awk

虽然sed编辑器是非常方便自动修改文本文件的工具，但其也有自身的限制。通常你需要一个用来处理文件中的数据的更高级工具，它能提供一个类编程环境来修改和重新组织文件中的数据。这正是gawk能够做到的。
gawk程序是Unix中的原始awk程序的GNU版本。gawk程序让流编辑迈上了一个新的台阶，它提供了一种编程语言而不只是编辑器命令。在gawk编程语言中，你可以做下面的事情：
- 定义变量来保存数据；
- 使用算术和字符串操作符来处理数据；
- 使用结构化编程概念（比如if-then 语句和循环）来为数据处理增加处理逻辑；
- 通过提取数据文件中的数据元素，将其重新排列或格式化，生成格式化报告。

gawk程序的报告生成能力通常用来从大文本文件中提取数据元素，并将它们格式化成可读的报告。其中最完美的例子是格式化日志文件。在日志文件中找出错误行会很难，gawk程序可以让你从日志文件中过滤出需要的数据元素，然后你可以将其格式化，使得重要的数据更易于阅读。
1. gawk命令格式
gawk程序的基本格式如下：
gawk options program file

gawk程序的可用选项。

选项	描述
-F fs	指定行中划分数据字段的字段分隔符
-f file	从指定的文件中读取程序
-v var=value	定义gawk程序中的一个变量及其默认值
-mf N	指定要处理的数据文件中的最大字段数
-mr N	指定数据文件中的最大数据行数
-W keyword	指定gawk的兼容模式或警告等级

命令行选项提供了一个简单的途径来定制gawk程序中的功能。我们会在探索gawk时进一步了解这些选项。
gawk的强大之处在于程序脚本。可以写脚本来读取文本行的数据，然后处理并显示数据，创建任何类型的输出报告。

2. 从命令行读取程序脚本
gawk程序脚本用一对花括号来定义。你必须将脚本命令放到两个花括号（{}）中。如果你错误地

使用了圆括号来包含gawk脚本，就会得到一条类似于下面的错误提示。
$ gawk '(print "Hello World!"}'
gawk: (print "Hello World!"}
gawk: ^ syntax error

由于gawk 命令行假定脚本是单个文本字符串，你还必须将脚本放到单引号中。下面的例子在命令行上指定了一个简单的gawk程序脚本：
 $ gawk '{print "Hello World!"}'
这个程序脚本定义了一个命令：print 命令。这个命令名副其实：它会将文本打印到STDOUT 。
如果尝试运行这个命令，你可能会有些失望，因为什么都不会发生。原因在于没有在命令行上指定文件名，所以gawk程序会从STDIN 接收数据。在运行这个程序时，它会一直等待从STDIN 输入的文本。
如果你输入一行文本并按下回车键，gawk会对这行文本运行一遍程序脚本。跟sed编辑器一样，gawk程序会针对数据流中的每行文本执行程序脚本。由于程序脚本被设为显示一行固定的文本字符串，因此不管你在数据流中输入什么文本，都会得到同样的文本输出。

```
$ gawk '{print "Hello World!"}'
This is a test
Hello World!
hello
Hello World!
This is another test
Hello World!
```

 要终止这个gawk程序，你必须表明数据流已经结束了。bash shell提供了一个组合键来生成EOF（End-of-File）字符。Ctrl+D组合键会在bash中产生一个EOF字符。这个组合键能够终止该gawk程序并返回到命令行界面提示符下。

3. 使用数据字段变量
 gawk的主要特性之一是其处理文本文件中数据的能力。它会自动给一行中的每个数据元素分配一个变量。默认情况下，gawk会将如下变量分配给它在文本行中发现的数据字段：
- $0 代表整个文本行；
- $1 代表文本行中的第1个数据字段；
- $2 代表文本行中的第2个数据字段；
- $n 代表文本行中的第n 个数据字段。

 在文本行中，每个数据字段都是通过字段分隔符 划分的。gawk在读取一行文本时，会用预定义的字段分隔符划分每个数据字段。gawk中默认的字段分隔符是任意的空白字符（例如空格或制表符）。
 在下面的例子中，gawk程序读取文本文件，只显示第1个数据字段的值。

```
$ cat data2.txt
One line of test text.
Two lines of test text.
Three lines of test text.
$
$ gawk '{print $1}' data2.txt
One
Two
Three
$
```

该程序用$1 字段变量来仅显示每行文本的第1个数据字段。

如果你要读取采用了其他字段分隔符的文件，可以用-F 选项指定。

```
$ gawk -F : '{print $1}' /etc/passwd
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
[...]
```

这个简短的程序显示了系统中密码文件的第1个数据字段。由于/etc/passwd文件用冒号来分隔数字字段，因而如果要划分开每个数据元素，则必须在gawk 选项中将冒号指定为字段分隔符。

4. 在程序脚本中使用多个命令

如果一种编程语言只能执行一条命令，那么它不会有太大用处。gawk编程语言允许你将多条命令组合成一个正常的程序。要在命令行上的程序脚本中使用多条命令，只要在命令之间放个分号即可。

```
$ echo "My name is Rich" | gawk '{$4="Christine"; print $0}'
My name is Christine
$
```

第一条命令会给字段变量$4 赋值。第二条命令会打印整个数据字段。注意， gawk程序在输出中已经将原文本中的第四个数据字段替换成了新值。

也可以用次提示符一次一行地输入程序脚本命令。

```
$ gawk '{
> $4="Christine"
> print $0}'
My name is Rich
My name is Christine
$
```

在你用了表示起始的单引号后，bash shell会使用次提示符来提示你输入更多数据。你可以每次在每行加一条命令，直到输入了结尾的单引号。因为没有在命令行
中指定文件名，gawk程序会从STDIN 中获得数据。当运行这个程序的时候，它会等着读取来自STDIN 的文本。要退出程序，只需按下Ctrl+D组合键来表明数据结束。

5. 从文件中读取程序

跟sed编辑器一样，gawk编辑器允许将程序存储到文件中，然后再在命令行中引用。

```
$ cat script2.awk
{print $1 "'s home directory is " $6}
$
$ gawk -F: -f script2.gawk /etc/passwd
root's home directory is /root
bin's home directory is /bin
daemon's home directory is /sbin
adm's home directory is /var/adm
lp's home directory is /var/spool/lpd
[...]
Christine's home directory is /home/Christine
Samantha's home directory is /home/Samantha
Timothy's home directory is /home/Timothy
$
```

script2.gawk程序脚本会再次使用print 命令打印/etc/passwd文件的主目录数据字段（字段变量$6 ），以及userid 数据字段（字段变量$1 ）。

可以在程序文件中指定多条命令。要这么做的话，只要一条命令放一行即可，不需要用分号。

```
$ cat script3.gawk
{
text = "'s home directory is "
print $1 text $6
}
$
$ gawk -F: -f script3.gawk /etc/passwd
root's home directory is /root
bin's home directory is /bin
daemon's home directory is /sbin
adm's home directory is /var/adm
lp's home directory is /var/spool/lpd
[...]
Christine's home directory is /home/Christine
Samantha's home directory is /home/Samantha
Timothy's home directory is /home/Timothy
$
```

script3.gawk程序脚本定义了一个变量来保存print 命令中用到的文本字符串。注意，gawk程序在引用变量值时并未像shell脚本一样使用美元符。

6. 在处理数据前运行脚本

gawk还允许指定程序脚本何时运行。默认情况下，gawk会从输入中读取一行文本，然后针对该行的数据执行程序脚本。有时可能需要在处理数据前运行脚本，比如为报告创建标题。BEGIN 关键字就是用来做这个的。它会强制gawk在读取数据前执行BEGIN 关键字后指定的程序脚本。

```
$ gawk 'BEGIN {print "Hello World!"}'
Hello World!
$
```

这次print 命令会在读取数据前显示文本。但在它显示了文本后，它会快速退出，不等待任何数据。如果想使用正常的程序脚本中处理数据，必须用另一个脚本区域来定义程序。

```
$ cat data3.txt
Line 1
Line 2
Line 3
$
$ gawk 'BEGIN {print "The data3 File Contents:"}
> {print $0}' data3.txt
The data3 File Contents:
Line 1
Line 2
Line 3
$
```

在gawk执行了BEGIN脚本后，它会用第二段脚本来处理文件数据。这么做时要小心，两段脚本仍然被认为是gawk 命令行中的一个文本字符串。你需要相应地加上单引号。

7. 在处理数据后运行脚本

与BEGIN 关键字类似，END 关键字允许你指定一个程序脚本，gawk会在读完数据后执行它。

```
$ gawk 'BEGIN {print "The data3 File Contents:"}
> {print $0}
> END {print "End of File"}' data3.txt
The data3 File Contents:
Line 1
Line 2
Line 3
End of File
$
```

当gawk程序打印完文件内容后，它会执行END脚本中的命令。这是在处理完所有正常数据后给报告添加页脚的最佳方法。

可以将所有这些内容放到一起组成一个漂亮的小程序脚本文件，用它从一个简单的数据文件中创建一份完整的报告。

```
$ cat script4.gawk
BEGIN {
print "The latest list of users and shells"
print " UserID \t Shell"
print "-------- \t -------"
FS=":"
}

{
print $1 "      \t "  $7
}

END {
print "This concludes the listing"
}
$
```

这个脚本用BEGIN脚本来为报告创建标题。它还定义了一个叫作FS 的特殊变量。这是定义字段分隔符的另一种方法。这样你就不用依靠脚本用户在命令行选项中定义字段分隔符了。

下面是这个gawk程序脚本的输出。

```
$ gawk -f script4.gawk /etc/passwd
The latest list of users and shells
 UserID          Shell
--------         -------
root             /bin/bash
bin              /sbin/nologin
daemon           /sbin/nologin
[...]
Christine        /bin/bash
mysql            /bin/bash
Samantha         /bin/bash
Timothy          /bin/bash
This concludes the listing
$
```

与预想的一样，BEGIN脚本创建了标题，程序脚本处理特定数据文件（/etc/passwd）中的信息，END脚本生成页脚。

这个简单的脚本让你小试了一把gawk的强大威力。