awk基本用法 awk高级应用

awk是一个强大的文本分析工具，相对于grep的查找，sed的编辑，awk在其对数据分析并生成报告时，显得尤为强大。简单来说awk就是把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。

awk有3个不同版本: awk、nawk和gawk，未作特别说明，一般指gawk，gawk 是 AWK 的 GNU 版本。

awk其名称得自于它的创始人 Alfred Aho 、Peter Weinberger 和 Brian Kernighan 姓氏的首个字母。实际上 AWK 的确拥有自己的语言： AWK 程序设计语言 ， 三位创建者已将它正式定义为“样式扫描和处理语言”。它允许您创建简短的程序，这些程序读取输入文件、为数据排序、处理数据、对输入执行计算以及生成报表，还有无数其他的功能。

**awk是行处理器**: 相比较屏幕处理的优点，在处理庞大文件时不会出现内存溢出或是处理缓慢的问题，通常用来格式化文本信息

**awk处理过程:** 依次对每一行进行处理，然后输出

**awk命令形式:**

awk [-F|-f|-v] ‘BEGIN{} //{command1; command2} END{}’ file

 [-F|-f|-v]   大参数，-F指定分隔符，-f调用脚本，-v定义变量 var=value

'  '          引用代码块

BEGIN   初始化代码块，在对每一行进行处理之前，初始化代码，主要是引用全局变量，设置FS分隔符

//           匹配代码块，可以是字符串或正则表达式

{}           命令代码块，包含一条或多条命令

；          多条命令使用分号分隔

END      结尾代码块，在对每一行进行处理之后再执行的代码块，主要是进行最终计算或输出结尾摘要信息

**特殊要点:**

$0           表示整个当前行

$1           每行第一个字段

NF          字段数量变量

NR          每行的记录号，多文件记录递增

FNR        与NR类似，不过多文件记录不递增，每个文件都从1开始

\t            制表符

\n           换行符

FS          BEGIN时定义分隔符

RS       输入的记录分隔符， 默认为换行符(即文本是按一行一行输入)

~            匹配，与==相比不是精确比较

!~           不匹配，不精确比较

==         等于，必须全部相等，精确比较

!=           不等于，精确比较

&&　     逻辑与

||             逻辑或

+            匹配时表示1个或1个以上

/[0-9][0-9]+/   两个或两个以上数字

/[0-9][0-9]\*/    一个或一个以上数字

FILENAME 文件名

OFS      输出字段分隔符， 默认也是空格，可以改为制表符等

ORS        输出的记录分隔符，默认为换行符,即处理结果也是一行一行输出到屏幕

-F'[:#/]'   定义三个分隔符

**print & $0**

print 是awk打印指定内容的主要命令

awk '{print}'  /etc/passwd   ==   awk '{print $0}'  /etc/passwd

awk '{print " "}' /etc/passwd                                           //不输出passwd的内容，而是输出相同个数的空行，进一步解释了awk是一行一行处理文本

awk '{print "a"}'   /etc/passwd                                        //输出相同个数的a行，一行只有一个a字母

awk -F":" '{print $1}'  /etc/passwd

awk -F: '{print $1; print $2}'   /etc/passwd                   //将每一行的前二个字段，分行输出，进一步理解一行一行处理文本

awk  -F: '{print $1,$3,$6}' OFS="\t" /etc/passwd        //输出字段1,3,6，以制表符作为分隔符

**-f指定脚本文件**

awk -f script.awk  file

BEGIN{

FS=":"

}

{print $1}               //效果与awk -F":" '{print $1}'相同,只是分隔符使用FS在代码自身中指定

awk 'BEGIN{X=0} /^$/{ X+=1 } END{print "I find",X,"blank lines."}' test

I find 4 blank lines.

 ls -l|awk 'BEGIN{sum=0} !/^d/{sum+=$5} END{print "total size is",sum}'                    //计算文件大小

total size is 17487

**-F指定分隔符**

$1 指指定分隔符后，第一个字段，$3第三个字段， \t是制表符

一个或多个连续的空格或制表符看做一个定界符，即多个空格看做一个空格

awk -F":" '{print $1}'  /etc/passwd

awk -F":" '{print $1 $3}'  /etc/passwd                       //$1与$3相连输出，不分隔

awk -F":" '{print $1,$3}'  /etc/passwd                       //多了一个逗号，$1与$3使用空格分隔

awk -F":" '{print $1 " " $3}'  /etc/passwd                  //$1与$3之间手动添加空格分隔

awk -F":" '{print "Username:" $1 "\t\t Uid:" $3 }' /etc/passwd       //自定义输出

awk -F: '{print NF}' /etc/passwd                                //显示每行有多少字段

awk -F: '{print $NF}' /etc/passwd                              //将每行第NF个字段的值打印出来

 awk -F: 'NF==4 {print }' /etc/passwd                       //显示只有4个字段的行

awk -F: 'NF>2{print $0}' /etc/passwd                       //显示每行字段数量大于2的行

awk '{print NR,$0}' /etc/passwd                                 //输出每行的行号

awk -F: '{print NR,NF,$NF,"\t",$0}' /etc/passwd      //依次打印行号，字段数，最后字段值，制表符，每行内容

awk -F: 'NR==5{print}'  /etc/passwd                         //显示第5行

awk -F: 'NR==5 || NR==6{print}'  /etc/passwd       //显示第5行和第6行

route -n|awk 'NR!=1{print}'                                       //不显示第一行

**//匹配代码块**

**//纯字符匹配   !//纯字符不匹配   ~//字段值匹配    !~//字段值不匹配   ~/a1|a2/字段值匹配a1或a2**

awk '/mysql/' /etc/passwd

awk '/mysql/{print }' /etc/passwd

awk '/mysql/{print $0}' /etc/passwd                   //三条指令结果一样

awk '!/mysql/{print $0}' /etc/passwd                  //输出不匹配mysql的行

awk '/mysql|mail/{print}' /etc/passwd

awk '!/mysql|mail/{print}' /etc/passwd

awk -F: '/mail/,/mysql/{print}' /etc/passwd         //区间匹配

awk '/[2][7][7]\*/{print $0}' /etc/passwd               //匹配包含27为数字开头的行，如27，277，2777...

awk -F: '$1~/mail/{print $1}' /etc/passwd           //$1匹配指定内容才显示

awk -F: '{if($1~/mail/) print $1}' /etc/passwd     //与上面相同

awk -F: '$1!~/mail/{print $1}' /etc/passwd          //不匹配

awk -F: '$1!~/mail|mysql/{print $1}' /etc/passwd

**IF语句**

**必须用在{}中，且比较内容用()扩起来**

awk -F: '{if($1~/mail/) print $1}' /etc/passwd                                       //简写

awk -F: '{if($1~/mail/) {print $1}}'  /etc/passwd                                   //全写

awk -F: '{if($1~/mail/) {print $1} else {print $2}}' /etc/passwd            //if...else...

**条件表达式**

**==   !=   >   >=**

awk -F":" '$1=="mysql"{print $3}' /etc/passwd

awk -F":" '{if($1=="mysql") print $3}' /etc/passwd          //与上面相同

awk -F":" '$1!="mysql"{print $3}' /etc/passwd                 //不等于

awk -F":" '$3>1000{print $3}' /etc/passwd                      //大于

awk -F":" '$3>=100{print $3}' /etc/passwd                     //大于等于

awk -F":" '$3<1{print $3}' /etc/passwd                            //小于

awk -F":" '$3<=1{print $3}' /etc/passwd                         //小于等于

**逻辑运算符**

**&&　||**

awk -F: '$1~/mail/ && $3>8 {print }' /etc/passwd         //逻辑与，$1匹配mail，并且$3>8

awk -F: '{if($1~/mail/ && $3>8) print }' /etc/passwd

awk -F: '$1~/mail/ || $3>1000 {print }' /etc/passwd       //逻辑或

awk -F: '{if($1~/mail/ || $3>1000) print }' /etc/passwd

**数值运算**

awk -F: '$3 > 100' /etc/passwd

awk -F: '$3 > 100 || $3 < 5' /etc/passwd

awk -F: '$3+$4 > 200' /etc/passwd

awk -F: '/mysql|mail/{print $3+10}' /etc/passwd                    //第三个字段加10打印

awk -F: '/mysql/{print $3-$4}' /etc/passwd                             //减法

awk -F: '/mysql/{print $3\*$4}' /etc/passwd                             //求乘积

awk '/MemFree/{print $2/1024}' /proc/meminfo                  //除法

awk '/MemFree/{print int($2/1024)}' /proc/meminfo           //取整

**输出分隔符OFS**

awk '$6 ~ /FIN/ || NR==1 {print NR,$4,$5,$6}' OFS="\t" netstat.txt

awk '$6 ~ /WAIT/ || NR==1 {print NR,$4,$5,$6}' OFS="\t" netstat.txt

//输出字段6匹配WAIT的行，其中输出每行行号，字段4，5,6，并使用制表符分割字段

**输出处理结果到文件**

①在命令代码块中直接输出    route -n|awk 'NR!=1{print > "./fs"}'

②使用重定向进行输出           route -n|awk 'NR!=1{print}'  > ./fs

**格式化输出**

netstat -anp|awk '{printf "%-8s %-8s %-10s\n",$1,$2,$3}'

printf表示格式输出

%格式化输出分隔符

-8长度为8个字符

s表示字符串类型

打印每行前三个字段，指定第一个字段输出字符串类型(长度为8)，第二个字段输出字符串类型(长度为8),

第三个字段输出字符串类型(长度为10)

netstat -anp|awk '$6=="LISTEN" || NR==1 {printf "%-10s %-10s %-10s \n",$1,$2,$3}'

netstat -anp|awk '$6=="LISTEN" || NR==1 {printf "%-3s %-10s %-10s %-10s \n",NR,$1,$2,$3}'

**IF语句**

awk -F: '{if($3>100) print "large"; else print "small"}' /etc/passwd

small

small

small

large

small

small

awk -F: 'BEGIN{A=0;B=0} {if($3>100) {A++; print "large"} else {B++; print "small"}} END{print A,"\t",B}' /etc/passwd

                                                                                                                  //ID大于100,A加1，否则B加1

awk -F: '{if($3<100) next; else print}' /etc/passwd                         //小于100跳过，否则显示

awk -F: 'BEGIN{i=1} {if(i<NF) print NR,NF,i++ }' /etc/passwd

awk -F: 'BEGIN{i=1} {if(i<NF) {print NR,NF} i++ }' /etc/passwd

另一种形式

awk -F: '{print ($3>100 ? "yes":"no")}'  /etc/passwd

awk -F: '{print ($3>100 ? $3":\tyes":$3":\tno")}'  /etc/passwd

**while语句**

awk -F: 'BEGIN{i=1} {while(i<NF) print NF,$i,i++}' /etc/passwd

7 root 1

7 x 2

7 0 3

7 0 4

7 root 5

7 /root 6

**数组**

netstat -anp|awk 'NR!=1{a[$6]++} END{for (i in a) print i,"\t",a[i]}'

netstat -anp|awk 'NR!=1{a[$6]++} END{for (i in a) printf "%-20s %-10s %-5s \n", i,"\t",a[i]}'

9523                               1

9929                               1

LISTEN                            6

7903                               1

3038/cupsd                   1

7913                               1

10837                             1

9833                               1

**应用1**

awk -F: '{print NF}' helloworld.sh                                                       //输出文件每行有多少字段

awk -F: '{print $1,$2,$3,$4,$5}' helloworld.sh                                 //输出前5个字段

awk -F: '{print $1,$2,$3,$4,$5}' OFS='\t' helloworld.sh                 //输出前5个字段并使用制表符分隔输出

awk -F: '{print NR,$1,$2,$3,$4,$5}' OFS='\t' helloworld.sh           //制表符分隔输出前5个字段，并打印行号

**应用2**

awk -F'[:#]' '{print NF}'  helloworld.sh                                                  //指定多个分隔符: #，输出每行多少字段

awk -F'[:#]' '{print $1,$2,$3,$4,$5,$6,$7}' OFS='\t' helloworld.sh   //制表符分隔输出多字段

**应用3**

awk -F'[:#/]' '{print NF}' helloworld.sh                                               //指定三个分隔符，并输出每行字段数

awk -F'[:#/]' '{print $1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12}' helloworld.sh     //制表符分隔输出多字段

**应用4**

计算/home目录下，普通文件的大小，使用KB作为单位

ls -l|awk 'BEGIN{sum=0} !/^d/{sum+=$5} END{print "total size is:",sum/1024,"KB"}'

ls -l|awk 'BEGIN{sum=0} !/^d/{sum+=$5} END{print "total size is:",int(sum/1024),"KB"}'         //int是取整的意思

**应用5**

统计netstat -anp 状态为LISTEN和CONNECT的连接数量分别是多少

netstat -anp|awk '$6~/LISTEN|CONNECTED/{sum[$6]++} END{for (i in sum) printf "%-10s %-6s %-3s \n", i," ",sum[i]}'

**应用6**

统计/home目录下不同用户的普通文件的总数是多少？

ls -l|awk 'NR!=1 && !/^d/{sum[$3]++} END{for (i in sum) printf "%-6s %-5s %-3s \n",i," ",sum[i]}'

mysql        199

root           374

统计/home目录下不同用户的普通文件的大小总size是多少？

ls -l|awk 'NR!=1 && !/^d/{sum[$3]+=$5} END{for (i in sum) printf "%-6s %-5s %-3s %-2s \n",i," ",sum[i]/1024/1024,"MB"}'

**应用7**

输出成绩表

awk 'BEGIN{math=0;eng=0;com=0;printf "Lineno.   Name    No.    Math   English   Computer    Total\n";printf "------------------------------------------------------------\n"}{math+=$3; eng+=$4; com+=$5;printf "%-8s %-7s %-7s %-7s %-9s %-10s %-7s \n",NR,$1,$2,$3,$4,$5,$3+$4+$5} END{printf "------------------------------------------------------------\n";printf "%-24s %-7s %-9s %-20s \n","Total:",math,eng,com;printf "%-24s %-7s %-9s %-20s \n","Avg:",math/NR,eng/NR,com/NR}' test0

[root@localhost home]# cat test0

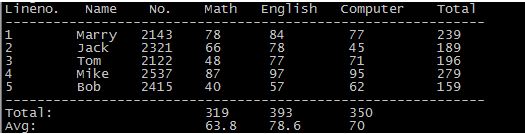
Marry   2143 78 84 77

Jack    2321 66 78 45

Tom     2122 48 77 71

Mike    2537 87 97 95

Bob     2415 40 57 62



**案例：awk综合脚本应用**

**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

图－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. **..** **..**

**awk使用shell变量及shell使用awk中的变量**

在写shell脚本时，经常会使用到awk程序。但是有些复杂的逻辑，可能需要在awk中使用在shell中定义的变量，而且awk程序处理之后，产生的中间变量，还需要在shell中继续处理。

**一、那如何在awk中使用在shell中定义的变量呢？**

**方法一：**使用"'把shell变量包起来，即"'$var'"；注意是“双引号+单引号+shell变量+单引号+双引号”的格式。

这种写法大家无需改变用'括起awk程序的习惯,是老外常用的写法；这种写法其实际是双括号变为单括号的常量,传递给了awk.

例如：

var="abc"

awk '*BEGIN*{print "'$var'"}'

**方法二：**和方法一类似，但使用"'"把shell变量包起来，即"'"$var"'"；注意是“双引号+单引号+双引号+shell变量+双引号+单引号+双引号”的格式。

如果变量的值中包含空格，为了shell不把空格作为分隔符，则应使用方法二。

例如：

var="this a test"

awk '*BEGIN*{print "'"$var"'"}'

**方法三：**export变量，然后在awk中使用ENVIRON["var"]形式获取环境变量的值

例如：

var="this a test"; export var;

awk '*BEGIN*{print ENVIRON["var"]}'  

**方法四：**使用awk -v选项；

当变量不是很多时，建议使用这种方式。

例如：

var="this a test"

awk -v awkVar="$var" '*BEGIN*{print awkVar}'

**二、那shell中如何使用awk程序中定义的变量呢？**

“由awk向shell传递变量”，其思想无非是用awk(sed/perl等也是一样)输出若干条shell命令，然后再用shell去执行这些命令。

**例如：**eval $(awk 'BEGIN{print "var1='str1';var2='str2'"}')  
或者eval $(awk '{printf("var1=%s; var2=%s; var3=%s;",$1,$2,$3)}' abc.txt)  
  
之后可以在当前shell中使用var1,var2等变量了。  
echo "var1=$var1 ----- var2=$var2"

**三、一个检测磁盘空间使用情况的脚本的例子**

#!/bin/sh  
#文件系统名字  
FILE\_SYSTEM\_NAME="rootfs"

#文件系统挂在的目录  
MOUNTED\_ON="/"  
  
# shell命令使用awk中定义的变量spaceSize  
eval $(df -P | awk '$1=="'"$FILE\_SYSTEM\_NAME"'" && $6=="'$MOUNTED\_ON'" {printf("spaceSize=%s;",$5)}')  
echo "主磁盘的使用空间为$spaceSize"  
  
spaceSize=`echo $spaceSize | cut -d% -f1`   
if [ aa$spaceSize = "aa" ]; then  
    spaceSize=-1  
fi  
  
if [ $spaceSize -le 85 ]; then  
    echo '主磁盘的使用空间充足'  
elif [ $spaceSize -eq -1 ]; then  
    echo '没有找到主磁盘使用空间，请检查脚本'  
else  
    echo '主磁盘的使用空间超过阈值'  
fi