正则表达式

Regular Expression 、 regex 或 regexp, 缩写为 RE 正则表达式这个概念最初是由 Unix 中的工具软件 (例如 sed 和 grep) 普及开的。

通常被用来检索、替换那些符合某个规则的文本。

许多程序设计语言都支持正则表达是进行字符串操作。例如，在perl中就内建了一个功能强大的正则表达式引擎，还有java语言自带的。

起源于科学家对人类神经系统工作原理的早期研究；Ken Thompson将其应用到计算搜索算法 ， Unix之父将此引入到编辑器QED，后来的ed，最终引入grep。

概念：

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、以及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。

什么是正则表达式

- 正则表达式就是记录文本规则的代码
- 和通配符类似,正则表达式也是用来进行文本匹配的工具,只不过比起通配符,它能更精确地描述你的需求

字符和字符串：

- 字符是计算机软件处理文字时最基本的单位,可能是字母,数字,标点符号,空格,换行符,汉字等等

- 字符串是0个或更多个字符的序列。

特点：

1. 灵活性、逻辑性和功能性非常强；
2. 可以迅速地用极简单的方式达到字符串的复杂控制；
3. 对于刚接触的人来说，比较晦涩难懂。

应用程序：
grep,egrep,awk,mysql,vim

特殊符号

```
特殊字符          代表意义
[:alnum:] 代表英文大小写字符及数字 ,0-9, A-Z, a-z
[:alpha:] 代表任何英文大小写字符 , A-Z, a-z
[:lower:] 代表小写字符 , a-z
[:upper:] 代表大写字符 ,A-Z
[:digit:] 代表数字而已 , 0-9
[:xdigit:] 代表 16 进制数字 , 因此包括 : 0-9, A-F, a-f
[:blank:] 代表空格键和 [Tab] 按键
[:space:] 任何会产生空白的字符,包括空格键 , [Tab], CR 等等
[:graph:] 除了空格键 ( 空格键和 [Tab] ) 外的其他所有按键
[:cntrl:] 代表键盘上面的控制按键 , 包括 CR, LF, Tab, Del.. 等等
[:print:] 代表任何可以被打印出来的字符
[:punct:] 代表标点符号 (punctuation symbol) :" ' ? ! ; : # $...
```

非打印字符

非打印字符也可以是正则表达式的组成部分。下表列出了表示非打印字符的转义序列：

```
字符     描述
\cx      匹配由x指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。
否则，将 c 视为一个原义的 'c' 字符。
\f       匹配一个换页符。等价于 \x0c 和 \cL。
\n       匹配一个换行符。等价于 \x0a 和 \cJ。
\r       匹配一个回车符。等价于 \x0d 和 \cM。
\s       匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [ \f\n\r\t\v]。
\S       匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\t       匹配一个制表符。等价于 \x09 和 \cI。
\v       匹配一个垂直制表符。等价于 \x0b 和 \cK
```

常见的正则表达式

```
^       行首定位符
$       行尾定位符
.       匹配除换行符之外的单个字符
*       匹配 0 个或多个前一字符
?       匹配 0 个或1个前一字符
+       匹配 1 个或多个前一个字符
[ ]     匹配指定字符组内的任一字符
[^]     匹配不在指定字符组内的任一字符
\<      单词起始边界匹配符
\>      单词结束边界匹配符
x\{m\}  连续 M 个字符 X
x\{m,\} 至少 M 个字符 X
x\{m,n\}至少 M 个最多 N 个字符 X
```

课堂练习

> 写一个测试脚本re.sh;带位置参数执行，例如 re.sh ab

代表以**a**开头的

```
#!/bin/bash
if [[ $1 =~ ^a  ]]
then
        echo ok
else
        echo no
fi
```

测试：

```
[root@rhel6 ~]# bash re.sh a
ok
[root@rhel6 ~]# bash re.sh b
no
[root@rhel6 ~]# bash re.sh ab
ok
```

代表以**a**结尾的

```
#!/bin/bash
if [[ $1 =~ a$  ]]
then
        echo ok
else
        echo no
fi
```

测试：

```
[root@rhel6 ~]# vim re.sh
[root@rhel6 ~]# bash re.sh a
ok
[root@rhel6 ~]# bash re.sh b
no
[root@rhel6 ~]# bash re.sh ab
no
[root@rhel6 ~]# bash re.sh ba
ok
```

代表**a**字符后面一定有两个字符

```
#!/bin/bash
if [[ $1 =~ a..  ]]
then
    echo ok
else
    echo no
fi
```

测试：

```
[root@rhel6 ~]# vim re.sh
[root@rhel6 ~]# bash re.sh a
no
[root@rhel6 ~]# bash re.sh axx
ok
[root@rhel6 ~]# bash re.sh baxx
ok
[root@rhel6 ~]# bash re.sh baxxx
ok
```

代表匹配**a**字符后面可以是**0**个**b**，也可以是多个**b**

```
#!/bin/bash
if [[ $1 =~ ab*  ]]
then
    echo ok
else
    echo no
fi
```

测试：

```
[root@rhel6 ~]# bash re.sh a
ok
[root@rhel6 ~]# bash re.sh ab
ok
[root@rhel6 ~]# bash re.sh abbbb
ok
[root@rhel6 ~]# bash re.sh abbbbxxx
ok
[root@rhel6 ~]# bash re.sh accc
ok
[root@rhel6 ~]# bash re.sh ccc
no
```

匹配 **0** 个或**1**个前一字符

```
#!/bin/bash
if [[ $1 =~ 1a?1 ]]
then
        echo ok
else
        echo no
fi
```

测试：

```
[root@rhel6 ~]# bash re.sh 11
ok
[root@rhel6 ~]# bash re.sh 1a1
ok
[root@rhel6 ~]# bash re.sh 1aa1
no
```

代表匹配**a**字符后面可以是**1**个**b**，也可以是多个**b**

```
#!/bin/bash
if [[ $1 =~ ab+ ]]
then
    echo ok
else
    echo no
fi
```

测试：

```
[root@rhel6 ~]# bash re.sh a
no
[root@rhel6 ~]# bash re.sh ab
ok
[root@rhel6 ~]# bash re.sh abbb
ok
[root@rhel6 ~]# bash re.sh ac
no
```

代表匹配**ab**字符，后面可以是任意字符

```
#!/bin/bash
if [[ $1 =~ ab.*  ]]
then
    echo ok
else
    echo no
fi
```

测试：

```
[root@rhel6 ~]# vim re.sh
[root@rhel6 ~]# bash re.sh a
no
[root@rhel6 ~]# bash re.sh ab
ok
[root@rhel6 ~]# bash re.sh abbbb
ok
[root@rhel6 ~]# bash re.sh accc
no
[root@rhel6 ~]# bash re.sh ccc
no
```

代表匹配**a**和**b**字符之间可以是任意字符

```
#!/bin/bash
if [[ $1 =~ a.*b  ]]
then
    echo ok
else
    echo no
fi
```

测试：

```
[root@rhel6 ~]# bash re.sh a
no
[root@rhel6 ~]# bash re.sh b
no
[root@rhel6 ~]# bash re.sh ab
ok
[root@rhel6 ~]# bash re.sh a1b
ok
[root@rhel6 ~]# bash re.sh 1a1b1
ok
```

代表匹配指定字符组内的任一字符，可以用逗号分割，或者不用，效果一样都代表匹配一个字符

```
#!/bin/bash
if [[ $1 =~ [Bb]ooboo ]]
then
        echo ok
else
        echo no
fi
```

测试：

```
[root@rhel6 ~]# bash re.sh booboo
ok
[root@rhel6 ~]# bash re.sh Booboo
ok
[root@rhel6 ~]# bash re.sh cooboo
no
```

代表匹配不在指定字符组内的任一字符

```
#!/bin/bash
if [[ $1 =~ [^Bb]ooboo ]]
then
        echo ok
else
        echo no
fi
```

测试：

```
[root@rhel6 ~]# bash re.sh booboo
no
[root@rhel6 ~]# bash re.sh Booboo
no
[root@rhel6 ~]# bash re.sh cooboo
ok
```

单词起始和结束边界匹配符与行首行尾的匹配对比

```
[root@rhel6 ~]# vim re.file
booboo  tom jack
jack    tom booboo
tom tom tom
jack
 jack
jack

[root@rhel6 ~]# grep "^booboo" re.file
booboo  tom jack
[root@rhel6 ~]# grep "\<booboo" re.file
booboo  tom jack
jack    tom booboo
[root@rhel6 ~]# grep "\>booboo" re.file
[root@rhel6 ~]# grep "\>jack" re.file
[root@rhel6 ~]# grep "jack$" re.file
booboo  tom jack
jack
 jack
jack
[root@rhel6 ~]# grep "jack\>" re.file
booboo  tom jack
jack    tom booboo
jack
 jack
jack
```

某个字符数量限定

- x{m}连续 M 个字符 X　　x{3} =3
    - x{m,} 至少 M 个字符 Xx{3,} >=3
- x{m,n} 至少 M 个最多 N 个字符 X x{3,4} >=3 <=4

```
[root@rhel6 ~]# vim re.file
booboo  tom jack
jack    tom booboo
tom tom tom
jack
 jack
jack
f   fo  foo fooo    foooo


[root@rhel6 ~]# grep "o\{0\}" re.file
booboo  tom jack
jack    tom booboo
tom tom tom
jack
 jack
jack
booo
boo
f   fo  foo fooo    foooo
[root@rhel6 ~]# grep "o\{1\}" re.file
booboo  tom jack
jack    tom booboo
tom tom tom
booo
boo
f   fo  foo fooo    foooo
[root@rhel6 ~]# grep "o\{4\}" re.file
f   fo  foo fooo    foooo



#!/bin/bash
if [[ $1 =~ o{2,3}$ ]]
then
        echo ok
else
        echo no
fi

[root@rhel6 ~]# bash re.sh foo
ok
[root@rhel6 ~]# bash re.sh fooo
ok
[root@rhel6 ~]# bash re.sh fooof
no
[root@rhel6 ~]# bash re.sh foof
no
```

[:digit:]代表数字而已 , **0-9**

```
[root@rhel6 ~]# cat re.file1
45aa
sdfsdf
4444
[root@rhel6 ~]# grep "[[:digit:]]" re.file1
45aa
4444
[root@rhel6 ~]# grep "^[[:digit:]]" re.file1
45aa
4444
[root@rhel6 ~]# grep "[^[:digit:]]" re.file1
45aa
sdfsdf
[root@rhel6 ~]# grep "^[^[:digit:]]" re.file1
sdfsdf
```

课堂作业

> 1.说出下面匹配的内容

```
#!/bin/bash
#^a 以a开头
#b? b的个数0或者1
#c+ c的个数>=1
#d.*e d和e之间可以是任意字符
#f$ 以f结尾
if [[ $1 =~ ^ab?c+d.*ef$ ]]
then
        echo ok
else
        echo no
fi
```

课后作业

> 1.if判断匹配ip地址

> 2.if判断匹配邮件地址格式为9aA@9aA.com

> 3.grep 过滤空白行

> 4.grep 过滤以空格开头的行

> 5.针对/usr/share/dict/words文件做过滤

> 1）列出文件中包含 先有字母t，然后中间有一个元音字母，之后是sh的单词；

> 2）列出文件中包含 先有字母t，然后中间有若干个元音字母，之后是sh的单词；

> 3）列出文件中刚好包含16个字母的单词。**

```
#!/bin/bash
if [[ $1 =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]
then
    IP=(${1//\./ })
    [ ${IP[0]} -gt 0 -a ${IP[0]} -lt 255 ] && [ ${IP[1]} -ge 0 -a ${IP[1]} -lt 255 ] && [
${IP[2]} -ge 0 -a ${IP[2]} -lt 255 ] && [ ${IP[3]} -gt 0 -a ${IP[3]} -lt 255 ] && echo ok || echo
no

else
        echo "this is not IPADDR!"
fi


#!/bin/bash
if [[ $1 =~ ^[0-9a-zA-Z]+@[0-9a-zA-Z]+\.com$  ]]
then
    echo ok
else
    echo no
fi


[root@rhel6 ~]# grep "^$" re.file2

grep '^t[a-zA-Z]sh' /usr/share/dict/words
grep '^t[a-zA-Z]\+sh' /usr/share/dict/words
grep -E '^[a-zA-Z0-9]{16}$' /usr/share/dict/words
grep '^[a-zA-Z0-9]\{16\}$' /usr/share/dict/words
```

晚自习作业

1. 完成所有练习
2. 帐号是否合法（字母开头，允许5-16位，只能包含字母数字下划线）
3. 密码是否合法（字母开头，允许6-18位，只能包含字母数字下划线和！@#）
4. 匹配日期2016-10-11

```
#!/bin/bash
#if [[ $1 =~ ^[0-9]{4}-[0-9]{2}-[0-9]{2}$ ]]
if [[ $1 =~ ^[0-9]{4}-(0[1-9]|1[0-2])-(0[1-9]|1[0-9]|2[1-9]|3[01])$ ]]
then
    echo ok
else
    echo no
fi
```