awk 编程入门

测试文档：(grade.txt)

```
M.Tansley    05/2013 48311    Green    8    90
J.Lulu       04/2012 48317    green    12   88
P.bunny      02/2013 48       Yellow   9    70
J.Troll      09/2013 4842     Brown-3  11   95
L.Tansley    05/2013 4712     Brown-2  10   85
Vincent      07/2012 4712     Brown    13   87
```

1，打印指定列（例如名字和年龄）

```
awk   '{print $1,$5}'   grade.txt
```

注意：其中单引号中的被大括号括着的就是 awk 的语句，只能被单引号包含。
其中的$1..$n 表示第几例。$0 表示整个行。

注意：awk 的工作次序是：首先读取 grade.txt 文件一个记录（即一行），然后根据单引号内部的指令工作，完了之后继续读下一行，以此类推。每读一行都会将单引号里面的语句从头到尾地应用到那一行中去。

2，格式化输出（和 C 语言一样一样的）

```
awk   '{printf"%-10s:%-d\n", $1,$5}'   grade.txt
```

3，过滤

```
awk   '$5==11 && $6>=90 {print $0}'   grade.txt
```

意思是==>打印出年龄等于 11 岁且分数大于 90 分的孩子的信息

4，打印表头，引入内建变量 NR

```
awk   'NR==1 || $6>=90 {print}'   grade.txt
```

注意：NR 是一个所谓的内建变量，表示已经读出的记录数（即行号）

其它有用的内建变量是：

$0	当前记录（这个变量中存放着整个行的内容）
$1~$n	当前记录的第 n 个字段，字段间由 FS 分隔
FS	输入字段分隔符 默认是空格或 Tab
NF	当前记录中的字段个数，就是有多少列
NR	行号，从1开始，如果有多个文件这个值将不断累加。
FNR	当前记录数，与 NR 不同的是，这个值会是各个文件自己的行号
RS	输入的记录分隔符， 默认为换行符
OFS	输出字段分隔符， 默认也是空格
ORS	输出的记录分隔符，默认为换行符

FILENAME	当前输入文件的名字

5，指定分隔符

awk ‘BEGIN{FS=”:”} {print $1}’ /etc/passwd

注意：BEGIN 意味着紧跟在它后面的语句{FS=”:”}会在 awk 读取第一行之前处理。

上面的语句等价于

awk -F: ‘{print $1}’ /etc/passwd

如果有多个分隔符，则可以写成

awk -F’[\t;:]’ ‘{print $1}’ /etc/passwd

6，使用正则表达式匹配字符串

awk '$0~/Brown.*/ {print}' grade.txt 意思是==>将所有包含 Brown 的行打印出来

注意：波浪号~后面紧跟着一对正斜杠，表示所指定的域（这里是$0)要匹配的规则

7，使用模式取反的例子

awk '$0!~/Brown.*/ {print}' grade.txt 意思是==>将所有不包含 Brown 的行打印出来

拆分文件

8，将各年龄段的孩子的信息分别存放在各个文件中

```
awk 'NR!=1 {print > $5}' grade.txt
```

意思是==>表头不处理，后面的每一行，都将被重定位到以第 5 个域（年龄）命名的文件中去。

也可以将指定的域重定位到相应的文件

```
awk 'NR!=1 {print $1, $6 > $5}' grade.txt
```

意思是==> 不处理表头，将每一行中的姓名和分数重定位到与其年龄相应的文件中去。

9，再复杂一点，按级别将信息分成三个文件：

```
awk 'NR!=1 {if($4~/Brown.*|Black/) print > "high.txt";
  > else if($4~/Yellow|[Gg]reen/) print > "midle.txt";
  > else print > "low.txt"}' grade.txt
```

意思是==> 如果记录中的第 4 个域（$4）匹配 Brown.*或者 Black，就将该记录重定位到文件 high.txt 中，如果匹配 Yellow 或者[Gg]reen，就重定位到 midle.txt 中，否则统统重定位到 low.txt 中。

统计

10，将所有孩子的分数累积起来并打印出来

```
awk '{sum+=$6} END{print sum}' grade.txt
```

注意：END 表示->紧跟其后的语句只会在 awk 处理完所有行之后才被执行。

11，统计各个级别的人数

```
awk 'NR!=1 {a[$4]++;} END{for(i in a) print i","a[i];}' grade.txt
```

注意：$4 是级别名称，例如 Yellow、Brown 等，a 是一个以这些级别为下标的数组，其值从零开始计算。

awk 脚本

12，想象我现在要打印出整个班级的所有孩子的信息，而且在最前面把表头也打印出来，而且下面要打印一行“==========”来跟具体内容加以划分。并且在最后一行，统计孩子们的平均年龄以及平均分数。将 awk 语句组织成脚本，如下：

```
#!/usr/bin/awk -f
#/usr/bin/awk 指定脚本解释器的位置，-f 表示运行该脚本需要指定一个文件作为输入
BEGIN{ #awk 开始运行之前的准备工作（定义了两个变量），此左花括号必须紧跟 BEGIN
    age = 0
    score = 0
}
{
    if(NR==1) #打印表头已经分割线
    {
        print $0
        printf "============================" #由于是 printf，没有\n 就不换行
        print  "============================" #由于是 print，会自动换行
    }
    else
    {
        age+=$5
        score+=$6
        print $0
    }
}
END{ #awk 处理完所有的记录之后，END 才开始运行
    printf "============================"
    print  "============================"
    print "Average:\t\t\t\t\t" age/NR  ",\t" score/NR
}
```