A+

awk从放弃到入门（1）：awk基础 （通俗易懂，快进来看）

50

在本博客中，AWK是一个系列文章，本人会尽量以通俗易懂的方式递进的总结awk命令的相关知识点。

awk系列博文直达链接：AWK命令总结之从放弃到入门

我们先来用专业的术语描述一下awk是什么，如果你看不懂，没关系，我们会再用"大白话"解释一遍。

awk是一个报告生成器，它拥有强大的文本格式化的能力，这就是专业的说法。

你可能不理解所谓的报告生成器中的"报告"是什么，你可以把"报告"理解为"报表"或者"表格",也就是说，我们可以利用awk命令，将一些文本整理成我们想要的样子，比如把一些文本整理成"表"的样子，然后再展示出来，刚才概念中提到的"文本格式化的能力"，也就是这个意思，其实这样说可能还是不太容易理解，不用着急，当你看到后面的"示例"时，自然会明白awk所擅长的"文本格式化"能力是什么。

awk是由Alfred Aho 、Peter Weinberger 和 Brian Kernighan这三个人创造的，awk由这个三个人的姓氏的首个字母组成。

awk早期是在unix上实现的，所以，我们现在在linux的所使用的awk其实是gawk，也就是GNU awk，简称为gawk，awk还有一个版本，New awk，简称为nawk，但是linux中最常用的还是gawk。

```
[www.zsythink.net]# ll /usr/bin/awk
lrwxrwxrwx. 1 root root 14 Aug  9  2016 /usr/bin/awk -> ../../bin/gawk
[www.zsythink.net]#
```

awk其实是一门编程语言，它支持条件判断、数组、循环等功能。所以，我们也可以把awk理解成一个脚本语言解释器。

grep 、sed、awk被称为linux中的"三剑客"。

我们总结一下这三个"剑客"的特长。

grep 更适合单纯的查找或匹配文本

sed 更适合编辑匹配到的文本

awk 更适合格式化文本，对文本进行较复杂格式处理

此处，我们只总结 awk

awk基础

awk基本语法如下，看不懂没关系，我们会慢慢举例。

awk [options] 'program' file1 , file2 , ```

对于上述语法中的program来说，又可以细分成pattern和action，也就是说，awk的基本语法如下

awk [options] 'Pattern{Action}' file

从字面上理解 ，action指的就是动作，awk擅长文本格式化，并且将格式化以后的文本输出，所以awk最常用的动作就是print和printf，因为awk要把格式化完成后的文本输出啊，所以，这两个动作最常用。

我们先从最简单用法开始了解awk，我们先不使用[options] ,也不指定pattern，直接使用最简单的action，从而开始认识awk，示例如下

```
[www.zsythink.net]# echo ddd > testd
[www.zsythink.net]# awk '{print}' testd
ddd
[www.zsythink.net]#
```

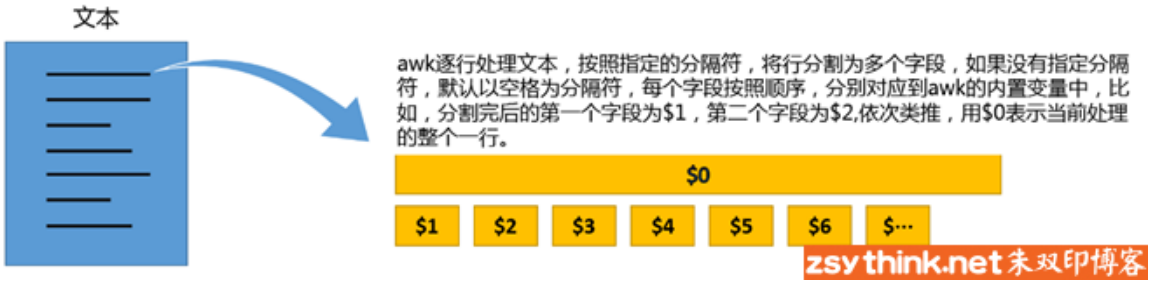
上图中，我们只是使用awk执行了一个打印的动作，将testd文件中的内容打印了出来。

好了，现在，我们来操作一下另一个类似的场景。

```
[www.zsythink.net]# df        3       4       5     6
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda2      103081248 10810004  87028364  12% /
tmpfs  1       2 502068      228    501840   1% /dev/shm
/dev/sda1         487652    34864    427188   8% /boot
/dev/sda3       30832636  9470688  19789084  33% /testdir
/dev/sr0         3824484  3824484         0 100% /mnt
[www.zsythink.net]#
[www.zsythink.net]# df | awk '{print $5}'
Use%
12%
1%
8%
33%
100%
[www.zsythink.net]#
```


上图中的示例没有使用到options和pattern，上图中的awk '{print $5}'，表示输出df的信息的第5列，$5表示将当前行按照分隔符分割后的第5列，不指定分隔符时，默认使用空格作为分隔符，细心的你一定发现了，上述信息用的空格不止有一个，而是有连续多个空格，awk自动将连续的空格理解为一个分割符了，是不是比cut命令要简单很多，这样比较简单的例子，有利于我们开始了解awk。

awk是逐行处理的，逐行处理的意思就是说，当awk处理一个文本时，会一行一行进行处理，处理完当前行，再处理下一行，awk默认以"换行符"为标记，识别每一行，也就是说，awk跟我们人类一样，每次遇到"回车换行"，就认为是当前行的结束，新的一行的开始，awk会按照用户指定的分割符去分割当前行，如果没有指定分割符，默认使用空格作为分隔符。



$0 表示显示整行 ，$NF表示当前行分割后的最后一列（$0和$NF均为内置变量）

注意，$NF 和 NF 要表达的意思是不一样的，对于awk来说，$NF表示最后一个字段，NF表示当前行被分隔符切开以后，一共有几个字段。

也就是说，假如一行文本被空格分成了7段，那么NF的值就是7，$NF的值就是$7，而$7表示当前行的第7个字段，也就是最后一列，那么每行的倒数第二列可以写为$(NF-1)。

我们也可以一次输出多列，使用逗号隔开要输出的多个列，如下，一次性输出第一列和第二列

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk '{print $1,$2}' test
abc 123
8ua 456
[www.zsythink.net]# zsythink.net朱双印博客
```

同理，也可以一次性输出多个指定的列，如下图

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk '{print $2,$4,$5}' test
123 ddd
456 ppp 7y7
[www.zsythink.net]# zsythink.net朱双印博客
```

我们发现，第一行并没有第5列，所以并没有输出任何文本，而第二行有第五列，所以输出了。

除了输出文本中的列，我们还能够添加自己的字段，将自己的字段与文件中的列结合起来，如下做法，都是可以的。

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk '{print $1,$2,"string"}' test
abc 123 string
8ua 456 string
[www.zsythink.net]# awk '{print $1,$2,666}' test
abc 123 666
8ua 456 666
[www.zsythink.net]# awk '{print "diyilie:"$1,"dierlie:"$2}' test
diyilie:abc dierlie:123
diyilie:8ua dierlie:456
[www.zsythink.net]# awk '{print "diyilie:" $1,"dierlie:" $2}' test
diyilie:abc dierlie:123
diyilie:8ua dierlie:456
[www.zsythink.net]# awk '{print "diyilie:" $1,"666","dierlie:" $2}' test
diyilie:abc 666 dierlie:123
diyilie:8ua 666 dierlie:456
[www.zsythink.net]# zsythink.net朱双印博客
```

从上述实验中可以看出，awk可以灵活的将我们指定的字符与每一列进行拼接，或者把指定的字符当做一个新列插入到原来的列中，也就是awk格式化文本能力的体现。

但是要注意，$1这种内置变量的外侧不能加入双引号，否则$1会被当做文本输出，示例如下

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# cat test | awk '{print $1}'
abc
8ua
[www.zsythink.net]# cat test | awk '{print "$1"}'
$1
$1
[www.zsythink.net]# cat test | awk '{print "firstF:"$1}'
firstF:abc
firstF:8ua
[www.zsythink.net]# cat test | awk '{print "firstF:$1"}'
firstF:$1
firstF:$1
[www.zsythink.net]#
```


我们也可以输出整行，比如，如下两种写法都表示输出整行。

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk '{print $0}' test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk '{print}' test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]#
```


我们说过，awk的语法如下

awk [options] 'Pattern{Action}' file

而且我们说过awk是逐行处理的， 刚才已经说过了最常用的Action：print

现在，我们来认识下一Pattern，也就是我们所说的模式

不过，我们准备先把awk中最特殊的模式展示给大家，以后再介绍普通的模式，因为普通模式需要的篇幅比较长，所以我们先来总结特殊模式。

AWK 包含两种特殊的模式：BEGIN 和 END。

BEGIN 模式指定了处理文本之前需要执行的操作：

END 模式指定了处理完所有行之后所需要执行的操作：

什么意思呢？光说不练不容易理解，我们来看一些小例子，先从BEGIN模式开始，示例如下

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk 'BEGIN{print "aaa","bbb"}' test
aaa bbb
[www.zsythink.net]#
```


上述写法表示，在开始处理test文件中的文本之前，先执行打印动作，输出的内容为"aaa","bbb".

也就是说，上述示例中，虽然指定了test文件作为输入源，但是在开始处理test文本之前，需要先执行BEGIN模式指定的"打印"操作

既然还没有开始逐行处理test文件中的文本，那么是不是根本就不需要指定test文件呢，我们来试试。

```
[www.zsythink.net]# awk 'BEGIN{print "aaa","bbb"}'
aaa bbb
[www.zsythink.net]#
```

经过实验发现，还真是，我们并没有给定任何输入来源，awk就直接输出信息了，因为，BEGIN模式表示，在处理指定的文本之前，需要先执行BEGIN模式中指定的动作，而上述示例没有给定任何输入源，但是awk还是会先执行BEGIN模式指定的"打印"动作，打印完成后，发现并没有文本可以处理，于是就只完成了"打印 aaa bbb"的操作。

这个时候，如果我们想要awk先执行BEGIN模式指定的动作，再根据执我们自定义的动作去操作文本，该怎么办呢？示例如下

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk 'BEGIN{print "aaa","bbb"} {print $1,$2}' test
aaa bbb
abc 123
8ua 456
[www.zsythink.net]#
```


上图中，蓝色标注的部分表示BEGIN模式指定的动作，这部分动作需要在处理指定的文本之前执行，所以，上图中先打印出了"aaa bbb"，当BEGIN模式对应的动作完成后，在使用后面的动作处理对应的文本，即打印test文件中的第一列与第二列，这样解释应该比较清楚了吧。

看完上述示例，似乎更加容易理解BEGIN模式是什么意思了，BEGIN模式的作用就是，在开始逐行处理文本之前，先执行BEGIN模式所指定的动作。以此类推，END模式的作用就一目了然了，举例如下。

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk '{print $1,$2} END{print "ccc","ddd"}' test
abc 123
8ua 456
ccc ddd
[www.zsythink.net]#
```


聪明如你一定明白了，END模式就是在处理完所有的指定的文本之后，需要指定的动作。

那么，我们可以结合BEGIN模式和END模式一起使用。示例如下

```
[www.zsythink.net]# cat test
abc 123 iuy ddd
8ua 456 auv ppp 7y7
[www.zsythink.net]# awk 'BEGIN{print "aaa","bbb"} {print $1,$2} END{print "ccc","ddd"}' test
aaa bbb
abc 123
8ua 456
ccc ddd
[www.zsythink.net]#
```


上述示例中返回的结果有没有很像一张"报表"，有"表头" 、"表内容"、 "表尾"，awk对文本的格式化能力你体会到了吗？

awk 常用命令