# 基础正则表达式介绍与练习

## 一、什么是正则表达式

在做文字处理或编写程序时，用到查找、替换等功能，使用正则表达式能够简单快捷的完成目标。简单而言，正则表达式通过一些特殊符号的帮助，使用户可以轻松快捷的完成查找、删除、替换等处理程序。例如 grep, expr, sed , awk. 或 Vi 中经常会使用到正则表达式，为了充分发挥 shell 编程的威力,需要精通正则表达式。正规表示法基本上是一种『表示法』， 只要工具程序支持这种表示法，那么该工具程序就可以用来作为正规表示法的字符串处理之用。 也就是说，例如 vi, grep, awk ,sed 等等工具，因为她们有支持正规表示法， 所以，这些工具就可以使用正规表示法的特殊字符来进行字符串的处理。

## 二、正则表达式特殊符号

首先是正则表达式的特殊符号：

[:alnum:]代表英文大小写字母及数字 [:alpha:]代表英文大小写字母[:blank:]代表空格和 tab 键[:cntrl:]键盘上的控制按键，如 CR,LF,TAB,DEL[:digit:]代表数字[:graph:]代表空白字符以外的其他[:lower:]小写字母[:print:]可以被打印出来的任何字符[:punct:]代表标点符号[:upper:]代表大写字符[:space:]任何会产生空白的字符如空格，tab,CR 等[:xdigit:]代表 16 进位的数字类型

### 特殊符号实例

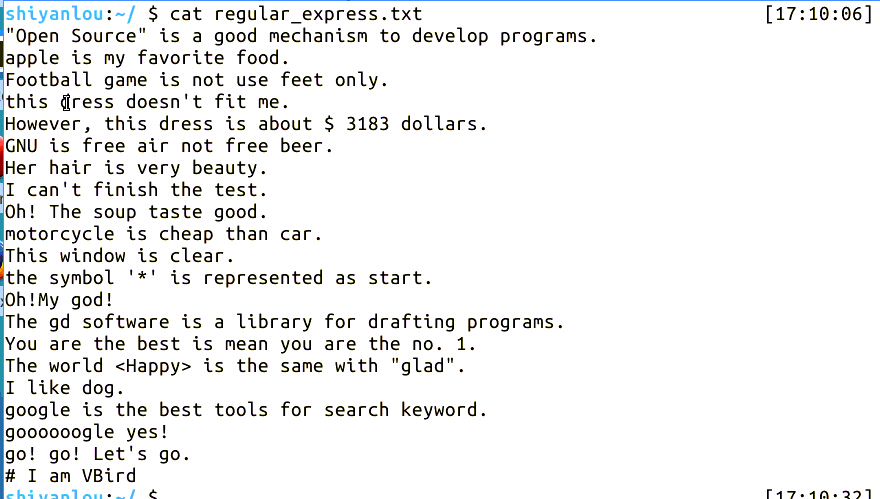
任意建立一个文本并编辑。或者以原有的文本做实验皆可。

$ touch regular\_express.txt

$ vi regular\_express.txt

//或者 sudo gedit regular\_express.txt

操作截图：



文本内容为：

"Open Source" is a good mechanism to develop programs.

apple is my favorite food.

Football game is not use feet only.

this dress doesn't fit me.

However, this dress is about $ 3183 dollars.

GNU is free air not free beer.

Her hair is very beauty.

I can't finish the test.

Oh! The soup taste good.

motorcycle is cheap than car.

This window is clear.

the symbol '\*' is represented as start.

Oh!My god!

The gd software is a library for drafting programs.

You are the best is mean you are the no. 1.

The world <Happy> is the same with "glad".

I like dog.

google is the best tools for search keyword.

goooooogle yes!

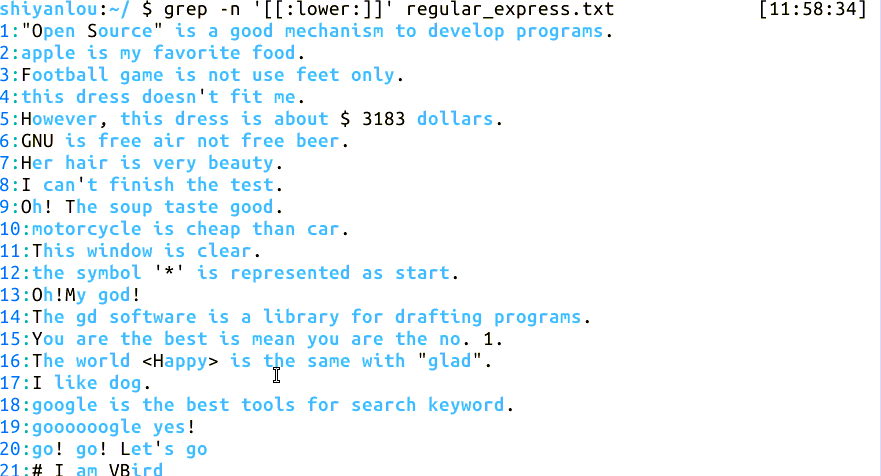
go! go! Let's go.# I am VBird

查找小写字母：

$ grep -n '[[:lower:]]' regular\_express.txt

操作截图：

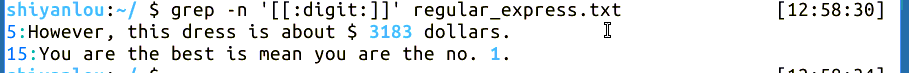
**蓝色为匹配成功的字符**



查找数字：

# grep -n '[[:digit:]]' regular\_express.txt

操作截图：



## 四、简单例子 ：ls 命令正则表达式应用

ls 命令是 linux 下最常用的命令。ls 命令是 list 的缩写,缺省下 ls 用来打印出当前目录的清单。

命令中‘ \* ’号是正则表达式的一种，代表任意字符串。

$ ls

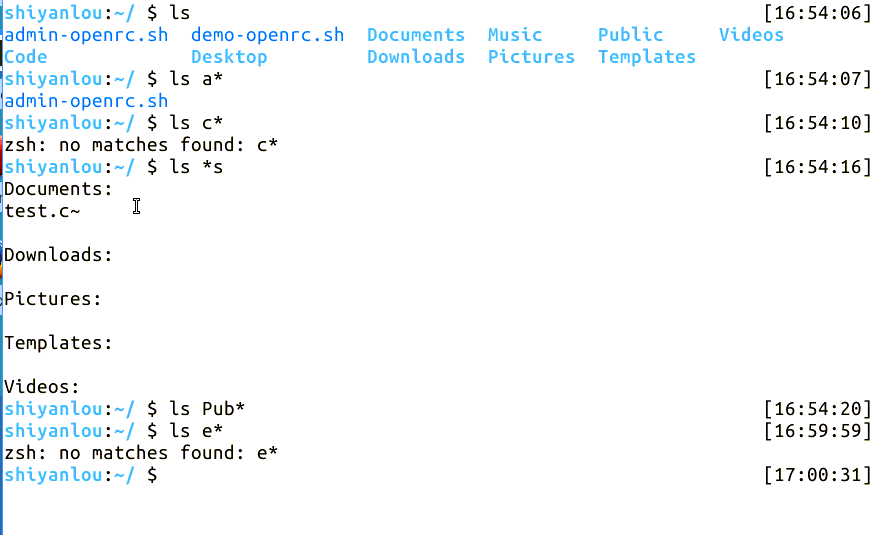
$ ls a\* //打印出以 a 开头的文件及内容或者目录

$ ls \*s //打印以 s 结尾的文件及内容或者目录

$ ls Pub\* // Public 文件为空，所有没有打印任何内容

$ ls e\* //没有以 e 开头的文件，所有显示没有找到匹配项

操作截图：



# grep 命令与正则表达式

## 一、搜寻特定字符串"the"

**参数说明：**

-a ：将 binary 档案以 text 档案的方式搜寻数据

-c ：计算找到 '搜寻字符串' 的次数

-i ：忽略大小写的不同，所以大小写视为相同

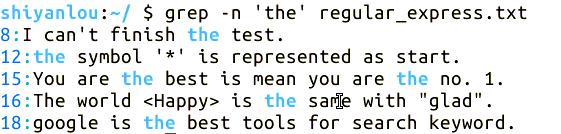
-n ：顺便输出行号

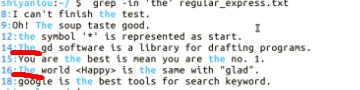
-v ：反向选择，亦即显示出没有 '搜寻字符串' 内容的那一行！

$ grep -n 'the' regular\_express.txt

$ grep -in 'the' regular\_express.txt

操作截图：





反向查找,也就是说当该行没有”the“ 字符时才显示在屏幕上。

$ grep -vn 'the' regular\_express.txt

## 二、字符组匹配:

[]中包含的任意一个字符。只能是一个。

字符组支持由连字符“ - ”来表示一个范围。当“ - ”前后构成范围时，要求前面字符的码位小于后面字符的码位。

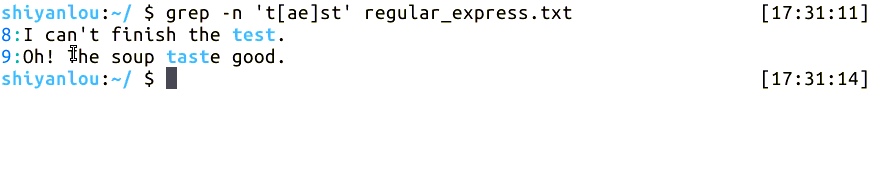
[^...] 排除型字符组。排除后面的字符。

[abc] ：表示“a”或“b”或“c”[0-9] ：表示 0~9 中任意一个数字，等价于[0123456789][\u4e00-\u9fa5] :表示任意一个汉字[^a1<] :表示除“a”、“1”、“<”外的其它任意一个字符[^a-z] :表示除小写字母外的任意一个字符

查找“tast”或者“test”两个字符串。# grep -n 't[ae]st' regular\_express.txt

# grep -n '^#' regular\_express.txt

操作截图：



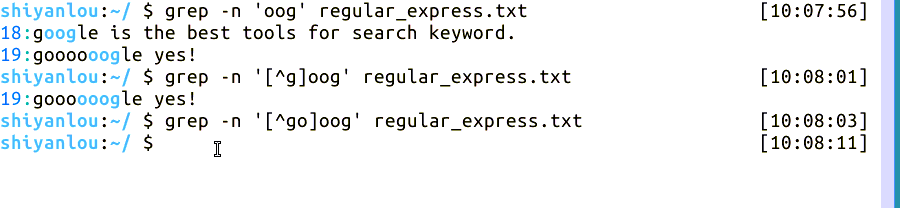
查找字符“oog”，如果我不想要“oog”字符前面有 g，则查找“[^g]oog”，同理，查找“[^go]oog”是指字符“oog”前面不能为 g 或者 o。

$ grep -n 'oog' regular\_express.txt

$ grep -n '[^g]oog' regular\_express.txt

$ grep -n '[^go]oog'regular\_express.txt

操作截图：



grep -n '[^go]oog'regular\_express.txt 结果返回为空，表明没有匹配到满足要求的字符串。

## 三、行首符: ^ 与行尾符: $

在第一个小实验中，查找了一行字符中含有“the”的，如果你想要只查找 行首为“the”的字符行，则使用以下命令：

$ grep -n '^the' regular\_express.txt

查找行首为大写字母的所有行：

'^[A-Z]' 表示以大写字母开头。

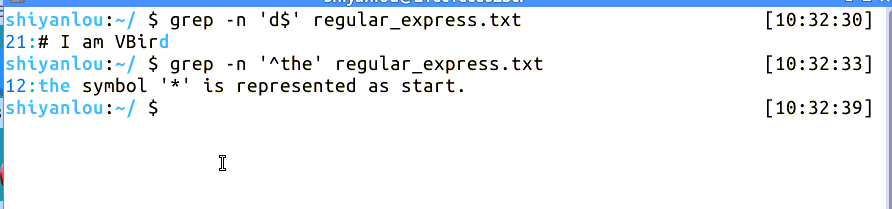
'[^A-Z]' 表示除了大写字母 A-Z 的所有字符。

# grep -n '^[A-Z]' regular\_express.txt

查找以 d 字母结尾的行：

# grep -n 'd$' regular\_express.txt

操作截图：



查找空行：

$ grep -n '^$' regular\_express.txt

应用实例：

查看/etc/insserv.conf 文档

'^$' : 过滤掉空白行

'^#' ：过滤掉注释行（以#号开头）

$ cat -n /etc/insserv.conf

$ grep -v '^$' /etc/insserv.conf | grep -v '^#'

## 四、任意一个字符: " . "（小数点）与重复字符 “ \* ”（星号） 。

查找 a？ou？类型的字符。

$ grep -n 'a.ou.' regular\_express.txt

'a.ou.' : 小数点表示任意一个字符，一个小数点只能表示一个未知字符。

\*（星号）：代表重复前面 0 个或者多个字符。

e\*： 表示具有空字符或者一个以上 e 字符。

ee\*，表示前面的第一个 e 字符必须存在。第二个 e 则可以是 0 个或者多个 e 字符。

eee\*，表示前面两个 e 字符必须存在。第三个 e 则可以是 0 个或者多个 e 字符。

ee\*e ：表示前面的第一个与第三个 e 字符必须存在。第二个 e 则可以是 0 个或者多个 e 字符。

下面的第一条命令与第二条命令由于允许存在空字符，所以会打印所有文本。

$ grep -n 'e\*' regular\_express.txt

$ grep -n '@\*' regular\_express.txt

$ grep -n 'eee\*' regular\_express.txt

## 五、限定连续字符范围{ }：

{ }可限制一个范围区间内的重复字符数。举个例子，若要找出 2~5 个 o 的连续字符串，如何做？ 此时便要用到{}了。由于 { 与 } 在 shell 中有特殊意义，需要用到转义字符\。

查找连续的两个 o 字符：

$ grep -n 'o\{2\}' regular\_express.txt

结果与命令 grep -n 'ooo\*' regular\_express.txt 的结果相同。

查找 g 后面接 2 到 5 个 o，然后再接 g 的字符串

$ grep -n 'go\{2,5\}g' regular\_express.txt

总结：

^word 表示带搜寻的字符串(word)在行首

word$ 表示带搜寻的字符串(word)在行尾

.(小数点) 表示 1 个任意字符

\ 表示转义字符，在特殊字符前加\会将特殊字符意义去除

\* 表示重复 0 到无穷多个前一个 RE(正则表达式)字符

[list] 表示搜索含有 list 的字符串

[n1-n2] 表示搜索指定的字符串范围,例如[0-9] [a-z] [A-Z]等

[^list] 表示反向字符串的范围,例如[0-9]表示非数字字符，[A-Z]表示非大写字符范围

\{n,m\} 表示找出 n 到 m 个前一个 RE 字符

\{n,\} 表示 n 个以上的前一个 RE 字符

# 正则表达式运用之 sed 工具命令

本实验继续以上一个实验的 regular\_express.txt 为实验文本。

## 一、概念

sed 是非交互式的编辑器。它不会修改文件，除非使用 shell 重定向来保存结果。默认情况下，所有的输出行都被打印到屏幕上。

sed 编辑器逐行处理文件（或输入），并将结果发送到屏幕。具体过程如下：首先 sed 把当前正在处理的行保存在一个临时缓存区中（也称为模式空间），然后处理临时缓冲区中的行，完成后把该行发送到屏幕上。sed 每处理完一行就将其从临时缓冲区删除，然后将下一行读入，进行处理和显示。处理完输入文件的最后一行后，sed 便结束运行。sed 把每一行都存在临时缓冲区中，对这个副本进行编辑，所以不会修改原文件。

如果要修改原文件，可使用-i 选项。

## 二、实验

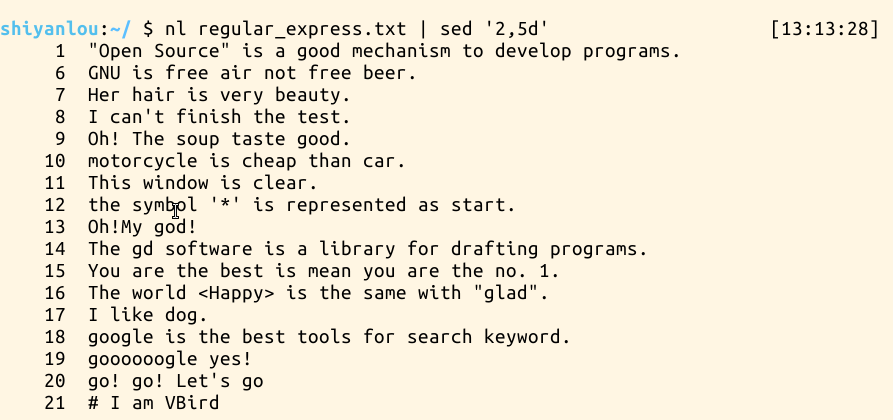
### 2.1 将 regular\_express.txt 的内容列出并打印行号，同时,将 2-5 行删除显示

代码如下:

$ nl regular\_express.txt | sed '2,5d'

命令解释：'2,5d' 表示 2~5 行，d 表示删除。

操作截图：



注: sed 是 sed -e 的简写, 后接单引号

同上删除第 2 行

$ nl regular\_express.txt | sed '2d'

同上删除第三行到最后一行, $定位到最后一行

$ nl regular\_express.txt | sed '3,$d'

在原文件中删除第 1 行：

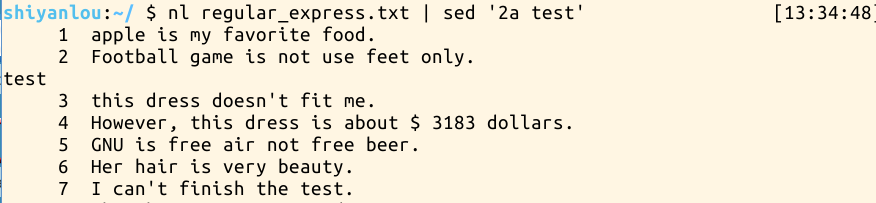
$ sed -i '1d' regular\_express.txt

### 2.2 a表示在行后加上字符串，i表示在行前添加字符串

在第二行后添加 test 字符串

$ nl regular\_express.txt | sed '2a test'

操作截图：



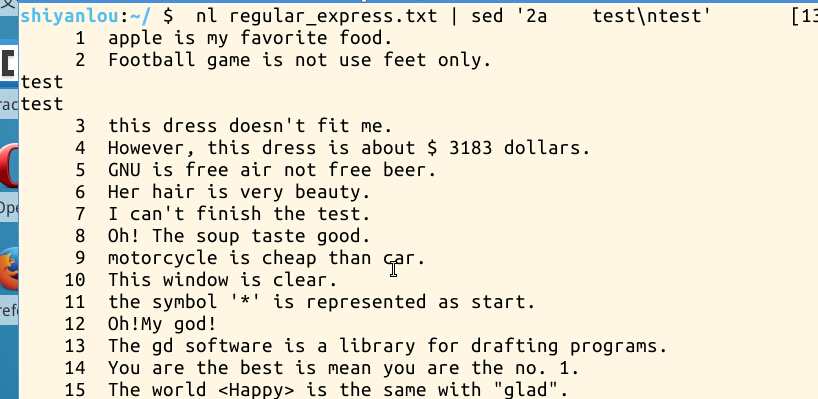
在第二行前添加 test 字符串

$ nl regular\_express.txt | sed '2i test'

在第二行后加入两行 test，“\n”表示换行符

$ nl regular\_express.txt | sed '2a test\ntest'

操作截图：



### 2.3将 2-5 行内容取代为 No 2-5 number

c 为替换内容选项。

$ nl regular\_express.txt | sed '2,5c No 2-5 number'

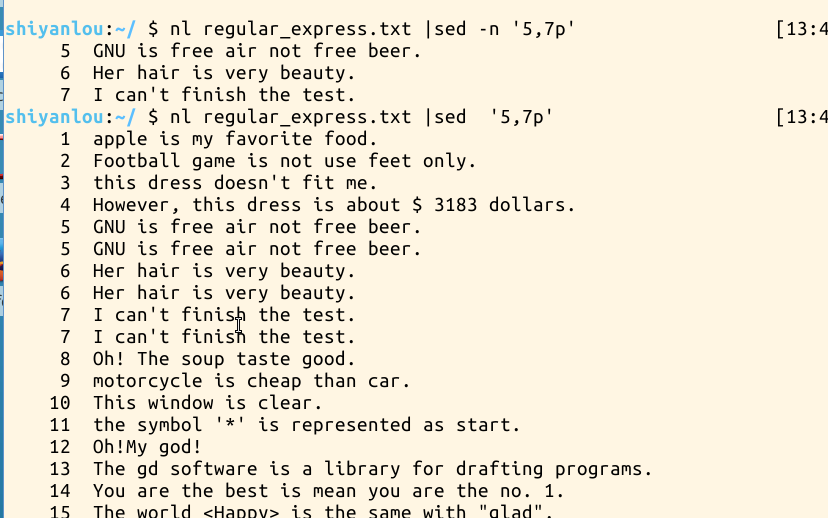
### 2.4 列出 regular\_express.txt 内第 5-7 行

sed 命令中-n 为安静模式选项。以下两条命令执行结束后可对比结果。

$ nl regular\_express.txt |sed -n '5,7p'

$ nl regular\_express.txt |sed '5,7p'

操作截图：



### 2.5 替换字符串:

sed 's/被替换字符串/新字符串/g'

1.获取本机 IP 的行

$ /sbin/ifconfig eth0 |grep 'inet '

inet 后面的空格不能少

命令详解： 在/sbin/ifconfig eth0 的结果中查找‘inet’，打印至终端

将 IP 前面的部分予以删除，下面两条命令结果相同。

$ /sbin/ifconfig eth0 |grep 'inet '| sed 's/.inet...://g'

$ /sbin/ifconfig eth0 |grep 'inet '| sed 's/.\{0,9\}://'

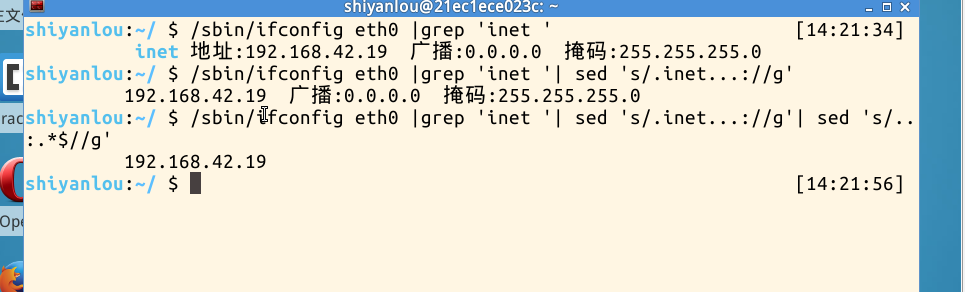
将 IP 后面的部分删除

$ /sbin/ifconfig eth0 |grep 'inet '| sed 's/.inet...://g'| sed 's/..:.\*$//g'

$ /sbin/ifconfig eth0 |grep 'inet '| sed 's/.inet...://g'| sed 's/.\{0,3\}:.\*$//g'

**192.168.42.19**

操作截图：



上述命令是比较复杂的正则表达式运用，熟悉正则表达式可以明显的简化命令，简单快捷的完成文件查询、修改等功能。

# 扩展正则表达式

事实上，一般实验者只需要熟悉基础的正则表达式就足够了。不过有时候为了简化命令操作，了解一些使用范围更广的扩展表达式，会更加方便。

## 一、简单对比

正规表示法:

$ grep -v '^$' regular\_express.txt |grep -v '^#'

需要使用到管线命令来搜寻两次！ 那么如果使用延伸型的正规表示法，我们可以简化为：

$ egrep -v '^$|^#' regular\_express.txt

利用支持延伸型正规表示法的 egrep 与特殊字符 “|” 的组功能来区隔两组字符串，如此一来，是否方便很多呢？

此外，grep 默认仅支持基础正则表达式，如果要使用扩展性正则表达式，可以使用 grep - E。grep -E 与 egrep 相当于命令别名关系。

## 二、扩展规则

### 1、 + ：表示重复****一个或一个以上****的前一个 RE 字符

$ egrep -n 'go+d' regular\_express.txt

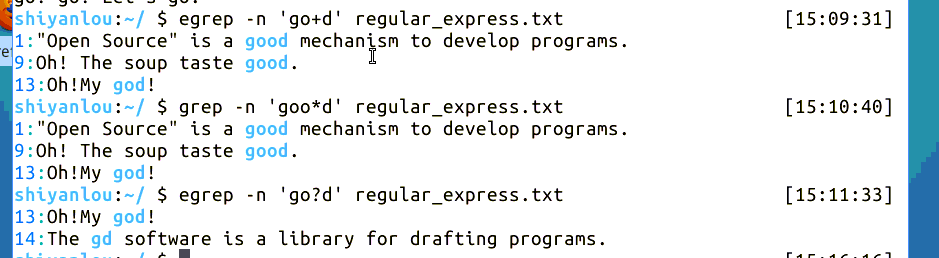
普通写法:

$ grep -n 'goo\*d' regular\_express.txt

### 2、 ? : 表示重复****零个或一个前一个****　RE 字符

$ egrep -n 'go?d' regular\_express.txt

操作截图：



上述三条命令结果如上，发现 ‘goo\*d’ 与 ‘go+d’ 等同，而 ‘go?d’ 结果不同。

### 3、 | :表示用或的方式找出数个字符串

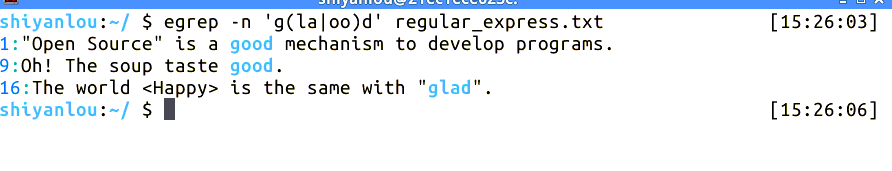
$ egrep -n 'gd|good' regular\_express.txt

### 4、 () : 表示找出群组字符串

$ egrep -n 'g(la|oo)d' regular\_express.txt

也就是搜寻(glad)或 good 这两个字符串

操作截图：



### 5、()+ : 多个重复群组判别

$　echo 'AxyzxyzxyzxyzC'|egrep 'A(xyz)+C'

$　echo 'AxyzxyzxyzxyzC'|egrep 'A(xz)+C'

也就是要找开头是 A 结尾是 C 中间有一个以上的 ‘xyz’ 或 ‘xz’ 字符串的意思。

结果显示 ‘A(xyz)+C’ 可以匹配， ‘A(xz)+C’ 没有匹配项。

操作截图：

