blog.magedu.com
3 study.magedu.com
4 linux.magedu.com
5 python.magedu.com
[19:31:25 root@centos8 ~]#awk -F"[. ]" '{print $2}' host_list.log
www
blog
study
linux
python
[19:31:28 root@centos8 ~]#awk -F"[. ]" '{print $2}' host_list.log >>host_list.log
[19:31:38 root@centos8 ~]#cat host_list.log
1 www.magedu.com
```

```
2 blog.magedu.com
3 study.magedu.com
4 linux.magedu.com
5 python.magedu.com
www
blog
study
linux
python
```

4.2 awk变量

awk中的变量分为：内置和自定义变量

4.2.1 常见的内置变量

- **FS：输入字段分隔符，默认为空白字符，功能相当于 -F**

范例：

```
#下面这四种写法输出的结果都差不多
[19:31:42 root@centos8 ~]#awk -v FS=':' '{print $1,FS,$3}' /etc/passwd
[19:34:37 root@centos8 ~]#awk -v FS=':' '{print $1FS$3}' /etc/passwd
[19:35:24 root@centos8 ~]#awk -F: '{print $1,$3,$7}' /etc/passwd
[19:36:22 root@centos8 ~]#S=:;awk -v FS=$S '{print $1FS$3}' /etc/passwd

#-F 和 FS变量功能一样，同时使用会冲突，排在后面的会生效
[09:01:27 root@centos8 ~]#awk -v FS=':' -F";" '{print $1FS$3}' /etc/passwd | head -n3
root:x:0:0:root:/root:/bin/bash;
bin:x:1:1:bin:/bin:/sbin/nologin;
daemon:x:2:2:daemon:/sbin:/sbin/nologin;
[09:01:47 root@centos8 ~]#awk -v FS=';' -F":" '{print $1FS$3}' /etc/passwd | head -n3
root:0
bin:1
daemon:2
```

- **OFS：输出字段分割符，默认为空白字符**

范例：

```
19:44:28 root@centos8 ~]#awk -v FS=":" '{print $1,$3,$7}' /etc/passwd | head -1
root 0 /bin/bash
[19:46:27 root@centos8 ~]#awk -v FS=":" -v OFS=':' '{print $1,$3,$7}' /etc/passwd | head -1
root:0:/bin/bash
```

- **RS：输入记录record分隔符，直到输入时的换行符**

范例：

```
[19:50:47 root@centos8 ~]#awk -v RS='' '{print $1,$2,$3}' /etc/passwd
root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin daemon:x:2:2:daemon:/sbin:/sbi
/nologin
```

- **ORS：输出记录分隔符，输出时用指定符号代替换行符**

范例：

```
[19:53:41 root@centos8 ~]#awk -v ORS='+++' -F':' '{print $1,$3,$7}' /etc/passwd | head -3
```

- **NF：字段数量**

范例：

```
#引用变量时，变量前不需加$
[19:55:02 root@centos8 ~]#awk -F: '{print NF}' /etc/passwd
7
[19:55:46 root@centos8 ~]#awk -F: '{print $(NF-1)}' /etc/passwd
/root
/bin

[19:58:58 root@centos8 Packages]#ls /mnt/AppStream/Packages/ | awk -F. '{print $(NF-1)}'|so
t|uniq -c
895 i686
1953 noarch
1 TRANS
2478 x86_64
```

面试题：主机连接数最多的前3个IP

```
[20:02:46 root@zhangzhuo ~]#ss -nta | awk -F" +|:" '{print $(NF-2)}' | sort |uniq -c |sort -nr | h
ad -n 3
5 172.18.0.2
5 127.0.0.1
5 127.0.0.1
[20:05:09 root@zhangzhuo ~]#ss -nta | awk -F" +|:" '/^ESTAB/{print $(NF-2)}' | sort |uniq -c |
ort -rn|head -n3
5 172.18.0.2
5 127.0.0.1]
5 127.0.0.1
```

范例：每十分钟检查将连接数超过100个以上的IP放入黑名单拒绝访问

```
[10:04:39 root@centos8 ~]#cat deny_dos.sh
LINK=100
ss -nt | awk -F"[[:space:]]+|:" '/^ESTAB/{print $(NF-2)}' | sort | uniq -c | while read count ip;do
if [ $count -gt $LINK ];then
iptables -A INPUT -s $ip -j REJECT
fi
done
[10:06:00 root@centos8 ~]#chmod +x /root/deny_dos.sh
[10:06:42 root@centos8 ~]#crontab -l
*/10 * * * * /root/deny_dos.sh
```

- **NR：记录的编号**

范例：

```
[20:06:59 root@centos8 Packages]#awk '{print NR,$0}' /etc/issue /etc/centos-release
1 \S
2 Kernel \r on an \m
3
4 CentOS Linux release 8.2.2004 (Core)
```

范例：取ifconfig输出结果中的IP地址

```
[10:12:16 root@centos8 ~]#ifconfig ens33 | awk '/netmask/{print $2}'
192.168.10.81
[10:12:40 root@centos8 ~]#ifconfig ens33 | awk 'NR==2{print $2}'
192.168.10.81
```

范例：

```
[10:13:15 root@centos8 ~]#awk -F: '{print NR}' /etc/passwd
1
2
3
.....
#文本内容结束后执行
[10:13:35 root@centos8 ~]#awk -F: 'END{print NR}' /etc/passwd
24
#文本内容开始前执行
[10:14:06 root@centos8 ~]#awk -F: 'BEGIN{print NR}' /etc/passwd
0
```

- **FNR：各文件分别计数，记录的编号**

范例：

```
[10:22:10 root@centos8 ~]#awk '{print NR,$0}' /etc/issue /etc/redhat-release
1 \S
2 Kernel \r on an \m
3
4 CentOS Linux release 8.2.2004 (Core)
[10:22:44 root@centos8 ~]#awk '{print FNR,$0}' /etc/issue /etc/redhat-release
1 \S
2 Kernel \r on an \m
3
1 CentOS Linux release 8.2.2004 (Core)
```

- **FILENAME：当前文件名**

范例：

```
[10:26:01 root@centos8 ~]#awk '{print FNR,FILENAME,$0}' /etc/issue
1 /etc/issue \S
2 /etc/issue Kernel \r on an \m
3 /etc/issue
[10:26:08 root@centos8 ~]#awk '{print FNR,FILENAME,$0}' /etc/issue /etc/redhat-release
1 /etc/issue \S
2 /etc/issue Kernel \r on an \m
3 /etc/issue
1 /etc/redhat-release CentOS Linux release 8.2.2004 (Core)
```

- **ARGC：命令参数的个数**

范例：

```
[10:26:24 root@centos8 ~]#awk '{print ARGC}' /etc/issue /etc/redhat-release
3
3
3
3
[10:27:34 root@centos8 ~]#awk 'BEGIN{print ARGC}' /etc/issue /etc/redhat-release
3
```

- **ARGV：数组，保存的是命令行所给定的各参数，每一个参数：ARGV[0],....**

范例：

```
[20:14:45 root@centos8 Packages]#awk 'BEGIN{print ARGV[0]}' /etc/issue /etc/centos-release
awk
[20:16:53 root@centos8 Packages]#awk 'BEGIN{print ARGV[1]}' /etc/issue /etc/centos-release
/etc/issue
[20:16:58 root@centos8 Packages]#awk 'BEGIN{print ARGV[2]}' /etc/issue /etc/centos-release
/etc/centos-release
[20:17:02 root@centos8 Packages]#awk 'BEGIN{print ARGV[3]}' /etc/issue /etc/centos-release
```

4.2.2 自定义变量

自定义变量是区分字符大小写的，使用下面方式进行赋值

- **-v var=value**
- **在program中直接定义**

范例：

```
[20:17:06 root@centos8 Packages]#awk -v test1=test2="hello,gawk" 'BEGIN{print test1,test2}'
test2=hello,gawk

[20:19:28 root@centos8 Packages]#awk -v test1=test2="hello1,gawk" 'BEGIN{test1=test2="he
lo2,gawk";print test1,test2}'
hello2,gawk hello2,gawk
#注意俩种方式的区别
```

范例：

```
[20:20:50 root@centos8 Packages]#awk -v test='hello gawk' '{print test}'  /etc/fstab
[20:22:37 root@centos8 Packages]#awk -v test='hello gawk' 'BEGIN{print test}'
[20:23:43 root@centos8 Packages]#awk 'BEGIN{test="hello,gawk";print test}'
[20:23:50 root@centos8 Packages]#awk -F: '{sex="male";print $1,sex,age;age=18}' /etc/passw

root male
bin male 18
daemon male 18

#脚本赋值引用
```

```
[20:27:58 root@centos8 ~]#cat awkscript
{print script,$1,$2}
[20:28:07 root@centos8 ~]#awk -F: -f awkscript script="awk" /etc/passwd
awk root x
awk bin x
```

4.3 动作printf

printf可以实现格式化输出

格式：

```
printf "FORMAT",item1,item2,...
```

说明：

- 必须指定FORMAT
- 不会自动换行，需要显示给出换行控制符\n
- FORMAT中需要分别为后面每个item指定格式符

格式符：与item一 一对应

```
%c：显示字符的ASCII码
%d,%i：显示十进制整数
%e,%E:显示科学计数法数值
%f：显示为浮点数
%g,%G：以科学计数法或浮点形式显示数值
%s：显示字符串
%%：显示%自身
```

修饰符

```
#[.#] 第一个数字控制显示的宽度；第二个#表示小数点后精度，如：%3.1f
- 左对齐（默认右对齐） 如：%-15%
+ 显示数值的正符号     如：%+d
```

范例：

```
[20:29:25 root@centos8 ~]#awk -F: '{printf "%s",$1}' /etc/passwd
[20:30:49 root@centos8 ~]#awk -F: '{printf "%s\n",$1}' /etc/passwd
root
bin
[20:30:54 root@centos8 ~]#awk -F: '{printf "+s\n",$1}' /etc/passwd
root
bin
[20:31:11 root@centos8 ~]#awk -F: '{printf "%-20s\n",$1}' /etc/passwd
root
bin
[20:31:49 root@centos8 ~]#awk -F: '{printf "%-20s %10d\n",$1,$3}' /etc/passwd
root                    0
bin                     1
[20:32:22 root@centos8 ~]#awk -F: '{printf "Username: %s\n",$1}' /etc/passwd
Username: root
```

```
Username: bin
[20:34:05 root@centos8 ~]#awk -F: '{printf "Username: %sUID:%d\n",$1,$3}' /etc/passwd
Username: rootUID:0
Username: binUID:1
[20:34:16 root@centos8 ~]#awk -F: '{printf "Username: %25sUID:%d\n",$1,$3}' /etc/passwd
Username:                      rootUID:0
Username:                       binUID:1
[20:34:54 root@centos8 ~]#awk -F: '{printf "Username: %-25sUID:%d\n",$1,$3}' /etc/passwd
Username: root                      UID:0
Username: bin                       UID:1
```

4.4 操作符

4.4.1 算数操作符：

x+y，x-y,x*y,x/y,x^2,x%y
-x:转换为负数
+x将字符串转换为数值

字符串操作符：没有符号的操作符，字符串连接

赋值操作符：

=, +=, -=, *=, /=, %=, ^=，++, --

范例：

```
[20:35:23 root@centos8 ~]#awk 'BEGIN{i=0;print i++,i}'
0 1
[20:48:34 root@centos8 ~]#awk 'BEGIN{i=0;print ++i,i}'
1 1
```

范例：'0{print "1"}'，' ""{print "1"}'表示假后面的不打印，其余为真打印，'n++'等于'n{print $0}表示打印所有行

```
[20:48:41 root@centos8 ~]#seq 10 | awk 'n++'
2
3
4
5
6
7
8
9
10
[20:49:49 root@centos8 ~]#awk -v n=0 '!n++' /etc/passwd
root:x:0:0:root:/root:/bin/bash
[20:49:54 root@centos8 ~]#awk -v n=0 '!n++{print n}' /etc/passwd
1
[20:50:26 root@centos8 ~]#awk -v n=1 '!n++{print n}' /etc/passwd
[20:51:02 root@centos8 ~]#awk -v n=0 '!++n{print n}' /etc/passwd

[20:51:06 root@centos8 ~]#awk -v n=0 '!++n' /etc/passwd
```

```
[20:52:04 root@centos8 ~]#awk -v n=-1 '!++n' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

4.4.2 比较操作符：

==, !=, >, >=, <, <=

范例：

```
[20:55:40 root@centos8 ~]#awk 'NR==2' /etc/issue
Kernel \r on an \m
[20:57:28 root@centos8 ~]#awk -F: '$3>=1000' /etc/passwd
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
zhang:x:1000:1000::/home/zhang:/bin/bash
```

范例：取奇，偶数行

```
[20:58:32 root@centos8 ~]#seq 10 | awk 'NR%2==0'
2
4
6
8
10
[20:58:38 root@centos8 ~]#seq 10 | awk 'NR%2==1'
1
3
5
7
9
[20:58:53 root@centos8 ~]#seq 10 | awk 'NR%2!=0'
1
3
5
7
9
```

4.4.3 模式匹配符：

~ 左边是否和右边匹配，包含关系
!~ 是否不匹配

范例：

```
[09:14:01 root@centos8 ~]#awk -F: '$0 ~ /root/{print $1}' /etc/passwd
root
operator
[09:14:34 root@centos8 ~]#awk -F: '$0 ~ "^root"{print $1}' /etc/passwd
root
[09:15:24 root@centos8 ~]#awk '$0 !~ /root/' /etc/passwd
[09:15:27 root@centos8 ~]#awk '/root/' /etc/passwd
[09:16:10 root@centos8 ~]#awk -F: '/r/' /etc/passwd
[09:16:47 root@centos8 ~]#awk -F: '$3==0' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

```
[09:16:51 root@centos8 ~]#df | awk -F" +|%" '$0 ~ /\/dev\//{print $5}'
0
14
14
[09:18:15 root@centos8 ~]#ifconfig eth0 | awk 'NR==2{print $2}'
192.168.10.81
```

4.4.4 逻辑操作符：

与:&& 并且关系
或:|| 或者关系
非:！取反

范例：！取反

```
[09:18:50 root@centos8 ~]#awk 'BEGIN{print i}'

[09:21:30 root@centos8 ~]#awk 'BEGIN{print !i}'
1
[09:21:39 root@centos8 ~]#awk -v i=10 'BEGIN{print !i}'
0
[09:21:52 root@centos8 ~]#awk -v i=-3 'BEGIN{print !i}'
0
[09:22:00 root@centos8 ~]#awk -v i=0 'BEGIN{print !i}'
1
[09:22:11 root@centos8 ~]#awk -v i=abc 'BEGIN{print !i}'
0
[09:22:15 root@centos8 ~]#awk -v i='' 'BEGIN{print !i}'
1
```

范例：

```
[09:23:25 root@centos8 ~]#awk -F: '$3>=0 && $3<=1000{print $1,$3}' /etc/passwd
root 0
bin 1
[09:23:29 root@centos8 ~]#awk -F: '$3==0 || $3>=1000{print $1,$3}' /etc/passwd
root 0
nobody 65534
zhang 1000
[09:25:11 root@centos8 ~]#awk -F: '!($3==0){print $1,$3}' /etc/passwd
bin 1
daemon 2
[09:26:24 root@centos8 ~]#awk -F: '!($3>=500){print $1,$3}' /etc/passwd
root 0
bin 1
```

4.4.5 条件表达式(三目表达式)

selector?if-true-expression:if-false-expression

范例：

```
[09:26:28 root@centos8 ~]#awk -F: '{$3>=1000?usertype="Common User":usertype="Sysuse
";printf "%-20s:%12s\n",$1,usertype}' /etc/passwd
root                 :     Sysuser
[09:35:19 root@centos8 ~]#df | awk -F"[ %]+" '/\/dev\//{$(NF-1)>10?disk="full":disk="OK";pr
nt $(NF-1),disk}'
0 OK
14 full
14 full
```

4.5 模式PATTERN

PATTERN：根据pattern条件，过滤匹配的行，在做处理

- **如果未指定：空模式，匹配每一行**

范例：

```
[09:35:43 root@centos8 ~]#awk -F: '{print $1,$3}' /etc/passwd
```

- /regular expression/:仅处理能够模式匹配到的行，需要用/ /括起来

范例

```
[09:39:26 root@centos8 ~]#awk '/^UUID/{print $1}' /etc/fstab
UUID=df5fee56-9dd8-4814-bed7-6449e315bae7
[09:41:11 root@centos8 ~]#awk '!/^UUID/{print $1}' /etc/fstab
[09:41:23 root@centos8 ~]#df | awk '/^\/dev\//'
/dev/mapper/cl-root   17G  2.3G   15G  14% /
/dev/sda1            976M  124M  786M  14% /boot
```

- relational expression:关系表达式，结果为“真”才会被处理
  - 真：结果为非0，非空字符串
  - 假：结果为空字符串或0值

范例：

```
[09:41:46 root@centos8 ~]#seq 10 | awk '1'
1
2
3
4
5
6
7
8
9
10
[09:44:26 root@centos8 ~]#seq 10 | awk '0'
[09:44:31 root@centos8 ~]#seq 10 | awk '"false"'
1
2
3
```

```
4
5
6
7
8
9
10
[09:44:41 root@centos8 ~]#seq 10 | awk '""'
[09:44:48 root@centos8 ~]#seq 10 | awk '"0"'
1
2
3
4
5
6
7
8
9
10
[09:44:53 root@centos8 ~]#seq 10 | awk 'true'    这里是当成变量了因为没有赋值所以为空假
[09:45:06 root@centos8 ~]#seq 10 | awk 'false'
[09:45:28 root@centos8 ~]#seq 10 | awk '0'
[09:45:35 root@centos8 ~]#seq 10 | awk '""'
[09:45:37 root@centos8 ~]#seq 10 | awk '" "'
1
2
3
4
5
6
7
8
9
10
[09:47:51 root@centos8 ~]#seq 10 | awk -v magedu=0 'magedu'
[09:47:58 root@centos8 ~]#seq 10 | awk -v magedu="0" 'magedu'
[09:48:11 root@centos8 ~]#seq 10 | awk -v magedu="" 'magedu'
[09:48:16 root@centos8 ~]#seq 10 | awk -v magedu="a" 'magedu'
1
2
3
4
5
6
7
8
9
10
```

范例：

```
[09:49:25 root@centos8 ~]#awk '1' /etc/passwd
[09:49:23 root@centos8 ~]#awk '!1' /etc/passwd
[09:49:27 root@centos8 ~]#awk '!0' /etc/passwd
```

范例

```
[09:50:46 root@centos8 ~]#seq 10 | awk 'i=0'
[09:51:48 root@centos8 ~]#seq 10 | awk 'i=1'
1
2
3
4
5
6
7
8
9
10
[09:51:50 root@centos8 ~]#seq 10 | awk 'i=!i'
1
3
5
7
9
[09:52:00 root@centos8 ~]#seq 10 | awk '{i=!i;print i}'
1
0
1
0
1
0
1
0
1
0
[09:52:15 root@centos8 ~]#seq 10 | awk '!(i=!i)'
2
4
6
8
10
[09:52:34 root@centos8 ~]#seq 10 | awk -v i=1 'i=!i'
2
4
6
8
10
```

范例：

```
[09:57:14 root@centos8 ~]#awk -F: 'i=1;j=1{print i,j}' /etc/passwd
[09:57:40 root@centos8 ~]#awk -F: '$NF=="/bin/bash"{print $1,$NF}' /etc/passwd
root /bin/bash
zhang /bin/bash
[09:58:47 root@centos8 ~]#awk -F: '$NF ~ /bash$/{print $1,$NF}' /etc/passwd
root /bin/bash
zhang /bin/bash
```

- line ranges:行范围

- 不支持直接用行号，但可以使用变量NR间接指定行号
  - /pat1/,/pat2/ 不支持直接给出数字格式

范例：

```
[09:59:24 root@centos8 ~]#seq 10 | awk 'NR>=3 && NR<=6'
3
4
5
6
[10:01:46 root@centos8 ~]#awk 'NR>=3 && NR<=6{print NR,$0}' /etc/passwd
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
6 sync:x:5:0:sync:/sbin:/bin/sync
[10:02:27 root@centos8 ~]#sed -n '3,6p' /etc/passwd
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
sync:x:5:0:sync:/sbin:/bin/sync
[10:02:50 root@centos8 ~]#awk '/^bin/,/^adm/' /etc/passwd
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
[10:03:12 root@centos8 ~]#sed -n '/^bin/,/^adm/p' /etc/passwd
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
```

- BEGIN/END模式
  - BEGIN{}:仅在开始处理文件中的文本之前执行一次
  - END{}:仅在文本处理完成之后执行一次

范例：

```
[10:03:51 root@centos8 ~]#awk -F: 'BEGIN{print "USER USERID"} {print $1":"$3} END{print "E
D FILE"}' /etc/passwd
USER USERID
root:0
[10:08:31 root@centos8 ~]#awk -F: '{print "USER USERID";print $1":"$3} END{print "END FILE"}
 /etc/passwd

[10:18:14 root@centos8 ~]#awk -F: 'BEGIN{printf "--------------------------------\n|%-20s|%1
s|\n--------------------------------\n","username","uid"}{printf "|%-20s|%10d|\n",$1,$3}END{pr
ntf "--------------------------------\n"}' /etc/passwd
```

4.6 条件判断if-else

语法：

```
#双分支判断
```

```
if(condition){statement;...}[else statement]
#多分支判断
if(condition1){statement1}else if(condition2){statement2}else if(condition3){statement3}..... els
 {statementN}
```

使用场景：对awk取得的整数行或某个字段做条件判断

范例：

```
[10:18:18 root@centos8 ~]#awk -F: '{if($3>=1000)print $1,$3}' /etc/passwd
[10:23:17 root@centos8 ~]#awk -F: '{if($NF=="/bin/bash")print $1,$3}' /etc/passwd
[10:23:35 root@centos8 ~]#awk '{if(NF>5)print $0}' /etc/fstab
#下面是俩种格式一样的功能
[10:30:38 root@centos8 ~]#awk -F: '{if($3>=1000) {printf "Common user: %s\n",$1} else{printf
"root or Sysuser: %s\n",$1}}' /etc/passwd
[10:30:38 root@centos8 ~]#awk -F: '{if($3>=1000) printf "Common user: %s\n",$1; else printf
root or Sysuser: %s\n",$1}' /etc/passwd

[10:35:49 root@centos8 ~]#df -h | awk -F% '/^\/dev\//{print $1}' | awk '$NF>=10{print $1,$5}'
[10:37:01 root@centos8 ~]#df | awk -F"[ %]+" '/^\/dev\//{if($5>10)print $1,$5}

[10:39:42 root@centos8 ~]#awk 'BEGIN{test=58;if(test>90){print "very good"}else if(test>60){
rint "good"}else{print "no pass"}}'
```

4.7 条件判断 switch

语法：

功能：类似于shell中的case判断语句

```
switch(expression) {case VALUE1 or /REGEXP/: statement1; case VALUE2 or /REGEXP2/: state
ent2; ...; default: statementn}
```

4.8 循环 while

语法：

```
while (condition) {statement;...}
```

条件“真”，进入循环；条件“假”，退出循环

使用场景：

- **对一行内的多个字段逐一类似处理时使用**
- **对数组中的各元素逐一处理时使用**

范例：

```
[10:56:45 root@centos8 ~]#awk -v i=1 -v sum=0 'BEGIN{while(i<=100){sum+=i;i++};print su
}'
5050
#使用awk要比shell中循环求和速度要快
[11:03:59 root@centos8 ~]#time (awk -v i=1 -v sum=0 'BEGIN{while(i<=1000000){sum+=i;i+
```

```
};print sum}')
500000500000

real    0m0.062s
user    0m0.059s
sys    0m0.002s
[11:04:32 root@centos8 ~]#time (./1000000sum.sh)
500000500000

real    0m4.441s
user    0m4.406s
sys    0m0.001s
```

示例：

```
#内置函数length()返回字符数，而非字节数
[11:06:54 root@centos8 ~]#awk 'BEGIN{print length("hello")}'
5
[11:07:07 root@centos8 ~]#awk 'BEGIN{print length("张卓")}'
2
[11:14:41 root@centos7 ~]#awk '/^[[:space:]]*linux16/{i=1;while(i<=NF){print $i,length($i);i++
}' /etc/grub2.cfg
linux16 7
/vmlinuz-3.10.0-1127.el7.x86_64 31
root=/dev/mapper/centos-root 28
ro 2
spectre_v2=retpoline 20
rd.lvm.lv=centos/root 21
rd.lvm.lv=centos/swap 21
rhgb 4
quiet 5
net.ifnames=0 13
biosdevname=0 13
linux16 7
/vmlinuz-0-rescue-89e3a70685b441a0a1e8546f09fcc342 50
root=/dev/mapper/centos-root 28
ro 2
spectre_v2=retpoline 20
rd.lvm.lv=centos/root 21
rd.lvm.lv=centos/swap 21
rhgb 4
quiet 5
net.ifnames=0 13
biosdevname=0 13

[11:18:02 root@centos7 ~]#awk '/^[[:space:]]*linux16/{i=1;while(i<=NF){if(length($i)>=10){pr
nt $i,length($i)};i++}}' /etc/grub2.cfg
/vmlinuz-3.10.0-1127.el7.x86_64 31
root=/dev/mapper/centos-root 28
spectre_v2=retpoline 20
rd.lvm.lv=centos/root 21
rd.lvm.lv=centos/swap 21
net.ifnames=0 13
biosdevname=0 13
```

```
/vmlinuz-0-rescue-89e3a70685b441a0a1e8546f09fcc342 50
root=/dev/mapper/centos-root 28
spectre_v2=retpoline 20
rd.lvm.lv=centos/root 21
rd.lvm.lv=centos/swap 21
net.ifnames=0 13
biosdevname=0 13
```

4.9 循环do-while

语法：

```
do {statement;...}while(condition)
```

意义：无论真假，至少执行一次循环体

范例：

```
[11:22:34 root@centos8 ~]#awk 'BEGIN{ total=0;i=1;do{total+=i;i++;}while(i<=100);print total
'
5050
```

4.10 循环for

语法：

```
for(expr1;expr2;expr3) {statement;...}
```

常见用法：

```
for(variable assignment;condition;iteration process) {for-body}
```

特殊用法：能够遍历数组中的元素

```
for(var in array) {for-body}
```

范例：

```
[11:22:49 root@centos8 ~]#awk 'BEGIN{sum=0;for(i=1;i<=100;i++){sum+=i};print sum}'
5050
[11:25:14 root@centos8 ~]#for((i=1,sum=0;i<=100;i++));do let sum+=i;done ;echo $sum
5050
```

范例：

```
[11:30:21 root@centos7 ~]#awk '/^[[:space:]]*linux16/{for(i=1;i<=NF;i++){if(length($i)>10){pr
nt $i,length($i)}}}' /etc/grub2.cfg
/vmlinuz-3.10.0-1127.el7.x86_64 31
root=/dev/mapper/centos-root 28
spectre_v2=retpoline 20
rd.lvm.lv=centos/root 21
rd.lvm.lv=centos/swap 21
net.ifnames=0 13
biosdevname=0 13
```

```
/vmlinuz-0-rescue-89e3a70685b441a0a1e8546f09fcc342 50
root=/dev/mapper/centos-root 28
spectre_v2=retpoline 20
rd.lvm.lv=centos/root 21
rd.lvm.lv=centos/swap 21
net.ifnames=0 13
biosdevname=0 13
```

4.11 continue和break

- **continue 中断本次循环**
- **break 中断整个循环格**

格式:

```
continue [n]
break [n]
```

范例:

```
[11:26:14 root@centos8 ~]#awk 'BEGIN{for(i=1;i<=100;i++){if(i==50)continue;sum+=i};print
um}'
5000
[11:33:27 root@centos8 ~]#awk 'BEGIN{for(i=1;i<=100;i++){if(i==50)break;sum+=i};print sum
'
1225
[11:33:41 root@centos8 ~]#awk 'BEGIN{for(i=1;i<=100;i++){if(i%2==0)continue;sum+=i};print
sum}'
2500
```

4.12 next

next 可以提前结束对本行处理而直接进入下一行处理(awk自身循环)

范例:

```
[11:37:09 root@centos8 ~]#awk -F: '{if($3%2!=0)next;print$1,$3}' /etc/passwd
root 0
daemon 2
lp 4
shutdown 6
mail 8
games 12
ftp 14
nobody 65534
polkitd 998
sssd 996
sshd 74
saslauth 994
rpc 32
tcpdump 72
zhang 1000
pcp 992
```

```
apache 48
cockpit-wsinstance 990
说明:本来条件是处理uid是奇数的行，假next之后不处理这行跳过直接打印下一行
```

4.13 数组

awk的数组为关联数组

格式：

```
array_name[index-expression]
```

范例：

```
weekdays["mon"]="Monday"
```

index-expression

- 利用数组，实现 k/v 功能
- 可使用任意字符串；字符串要使用双引号括起来
- 如果某数组元素事先不存在，在引用时，awk会自动创建此元素，并将其值初始化为“空串”
- 若要判断数组中是否存在某元素，要使用“index in array”格式进行遍历

范例：

```
[11:37:16 root@centos8 ~]#awk 'BEGIN{weekdays["mon"]="Monday";weekdays["tue"]="Tues
ay";print weekdays["mon"]}'
Monday
```

范例：很难理解多看下

```
[11:44:17 root@centos8 ~]#awk '!line[$0]++' dupfile
1
2
3
4
5
[11:44:21 root@centos8 ~]#awk '{print !line[$0]++,$0,line[$0]}' dupfile
1 1 1
1 2 1
1 3 1
1 4 1
1 5 1
[11:45:52 root@centos8 ~]#awk '{!line[$0]++;print $0,line[$0]}' dupfile
1 1
2 1
3 1
4 1
5 1
[11:47:19 root@centos8 ~]#cat dupfile
1
2
3
```

```
4
5
```

范例：判断数组索引是否存在

```
[11:48:06 root@centos8 ~]#awk 'BEGIN{array["i"]="x";array["j"]="y";print "i" in array,"y" in arr
y}'
1 0
[11:49:57 root@centos8 ~]#awk 'BEGIN{array["i"]="x";array["j"]="y";if("i" in array){print "存在"}
lse{print "不存在"}}'
存在
[11:51:58 root@centos8 ~]#awk 'BEGIN{array["i"]="x";array["j"]="y";if("a" in array){print "存在"
else{print "不存在"}}'
不存在
```

4.13.1 若要遍历数组中每个元素，要使用for循环

```
for(var in array) {for-body}
```

注意：var会遍历array的每个索引

范例：遍历数组

```
[11:54:23 root@centos8 ~]#awk 'BEGIN{zhang["mon"]="Monday";zhang["tue"]="Tuesday";for
i in zhang){print i,zhang[i]}}'
tue Tuesday
mon Monday

[11:56:13 root@centos8 ~]#awk 'BEGIN{students[1]="daizong";students[2]="junzong";studen
s[3]="xiaohong";for(x in students){print x,students[x]}}'
1 daizong
2 junzong
3 xiaohong

[12:01:43 root@centos8 ~]#awk 'BEGIN{ a["x"] = "welcome"
a["y"] = "to"
a["z"] = "zhangzhuo"
for (i in a){
print i,a[i]
}
}'
x welcome
y to
z zhangzhuo

[12:01:48 root@centos8 ~]#awk -F: '{user[$1]=$3}END{for(i in user){print "username: "i,"uid: "
ser[i]}}' /etc/passwd
username: tcpdump uid: 72
username: rpc uid: 32
username: sshd uid: 74
```

范例：显示主机的连接状态出现的次数

```
[14:08:30 root@centos8 ~]#ss -nta |awk 'NR!=1{print $1}' |sort |uniq -c
```

```
1 ESTAB
7 LISTEN
[14:10:20 root@centos8 ~]#ss -nta |awk 'NR!=1{state[$1]++}END{for(i in state){print i,state[i]}}

LISTEN 7
ESTAB 1
```

范例：查看nginx服务访问次数

```
[14:15:05 root@centos8 ~]#awk '{ip[$1]++}END{for(i in ip){print ip[i],i}}' access.log
[14:15:05 root@centos8 ~]#awk '{ip[$1]++}END{for(i in ip){print ip[i],i}}' access.log | sort -nr |
ead  -3
1586 114.246.83.58
1424 222.131.157.24
1092 123.123.104.28
[14:18:31 root@centos8 ~]#awk '{ip[$1]++}END{for(i in ip){print i,ip[i]}}' access.log | sort -k2rn
 head  -3
114.246.83.58 1586
222.131.157.24 1424
123.123.104.28 1092
```

范例：封掉查看访问日志中连接次数超过1000次的IP

```
[14:23:58 root@centos8 ~]#awk '{ip[$1]++}END{for(i in ip){if(ip[i]>=1000){system("iptables -A
NPUT -s "i" -j REJECT")}}}' access.log
```

范例：多维数组，遍历的话也得遍历2次

```
[14:24:36 root@centos8 ~]#awk 'BEGIN{

> array[1][1]=11
> array[1][2]=12
> array[1][3]=13
> for (i in array)
> for (j in array[i])
> print array[i][j]
> }'
> 11
> 12
> 13
```

范例：

```
[14:35:49 root@centos8 ~]#cat score.txt
name sex score
alice f 100
bob   m 90
ming  m 95
hong  f 90
[14:35:53 root@centos8 ~]#awk 'NR!=1{if($2=="m"){m_sum+=$3;m_num++}else{f_sum+=$3;
_num++}}END{print "男 生平均成绩="m_sum/m_num,"女生平均成绩="f_sum/f_num}' score.txt
男生平均成绩=92.5 女生平均成绩=95

[14:35:54 root@centos8 ~]#awk 'NR!=1{score[$2]+=$3;num[$2]++}END{for(i in score){print i,
core[i]/num[i]}}' score.txt
```

```
m 92.5
f 95

[14:40:15 root@centos8 ~]#awk 'NR!=1{num[$2]+=$3;sum[$2]++}END{for(i in num){if(i=="m
){print "男生平均成绩="num[i]/sum[i]}else{print "女生平均成绩="num[i]/sum[i]}}}' score.txt
男生平均成绩=92.5
女生平均成绩=95
```

4.14 awk函数

awk的函数分为内置和自定义函数

官方文档

https://www.gnu.org/software/gawk/manual/gawk.html#Functions

4.14.1 常见内置函数

- 数值处理：

```
rand()：返回0和1之间一个随机数
srand()：配合rand() 函数,生成随机数的种子
int()：返回整数
```

范例：

```
[14:42:24 root@centos8 ~]#awk 'BEGIN{srand();print rand()}'
0.534234
[14:45:31 root@centos8 ~]#awk 'BEGIN{srand();print rand()}'
0.642982
[14:45:33 root@centos8 ~]#awk 'BEGIN{srand();print rand()}'
0.741868
[14:46:41 root@centos8 ~]#awk 'BEGIN{srand();for (i=1;i<=10;i++)print int(rand()*100)}'
47
85
85
32
99
29
83
84
85
45
```

- **字符串处理：**

```
length([s])：返回指定字符串的长度
sub(r,s,[t])：对t字符串搜索r表示模式匹配的内容，并将第一个匹配内容替换为s
gsub(r,s,[t])：对t字符串进行搜索r表示的模式匹配的内容，并全部替换为s所表示的内容
split(s,array,[r])：以r为分隔符，切割字符串s，并将切割后的结果保存至array所表示的数组中，第一
索引值为1,第二个索引值为2,...
```

范例：统计用户名的长度

```
[14:46:44 root@centos8 ~]#cut -d: -f1 /etc/passwd | awk '{print length()}'
[14:49:03 root@centos8 ~]#awk -F: '{print length($1)}' /etc/passwd
```

范例：替换字符串字符，替换的是第一个

```
[14:49:56 root@centos8 ~]#echo "2008:08:08 08:08:08" | awk 'sub(/:/,"-",$1)'
2008-08:08 08:08:08
[14:51:08 root@centos8 ~]#echo "2008:08:08 08:08:08" | awk '{sub(/:/,"-",$1);print $0}'
2008-08:08 08:08:08
```

范例：指定字符串全局替换

```
[14:51:20 root@centos8 ~]#echo "2008:08:08 08:08:08" | awk 'gsub(/:/,"-",$1)'
2008-08-08 08:08:08
[14:51:56 root@centos8 ~]#echo "2008:08:08 08:08:08" | awk '{gsub(/:/,"-",$1);print $0}'
2008-08-08 08:08:08
```

范例：

```
[14:53:32 root@centos8 ~]#netstat -tn | awk '/^tcp/{split($5,ip,":");count[ip[1]]++}END{for(i in
count){print i,count[i]}}'
192.168.10.1 1
```

- 可以awk中调用shell命令

```
system('cmd')
```

空格是awk中的字符串连接符，如果system中需要使用awk中的变量可以使用空格分隔，或者说了awk的变量外其他一律用""引用起来

```
[14:54:58 root@centos8 ~]#awk 'BEGIN{system("hostname")}'
centos8
[14:57:28 root@centos8 ~]#awk 'BEGIN{score=100;system("echo your score is "score"")}'
your score is 100

[15:01:54 root@centos8 ~]#netstat -nt | awk '/^tcp/{split($5,ip,":");count[ip[1]]++}END{for(i in
count){if(count[i]>=10){system("iptables -A INPUT -s "i" -j REJECT")}}}'
```

- **时间函数：**

官方文档：时间函数

```
https://www.gnu.org/software/gawk/manual/gawk.html#Time-Functions

systime()  当前时间到1970年1月1日的秒数
strftime() 指定时间格式
```

范例：

```
[15:07:03 root@centos8 ~]#awk 'BEGIN{print strftime("%Y-%m-%dT%H:%M")}'
2021-01-07T15:07
[15:07:05 root@centos8 ~]#awk 'BEGIN{print systime()}'
1610003233
#前一个小时
[15:18:00 root@centos8 ~]#awk 'BEGIN{print strftime("%Y-%m-%dT%H:%M",systime()-3600)}'
```

2021-01-07T14:21

4.14.2 自定义函数

自定义函数格式：

```
function name ( parameter, parameter,... )
{ statements
return expression
}
```

范例：

```
[15:10:23 root@centos8 ~]#cat func.awk
function max(x,y) {
x>y?var=x:var=y
return var
}
BEGIN{print max(a,b)}
[15:10:27 root@centos8 ~]#awk -v a=80 -v b=40 -f func.awk
80
```

4.15 awk脚本

将awk程序写成脚本，直接调用或执行

范例：

```
[15:12:54 root@centos8 ~]#cat passwd.awk
#! /bin/awk -f
{if($3>=1000)print $1,$3}
[15:12:59 root@centos8 ~]#awk -F: -f passwd.awk /etc/passwd
nobody 65534
zhang 1000

[15:14:57 root@centos8 ~]#cat test.awk
#! /bin/awk -f
{if($3>=1000)print $1,$3}
[15:15:02 root@centos8 ~]#chmod +x test.awk
[15:15:05 root@centos8 ~]#./test.awk -F: /etc/passwd
nobody 65534
zhang 1000
```

向awk脚本传递参数

格式：

```
awkfile  var=value  var2=value2... Inputfile
```

注意：在BEGIN过程中不可用。直到首行输入完成以后，变量才可用。可以通过-v 参数，让awk在执行BEGIN之前得到变量的值。命令行中每一个指定的变量都需要一个-v参数

范例：

```
[15:17:28 root@centos8 ~]#cat test2.awk
#! /bin/awk -f
{if($3>=min && $3<=max)print $1,$3}
[15:17:34 root@centos8 ~]#chmod +x test2.awk
[15:17:39 root@centos8 ~]#./test2.awk -F: min=100 max=200 /etc/passwd
systemd-resolve 193
```

练习：

1. 文件host_list.log 如下格式，请提取”.magedu.com”前面的主机名部分并写入到回到该文件中

```
1  www.magedu.com
2  blog.magedu.com
3  study.magedu.com
4  linux.magedu.com
5  python.magedu.com
.....
999 study.magedu.com

[16:04:41 root@centos8 ~]#awk -F"[ .]" '{print $2}' host_list.log >host_list.log
```

2. 统计/etc/fstab文件中每个文件系统类型出现的次数

```
[16:08:36 root@centos8 ~]#awk '!/^#/ && !/^$/{type[$3]++}END{for(i in type){print i,type[i]} }
 /etc/fstab
swap 1
ext4 1
xfs 1
```

3. 统计/etc/fstab文件中每个单词出现的次数

```
[16:41:14 root@centos8 ~]#grep -oE "[[:alpha:]]+" /etc/fstab | awk '{num[$1]++}END{for(i in
um)print i,num[i]}'
```

4. 提取出字符串Yd$C@M05MB%9&Bdh7dq+YVixp3vpw中的所有数字

```
[16:57:25 root@centos8 ~]#echo "Yd$C@M05MB%9&Bdh7dq+YVixp3vpw" | awk -F"[^0-9 ]+
 '{for(i=1;i<=NF;i++){printf "%s",$i}}'
05973
```

5. 有一文件记录了1-100000之间随机的整数共5000个，存储的格式100,50,35,89...请取出其中最大最小的整数

```
#先生成文件
[16:57:36 root@centos8 ~]#awk  'BEGIN{for(i=1;i<=5000;i++){OSF=",";printf "%s,",int(rand()*
000000)}}' >num.bak
#取最大最小
[17:06:38 root@centos8 ~]#awk -F"," -v min=1000000 '{for(i=1;i<=5000;i++){if($i>max){max
$i}else if($i
```

6. 解决Dos攻击生产案例：根据web日志或者或者网络连接数，监控当某个IP并发连接数或者短时内P达到100，即调用防火墙命令封掉对应的IP，监控频率每隔5分钟。防火墙命令为：iptables -AINPUT s IP -j REJECT

```
[17:25:15 root@centos8 ~]#ss -nt | awk -F"[ :]+" 'NR>1{ip[$6]++}END{for(i in ip){if(ip[i]>100){
ystem("iptables -A INPUT -s "i" -j REJECT")}}}'

[17:30:28 root@centos8 ~]#crontab -l
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
*/5 * * * * ss -nt | awk -F"[ :]+" 'NR>1{ip[$6]++}END{for(i in ip){if(ip[i]>100){system("iptables -
 INPUT -s "i" -j REJECT")}}}
```

7. 将以下文件内容中FQDN取出并根据其进行计数从高到低排序

```
http://mail.magedu.com/index.html
http://www.magedu.com/test.html
http://study.magedu.com/index.html
http://blog.magedu.com/index.html
http://www.magedu.com/images/logo.jpg
http://blog.magedu.com/20080102.html
http://www.magedu.com/images/magedu.jpg

[17:35:11 root@centos7 ~]#awk -F/ '{num[$3]++}END{for(i in num){print i,num[i]}}' a |sort -k2
r
www.magedu.com 3
blog.magedu.com 2
mail.magedu.com 1
study.magedu.com 1
```

8. 将以下文本文件awktest.txt中 以inode列为标记，对inode列相同的counts列进行累加，并且统计同一inode中，beginnumber列中的最小值和endnumber列中的最大值

```
inode|beginnumber|endnumber|counts|
106|3363120000|3363129999|10000|
106|3368560000|3368579999|20000|
310|3337000000|3337000100|101|
310|3342950000|3342959999|10000|
310|3362120960|3362120961|2|
311|3313460102|3313469999|9898|
311|3313470000|3313499999|30000|
311|3362120962|3362120963|2|

[17:56:58 root@centos7 ~]#awk -F"|" 'NR>1{num[$1]+=$4;if(!begin[$1])begin[$1]=$2;else if(
egin[$1]>$2)begin[$1]=$2;if(!end[$1])end[$1]=$3;else if(end[$1]<$3)end[$1]=$3}END{for(i in
num)print i"|"begin[i]"|"end[i]"|"sum[i]}' awktest.txt
310|3337000000|3362120961|
311|3313460102|3362120963|
106|3363120000|3368579999|
```