awk简介

awk是Linux Shell中一种非常强大的文本处理工具，数据可以来自**标准输入或者管道**；

awk不仅仅是一个命令，更是一门**编程语言**，支持选择结构、循环结构、以及丰富的逻辑运算符，便于进行复杂的文本处理；

awk按照行处理文本，逐行扫描，默认从第一行到最后一行，找到匹配到特定行，并进行相关操作；

awk分别代表其作者姓氏的第一个字母。因为它的作者是三个人，分别是Alfred Aho、Brian Kernighan、Peter Weinberger。

语法格式：

```
awk 选项 'commands' 文件名

常用选项：
-F  定义字段分割符号，默认的分隔符是空格
-v  定义变量并赋值
```

```
hioier@yunpc:~/scripts$ awk -F: '/root/{print FILENAME}' input.txt
input.txt
hioier@yunpc:~/scripts$ awk -F: '/root/{print $0}' input.txt
root:x:0:0:root:/root:/bin/bash
```

-F: 是以:将源文件分隔

/root/匹配源文件中包括root的行，打印出文件名，如果不进行匹配，则源文件有多少行就打印出多少行文件名

$0打印出匹配到行的全部内容

awk内置变量

变量	变量说明
$0	当前处理行的所有记录
`$1,$2,$3...$n`	文件中每行以间隔符号分割的不同字段
NF	当前记录的字段数（列数）
$NF	最后一列
NR	行号
FS	定义间隔符
OFS	定义输出字段分隔符，默认空格
RS	输入记录分割符，默认换行
ORS	输出记录分割符，默认换行
FILENAME	当前输入的文件名

```
hioier@yunpc:~/scripts$ awk -F: '{print NF,$NF,$1,$2,$(NF-1)}' input.txt
7 /bin/bash root x /root
7 /bin/bash root x /root
7 /bin/bash root x /root
7 /usr/sbin/nologin daemon x /usr/sbin
7 /usr/sbin/nologin bin x /bin
```

打印列数、最后一列、第1列、第2列、倒数第二列。

```
hioier@yunpc:~/scripts$ awk 'NR==1,NR==5' input.txt
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

打印第1行到第5行。

```
hioier@yunpc:~/scripts$ awk 'NR==1,NR==5;/^root/{print $0}' input.txt
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

用分号;分隔开两部分匹配规则，可以理解为或的关系，前面是匹配1-5行，后面是匹配以root开头的行，是前3行。一共输出8行。

格式化输入printf

```
hioier@yunpc:~/scripts$ awk -F: '{printf "%-10s %-10s %-10s\n",$1,$2,$3}' input.txt
root       x          0
root       x          0
root       x          0
daemon     x          1
bin        x          2
```

%s字符串类型占位符，默认右对齐，前面加负号-左对齐。

awk中BEGIN...END使用

BEGIN：表示在程序开始前执行

END：表示所有文件处理完后执行

用法：`'BEGIN{开始处理之前};{处理中};END{处理结束后}'`

```
hioier@yunpc:~/scripts$ awk -F: 'BEGIN{print "NAME\tDIR\tSHELL\n****************************************"}{printf "%-10s %-10s %-10s\n",$1,$(NF-1),$NF}END{print "****************************************"}' input.txt
NAME    DIR     SHELL
****************************************
root       /root      /bin/bash
root       /root      /bin/bash
root       /root      /bin/bash
daemon     /usr/sbin  /usr/sbin/nologin
bin        /bin       /usr/sbin/nologin
****************************************
```

```
awk -F: 'BEGIN{print "NAME\tDIR\tSHELL\n****************************************"}{printf "%-10s %-10s %-10s\n",$1,$(NF-1),$NF}END{print "****************************************"}' input.txt
```

间隔符

```
hioier@yunpc:~/scripts$ awk 'BEGIN{FS=":"};/^root/{print $1,$NF}' input.txt
root /bin/bash
root /bin/bash
root /bin/bash
```

BEGIN{FS=":"}相当于使用选项-F:

```
hioier@yunpc:~/scripts$ awk -F: 'BEGIN{OFS="\t\t"};/^root/{print $1,$NF}' input.txt
root            /bin/bash
root            /bin/bash
root            /bin/bash
```

OFS输出间隔为两个制表符\t。

```
hioier@yunpc:~/scripts$ awk 'BEGIN{RS="@"};{print $0}' input.txt
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
hioier.com
blog.hioier.com
noi.hioier.com
```

输入内容以@分隔。

```
hioier@yunpc:~/scripts$ awk 'BEGIN{RS="@";ORS="++++"};{print $0}'
input.txt
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
hioier.com++++blog.hioier.com++++noi.hioier.com
```

输出内容指定以"++++"分隔。

awk工作原理

1. awk使用一行作为输入，并将这一行赋给内部变量$0，每一行也可称为一个记录，以换行符(RS)结束
2. 每行被间隔符==:==(默认为空格或制表符)分解成字段，每个字段存储在已编号的变量中，从$1开始

   问：awk如何知道用空格来分隔字段的呢?

   答：因为有一个内部变量FS来确定字段分隔符。初始时，FS赋为空格。
3. awk使用print函数打印字段，打印出来的字段会以空格分隔，比如1,1,3之间有一个逗号，但是逗号比较特殊，它映射为另一个内部变量，称为输出字段分隔符OFS，OFS默认为空格。
4. awk处理完一行后，将从文件中获取另一行，并将其存储在$0中，覆盖原来的内容，然后将新的字符串分隔成字段并进行处理，该过程将持续到所有行处理完毕。

awk变量定义

```
hioier@yunpc:~/scripts$ awk -v NUM=3 -F: '/^root/{print NUM}' inpu
t.txt
3
3
3
hioier@yunpc:~/scripts$ awk -v NUM=3 -F: '/^root/{print $NUM}' inp
ut.txt
0
0
0
```

调用awk中变量不需要加$。

逻辑运算符

```
hioier@yunpc:~/scripts$ awk 'NR≥1 && NR≤4' input.txt
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
hioier@yunpc:~/scripts$ awk 'NR==1 || NR==4' input.txt
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

使用或||运算符打印第1行和第4行。

```
hioier@yunpc:~/scripts$ awk 'NR≥1 && NR≤4 || /^hioier/' input.txt
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
hioier.com@blog.hioier.com@noi.hioier.com
```

匹配1-4行或者以hioiei开头的行。

流程控制

awk的流程控制和一般编程语言中的逻辑一样，包括选择结构和循环结构，用于处理复杂的命令逻辑。

if条件判断

```
单分支结构：
{if(表达式){语句1;语句2;...}}
```

```
hioier@yunpc:~/scripts$ awk -F: '{if($3==0){print $0,$1"是超级管理员
"}}' input.txt
root:x:0:0:root:/root:/bin/bash root是超级管理员
root:x:0:0:root:/root:/bin/bash root是超级管理员
root:x:0:0:root:/root:/bin/bash root是超级管理员
```

```
双分支结构：
{if(表达式){语句;语句;...}else{语句;语句;...}}
```

```
hioier@yunpc:~/scripts$ awk -F: '{if($3==0){print $1"是超级管理
员"}else{print $1"不是超级管理员"}}' input.txt
root是超级管理员
root是超级管理员
root是超级管理员
daemon不是超级管理员
bin不是超级管理员
oyhx不是超级管理员
zzx不是超级管理员
mongodb不是超级管理员
```

```
多分支结构：
{ if(表达式1){语句;语句；...} else if(表达式2){语句;语句；...} else if(表达式3){语
句;语句；...} else{语句;语句；...} }
```

```
hioier@yunpc:~/scripts$ awk -F: '{if($3==0){print $1"是超级管理
员"}else if($3 <= 999){print $1"是系统用户"}else{print $1"是普通
用户"}}' input.txt
root是超级管理员
root是超级管理员
root是超级管理员
daemon是系统用户
bin是系统用户
oyhx是普通用户
zzx是普通用户
mongodb是系统用户
```

循环结构

```
hioier@yunpc:~/scripts$ awk 'BEGIN{i=1;while(i<=10){print i;i++}
}' input.txt
1
2
3
4
5
6
7
8
9
10
```

while循环打印1-10

```
hioier@yunpc:~/scripts$ awk 'BEGIN{for(i=1;i<=10;i++){sum+=i;pri
nt i};{print sum}}' input.txt
1
2
3
4
5
6
7
8
9
10
55
```

for循环打印1-10并求和，结果55。

```
hioier@yunpc:~/scripts$ awk 'BEGIN{i=1;while(i<=10){if(i==3)brea
k;print i;i++}}' input.txt
1
2
hioier@yunpc:~/scripts$ awk 'BEGIN{i=1;while(i<=10){if(i==3)cont
inue;print i;i++}}' input.txt
1
2
```

break跳出整个循环，continue跳出本次循环，继续进入下一次循环，由于本次结束后，i并没有加1，因此程序一直卡住。

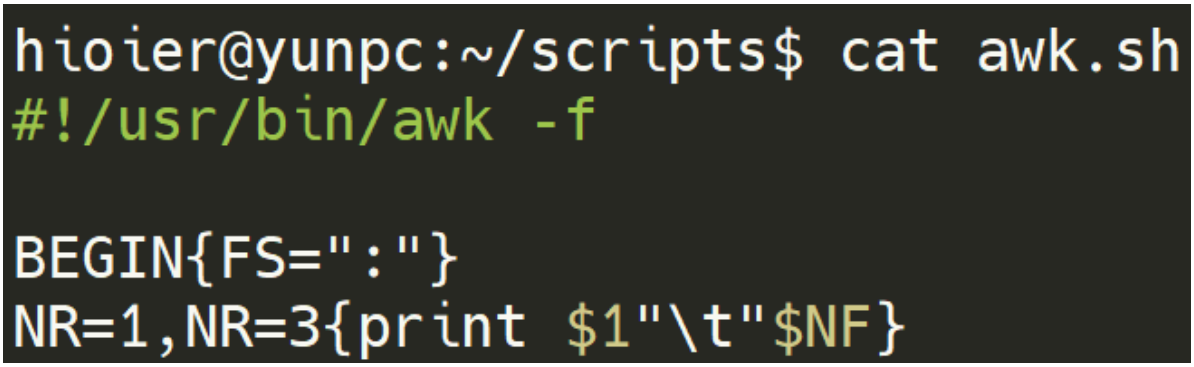
思考，如何调整循环体的顺序？使得跳出本次循环后，进入下一次循环。

算术运算

```
hioier@yunpc:~/scripts$ awk 'BEGIN{print 1+2}'
3
hioier@yunpc:~/scripts$ awk 'BEGIN{print 1-2}'
-1
hioier@yunpc:~/scripts$ awk 'BEGIN{print 1*2}'
2
hioier@yunpc:~/scripts$ awk 'BEGIN{print 1/2}'
0.5
```

awk按照**浮点数**进行数学运算。

脚本运行




```
#!/usr/bin/awk -f

BEGIN{FS=":"}
NR==1,NR==3{print $1"\t"$NF}
```

运行：

```
1.awk -f awk.sh input.txt
./awk.sh input.txt
```