20. 正则三剑客 - awk

笔记本:	优秀笔记		
创建时间:	2018/12/28 9:43	**更新时间:**	2018/12/28 9:58
作者:	306798658@qq.com		

awk工具介绍

> awk也是流式编辑器，针对文档中的行来操作，一行一行的操作；
> 前面介绍的grep、sed所有的功能，awk基本上都可以实现。

awk截取文档中的某个段落，示例如下：

- -F选项的作用是指定分隔符，如果不加-F选项，以空格或者tab为分隔符。
- print为打印的动作，用来打印某个字段。$1为第一段，$2为第二段，$0表示整行。
  - 示例1

```
[root@localhost ~]# mkdir awk #创建个awk目录
[root@localhost ~]# cp /etc/passwd awk/test.txt #拷贝个文件过来并且修改名字
[root@localhost ~]# cd awk/
[root@localhost awk]# ls
test.txt
[root@localhost awk]# awk -F ':' '{print $1}' test.txt #打印出来第一段，分隔符为冒号“：”
root
bin
daemon
adm
lp
sync
shutdown
halt

mail
```

  - 示例2，打印出来所有的内容：$0

```
[root@localhost awk]# awk '{print $0}' test.txt #$0表示整行，所有。
root:x:0:0:root:/root:/bin/bash
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
```

匹配字符或者字符串

- 打印出来包含oo的行。

```
[root@localhost awk]# awk '/oo/' test.txt
root:x:0:0:root:/root:/bin/bash
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
```

```
operator:x:11:0:operator:/root:/sbin/nologin

postfix:x:89:89::/var/spool/postfix:/sbin/nologin
```

- 打印第一行的第一段包含oo的行。

```
[root@localhost awk]# awk -F ':' '$1 ~ /oo/' test.txt

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

- 多次打印，多次匹配test，示例如下：

```
[root@localhost awk]# awk -F ':' '/root/ {print $1,$3} /bash/ {print $1,$3,$7}' test.txt
root 0
root 0 /bin/bash
operator 11

#这命令的意思是 第一和第三段，包含root的字符，和第一段 第三段 第七段包含bash的字符打印出来。
```

条件操作符

```
awk中可以用逻辑符号进行判断，比如==就是等于，也可以理解为精确匹配；
另外还有>、>=、<等，在和数字比较时，若把比较的数字用双引号括起来，那么awk不会认为是数字，
而会认为是字符，不加双引号则会认为是数字。
```

- 示例1，精确匹配：

```
[root@localhost awk]# awk -F ':' '$3=="0"' test.txt #数字0

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

- 示例2，列出来第三段大于等于500的行打印出来

```
[root@localhost awk]# awk -F ':' '$3>=500' test.txt
polkitd:x:999:997:User for polkitd:/:/sbin/nologin

chrony:x:998:996::/var/lib/chrony:/sbin/nologin
```

- 示例3，！=表示不匹配，第七段不等于/sbin/nologin的有哪些，如下：

```
[root@localhost awk]# awk -F ':' '$7!="/sbin/nologin"' test.txt
root:x:0:0:root:/root:/bin/bash
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

halt:x:7:0:halt:/sbin:/sbin/halt
```

- 可以使用&&和||，他们分别表示并且和或者，用法如下：

```
[root@localhost awk]# awk -F ':' '$3<$4' test.txt
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
```

- ||或者的用法如下：

```
[root@localhost awk]# awk -F ':' '$3>100 || $7=="/sbin/nologin"' test.txt
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
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin
```

awk的内置变量

```
awk常用的变量有OFS，NF和NR
OFS和-F选项有类似的功能，也是用来定义分隔符的，但是它实在输出的时候定义；
NF表示用分割符分割后一共有多少段；
NR表示行号。
```

- OFS的用法如下，print打印时的分割符：

```
[root@localhost awk]# awk -F ':' '{OFS="#"} $3>1000 || $7 ~ /bash/ {print $1,$3,$7}' test.txt
root#0#/bin/bash

#OFS分隔符，以#号为单位，再写条件$3>1000, 后面在跟print语法。
```

- NR表示行，NF表示段，命令介绍如下：

```
[root@localhost awk]# awk -F ':' '{print NF":"$0}' test.txt #NF表示列出来有多少段。
7:root:x:0:0:root:/root:/bin/bash
7:bin:x:1:1:bin:/bin:/sbin/nologin
7:daemon:x:2:2:daemon:/sbin:/sbin/nologin
7:adm:x:3:4:adm:/var/adm:/sbin/nologin

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

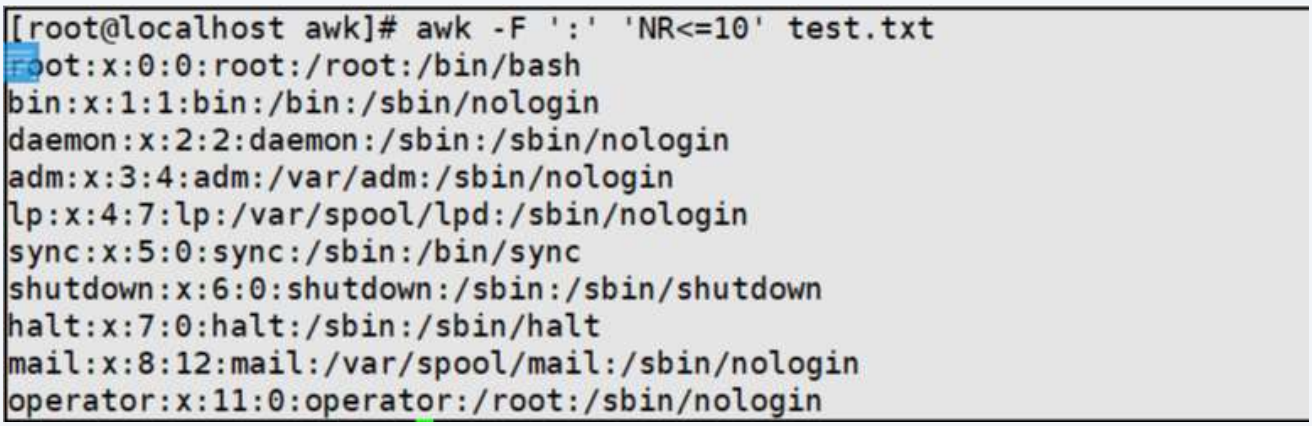
```
[root@localhost awk]# awk -F ':' '{print NR":"$0}' test.txt #NR表示把行号标出来。
1:root:x:0:0:root:/root:/bin/bash
2:bin:x:1:1:bin:/bin:/sbin/nologin
```

```
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
```

- 用NR列出前10行

```
[root@localhost awk]# awk -F ':' 'NR<=10' test.txt #:表示分割符，NR列行小于10的列出来
root:x:0:0:root:/root:/bin/bash
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
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
```



- 还可以指定条件匹配一起使用

```
[root@localhost awk]# awk -F ':' 'NR<=10 && $1 ~ /root|sync/' test.txt
#awk -F指定分割符为“:”，NR列出来小于10的行，其中$1第一行 ~包含 /root|或者sync/的列出来。
root:x:0:0:root:/root:/bin/bash

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

awk中的数学运行

- awk可以更改段值，示例如下：

```
[root@localhost awk]# head -n 3 test.txt |awk -F ':' '$1="root"'  # 这里的=相当于赋值，第一段都赋值成root，列出前三段。
root x 0 0 root /root /bin/bash
root x 1 1 bin /bin /sbin/nologin

root x 2 2 daemon /sbin /sbin/nologin
```

- 上例中切割没有分割符，可以OFS定义一下分割符，示例如下：

```
[root@localhost awk]# head -n 3 test.txt |awk -F ':' '{OFS=";"} $1="root"'
root;x;0;0;root;/root;/bin/bash
root;x;1;1;bin;/bin;/sbin/nologin

root;x;2;2;daemon;/sbin;/sbin/nologin
```

- awk匹配root的第一行有哪些，示例如下：

```
[root@localhost awk]# head -n 3 test.txt |awk -F ':' '$1=="root"'   # 这里的==是精准匹配，也就是所谓的等于

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

- awk计算某个段的总和，示例如下：

```
[root@localhost awk]# awk -F ':' '{{tot=tot+$3}}; END {print tot}' test.txt

2605


tot=tot+$3 这段的意思是每一次的值，都会跟第三段相加，实现一个循环，默认是从0开始。这样可以列出来一个列的总和。
```