# 文本处理

背景：需要自动处理文本文件，能够轻松实现自动格式化，插入，修改，或删除文本元素，linux 系统中提供两个常见的具备上述功能的工具，

## sed

sed = stream editor,流编辑器，特点：在编辑器处理数据之前基于预先提供的一组规则来编辑数据流。与交互式文本编辑器(vim)相反。

Sed可以根据命令来处理数据流中数据，这些命令要么从命令行中输入，要么存储在一个命令文本文件中。Sed会执行下列操作

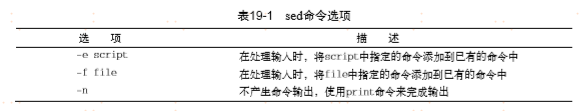
1. 一次从输入中读取一行数据
2. 根据所提供的编辑器命令匹配数据
3. 按照命令修改流中的数据
4. 将新的数据输入到到STDOUT

在流编辑器将所有命令与一行数据匹配完毕后，它会读取下一行数据并重复这个过程。在流编辑器处理完流中所有数据行后，它将会终止。由于命令是按照顺序逐行给出的，sed编辑器要比交互式编辑器快得多，可以快速完成对数据的自动修改

Sed命令的格式如下

Sed options script file

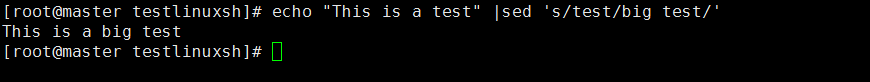
选项允许修改sed命令的行为



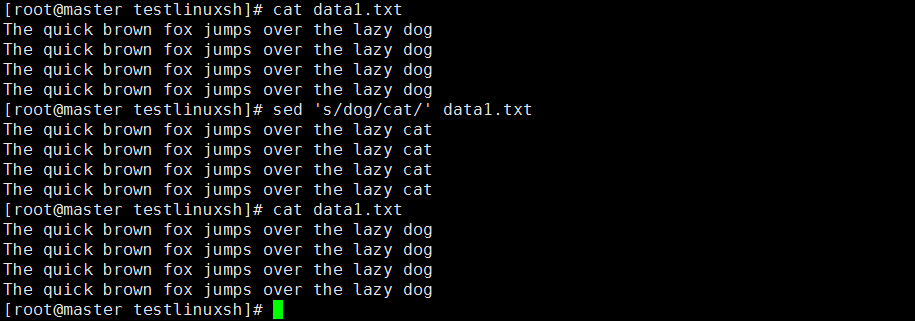
Script参数指定了应用于流数据上的单个命令，如果需要用多个命令，那么使用-e选项在命令行中指定，要么使用-f 选项在单独的文件中指定。有大量的命令可以用来处理数据

1. 在命令行定义编辑器命令

默认情况下，sed编辑器会将指定的命令应用到STDIN输入流中，这样就可以直接将数据通过管道输入sed编辑器处理



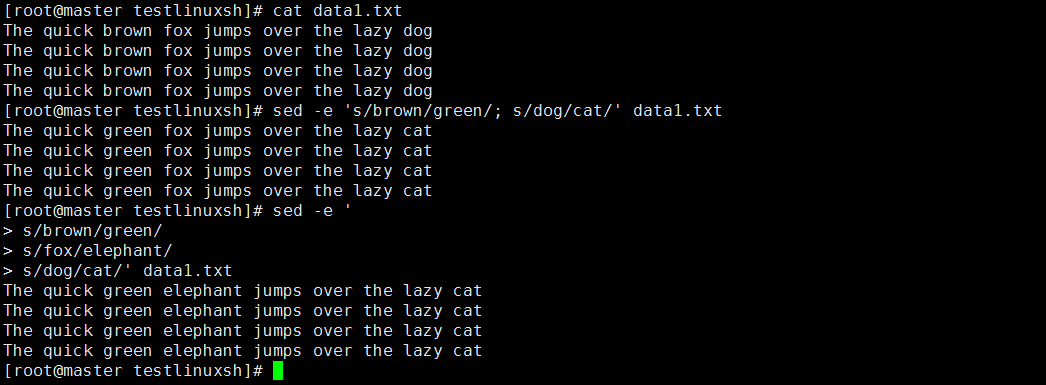
在这个例子中，sed编辑器使用了s命令，s命令会用斜线间指定的第二个文本字符串来替换第一个文本字符串模式。本例中就是用big test替换test.这个例子中只是修改一行数据，但也可以编辑整个文件，处理速度也很快。



重点： sed编辑器并不会修改文本文件中的数据，它只会将修改后的数据发送到STDOUT。上例中可以明显发现。

1. 在命令行使用多个编辑器命令

要在sed命令行中执行多个命令时，只要用-e选项就可以了



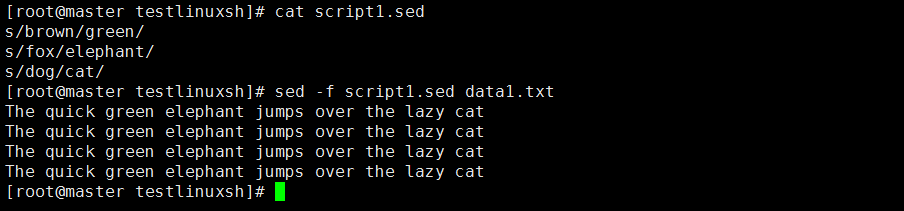
两个命令都作用到文件中每行数据中。命令之间必须用分号隔开，并且在命令行末尾和分号之间有空格。

如果不用分号，也可以用bash shell中的次提示符来分隔命令，只要输入第一个单引号标示出sed程序脚本的其实，bash会继续提示你输入更多命令，知道输入标示结束的单引号。

重点：要在封尾单引号所在行结束命令。Bash shell一旦发现了封尾的单引号，就会执行命令。开始后，sed命令就会将你指定的每条命令应用到文本文件中的每一行上。

1. 从文件中读取编辑器命令

如果有大量需要处理的命令，那么将它们放进一个单独文件中通常更方便一些，可以在sed命令中用-f选项指定文件。



在这种情况下，不用在每条命令后面放一个分号。Sed编辑器知道每行都是一条单独的命令。跟在命令行输入命令一样，sed编辑器会在指定文件中读取命令，并将它们应用到数据文件中每一行上。

窍门：通常使用.sed作为sed脚本文件的扩展名

## 2.gawk

背景：通常你需要一个用来处理文件中数据的更高级的工具，它能提供一个类编程环境来修改和重新组织文件中数据。Gawk程序让流编辑器迈上一个新的台阶，它提供了一种编程语言而不只是编辑器命令。

Gawk编程语言中，可以完成下面功能

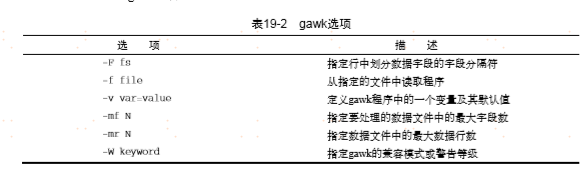
1. 定义变量来保存数据
2. 使用算术和字符串操作符来处理数据
3. 使用结构化编程概念【比如if-then语句和循环】来为数据处理增加逻辑
4. 通过提取数据文件中的数据元素，将其重新排列或格式化，生成格式化报告。

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。

### gawk命令格式

gawk程序的基本格式如下:

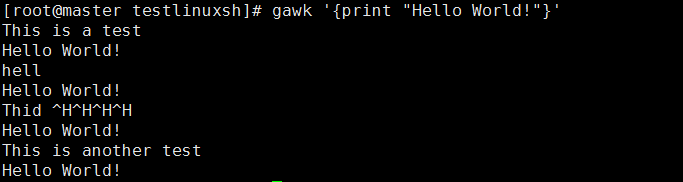
gawk options program file



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。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。

### 从命令行读取程序脚本

Gawk程序脚本用一对花括号定义。必须将脚本命令放到两个花括号（{}）中，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。



### 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：

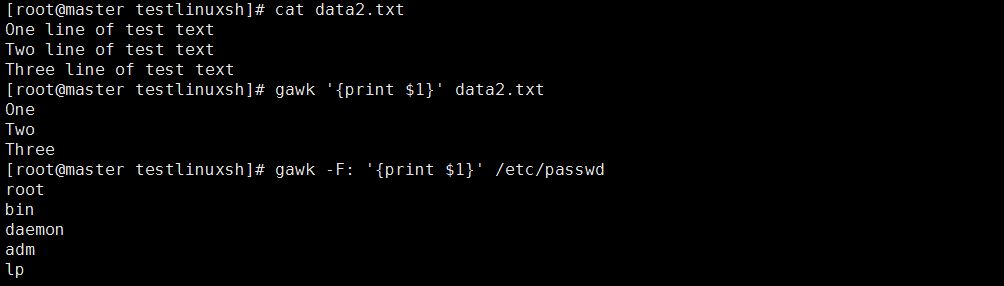
$0代表整个文本行

$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

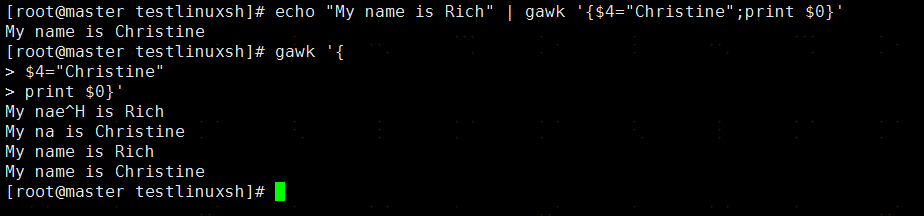
在文本行中，每个数据字段都是通过字段分割符划分的，gawk在读取一行文本时，会用预定义的字段分割符划分每个数据字段。Gawk中默认的字段分割符是任意的空白字符（例如空格或制表符）



说明：第一个程序中用$1字段变量来仅显示每行第1个数据字段 ，第二个程序中由于采用了冒号来分割数字字段，所以用-F选项指定。

### 在程序脚本中使用多个命令

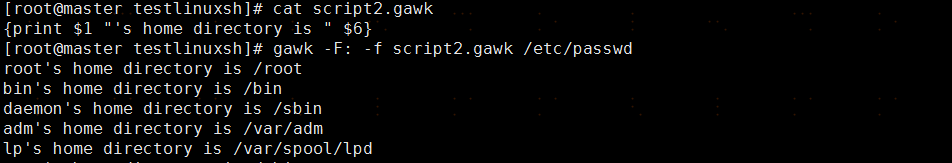
Gawk编程语言允许将多条命令组合成一个正常的程序，要在命令行上的程序脚本中使用多条命令，只要在命令之间放多个分号即可



说明: $4=”Christne”,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中的第四个数据字段替换了新值。

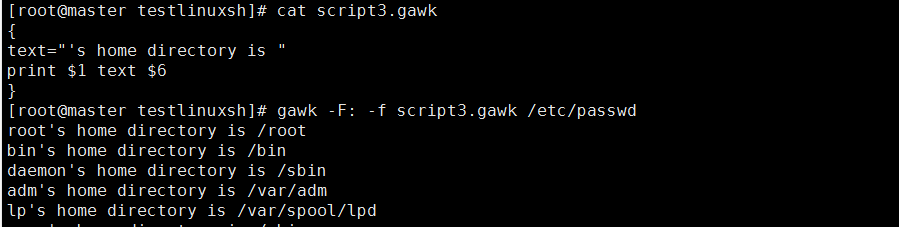
### 从文件中读取程序

与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



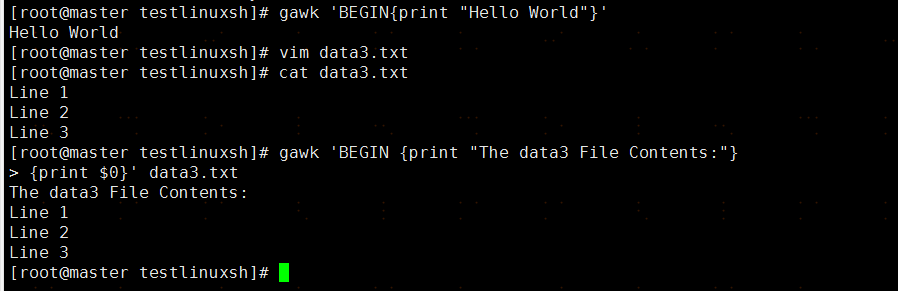
Script2.gawk程序脚本会再次使用print命令打印/etc/passwd文件的主目录数据字段【字段变量$6】,以及userid数据字段【字段变量$1】.

可以在程序文件中指定多条命令，如果这样的，只要一条命令放一行即可，不需要用分号。



### 在处理数据前运行脚本

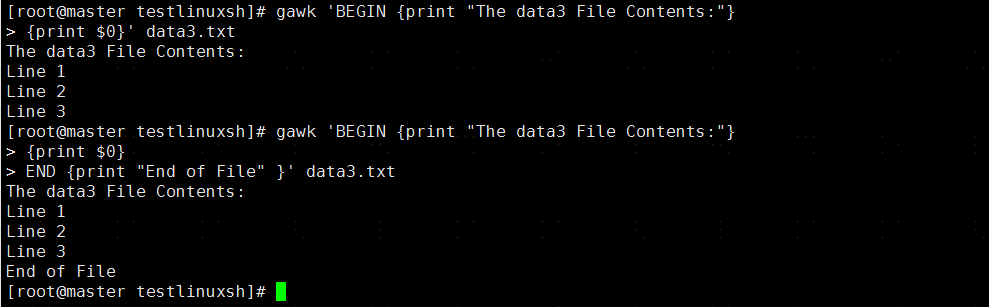
Gawk还允许指定程序脚本何时运行。默认情况下，gawk会从输入中读取一行文本，然后针对该行的数据执行程序脚本。有时可能需要在处理数据前运行脚本，比如为报告创建标题。BEGIN关键字就是用来做这个的，它会强制gawk读取数据前执行BEGIN关键字后指定的程序脚本。



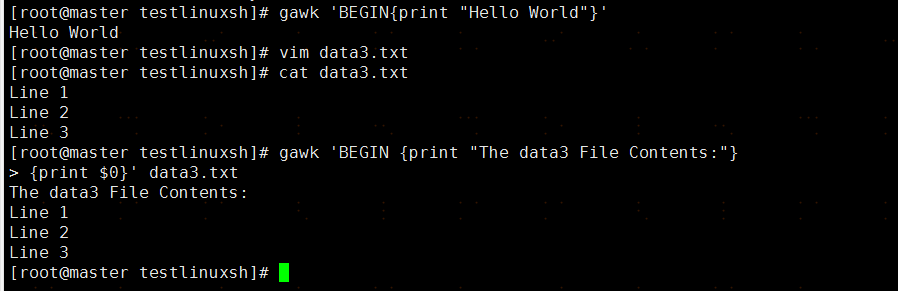
说明：第一个程序中，print命令会在读取数据前显示文本。但在显示文本后，它会快速退出，不等待任何数据。如果想要使用正常的程序脚本处理数据，必须用另一个脚本区域来定义程序。如第二个程序中所显示，在gawk执行BEGIN脚本后，它会用第二段脚本处理文件数据，这样做必须小心，两段脚本仍然被认为时gawk命令行中的一个文本字符串。需要加上相应的单引号。

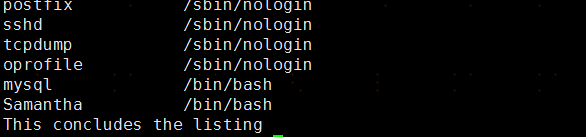
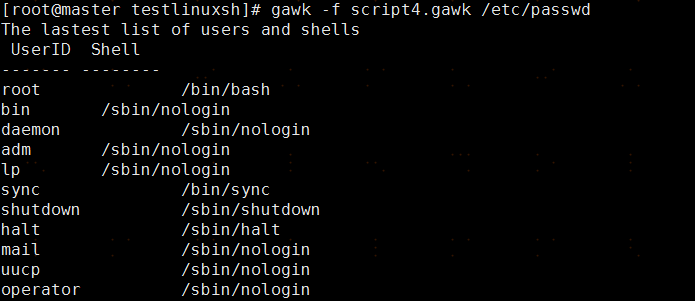
### 在处理数据前运行脚本

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。



说明：当gawk程序打印完文件内容后，它会执行END脚本的名利，这是处理完所有正常数据后给报告添加页脚的最佳方法





说明：还可以将所有内容放到一起组成一个小程序脚本文件，用它从一个简单的数据文件中创建一份完整的报告

# Sed编辑器基础

### 更多的替换项

除了使用s命令（substitute）来在行中替换命令，还有其他选项

#### 替换标记

替换命令在替换多行中的文本时能正常工作，但默认情况下它只替换每行中出现的第一处。要让替换命令能够替换一行中不同地方出现的文本必须使用替换标记，替换标记会在替换命令字符串之后设置。

s/pattern/replacement/flags

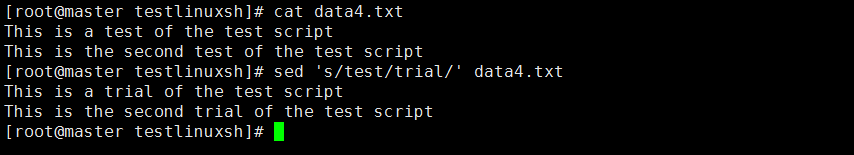
有4种可用的替换标记：

数字：表明新文本将替换第几处模式匹配的地方

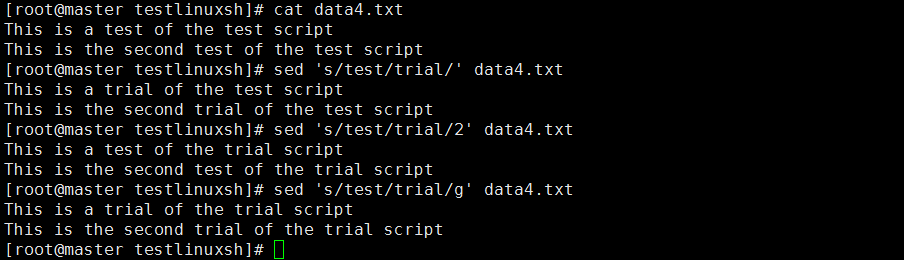
G:表明新文本将会替换所有匹配的文本

P:表明原先行的内容要打印出来

W file 将替换的结果写到文件中

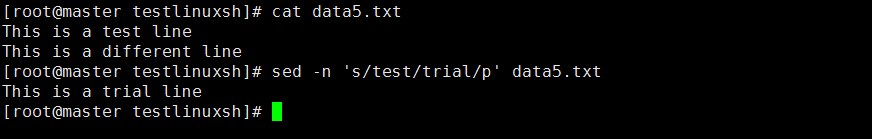


备注：默认情况下只是替换出现的第一处

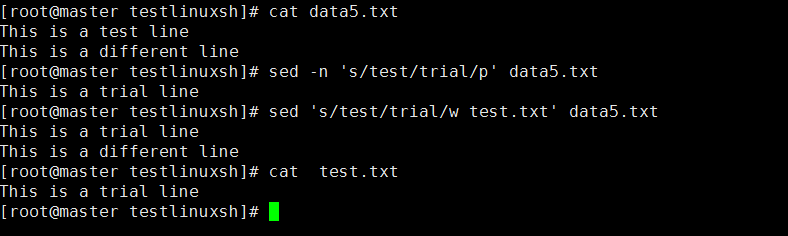


说明：将替换标记指定为2的结果是：编辑器只替换每行中第二次出现的匹配模式

G替换标记使你能替换文本中匹配模式所匹配的每处地方



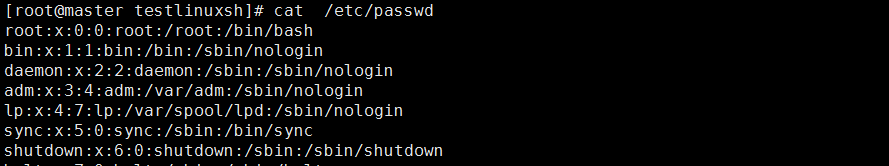
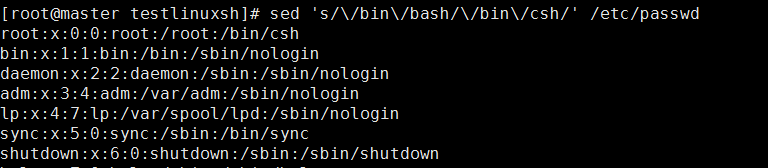
说明：p替换标记会打印与替换命令中指定的模式匹配的行，这通常会和sed的-n选项一起使用， -n选项将禁止sed编辑器输出，但p醍醐标记会输出修改过的行，将二者配合使用的效果就是只输出被替换命令修改过的行



说明：w替换标记会产生同样的输出，不过会将输出保存到指定文件中

#### 替换字符

有时会在文本字符串中遇到一些不太方便在替换模式中使用的字符，Linux中一个常见的例子时就是正斜线，替换文件名中的路径名会比较麻烦。由于正斜线通常用作字符串分割符，因而如果它出现在模式文本中的话，必须用反斜线来转义，要解决这个问题，sed编辑器允许选择其他字符来作为替换命令中的字符串分隔符



### 2.使用地址

默认情况下，在sed编辑器中使用命令会作用与文本数据的所有行，如果只想命令作用于特定行或某些行，则必须用行寻址(line addressing)

在sed编辑器中有两种形式的行寻址

以数字形式表示行区间

用文本模式来过滤出行

两种形式都使用相同的格式来指定地址：[address]command,也可将特定地址的多个命令分组：

Address{

Command1

Command2

Command3

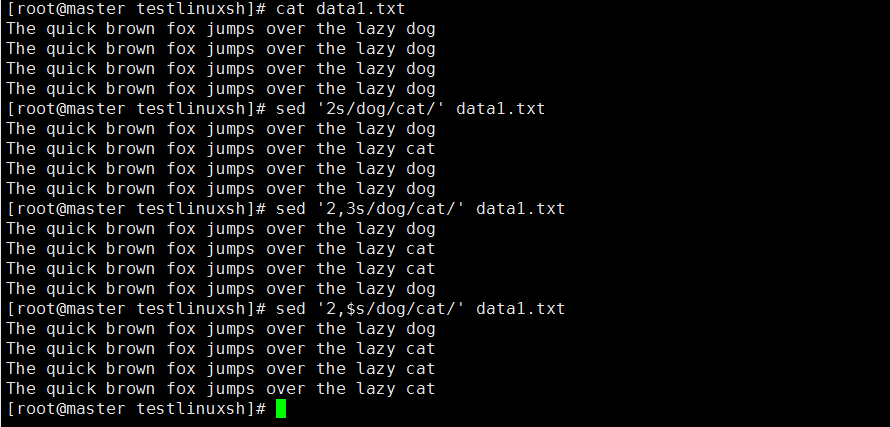
}

Sed编辑器会将指定的每条命令作用到匹配指定地址的行上。

#### 数字方式的行寻址

当使用数字方式的行寻址时，可以用行在文本流种的行位置来引用，sed编辑器会将文本流中第一行编号为1，然后继续按照顺序为接下来的行分配行号。

在命令行中指定的地址可以是单个行号，或是用起始行号，逗号以及结尾行号指定的一定区间范围内的行。



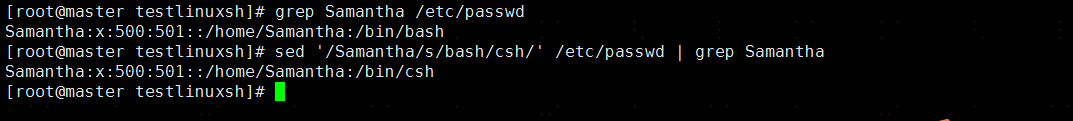
说明：第一个程序只修改地址指定的第二行的文本，第二个程序使用行地址区间，修改第2行到第3行的文本，第三个程序表明如果想命令作用到从某行开始的所有行，可以用特殊地址—美元符

#### 使用文本模式过滤器

Sed编辑器允许指定文本模式来过滤出命令要作用的行，格式如下：/pattern/command

必须用正斜线将要指定的pattern封起来。Sed编辑器会将该命令作用到包含指定文本模式的行中

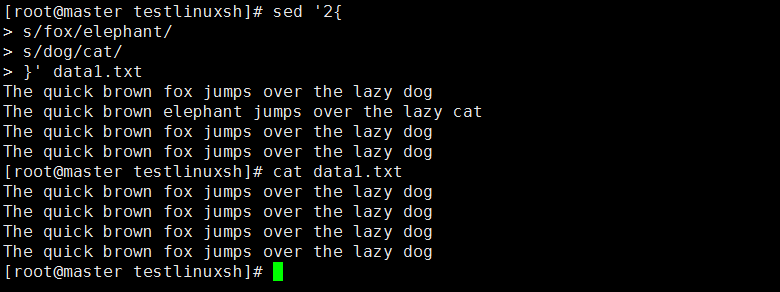
实例：修改用户Samantha的默认shell



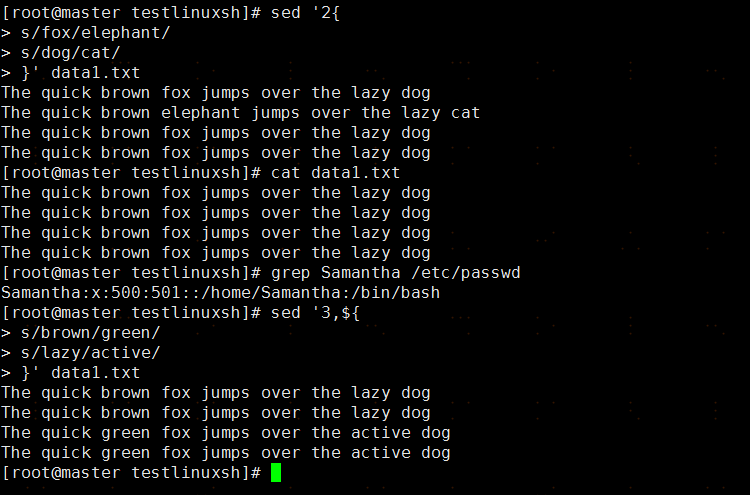
说明：该命令只作用到匹配文本模式的行上。

#### 命令组合

如果在单行上执行多条命令，可以用花括号将多条命令组合在一起，sed编辑器会处理地址行处列出的每条命令



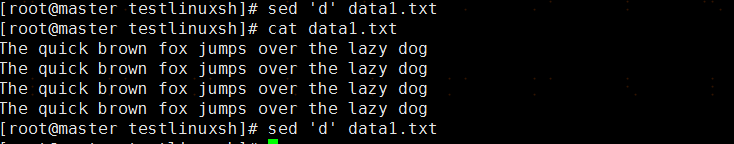
说明：可以在一组命令前指定一个区间，该程序在替换第二行文本

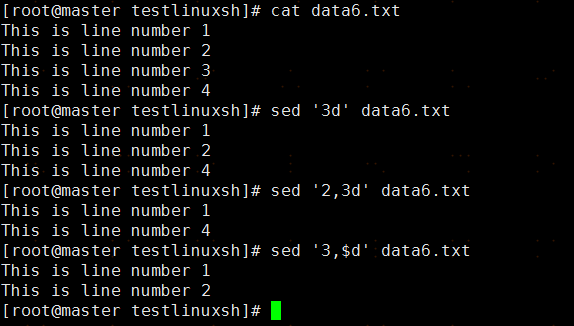


说明：将命令作用于第三行到最后一行中

### 3.删除行

删除命令d，它会删除匹配指定寻址模式的所有行。使用该命令时要特别小心，如果忘记加入寻址模式，流中所有文本行会被删除。如下图所示

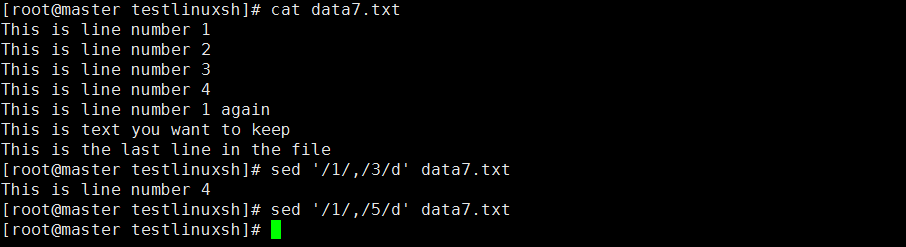




当删除命令和指定地址一起使用时，删除命令显而易见能发挥更大的作用，可从数据流中删除特定的文本行，如第一个程序中删除第三行文本，第二个程序中删除第二到第三行文本，最后一个程序删除从第三行到最后一行的文本

可以使用两个文本模式来删除某个区间的行，但是这时需要小心，指定的第一个模式会打开删除的功能，第二个模式会关闭行删除功能。Sed编辑器会删除两个指定行之间的所有行（包括指定的行）

另外，要特别小心的是，因为只要sed编辑器在数据流中匹配到开始模式，删除功能就会打开，这可能导致意外的结果发生。



说明：当出现第二个数字1的行再次触发删除命令，因为没有找到停止模式，所以将数据流中剩余行全部删除。当然如果指定了一个从未在文本中出现的停止模式，显而易见会出现另外一个问题，如第二个程序所示，因为删除功能在匹配到第一个模式的时候打开了，但一直没有匹配到结束模式，所以整个数据流都被删掉。

### 4.插入和附加文本

Sed编辑器允许向数据流插入和附加文本行，两个操作的区别如下：

插入(insert)命令(i)会在指定行前增加一个新行

附加(append)命令(a)会在指定行后增加一个新行

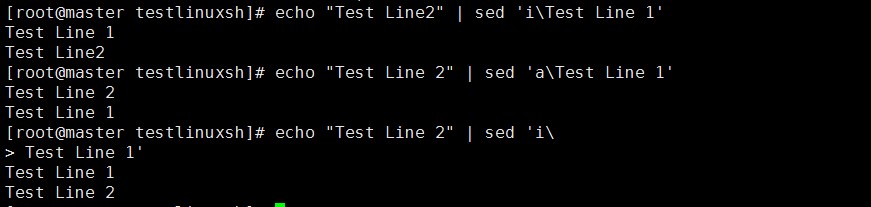
这两条命令的费解之处在于他们的格式。它们不能在单个命令行上使用，你必须指定是要将行插入还是附加到另一行。格式如下：

Sed ‘[address]command\new line’

New line中的文本将会出现在sed编辑器输出中你指定的位置。

记住，当使用插入命令时，文本会出现在数据流文本的前面。当使用附加命令时，文本会出现在数据流文本后面：如下图实例所示。

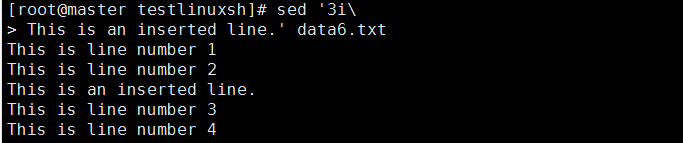
注意: 在命令行界面提示符上使用sed编辑器时，会看到次提示符提醒输入新的行数据，必须在该行完成sed编辑器命令，一旦输入了结尾的单引号，bash shell就会执行该命令



在数据流中文本前面或后面添加文本，但是如何在数据流内部添加文本？

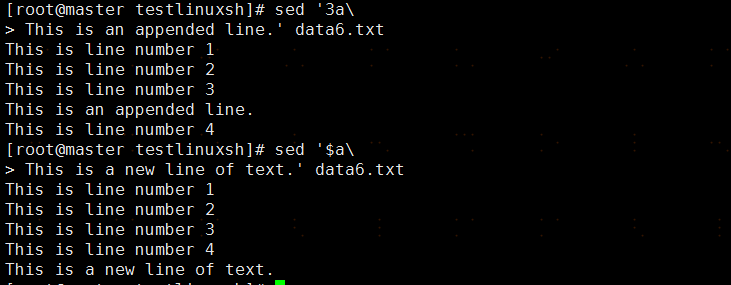
标准做法：通过寻址来告诉编辑器你想让数据出现在什么位置，可以在用这些命令时只指定一个行地址。可以匹配一个数字行号或文本模式，但不能用地址区间。

实例 1：将一个新行插入数据流第三行前

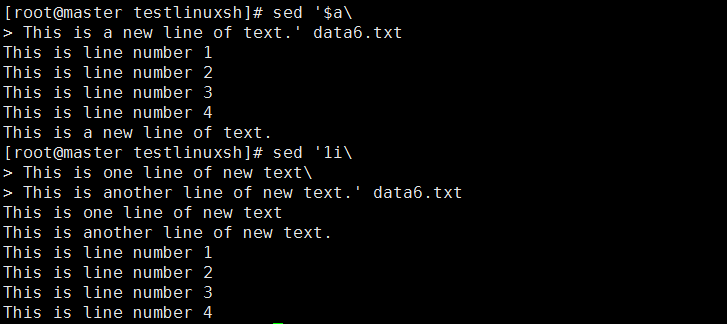


实例2：将一个新行附加到数据流中第三行后

实例3：将一个新行附加到数据流的末尾，只要用代表数据最后一行的美元符就可以了



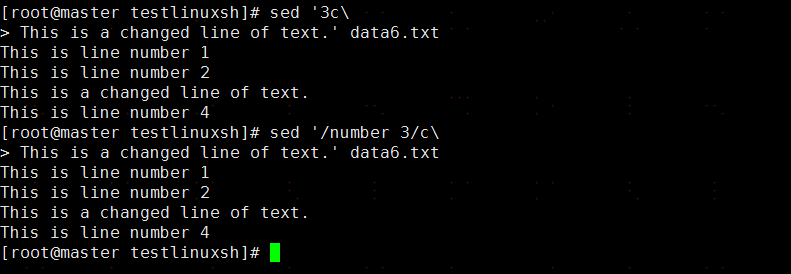
实例4：要插入或附加多行文本是，必须对新文本中的每一行使用反斜线，直到最后一行



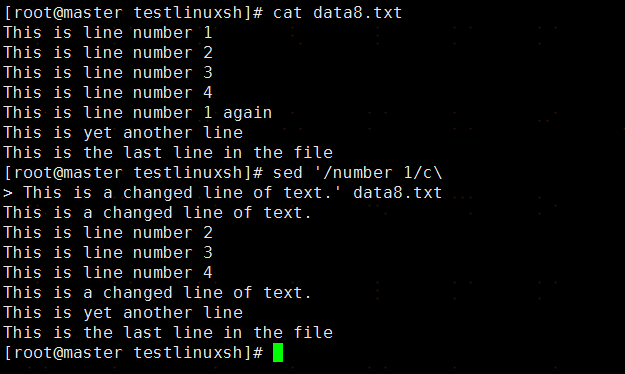
### 5.修改行

修改[change]命令允许修改数据流中整行文本的内容，它跟插入和附加命令的工作机制一样，必须在sed中单独指定新行

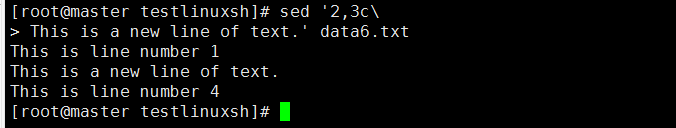
实例1. Sed编辑器会修改第三行中的文本，也可以用文本模式来寻址



实例2.文本模式修改命令会修改匹配数据流中任意文本行



实例3.特别注意：修改命令行中使用地址区间，sed编辑器会用一行文本替换数据流中两行文本，而不是逐一修改这两行文本

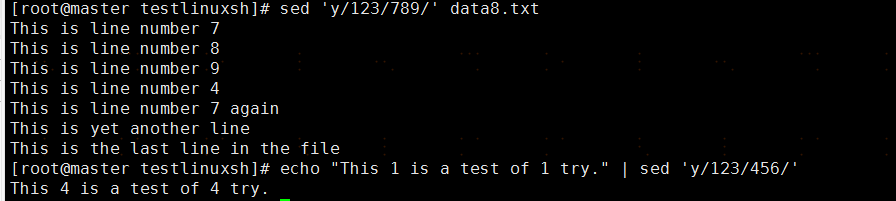


### 6.转换行

转换(transform)命令(y)是唯一可以处理单个字符sed编辑器命令，转换命令格式如下：

[address]/inchars/outchars,

转换命令会对inchars 和outchars值进行一对一的映射。Inchars与outchars的长度必须相同，否则会产生一条错误



### 7.打印

Sed中除了使用p标记和替换命令显示sed编辑器修改过的行，另外3个命令也能用来打印数据流中信息

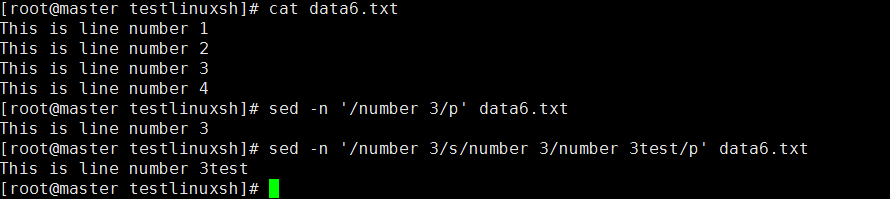
P命令用来打印文本行

等号(=)命令用来打印行号

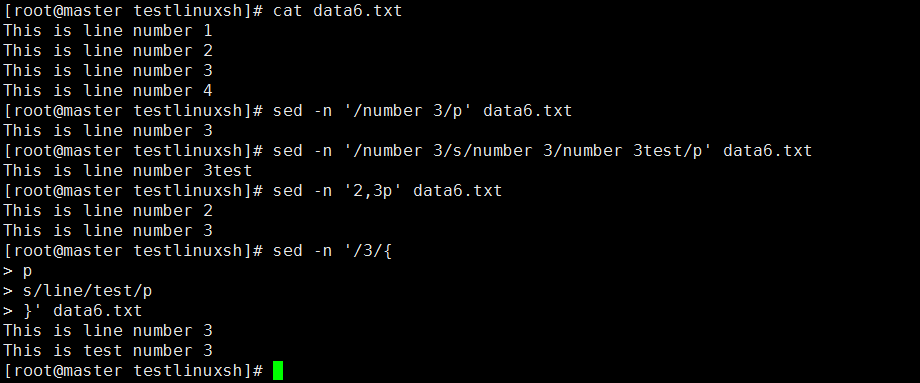
L(小写的L)命令用来列出行

#### 1.打印行

实例1：打印包含匹配文本模式的行，-n禁止输出其他行，只打印包含匹配文本模式的行



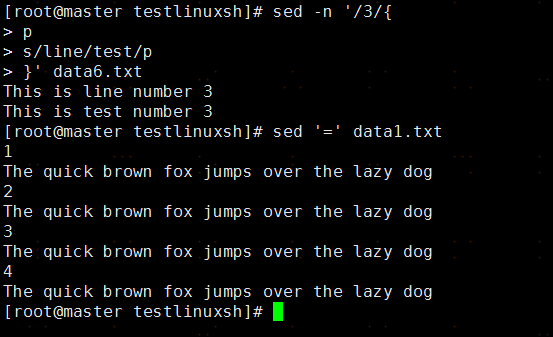
实例2：编辑器会查找包含数字3的行，然后执行两条命令，首先用p命令打印出原来的行，然后它用s命令替换文本，并用p标记打印出替换结果，输出同时显示出原来的行文本和新的行文本



#### 2.打印行号

等号命令会打印在数据流中当前行号，行号由数据流中的换行决定，每次数据流中出现一个换行符，sed编辑器会认为一行文本结束了。

实例 1：打印行号



Sed编辑器在实际的文本行出现前打印行号

实例2： 查找特定文本模式的行号



#### 3.列出行

### 8.使用sed处理文件

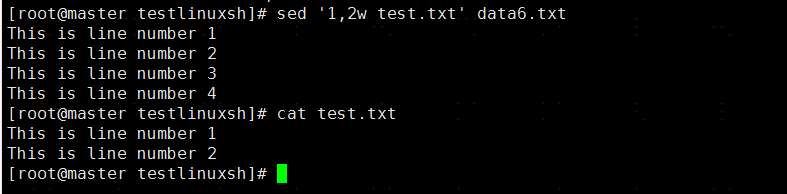
替换命令包含一些可以用于文件的标记。还有一些sed编辑器命令也可以实现同样的目标，

#### 8.1.写入文件

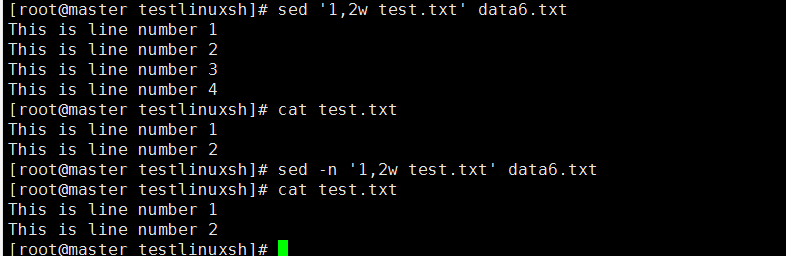
W命令用来向文件中写入行，该命令的格式如下： [address ] w filename

Filename可以使用相对路径或绝对路径，但是不管是哪种，运行sed编辑器的用户必须有文件写的权限。地址可以是sed中支持的任意类型的寻址方式，如单个行号，文本模式，行区间，文本模式

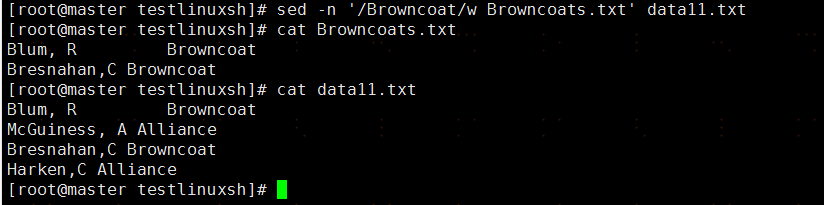
实例1： 将数据流中前两行打印到一个文本文件中



如果不想让行显示到STDOUT上，可以用sed命令中-n的选项



实例2：根据一些公用的文本值从主文件中创建一份数据文件

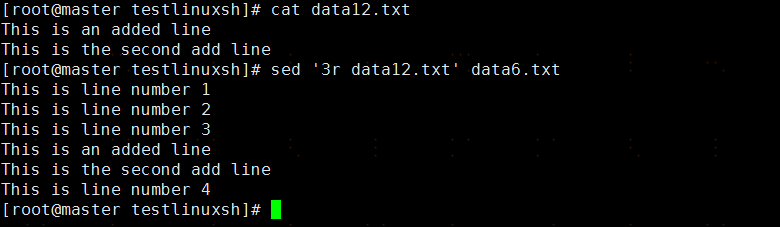


#### 8.1.读取文件

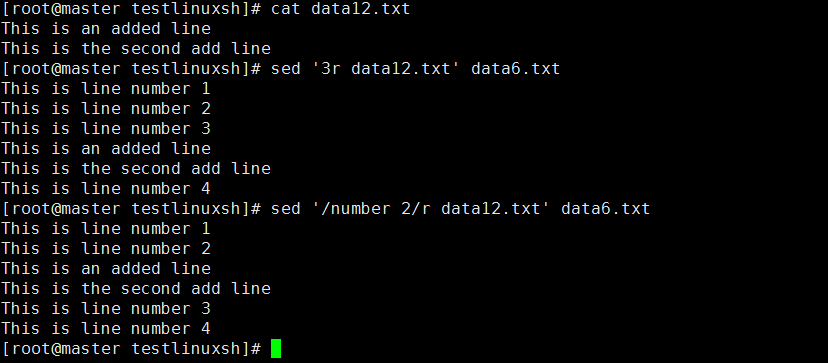
读取命令允许将一个独立文件中数据插入到数据流中，读取的命令格式如下:

[address] r filename

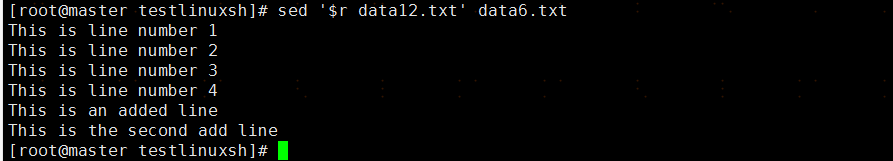
实例1：将数据中所有文件插入到数据流中



实例2：使用文本模式读取



实例3： 数据流末尾添加文本



实例4：与删除命令配合使用

