# 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。

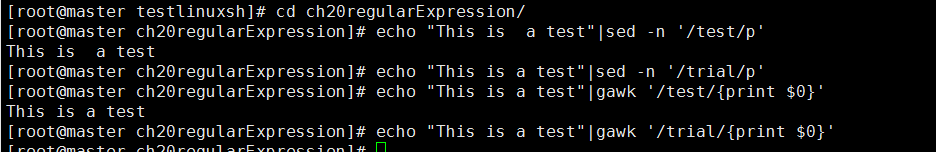
POSIX ERE引擎通常出现在依赖正则表达式进行文本过滤的编程语言中，它为常见模式提供了高级模式符号和特殊符号，比如匹配数字，单词，以及字母的字符。Gawk程序用ERE引擎来处理他的正则表达式模式。

## 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。

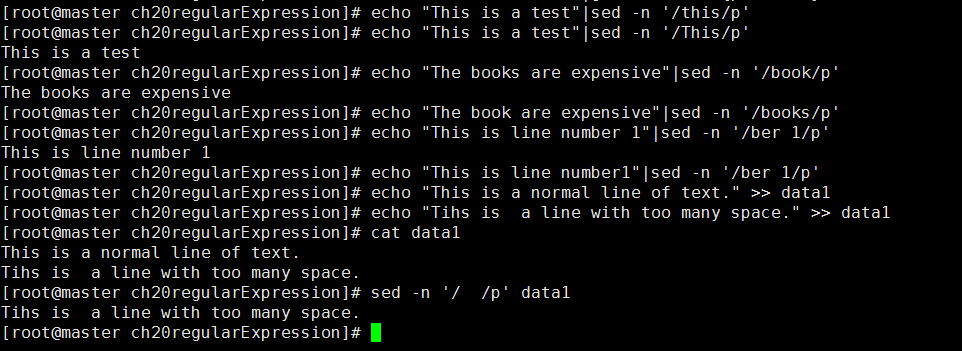
注意点：

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()

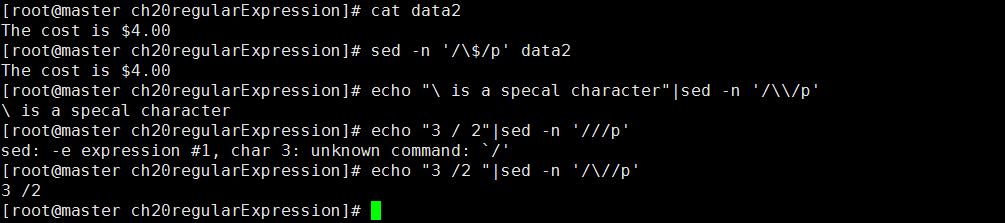
点号与问号： .? 备注：这两个符号在键盘上靠在一起

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：

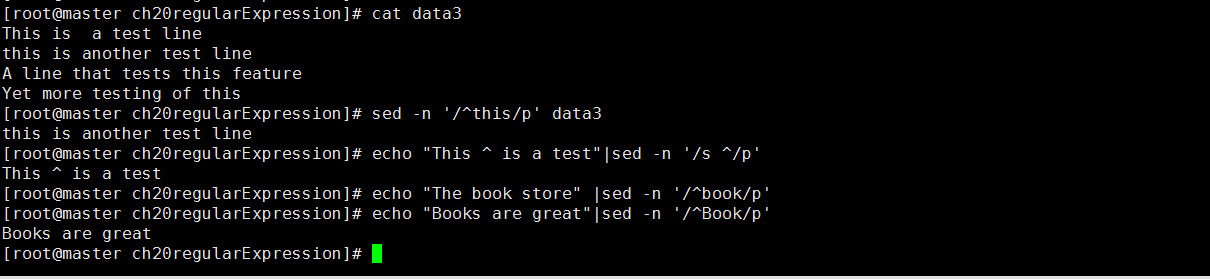
1. 第一个例子中查找文本中的美元符，只要在它前面加一个反斜线
2. 第二个例子中，由于反斜线也是特殊字符，所以模式中要用它，必须转义，这样产生了两个反斜线
3. 第三个例子中，特别注意，尽管正斜线不是正则表达式中的特殊字符，但如果出现在正则表达式中，仍然需要转义

### 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。

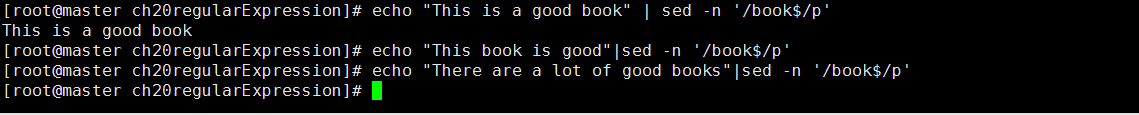


说明：

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。



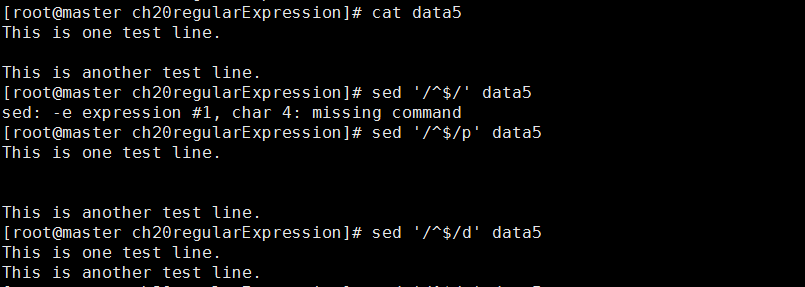
说明：使用结尾文本模式的问题在于你必须留意到底要查找什么

#### 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。

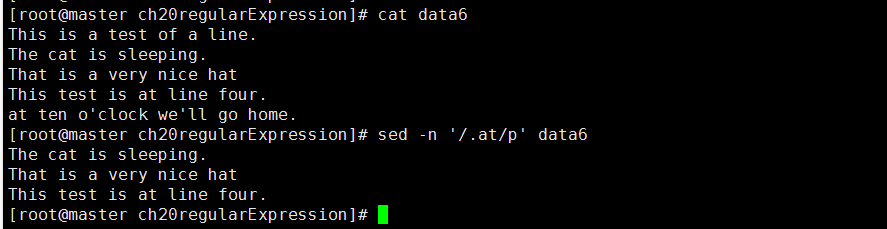
第一种情况中：假定你要查找只含有特定文本模式的数据行

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。

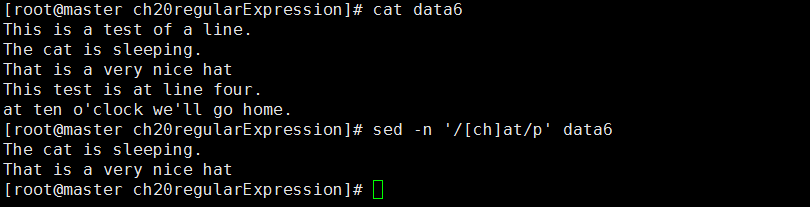


说明：在正则表达式中，空格也是字符，因此at前面的空格刚好匹配了该模式

### 字符数组

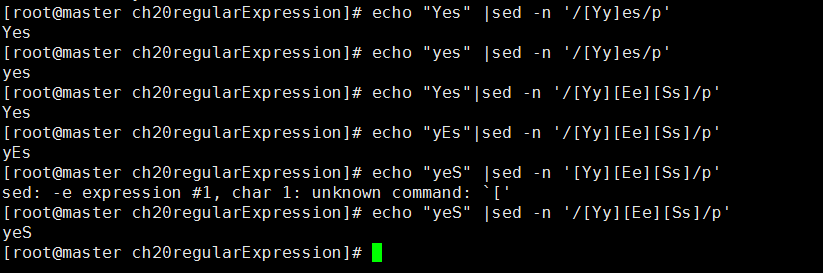
背景：点号字符在匹配某个字符位置上的任意字符时有用，但如果要限定待匹配的具体字符时则用[]，在正则表达式中，称为字符组。可以定义用来匹配文本模式中某个位置的一组字符。如果字符组中某个字符出现在数据流中，那它就匹配该模式

实例1



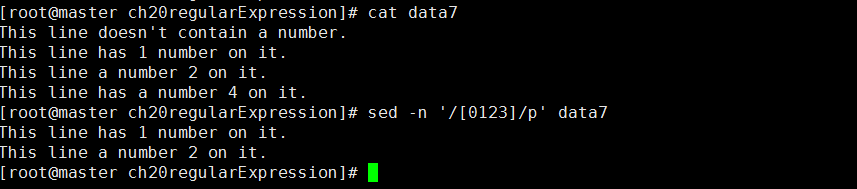
这里用到的数据文件和点号特殊字符例子中的一样，但得到的结果却不一样。这次我们成功 滤掉了只包含单词at的行。匹配这个模式的单词只有cat和hat。还要注意以at开头的行也没有 匹配。字符组中必须有个字符来匹配相应的位置。

实例2：在不太确定某个字符的大小写时，字符组会非常有用

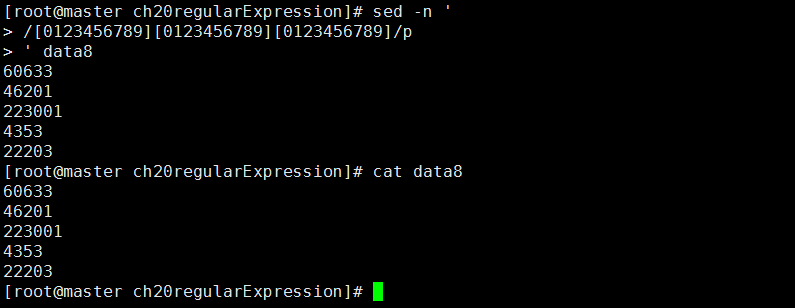


实例3：使用数字

这个正则表达式模式匹配了任意含有数字0、1、2或3的行。含有其他数字以及不含有数字的行都会被忽略掉。

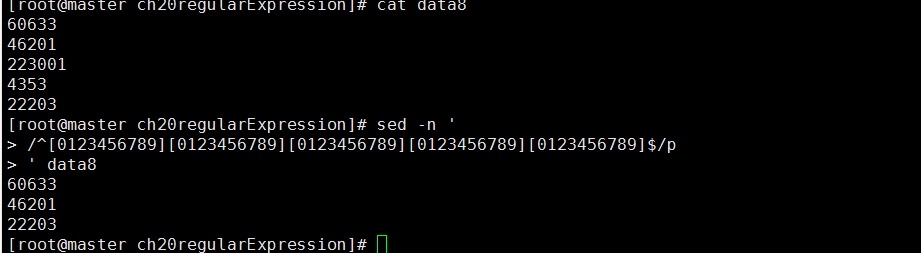


实例4.匹配邮编



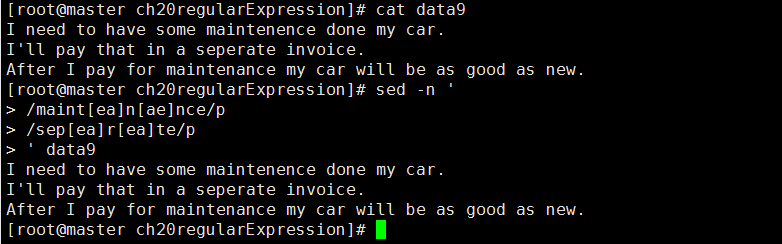
这个结果出乎意料。它成功过滤掉了不可能是邮编的那些过短的数字，因为后一个字符组 没有字符可匹配。但它也通过了那个六位数，尽管我们只定义了3个字符组。 记住，正则表达式模式可见于数据流中文本的任何位置。经常有匹配模式的字符之外的其他 字符。如果要确保只匹配五位数，就必须将匹配的字符和其他字符分开，要么用空格，要么像这 个例子中这样，指明它们就在行首和行尾。

改编



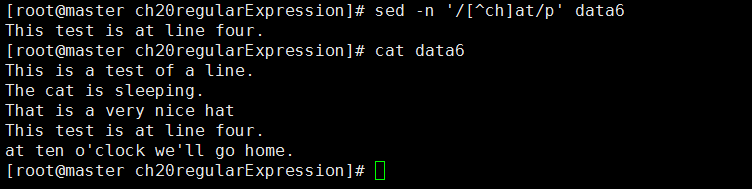
实例4.

字符组的一个极其常见的用法是解析拼错的单词，比如用户表单输入的数据。你可以创建正 则表达式来接受数据中常见的拼写错误。



### 排除型字符组

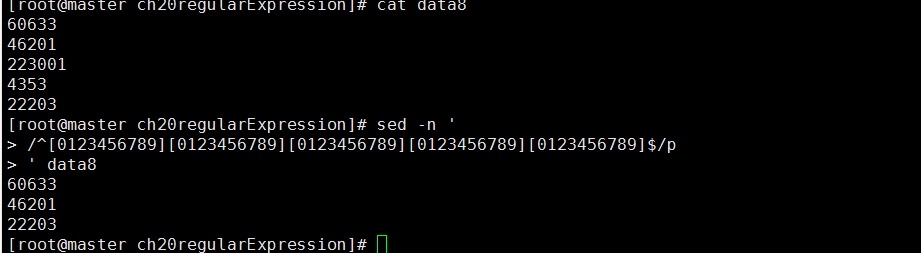
在正则表达式模式中，也可以反转字符组的作用。可以寻找组中没有的字符，而不是去寻找组中含有的字符。要这么做的话，只要在字符组的开头加个脱字符



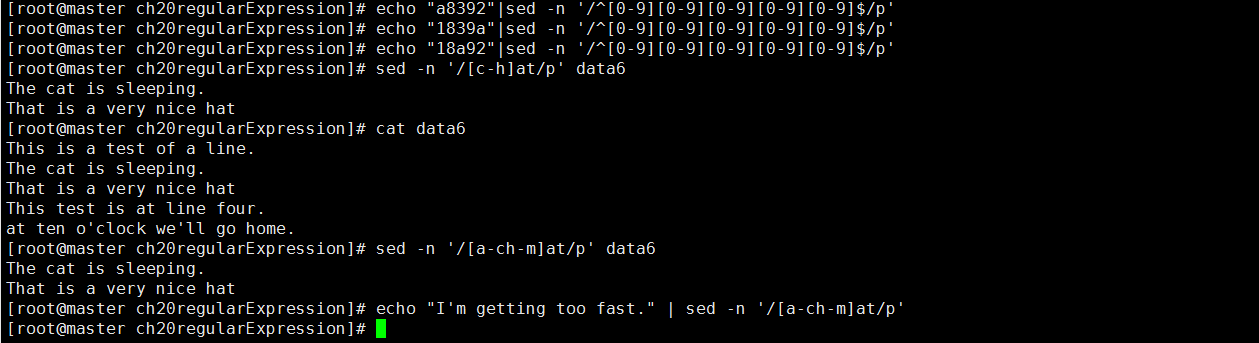
通过排除型字符组，正则表达式模式会匹配c或h之外的任何字符以及文本模式。由于空格字 符属于这个范围，它通过了模式匹配。但即使是排除，字符组仍然必须匹配一个字符，所以以at开头的行仍然未能匹配模式。

### 区间

使用区间进行过滤，。可以用单破折线符号在字符组 中表示字符区间。只需要指定区间的第一个字符、单破折线以及区间的后一个字符就行了。根据Linux系统采用的字符集，正则表达式会包括此区间内的任意字符。如下所示



每个字符组都会匹配0~9的任意数字。如果字母出现在数 据中的任何位置，这个模式都将不成立。如下所示。

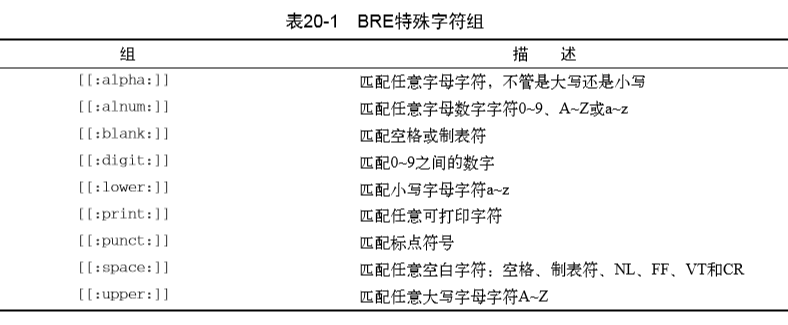


说明：新的模式[c-h]at匹配了首字母在字母c和字母h之间的单词。这种情况下，只含有单词at 的行将无法匹配该模式。 还可以在单个字符组指定多个不连续的区间。

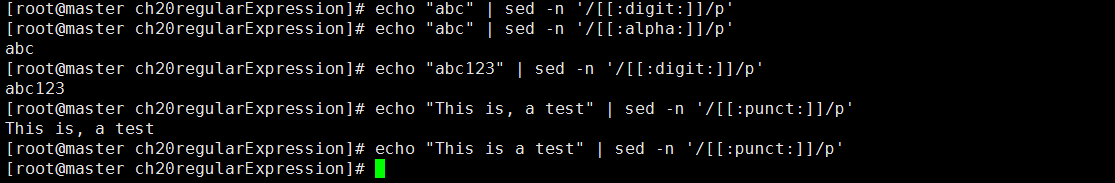
$ sed -n '/[a-ch-m]at/p' data6该字符组允许区间a~c、h~m中的字母出现在at文本前，但不允许出现d~g的字母。

### 特殊的字符组

BRE还包含了一些特殊的字符组，可用来匹配特定类型的字符。

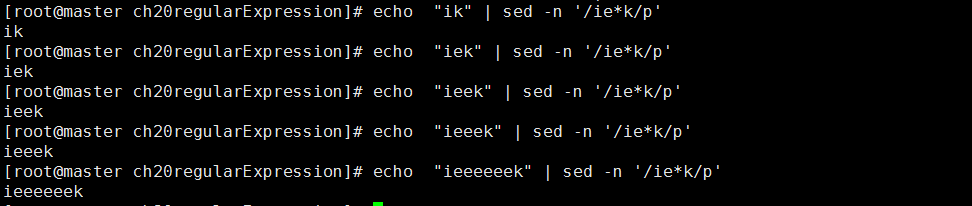


可以在正则表达式模式中将特殊字符组像普通字符组一样使用。使用特殊字符组可以很方便地定义区间。可以用[[:digit:]]来代替区间[0-9]。



### 星号

在字符后面放置星号表明该字符必须在匹配模式的文本中出现0次或多次。

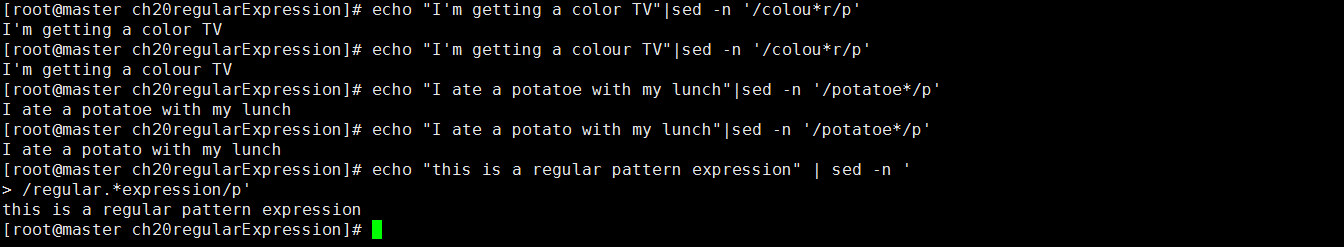


应用场景：

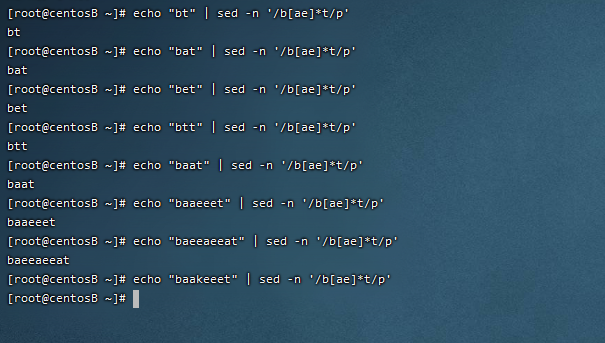
1. 这个模式符号广泛用于处理有常见拼写错误或在不同语言中有拼写变化的单词。

模式中的u\*表明字母u可能出现或不出现在匹配模式的文本中。类似地，如果你知道一个单词经常被拼错，你可以用星号来允许这种错误。

2. 另一个方便的特性是将点号特殊字符和星号特殊字符组合起来。这个组合能够匹配任意数量的任意字符。它通常用在数据流中两个可能相邻或不相邻的文本字符串之间。



1. 应用于字符组中



只要a和e字符以任何组合形式出现在b和t字符之间（就算完全不出现也行），模式就能够匹配。如果出现了字符组之外的字符，该模式匹配就会不成立。

# 扩展正则表达式

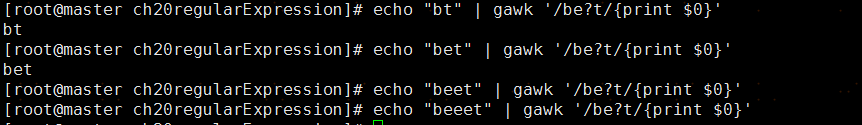
POSIX ERE模式包括了一些可供Linux应用和工具使用的额外符号。gawk程序能够识别ERE 模式，但sed编辑器不能。

警告，sed编辑器和gawk程序的正则表达式引擎之间是有区别的。

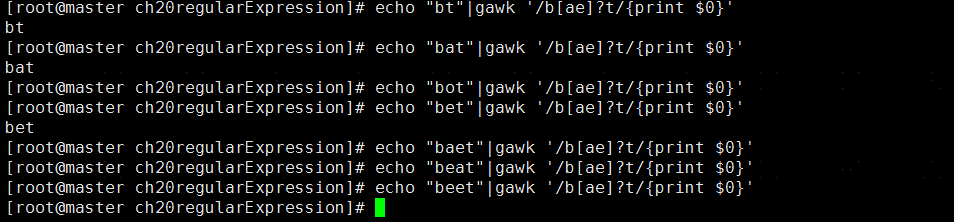
gawk程序可以使用大多数扩展正则表达式模式符号，并且能提供一些额外过滤功能，而这些功能都是sed编辑器所不具备的。但正因为如此，gawk程序在处理数据流时通常才比较慢。

### 问号

问号类似于星号，不过有点细微的不同。问号表明前面的字符可以出现0次或1次，但只限于此。它不会匹配多次出现的字符。



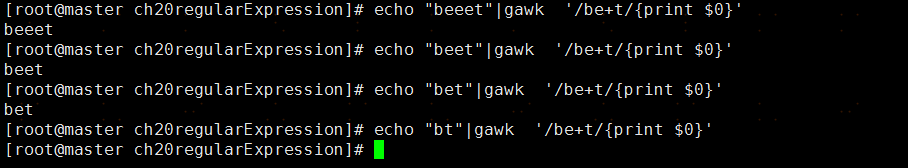
如果字符e并未在文本中出现，或者它只在文本中出现了1次，那么模式会匹配。 与星号一样，你可以将问号和字符组一起使用。



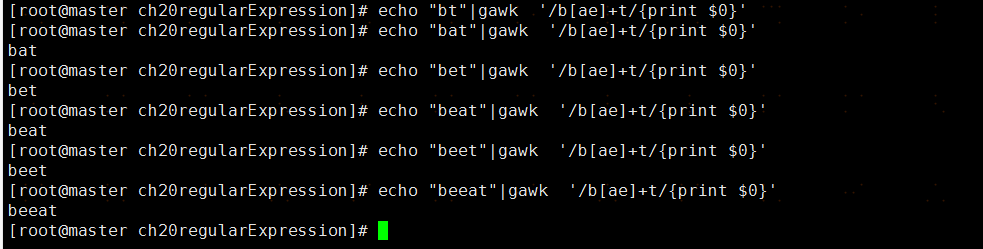
如果字符组中的字符出现了0次或1次，模式匹配就成立。但如果两个字符都出现了，或者其中一个字符出现了2次，模式匹配就不成立。

### 2.加号

加号是类似于星号的另一个模式符号，但跟问号也有不同。加号表明前面的字符可以出现1 次或多次，但必须至少出现1次。如果该字符没有出现，那么模式就不会匹配。



如果字符e没有出现，模式匹配就不成立。加号同样适用于字符组，与星号和问号的使用方式相同



### 3.使用花括号

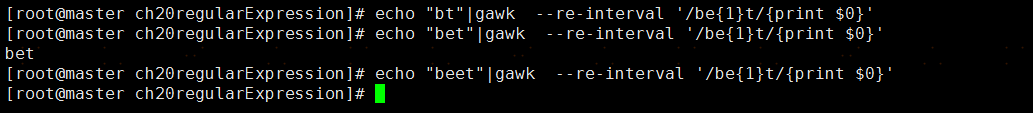
ERE中的花括号允许你为可重复的正则表达式指定一个上限。这通常称为间隔（interval）。 可以用两种格式来指定区间。 

m：正则表达式准确出现m次。 

m, n：正则表达式至少出现m次，至多n次。

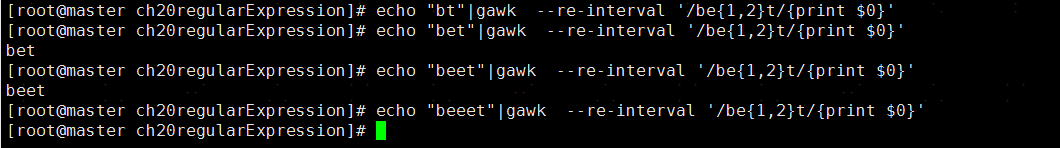
这个特性可以精确调整字符或字符集在模式中具体出现的次数。

警告 默认情况下，gawk程序不会识别正则表达式间隔。必须指定gawk程序的--re- interval 命令行选项才能识别正则表达式间隔。

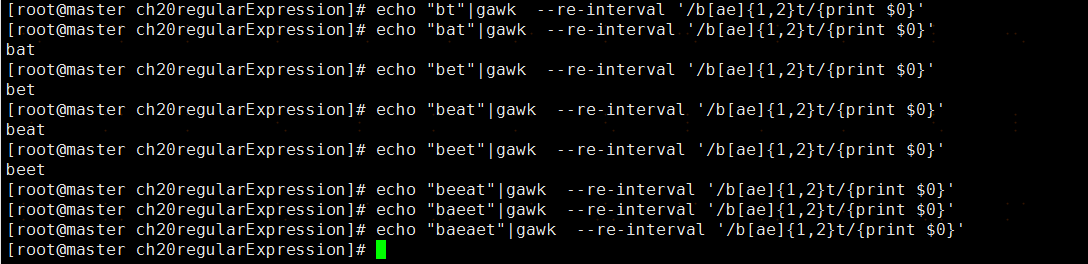


通过指定间隔为1，限定了该字符在匹配模式的字符串中出现的次数。如果该字符出现多次， 模式匹配就不成立。

设置上下限制



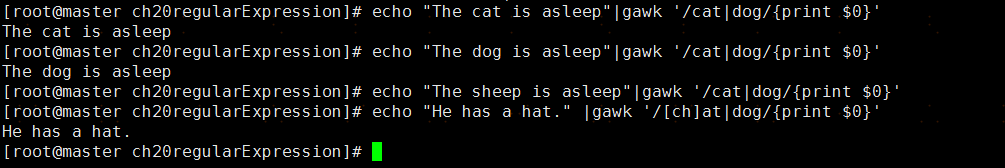
分组中使用



### 管道符号

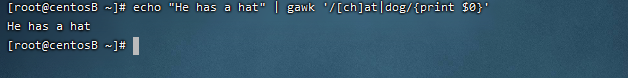
管道符号允许你在检查数据流时，用逻辑OR方式指定正则表达式引擎要用的两个或多个模 式。如果任何一个模式匹配了数据流文本，文本就通过测试。如果没有模式匹配，则数据流文本 匹配失败。

使用管道符号的格式如下： expr1 | expr2|…



上面的例子中会在数据流中查找正则表达式cat或dog，正则表达式和管道符号之间不能有空格，否则他们会被认为是正则表达式模式的一部分。

管道符号两侧的正则表达式可以采用任何正则表达式模式(包括字符组)来定义文本。

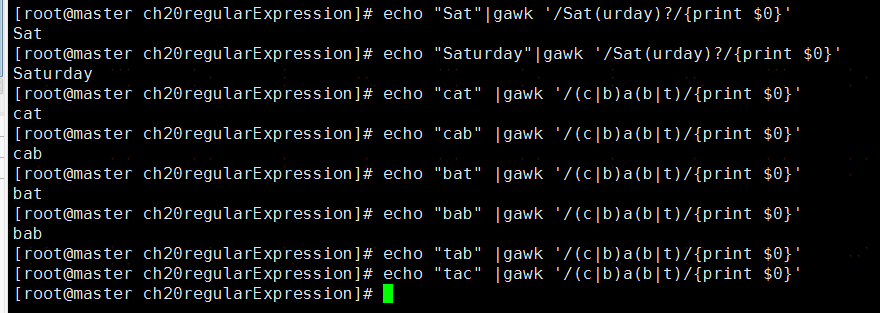


### 5.表达式分组

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

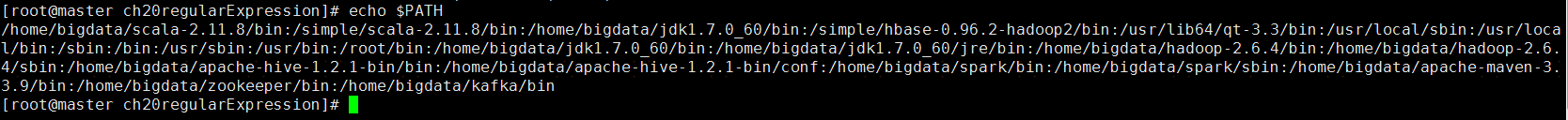
将分组和管道符号一起使用来创建可能的模式匹配，是最常见的做法。



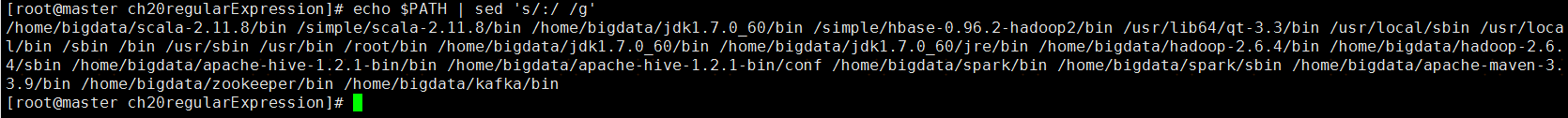
# 正则表达式实战

### 目录文件计数

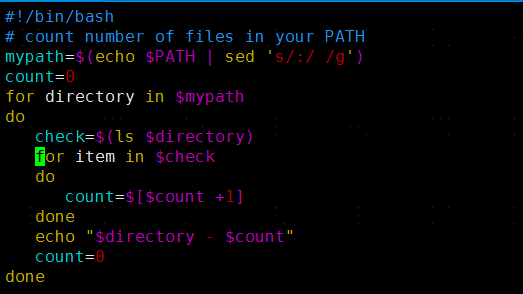
需求：对PATH环境变量中定义的目录的可执行文件进行计数



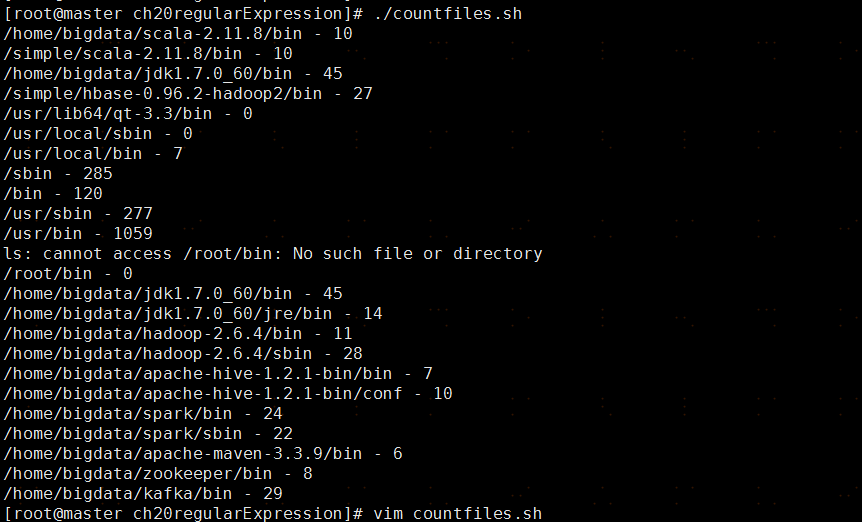
分析1.进行分离



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}

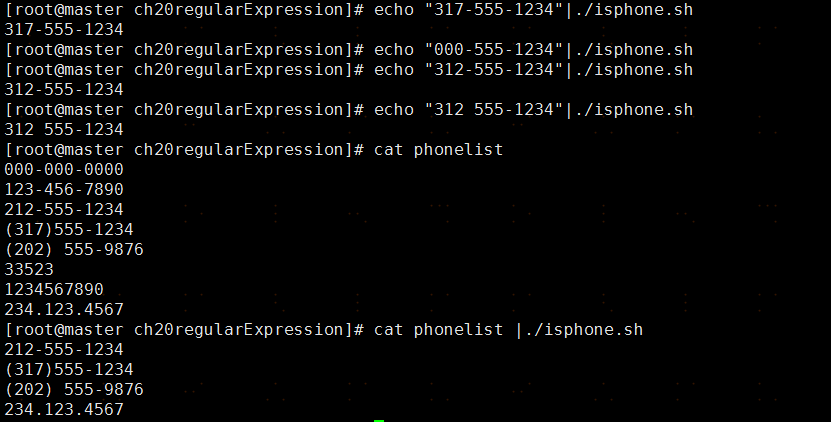
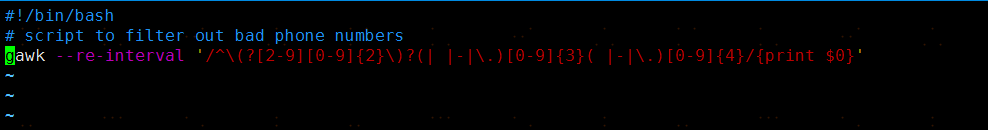
1. 交换机号码后面的

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，

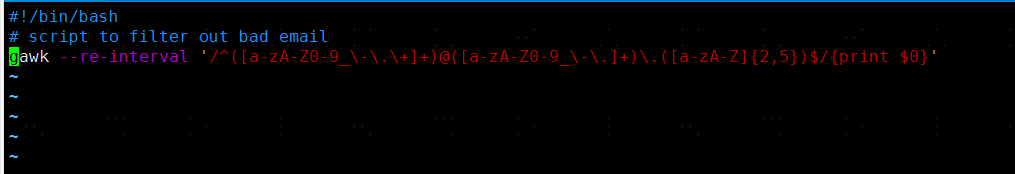
所以，服务器名与域名的正则表达式模式为

([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：

