awk也是一个数据处理工具，sed其实是以行为单位的文本处理工具，而awk则是基于列的文本处理工具，它的工作方式是按行读取文本并视为一条记录，每条记录以字段分割成若干字段，然后输出各字段的值。awk语言的最基本功能是在文件或字符串中基于指定规则来分解抽取信息，也可以基于指定的规则来输出数据。其实他更像一门编程语言，他可以自定义变量，有条件语句，有循环，有数组，有正则，有函数等。

awk认为文件都是结构化的，也就是说都是由单词和各种空白字符组成的，这里的“空白字符”包括空格、Tab，以及连续的空格和Tab等。每个非空白的部分叫做“域”，从左到右依次是第一个域、第二个域，等等。$1、$2分别用于表示域，$0则表示全部域。

有三种方式调用awk

1．命令行方式

awk [-F field-separator] 'commands' input-files

其中，[-F域分隔符]是可选的，因为awk使用空格或tab键作为缺省的域分隔符，因此如果要浏览域间有空格的文本，不必指定这个选项，如果要浏览诸如passwd文件，此文件各域以冒号作为分隔符，则必须指明-F选项，如：awk -F: 'commands' input-file。

注:在linux系统中用环境变量IFS存储分隔符,但根据实际应用也可以改变IFS的值.

例如:

```
data="name,sex,rollno,location"
oldIFS=$IFS
IFS=","
for item in $data
do
  echo item: $item
done
IFS=$oldIFS
```

脚本执行结果如下:

```
item: name
item: sex
item: rollno
item: location
```

commands 是真正awk命令， input-files 是待处理的文件。

iput_files可以是多于一个文件的文件列表，awk将按顺序处理列表中的每个文件。

在awk中，文件的每一行中，由域分隔符分开的每一项称为一个域。通常，在不指名-F域分隔符的情况下，默认的域分隔符是空格或tab键。

2．shell脚本方式

将所有的awk命令插入一个文件，并使awk程序可执行，然后awk命令解释器作为脚本的首行，以便通过键入脚本名称来调用。

相当于shell脚本首行的：#!/bin/sh可以换成：#!/bin/awk

3．将所有的awk命令插入一个单独文件，然后调用：

awk -f awk-script-file input-files

其中，-f选项加载awk-script-file中的awk脚本，input-files跟上面的是一样的。

awk的模式和动作

任何awk语句都由模式和动作组成（awk_pattern { actions }）。

在一个awk脚本中可能有许多语句。

模式部分决定动作语句何时触发及触发事件。处理即对数据进行的操作。如果省略模式部分，动作将时刻保持执行状态。即省略时不对输入记录进行匹配比较就执行相应的actions。

模式可以是任何条件语句或正则表达式等。awk_pattern可以是以下几种类型:

1) 正则表达式用作awk_pattern: /regexp/

例如:awk '/ ^[a-z]/' input_file

2) 布尔表达式用作awk_pattern，表达式成立时，触发相应的actions执行。

① 表达式中可以使用变量(如字段变量$1,$2等)和/regexp/

② 布尔表达式中的操作符:

关系操作符: < > <= >= == !=

匹配操作符: value ~ /regexp/ 如果value匹配/regexp/，则返回真

value !~ /regexp/ 如果value不匹配/regexp/，则返回真

例如: awk '$2 > 10 {print "ok"}' input_file

awk '$3 ~ /^d/ {print "ok"}' input_file

③ &&(与) 和 ||(或) 可以连接两个/regexp/或者布尔表达式，构成混合表达式。!(非)可以用于布尔表达式或者/regexp/之前。

例如: awk '($1 < 10) && ($2 > 10) {print "ok"}' input_file

awk '/^d/ || /x$/ {print "ok"}' input_file

模式包括两个特殊字段 BEGIN和END。使用BEGIN语句设置计数和打印头。BEGIN语句使用在任何文本浏览动作之前，之后文本浏览动作依据输入文本开始执行。END语句用来在awk完成文本浏览动作后打印输出文本总数和结尾状态标志。

实际动作在大括号{ }内指明。动作大多数用来打印，但是还有些更长的代码诸如if和循环语句及循环退出结构。如果不指明采取动作，awk将打印出所有浏览出来的记录。

awk执行时，其浏览域标记为$1，$2...$n。这种方法称为域标识。使用这些域标识将更容易对域进行进一步处理。使用$1，$3表示参照第1和第3域，注意这里用逗号做域分隔。如果希望打印一个有5个域的记录的所有域，不必指明$1，$2，$3，$4，$5，可使用$0，意即所有域。为打印一个域或所有域，使用print命令。这是一个awk动作

awk的运行过程:

① 如果BEGIN 区块存在，awk执行它指定的actions。

② awk从输入文件中读取一行，称为一条输入记录。(如果输入文件省略，将从标准输入读取)

③ awk将读入的记录分割成字段，将第1个字段放入变量$1中，第2个字段放入$2，以此类推。$0表示整条记录。

④ 把当前输入记录依次与每一个awk_cmd中awk_pattern比较，看是否匹配，如果相匹配，就执行对应的actions。如果不匹配，就跳过对应的actions，直到比较完所有的awk_cmd。

⑤ 当一条输入记录比较了所有的awk_cmd后，awk读取输入的下一行，继续重复步骤③和④，这个过程一直持续，直到awk读取到文件尾。

⑥ 当awk读完所有的输入行后，如果存在END，就执行相应的actions。

入门实例:

例1：显示/etc/passwd文件中的用户名和登录shell

```
[root@vcloud ~]# awk -F : '{print $1,$7}' /etc/passwd
```

如果只是显示/etc/passwd的账户和账户对应的shell,而账户与shell之间以tab键分割

```
[root@vcloud ~]# awk -F : '{print $1 "\t" $7}' /etc/passwd
```

如果只是显示/etc/passwd文件中的用户名和登录shell, 而账户与shell之间以逗号分割

```
[root@vcloud ~]# awk -F : '{print $1 "," $7}' /etc/passwd
```

注：awk的总是输出到标准输出，如果想让awk输出到文件，可以使用重定向。

例2：显示/etc/passwd文件中的UID大于500的所有用户的用户名和登录shell

```
[root@vcloud ~]# awk -F : '$3>500 {print $1,$7}' /etc/passwd
nfsnobody /sbin/nologin
vcloud /sbin/nologin
monitor /bin/bash
```

例3：如果只是显示/etc/passwd文件中的UID大于500的用户名和登录shell,而账户与shell之间以逗号分割,而且在所有行添加列名name,shell,在最后一行添加"blue,/bin/nosh"。

```
[root@vcloud ~]# awk -F : 'BEGIN {print "name,shell"} $3>500 {print $1","$7} END
 {print "blue,/bin/bash"}' /etc/passwd
name,shell
nfsnobody,/sbin/nologin
vcloud,/sbin/nologin
monitor,/bin/bash
blue,/bin/bash
```

注:

1.awk 后面接两个单引号并加上大括号 {} 来设定想要对数据进行的处理动作

2.awk工作流程是这样的：先执行BEGING，然后读取文件，读入有\n换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0则表示所有域,$1表示第一个域,$n表示第n个域,随后开始执行模式所对应的动作。接着开始读入第二条记录……直到所有的记录都读完，最后执行END操作。

思考题：如何打印所有记录（以/etc/passwd中的内容为例）

```
[root@vcloud ~]# awk '{print $0}' /etc/passwd
```

例4：搜索/etc/passwd有root关键字的所有行

```
[root@vcloud ~]# awk -F : '/root/' /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

这种是pattern（模式）的使用示例，匹配了pattern(这里是root)的行才会执行action(没有指定action，默认输出每行的内容)。

搜索支持正则表达式，例如找root开头的:

```
[root@vcloud ~]# awk -F : '/^root/' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

搜索/etc/passwd有root关键字的所有行，并显示对应的shell

```
[root@vcloud ~]# awk -F : '/root/ {print $7}' /etc/passwd
```

这里指定了action是{print $7}

例5：显示最近登录系统的5个用户信息，只显示用户名和IP地址

使用last命令可以查看最近登录的用户信息。如下图所示:

```
[root@vcloud ~]# last -n 5
root     pts/0        192.168.10.144   Fri Feb 15 20:07   still logged in
root     tty1                          Fri Feb 15 20:02   still logged in
reboot   system boot  2.6.32-71.el6.x8 Fri Feb 15 20:01 - 20:07  (00:05)
root     pts/1        vcloud.benet.com Fri Feb 15 14:09 - 14:09  (00:00)
root     pts/0        :0.0             Fri Feb 15 13:16 - down   (01:40)
```

使用awk命令抽取用户名和IP区域的数据

```
[root@vcloud ~]# last -n 5 | awk '{print $1,$3}'
root 192.168.10.144
```

或

```
[root@vcloud ~]# last -n 5 | awk '{print $1 "\t" $3}'
root    192.168.10.144
```

awk内置变量

awk有许多内置变量用来设置环境信息，下面给出了最常用的一些变量。

FILENAME awk浏览的文件名

FS 设置输入域分隔符，等价于命令行-F选项

NF 浏览记录的域个数（每一行（$0）拥有的字段总数）

NR 已读的记录数（awk所处理的是第几行数据）

例6：统计/etc/passwd:文件名，每行的行号，每行的列数，对应的完整行内容:

```
[root@vcloud ~]# awk -F : '$3>500 {print "filename:" FILENAME ",linenumber:" NR
",columns:" NF ",linecontent:"$0}' /etc/passwd
filename:/etc/passwd,linenumber:32,columns:7,linecontent:nfsnobody:x:65534:65534
:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
filename:/etc/passwd,linenumber:38,columns:7,linecontent:vcloud:x:501:501::/opt/
vmware/vcloud-director:/sbin/nologin
filename:/etc/passwd,linenumber:39,columns:7,linecontent:monitor:x:502:502::/hom
e/monitor:/bin/bash
```

显示所有账户的记录，并带有其记录号，并在END部分打印输入文件名

```
[root@vcloud ~]# awk -F : '{print NF,NR,$0} END {print FILENAME}' /etc/passwd _
```

除了awk的内置变量，awk还可以自定义变量

例7：统计/etc/passwd的账户人数

```
[root@vcloud ~]# awk '{count++;print $0;} END {print "user count is:",count}' /e
tc/passwd_
```

count是自定义变量。之前的action{}里都是只有一个print,其实print只是一个语句，而action{}可以有多个语句，以;号隔开。

这里没有初始化count，虽然默认是0，但是妥当的做法还是初始化为0:

```
[root@vcloud ~]# awk 'BEGIN {count=0; print "[start]user count is:", count} {cou
nt=count+1;print $0;} END {print "[end]user count is:",count}' /etc/passwd_
```

例8：统计某个文件夹下的文件占用的字节数

```
[root@vcloud ~]# ls -l /etc/ | awk 'BEGIN {size=0;} {size=size+$5;} END {print "
[end]size is:", size}'
```

如果以M为单位显示:

```
[root@vcloud ~]# ls -l /etc/ | awk 'BEGIN {size=0;} {size=size+$5;} END {print "
[end]size is:", size/1024/1024,"M"}'
```

注意：以上统计没有包括子目录中的文件。

如果想快速查看所有文件的长度及其总和，但要排除子目录，如何实现：

```
[root@server2 ~]# ls -l /aa
total 16
drwxr-xr-x 2 root root 4096 Jul 15 08:04 grub
-rw------- 1 root root  819 Jul 15 08:04 grub.conf
-rw-r--r-- 1 root root  198 Jul 15 08:03 hosts
-rw-r--r-- 1 root root 2005 Jul 15 08:03 passwd
[root@server2 ~]# ls -l /aa/ | awk '/^[^d]/ {print $9 "\t" $5;sum+=$5} END {prin
t "total KB:" sum}'

grub.conf       819
hosts   198
passwd  2005
total KB:3022
[root@server2 ~]#
```