# 正则表达式

背景：在shell脚本中成功使用 sed编辑器和gawk程序的关键是在于熟练使用正则表达式

## 正则表达式基本概念

### 定义

正则表达式是你所定义的模式模板[pattern template]，linux工具可以用它来过滤文本。Linux工具[sed编辑器或gawk程序]能够在处理数据时使用正则表达式对数据进行模式匹配。如果数据匹配，它将会被进一步接受处理，如果不匹配，它就会过滤掉

### 2.正则表达式类型

正则表达式是通过正则表达式引擎[regular expression engine]来实现的。正则表达式引擎是一套底层软件，负责解释正则表达式模式并使用这些模式进行文本匹配的。

在Linux中，有两种流行的正则表达式引擎:

1. POSIX基础正则表达式(basic regular expression BRE)引擎
2. POSIX扩展正则表达式(extended regular expression ERE)引擎

大多数Linux工具都至少符合POSIX BRE引擎规范，能够识别该规范定义的所有模式符号，不过，有些工具【sed】只符合BRE引擎规范的子集，这是出于速度方面的考虑导致的，因为sed希望能够尽快处理数据流中文本。

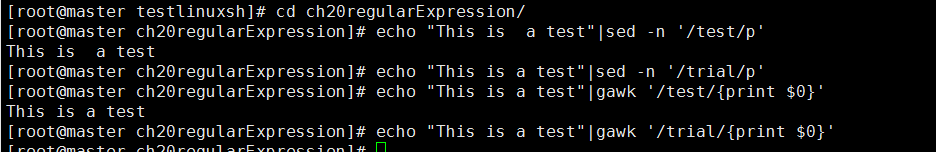
POSIX ERE引擎通常出现在依赖正则表达式进行文本过滤的编程语言中，它为常见模式提供了高级模式符号和特殊符号，比如匹配数字，单词，以及字母的字符。Gawk程序用ERE引擎来处理他的正则表达式模式。

## 2.定义BRE模式

背景：最基本的BRE模式是匹配数据流中文本字符。

### 纯文本

复习：在sed与gawk程序中使用标准文本字符串来过滤数据



说明：

第一个模式定义了一个单词test,sed编辑器和gawk程序脚本用他们各自的print命令打印出匹配该正则表达式模式的所有行。由于echo语句在文本字符串中包含了单词test，数据流文本能够匹配所定义的正则表达式模式，因此sed编辑器显示了该行

第二个模式也定义一个单词trial,因为echo语句文本字符串中没有包含该单词，所以正则表达式模式没有匹配，因此sed编辑器和gawk程序都没有打印该行。

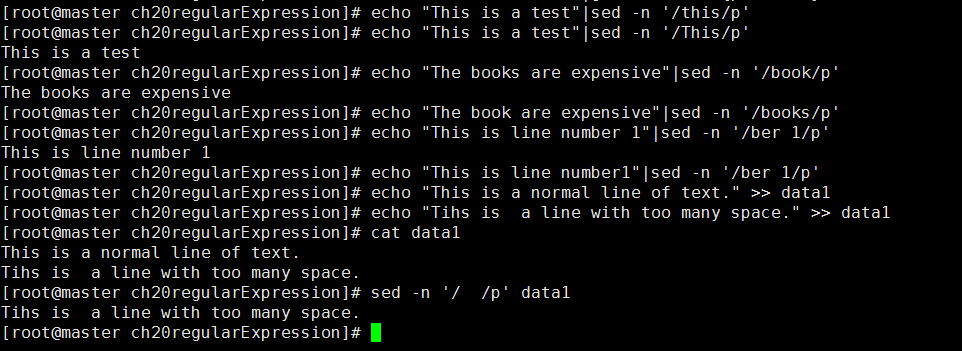
注意点：

正则表达式并不关心模式在数据流中的位置，它也不关心模式出现了多少次，一旦正则表达式匹配了文本字符串中任意位置上的模式，它就会将该字符串传回Linux工具。

关键在于将正则表达式模式匹配到数据流文本上，

重点：正则表达式对匹配的模式非常挑剔，第一条原则就是：正则表达式模式都是区分大小写的

实例如下：



说明：

1. 第一个例子中没有能匹配成功，因为this在字符串中并不都是小写，而第二个例子中使用大写字母，所有能正常工作
2. 第三个例子中：在正则表达式中，不用写出整个单词，只要定义的文本出现在数据流中，正则表达式就能够匹配，
3. 第四个例子中：反之不成立，与第三个例子中相反。
4. 第五个例子中：正则表达式中可以使用空格和数字，空格和其他字符并没有什么区别。
5. 第六个例子中，单词中间有两个空格的行匹配正则表达式模式，这是用来查看文本文件空格问题的好办法

### 特殊字符

在正则表达式模式中使用文本字符时，有些字符在正则表达式中有特别的含义，有如下特殊字符

括号: 方括号：[]，花括号:{}, 小括号()

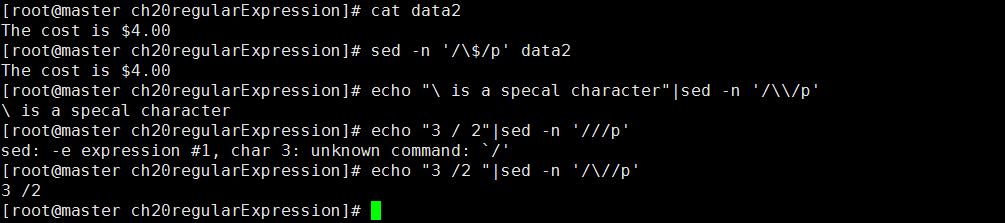
点号与问号： .? 备注：这两个符号在键盘上靠在一起

分号与反斜线：|，\：备注：这两个符号在键盘上位于同一个键

美元符号$

锚字符^

如果要用某个特殊字符作为文本字符，就必须转义。在转义特殊字符时，需要在它前面加一个特殊字符来告诉正则表达式引擎接下来的字符当做普通的文本字符，这个特殊字符就是反斜线。



说明：

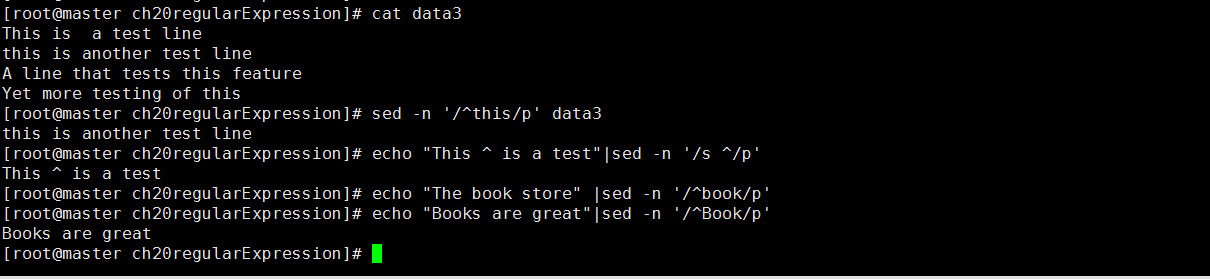
1. 第一个例子中查找文本中的美元符，只要在它前面加一个反斜线
2. 第二个例子中，由于反斜线也是特殊字符，所以模式中要用它，必须转义，这样产生了两个反斜线
3. 第三个例子中，特别注意，尽管正斜线不是正则表达式中的特殊字符，但如果出现在正则表达式中，仍然需要转义

### 锚字符

背景产生： 在默认的情况下，当指定一个正则表达式模式中，只要模式出现在数据流中任何地方，它就能匹配。有两个特殊字符可以用来将模式锁定在数据流中的行首或行尾。

#### 锁定在行首

脱字符【^】定义从数据流中文本行的行首开始的模式。如果模式出现在行首之外的位置，正则表达式模式则无法匹配。要用脱字符，就必须将它放在正则表达式中指定的模式前面。

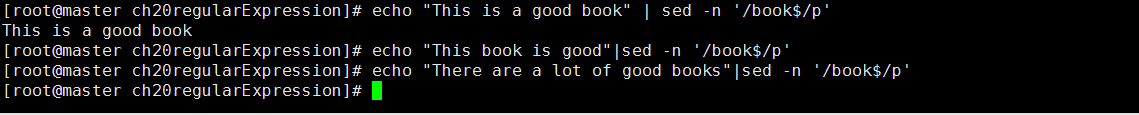


说明：

1. 第一个例子：脱字符会在每个由换行符决定的新数据行的行首检查模式，只要模式出现在新行的行首，脱字符就能够发现它。
2. 第二个例子：如果将脱字符放到模式开头之外的其他位置，那么它跟普通字符一样，不再是特殊字符了。Sed 将它当做普通字符来匹配。
3. 第三个例子与第四个例子中，要使用脱字符，必须将它放在正则表达式中指定的模式前面。

#### 锁定在行尾

特殊字符美元符定义了行尾锚点，将这个特殊字符放在文本模式之后来指明数据行必须以该文本模式结尾。



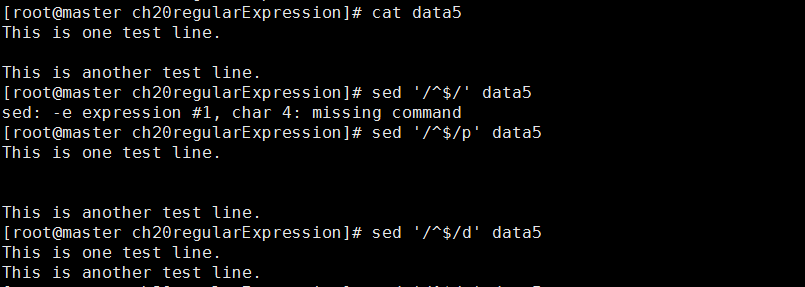
说明：使用结尾文本模式的问题在于你必须留意到底要查找什么

#### 3.组合锚点

背景：在一些常见的情况下，可以在同一行中将行首锚点和行尾锚点组合在一起使用。

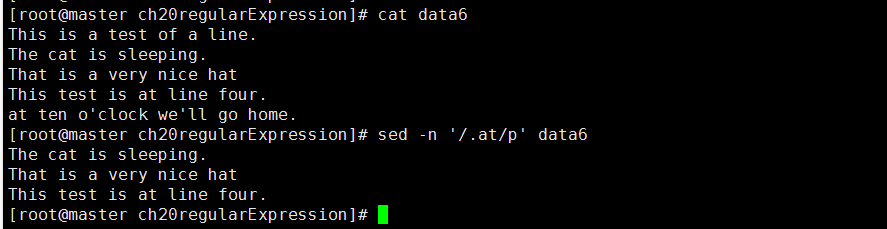
第一种情况中：假定你要查找只含有特定文本模式的数据行

第二种情况中有些怪异，但极其有用，将两个锚点直接组合在一起，之间不加任何文本，这样过滤出数据流中的空白行



### 点号字符

特殊字符点号用来匹配换行符之外的任意单个字符，它必须匹配一个字符，如果在点号字符的位置没有字符，那么模式就不成立。



说明：在正则表达式中，空格也是字符，因此at前面的空格刚好匹配了该模式

### 字符数组

背景：点号字符在匹配某个字符位置上的任意字符时有用，但如果要限定待匹配的具体字符时则用[]，在正则表达式中，称为字符组。可以定义用来匹配文本模式中某个位置的一组字符。如果字符组中某个字符出现在数据流中，那它就匹配该模式

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。

### 排除型字符组

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。

### 区间

### 特殊的字符组

### 星号

# 3.扩展正则表达式

### 问号

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.加号

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

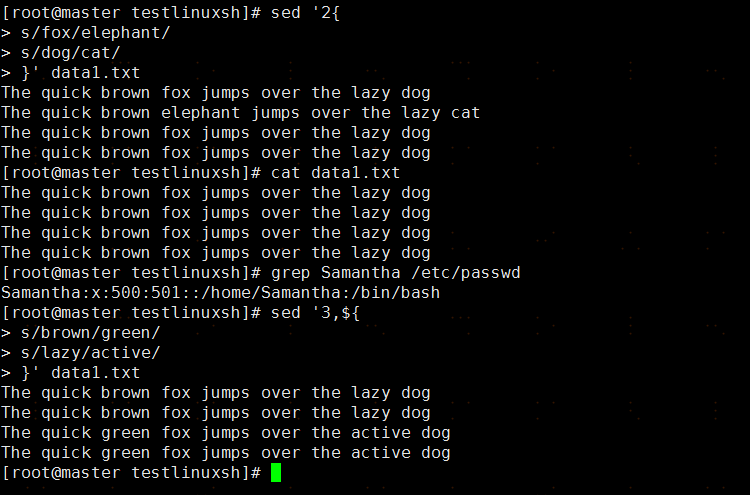
实例：修改用户Samantha的默认shell

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

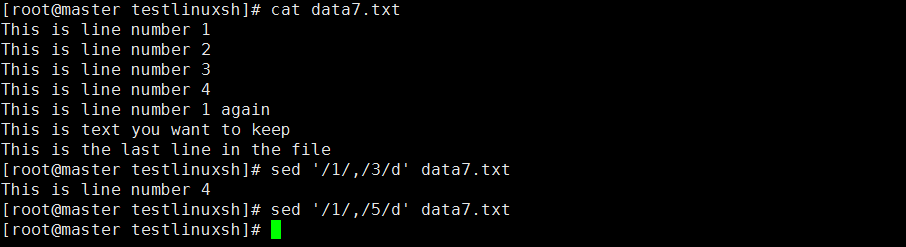
### 3.使用花括号

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.管道符号

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

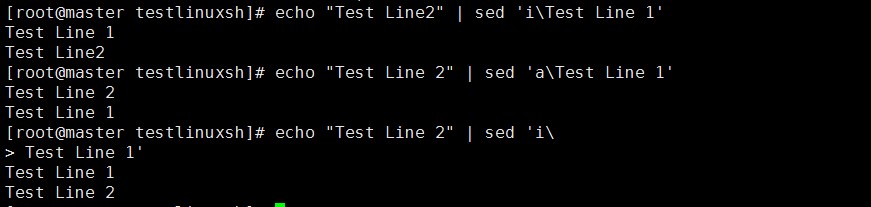
这两条命令的费解之处在于他们的格式。它们不能在单个命令行上使用，你必须指定是要将行插入还是附加到另一行。格式如下：

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。

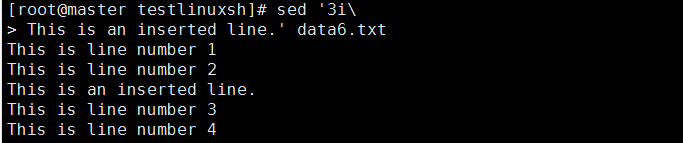
注意: 在命令行界面提示符上使用sed编辑器时，会看到次提示符提醒输入新的行数据，必须在该行完成sed编辑器命令，一旦输入了结尾的单引号，bash shell就会执行该命令



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？

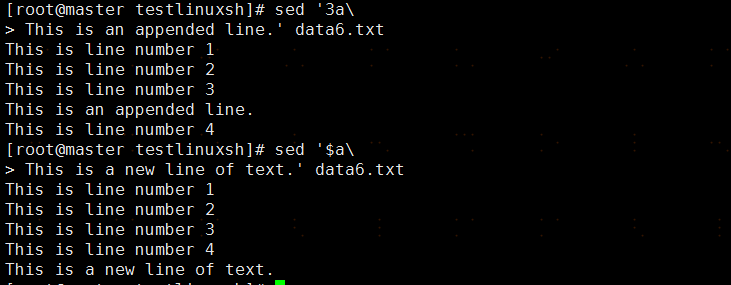
标准做法：通过寻址来告诉编辑器你想让数据出现在什么位置，可以在用这些命令时只指定一个行地址。可以匹配一个数字行号或文本模式，但不能用地址区间。

实例 1：将一个新行插入数据流第三行前

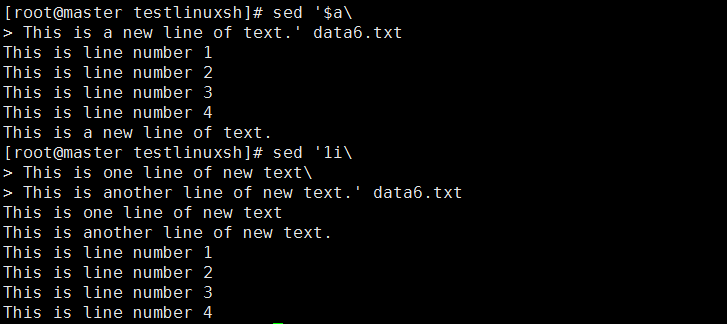


实例2：将一个新行附加到数据流中第三行后

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

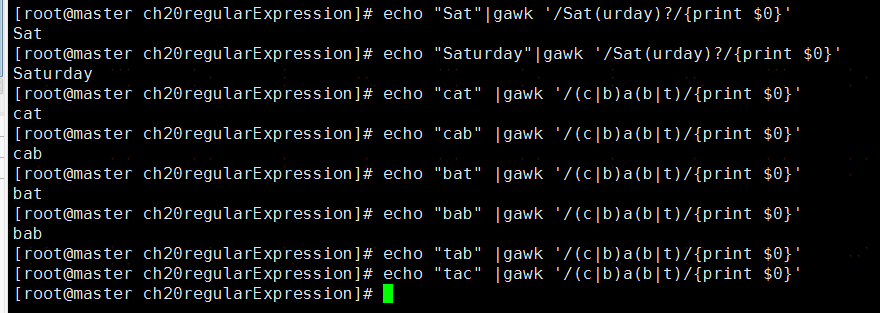


### 5.表达式分组

正则表达式模式可以使用圆括号进行分组，当正则表达式模式分组时，该组会被视为一个标准字符。可以像对普通字符一样给该组使用特殊字符

如下例子1. 结尾的urday分组以及问号，使得模式能够匹配完整的Saturday或者缩写Sat

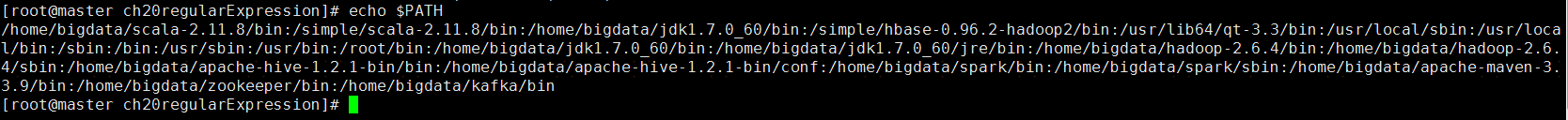
将分组和管道符号一起使用来创建可能的模式匹配，是最常见的做法。



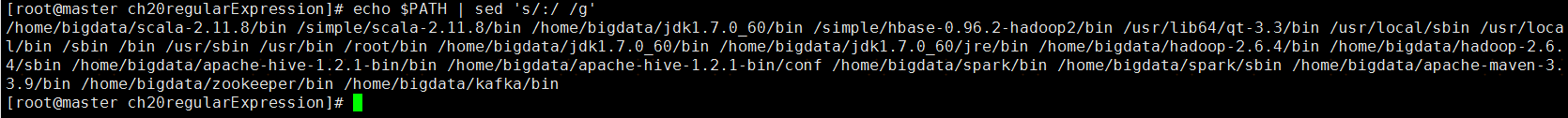
# 正则表达式实战

### 目录文件计数

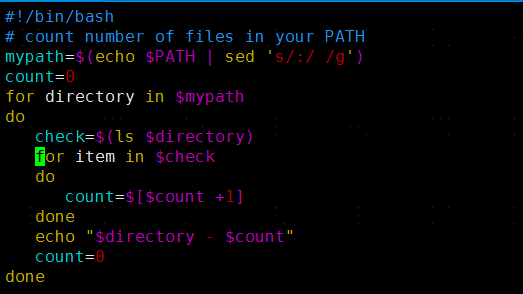
需求：对PATH环境变量中定义的目录的可执行文件进行计数



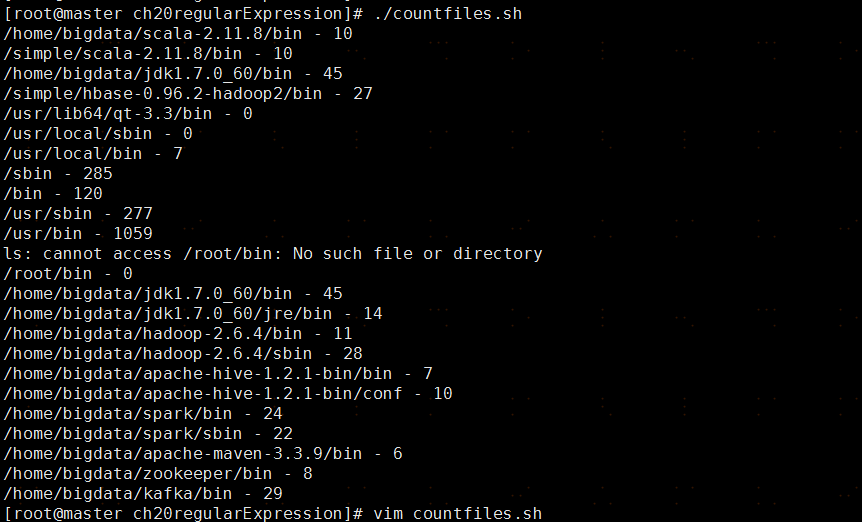
分析1.进行分离



使用for命令，遍历每个目录，获得每个目录后，再用ls列出每个目录的文件，并用另一个for语句遍历每个文件，为每个文件计数器增值。



测试



### 验证电话号码

分析：1.列出电话号码的常见形式

(123)456-7890

(123) 456-7890

123-456-7890

123.456.7890

2.从最左手边开始分析，然后构建用来匹配可能遇到的字符模式

电话号码开始有可能有左圆括号，也可能没有，其圆括号是特殊字符，所以必须使用转义转为普通字符。所以模式为^\(?

紧接着是三位区号，区号在美国是以数字2开始的，最大可以到9，后面跟任意两位数字，

所以模式为[2-9][0-9]{2}

结尾的括号可能存在也可能不存在,所以模式为\)?

所以区号的整个模式为^\(?[2-9][0-9]{2}\)?

1. 区号后面的模式

区号后面存在以下几种情况：有一个空格，没有空格，有一条单破折线，有一个点，所以使用管道符号，并用圆括号分组(| |-|\.)

说明：第一个管道符号紧跟左圆括号后， 用来匹配没有空格的情形，

第二个管道符号与第一个管道符号中间的空格，用来匹配空格的情形

第三个管道符号与第二个管道符号中间的单破折线，用来匹配单破折线

第三个用来匹配点号，因为是特殊字符，所以必须转义

1. 交换机号码

交换机号码为3位数字，所以模式[0-9]{3}

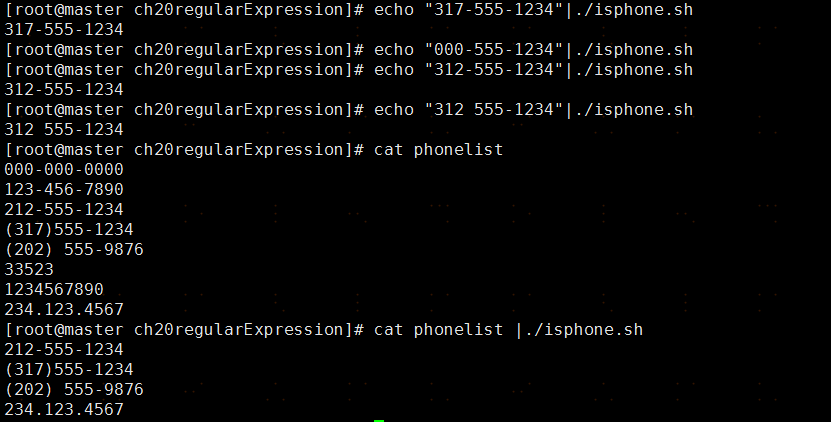
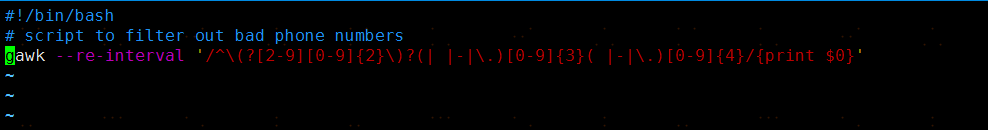
1. 交换机号码后面的

交换机号码后面的则会出现一个空格， 一条单破折线，一个点，所以模式为( |-|\.)

1. 电话分机号

电话分机号为4为数字，并结尾，所以[0-9]{4}$

所以整个模式为^\(?[2-9][0-9]{2}\)?(| |-|\.)[0-9]{3}( |-|\.)[0-9]{4}$



### 3.解析电子邮件地址

分析：邮件地址基本格式如下username@hostname

Username值可以为字母， 数字，点号，单破折号，加号，下划线

则正则表达式模式则如下^([a-zA-Z0-9\-\.\+]+), 最后一个加号表示至少有一个字符。

Hostname值由一个或多个域名和一个服务器名组成，只允许字母，数字，点号，下划线字符， 服务器名和域名都用点隔离，先指定服务器名，紧接着指定子域名，最后是后面不带点号的顶级域名。

顶级域名只能是字母字符，必须不少于两个字符，并且长度不超过5个字符，

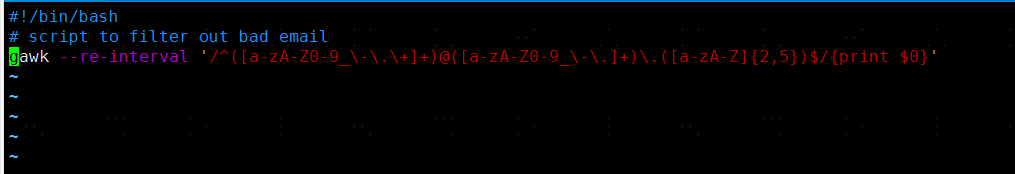
所以，服务器名与域名的正则表达式模式为

([a-zA-Z0-9\_\-\.]+)

顶级域名正则表达式模式为:\.([a-zA-Z]{2,5})$

所以整个模式为：^([a-zA-Z0-9\-\.\+]+)@([a-zA-Z0-9\_\-\.]+)\.([a-zA-Z]{2,5})$

编写脚本如下：



测试结果如下：

