**awk格式**

awk参数 模式动作

awk options 'pattern {action}' file

（条件）找谁 干啥

　‘pattern {action} ’

**域(filed)和记录(record**)

filed:域，区域，字段（类似列）。如$1,$2,$3,$0（$0是整行，$1或以上就是列数）

record:记录，默认一整行

NR：行号（是默认是\n换行回车分隔），number of record 记录的数。awk当前处理的，记录的数

RS:record sep(记录分隔符)每个记录（类似行）的分隔符。（默认是\n换行回车分隔的

ORS：output，输出时候的分隔符）

FS: （字段分隔符，类似列分隔符） 就是awk -F":" 的-F， -F"[:/]"表示以冒号或斜杆分隔

OFS：（输出），通过BEGIN模块输出

**小结：**

一、$:表取区域，$0(1整行),$1（第1列）,$2 NF,$NF,$(NF-3)

二、NR：存放着每个记录的行号，读取新行时会自动+1，NR是从一到尾行显示行数。$NR是第x行，第x列的数， 如当NR=3时，$NR显示的数是第3行，第3列的数值。

三、NF：记录中的区域数量，NF是每行的总区域数（列数），$NF是最后一区域（列数），$(NF-2)是倒数第3区 域数（列数）

四、FS（重点，就是-F，一把选好分隔的刀）: 就是awk -F":" 的-F， -F"[:/]"表示以冒号或斜杆分隔

五、RS：是记录(类似行)的分隔符，是指定每个记录的结束标志（默认是\n回车换行）

六、print $0 特殊问题，建立索引即可，如awk -F"[:/]" 'BEGIN{OFS="###"}{$1=$1;print $0}' fs-example.txt

七、BEGIN是在第1行读取前处理，END是所有行读取完了处理

RS案例

[root@shl ~]# head -1 passwd2.txt

root x 0 0 root /root /bin/bash

[root@shl ~]# awk 'BEGIN{RS="/"}{print NR,$0}' passwd2.txt

1 root x 0 0 root

2 root

3 bin

4 bash

OFS案例

[root@shl ~]# head -2 fs-example.txt

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

[root@shl ~]# awk -F"[:/]" 'BEGIN{OFS="###"}{print $1,$NF}' fs-example.txt

root###bash

bin###nologin

注意$0都是是默认整行

[root@shl ~]# awk -F"[:/]" 'BEGIN{OFS="###"}{print $0}' fs-example.txt

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

加特殊条件

t@shl ~]# awk -F"[:/]" 'BEGIN{OFS="###"}{$1=$1;print $0}' fs-example.txt

root###x###0###0###root######root######bin###bash

bin###x###1###1###bin######bin######sbin###nologin

加文本注释输出

~]# awk -F: '{print "user:"$1}' fs-example.txt

user:root

user:bin

注意：（想文本输出就要加双引号，再加逗号就是一空格分隔，不加是紧靠）

+++++++++++++++++++++++华丽的分隔线++++++++++++++++++++

~]# cat >rs.txt <<EOF

> 2

> 3

> 1

> 4

> 5

> 1

> 6

> 7

> EOF

~]# awk 'BEGIN{RS="1"}{print $0}' rs.txt

2

3

4

5

6

7

[root@shl ~]# awk 'BEGIN{RS="1"}{print NR,$0}' rs.txt

1 2

3

2

4

5

3

6

7

[root@shl ~]# awk 'BEGIN{RS="1"}{print "num:"NR,$0}' rs.txt

num:1 2

3

num:2

4

5

num:3

6

7

t@shl ~]# awk 'BEGIN{RS="1"}{$1=$1;print "num:"NR,$0}' rs.txt

num:1 2 3

num:2 4 5

num:3 6 7

（$0特殊，要加"$1=$1;"才显示 以上效果）

t@shl ~]# awk 'BEGIN{RS="1";FS="\n"}{$1=$1;print "num:"NR,$0}' rs.txt

num:1 2 3

num:2 4 5

num:3 6 7

终极~]# awk 'BEGIN{RS="1";FS="\n"}{$1=$1;print "num:"NR,"1fi"$1,"2se"$2,"3tr"$3}' rs.txt

num:1 1fi2 2se3 3tr

num:2 1fi 2se4 3tr5

num:3 1fi 2se6 3tr7

讲解：

2 2\n

3 3\n

1------》一记录结束\n

4 4\n ----》 2 3 （记录空）

5 5\n \n 4 5 (记录空)

1------》二记录结束\n \n 6 7 (记录空)

6 6\n

7 7 \n

+++++++++++++++++++++++华丽的分隔线++++++++++++++++++

[root@shl ~]# awk '{print "daxiuge888"}'

^C （一直卡在这里，ctrl+终止）

[root@shl ~]# awk '{print "daxiuge888"}' fs-example.txt （文本有4行内容）

daxiuge888

daxiuge888

daxiuge888

daxiuge888

（文本有多少行就输出多少次）

但是加了BEGIN就可以输出一行

[root@shl ~]# awk '{print "daxiuge888"}'

^C

[root@shl ~]# awk 'BEGIN{print "daxiuge888"}'

daxiuge888

**正则表达式**

|  |  |  |  |
| --- | --- | --- | --- |
| 元字符 | 功能 | 示例 | 解释 |
| **^** (尖角号) | 字符串开关 | **/^cool/** | 匹配所有以cool开头的字符串 |
| **$** | 字符串的结尾 | **/cool$/** | 表匹配所有以cool结尾的字符串 |
| **.** (点) | 匹配任意单个字符（包括回车符） | **/c..l/** | 表示匹配字母c然后两个任意字符，再以匹配l的字符的行。 |
| **\*** | 匹配0个或多个前导 | **/a\*cool/** | 匹配0个或多个a之后 紧跟着cool的行，比如cool,aaacool |
| **?** | 匹配0个或1个前导字符，最多1个 | **/a?b/** | 匹配了或ab的行 |
| **+** | 匹配1个或1个以上前导字符，至少1个 | **/a+b/** | 匹配一个或多个a加b的行，如包含ab,aab的行 |
| **[]** | 匹配指定字符组内的任一个字符 | **/a-zA-Z0-9/** | 匹配任意大小写字母及0到9字数字符的行 |
|  |  | **/[^abc]/** | 匹配以字母a或b或c开头的行 |
|  |  | **/^[^abc]/** | 匹配不以字母a或b或c开头的行 |
| **()** | 子表达式组合 | **/（cool）+/** | 表示一个或多个cool组合，当有一些字符需要组合时，使用括号括起来 |
| **|** | 或者的意思 | **/（cool）|B/** | 匹配cool或字母B的行 |

awk不支持的元字符，和需要添加参数才能支持的元字符

|  |  |  |
| --- | --- | --- |
| **x{m}** | **/cool{5}/** | x重复m次 |
| **x{m,}** | **/cool{5,}/** | x重复至少m次 |
| **x{m,n}** | **/cool{5,6}/** | x重复至少m次,但不起过n次 |

**注意：awk --posix ‘$0~/o{1,2}/’{print $0} /etc/passswd的用法**

**（--posix 与 {m,n}连用）**

awk支持的关系运算符

|  |  |  |
| --- | --- | --- |
| 运算符 | 含义 | 示例 |
| **>** | 大于 | awk -F 'NR>2{print NF}' |
| **>=** | 小于或等于 | awk -F 'NR>=2{print NF}' |
| **==** | 等于 | awk -F 'NR==2{print NF}' |
| (小于的也一样，上面的针对数字，下面的针对字符) | | |
| **~** | 与正则表达式匹配（包含） | awk -F '$5~/^s/{print }' |
| **！~** | 与正则表达式不匹配（不包含） | awk -F ’$5！~/^s/{print }' |

**注意：只有！~和！=**

**正则表达式和通配符的区别**

通配符：匹配文件名的

正则表达式：匹配字符串

**awk数组**

**定义数组：数组名[****下标]=元素值** 也叫**数组名[元素名]=元素值**

**调用数组：数组名[下标]**

**遍历数组：for(变量 in 数组名){print 数组名[变量]}**  //这里的变量in是取数组名下标（或元素名）

数组==》元素值

一个筐==》苹果数

array[www.a.com] ==》红苹果数量

array[www.b.com] ==》青苹果数量

[www.a.com](http://www.a.com) ==>下标(或元素名)

[www.b.com](http://www.b.com) ==>下标(或元素名)

**数组思想：**1.awk核心思想是“先处理”，然后“END模块输出”（awk数组）。

2.BEGIN模块用于awk内置变量FS,RS的赋值，打印标题头的信息。要在awk执行前定义好。

3.END模块用来最后输出，统计信息，awk数组信息。

~]# **grep -c "^$" /etc/init.d/network //grep -c 是匹配条件的总行数**

42 //找出/etc/init.d/network 空行的次数

~]# awk '/^$/{x=x+1}END{print x}' /etc/init.d/network

42

~]# awk '/^$/{x=x+1;print NR}END{print x}' /etc/init.d/network 加多个NR显示 那行是空行

shl ~]# awk '/^$/{x=x+1}END{print "total:"x}' /etc/init.d/network

total:42

~]# awk '/^$/{x=x+1;print x}END{print "total:"x}' /etc/init.d/network

1

2

..

42

total:42

**（注意 /^$/{x=x+1}相等/^$/{x+=1}相等/^$/{x++}）**

**数组案例一**：找出环境变量$PATH中，所有只有三个任意字符的命令，例如scp，(并把它们重定向到command.txt中，要求一行显示1个，)并在尾部统计他们的个数！(括号里的可做可不做)

（错误做法，不是找文件，找文本内容了）

~]# find `echo $PATH |awk 'BEGIN{RS=":"}{print NR,$0}'` |awk -F/ '{print $NF}' |grep -c "^...$" find: ‘1’: 没有那个文件或目录

find: ‘2’: 没有那个文件或目录

find: ‘3’: 没有那个文件或目录

find: ‘4’: 没有那个文件或目录

find: ‘5’: 没有那个文件或目录

find: ‘/root/bin’: 没有那个文件或目录

51

正确方法 ~]# find `echo $PATH |awk 'BEGIN{RS=":"}{print $0}'` -type f -name "???" |wc -l

find: ‘/root/bin’: 没有那个文件或目录

46

标准方法~]# **find $(echo $PATH|tr ":" " ") -type f -name "???" | awk '{a++}END{print “total:”a}'**

find: ‘/root/bin’: 没有那个文件或目录

46

**数组案例二**

[root@shl ~]# cat http.txt

http://www.a.com/index.html

http://www.daxiuge888.com/index.html

http://www.daxiuge888.com/index.htmli

http://www.daxiuge888.com/index.htmlfasf

http://www.b.com/index.html

[root@shl ~]# **awk -F"/+" '{print $2}' http.txt |sort |uniq -c**

1 www.a.com

1 www.b.com

[root@shl ~]# **awk -F"/+" '{array[$2]++;print array[$2],$2}' http.txt**

1 www.a.com

1 www.daxiuge888.com

2 www.daxiuge888.com

3 www.daxiuge888.com

1 www.b.com

（左排是元素值（数值），右边是数组下标）

[root@shl ~]#**awk -F"/+" '{array[$2]++}END{for(key in array)print key,array[key]}' http.txt**

www.daxiuge888.com 3

www.b.com 1

www.a.com 1

**awk的BEGIN模块和END模块**

**BEGIN模块输出一些提示性文字。awk内置变量FS,RS,ORS,OFS （FS最常用），BEGIN模块在awk读取文件之前就执行。**

**END模块输出一些提示性文字，显示最后的结果，计算空行，awk数组，取苹果、IP的过程。END模块在读取所有文件的时候，再执行END模块，一般用来输出一个结果（累加、数组结果 ）**

**BEGIN案例**

**作屏幕输出**

[root@shl awk]# head -2 /etc/passwd |awk '**BEGIN{print "hello world"}**{print }'

hello world

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

**作RS=“分隔符”分隔每行**

[root@shl awk]# head -2 /etc/passwd

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

[root@shl awk]# **head -2 /etc/passwd |awk 'BEGIN{RS="/"}{print }'**

root:x:0:0:root:

root:

bin

bash

bin:x:1:1:bin:

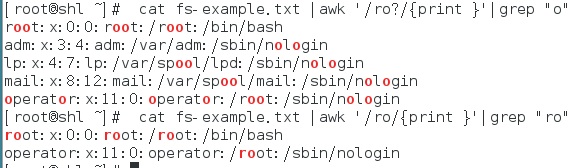
bin:

sbin

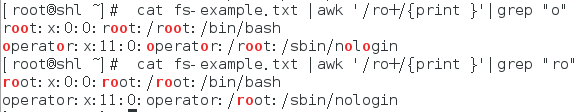
nologin

**awk综合案例**

?用法的案例



+用法的案例



（{print }相当于{print $0}）

~]# awk -F":" '$5~/^s/{print }' /etc/passwd

sync:x:5:0:sync:/sbin:/bin/sync

shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin

~]# awk -F: '$5~/c$/{print $0}' /etc/passwd

sync:x:5:0:sync:/sbin:/bin/sync

**取IP案例**

~]# ifconfig eth0 |awk 'NR==2{print $0}'

inet 192.168.1.21 netmask 255.255.255.0 broadcast 192.168.1.255

~]# ifconfig eth0 |awk -F"[ ]|:" 'NR==2{print NF}'

16

~]# ifconfig eth0 |awk -F"[ ]+" 'NR==2{print NF}'

7

~]# ifconfig eth0 |awk -F"[ ]+" 'NR==2{print $3}'

192.168.1.21

（注意：awk -F"[ ]+"和awk -F “ +”都表示匹配至少一个空格分隔，要不inet前有好多空格分隔为多个空白字段（空格也有一个字符分配的），只有方法把字段分隔越少，你想要过滤的内容越容易过滤出来）

方法二 ~]# ifconfig eth0 | awk -F"inet|netmask" 'NR==2{print NF}'

3

[root@shl ~]# ifconfig eth0 | awk -F"inet|netmask" 'NR==2{print $2}'

192.168.1.21

**awk运算符运用**

（取行数）用awk取出/etc/passwd的第2行到第5行

方法一~]# awk 'NR>=2&&NR<=5{print }' /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

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

方法二 ~]# awk '/^bin/,/^lp/{print }' /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

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

（ awk '/start开始位置/,/end结束位置/{print }' 文件名），不过要请注意awk -F: '/^bin/,/^lp/{print $1,$7}' /etc/passwd里的$1和$7是第1、第7字段（列），不是连续的1到7字段（列））