正则表达式就是能用某种模式去匹配一类字符串的公式，它是由一串字符和元字符构成的字符串。所谓元字符，就是用以阐述字符表达式的内容、转换和描述各种操作信息的字符。

正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。简单的说，正则表示式就是处理字符串的方法，它是以行为单位来进行字符串的处理行为， 正则表示式通过一些特殊符号的辅助，可以让使用者轻易的达到搜寻/删除/取代某特定字符串的处理程序。vim、grep、find、awk、sed等命令都支持正则表达式。

基础的正则表达式

1、 “.”(一个点)符号

点符号用于匹配除换行符之外的任意一个字符。例如：r.t可以匹配rot、rut，但是不能匹配root,如果使用r..t就可以匹配root、ruut、r t(中间是两个空格)等。

[root@sunday-test ~]# grep "r..t" /etc/passwd

root:x:0:0:root:/root:/bin/bash

operator:x:11:0:operator:/root:/sbin/nologin

ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin

2、 “*”符号

“*”符号用于匹配前一个字符0次或任意多次，比如ab*，可以匹配a、ab、abb等。 “*”号经常和 “.”符号加在一起使用。比如 “.*”代表任意长度的不包含换行的字符。

例如：

[root@sunday-test ~]# grep "rr*t" /etc/passwd

abrt:x:173:173::/etc/abrt:/sbin/nologin

rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin

查找包含字母r，后面紧跟任意长度的字符，再跟一个字母t的行：

```
[root@sunday-test ~]# grep "r.*t" /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
systemd-bus-proxy:x:999:997:systemd Bus Proxy:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
polkitd:x:998:996:User for polkitd:/:/sbin/nologin
abrt:x:173:173::/etc/abrt:/sbin/nologin
unbound:x:997:994:Unbound DNS resolver:/etc/unbound:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
libstoragemgmt:x:996:993:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
saslauth:x:994:76:Saslauthd user:/run/saslauthd:/sbin/nologin
setroubleshoot:x:992:989::/var/lib/setroubleshoot:/sbin/nologin
rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
gnome-initial-setup:x:989:984::/run/gnome-initial-setup/:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
```

3、 “\{n,m\}”符号

使用 “\{n,m\}”符号可以灵活的控制字符的重复次数，典型的有以下3种形式：

\{n\}　匹配前面的字符n次。

[root@sunday-test ~]# grep "ro\{2\}t" /etc/passwd（r和t中包含两个o）

```
[root@sunday-test ~]# grep "ro\{2\}t" /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

\{n,\}　匹配前面的字符至少n次以上（含n次）

[root@sunday-test ~]# grep "ro\{2,\}t" /etc/passwd

```
[root@sunday-test ~]# grep "ro\{2,\}t" /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
rooot:x:1007:1008::/home/rooot:/bin/bash
```

\{n,m\}　匹配前面的字符n到m次。

4."^"符号

用于匹配开头的字符。

[root@sunday-test ~]# grep "^root" /etc/passwd

root:x:0:0:root:/root:/bin/bash

5、“$”符号

用于匹配结尾的字符。

“abc$”代表以abc结尾

“^$”　代表空行

下面的例子：以r开头,中间有一串任意字符，以h结尾的行。

[root@sunday-test ~]# grep "^r.*h$" /etc/passwd

```
[root@sunday-test ~]# grep "^r.*h$"  /etc/passwd
root:x:0:0:root:/root:/bin/bash
rooot:x:1007:1008::/home/rooot:/bin/bash
rooat:x:1008:1009::/home/rooat:/bin/bash
```

6、“[]”符号

一对方括号，用于匹配方括号内出现的任一字符。

[ABCD]　就是A、B、C、D中的任意一个

[A-Z]　匹配任意一个大写字母

[A-Za-z]　匹配任意一个字母

[^A-Z]　代表不是大写字母　“^”符号在[]里面就是取反

匹配手机号。手机号是11位连续的数字，第一位一定是1，所以表示为“^1”;第二位有可能是3（移动）或8（联通），表示为“[38]”；后面连续9个任意数字，表示为“[0-9]\{9\}”;所以整个表达式为：

“^1[38][0-9]\{9\}”

7、“\”符号　转义符号

例如：021-88888888 和021 88888888　两种不同的电话号码写法，如何匹配呢?

[-]是错误的

[\ \-]是对的　\对空格和-进行了转义

如何匹配任意长度的点号呢？　“\.*”

如果要对“\”符号进行转义，就可以写成：　“\\”

8、 “\<”符号和“\>”符号

这两个符号分别用于界定单词的左边界和右边界。

“\<hello”用于匹配以“hello”开头的单词

“hello\>”则用于匹配以“hello”结尾的单词

“\<\>”用于精确匹配一个字符串。

“\<hello\>”精确匹配单词hello，而不是helloworld等

例如：

```
[root@sunday-test ~]# echo "hello" | grep "\<hello\>"
hello
[root@sunday-test ~]# echo "helloworld" | grep "\<hello\>"
[root@sunday-test ~]#
```

以上8中元字符是最常见的，还有一些不太常用的字符，这些字符中有不少可以使用前面8种基础的元字符来表示。

9、 “\b”符号

匹配单词的边界，比如“\bhello\b”可以精确匹配“hello”单词

10、 “\B”符号

匹配非单词的边界，比如hello\B可以匹配“helloworld”中的"hello"。

11、 “\w”符号

匹配字母、数字和下划线，等价于[A-Za-z0-9]

12、 “\W”符号

匹配非字母、非数字、非下划线，等价于[^A-Za-z0-9]

13、 “\n”符号

匹配一个换行符

14、 “\r”符号

匹配一个回车符

15、 “\t”符号

匹配一个制表符

16、 “\f”符号

匹配一个换页符

17、 “\s”符号

匹配任何空白字符

18、 “\S”符号

匹配任何非空白字符

扩展的正则表达式

在基础正则表达式上做了一些补充。实际上，扩展正则表达式比基础正则表达式多了几个重要的符号。注意，在使用这些扩展符号时，需要使用egrep命令

1、“？”符号

用于匹配前一个字符0次或1次，所以"ro?t" 仅可以匹配rot、rt。

2、“+”符号

用于匹配前一个字符1次以上，所以“ro+t”可以匹配rot、root等。

3、“|”符号

“|”符号是“或”的意思，即多种可能的罗列，彼此间是一种分支关系。比如

区号是3位的固定电话的正则表达式方式

^0[0-9]\{2\}-[0-9]\{8\}

区号是4位的固定电话的正则表达式方式

^0[0-9]\{3\}-[0-9]\{8\}

两种区号的固定电话号码可以这样写：

^0[0-9]\{2,3\}-[0-9]\{8\}

使用“|”符号也可以，但是 显然比上面的方式麻烦

^0[0-9]\{2\}-[0-9]\{8\}|^0[0-9]\{3\}-[0-9]\{8\}

4、“()”符号

“()”符号通常需要和“|”符号联合使用，用于枚举一系列可替换的字符。

使用“()”和“|”匹配hard、hold或hood

h(ar|ol|oo)d

通配符

通配符和正则表达式之间存在一些差异，通配符主要用在文件名的匹配上，正则表达式主要使用在对文件内容的匹配上。

1、“*”符号

代表0个或多个字符

列出当前目录下以“.doc”结尾的文件

[root@sunday-test ~]# ls -l *.doc

2、“?”符号

代表任意一个字符

3、“{}”符号

匹配所有括号内包含的以逗号隔开的字符

[root@sunday-test ~]# touch file{1,2,3}.doc

[root@sunday-test ~]# ls -l file{1..3}.doc

-rw-r--r--. 1 root root 0 Nov 2 16:51 file1.doc

-rw-r--r--. 1 root root 0 Nov 2 16:51 file2.doc
-rw-r--r--. 1 root root 0 Nov 2 16:51 file3.doc
4、“^”符号和“!”符号
这两个符号往往和“[]”一起使用，当出现在“[]”中的时候，代表取反。所以[^A]或[!A]都代表不是A

正则表达式实例

正则表达式和grep结合后会产生强大的搜索功能，由于正则表达式中含有较多特殊的字符，所以结合grep时，最好使用单引号将正则表达式括起来，以免造成错误。

首先创建一个文件RegExp.txt，内容如下：

[root@sunday-test ~]# cat RegExp.txt
----TEXT BEGIN----

good morning teacher
hello world is a script
gold sunshine looks beautiful
golden time flies
god bless me
what a delicious food
they teast Good
you fell glad
wrong word gooood
wrong word g10d
wrong word g12d
wrong word g13d
www.helloworld.com
www@helloworld@com
Upper case
100% means pure
php have a gd module

-----TEXT END-----
搜索含有good单词的行，grep默认是区分大小写的
[root@sunday-test ~]# grep 'good' RegExp.txt
good morning teacher

搜索含有good单词的行，不区分大小写
[root@sunday-test ~]# grep -i 'good' RegExp.txt
good morning teacher
they teast Good
搜索以good开头的行
[root@sunday-test ~]# grep '^good' RegExp.txt
good morning teacher
搜索以Good结尾的行
[root@sunday-test ~]# grep 'Good$' RegExp.txt
they teast Good
统计文件中共有多少行空行
[root@sunday-test ~]# grep -c '^$' RegExp.txt
3
搜索包含good和Good的行
[root@sunday-test ~]# grep '[gG]ood' RegExp.txt
good morning teacher
they teast Good
搜索一个包含ood的行，但是不能是Good或good
[root@sunday-test ~]# grep '[^Gg]ood' RegExp.txt
what a delicious food
wrong word gooood
搜索包含一个词，该词以g开头、紧跟着是两个任意字符、再接着是一个d的行
[root@sunday-test ~]# grep 'g..d' RegExp.txt
good morning teacher
gold sunshine looks beautiful
golden time flies
you fell glad
wrong word g10d
wrong word g12d
wrong word g13d
搜索包含一个词，该词以g或G开头、紧跟着是两个任意字符、再接着是一个d的行
[root@sunday-test ~]# grep '[Gg]..d' RegExp.txt
good morning teacher
gold sunshine looks beautiful
golden time flies

they teast Good

you fell glad

wrong word g10d

wrong word g12d

wrong word g13d

搜索这样一些行，该行包含某个单词，该词满足如下条件：

1、第一个字符可以是G或g

2、第二个字符可以是1或o

3、第三个字符可以是换行符之外的任意字符

4、第四个字符一定是d

[root@sunday-test ~]# grep '[Gg][1o].d' RegExp.txt

```
[root@sunday-test ~]# grep '[Gg][1o].d' RegExp.txt
good morning teacher
gold sunshine looks beautiful
golden time flies
they teast Good
wrong word g10d
wrong word g12d
wrong word g13d
[root@sunday-test ~]#
```

搜索精确匹配含有gold这个单词的行

[root@sunday-test ~]# grep '\<gold\>' RegExp.txt

gold sunshine looks beautiful

[root@sunday-test ~]# grep '\bgold\b' RegExp.txt

gold sunshine looks beautiful

[root@sunday-test ~]# egrep '\bgold\b' RegExp.txt

gold sunshine looks beautiful

搜索这样一些行，该行包含某个单词，该词满足如下条件：

1、以g开头

2、g后面一定有字符

3、最后是d

[root@sunday-test ~]# grep 'g..*d' RegExp.txt

```
[root@sunday-test ~]# grep 'g.*d' RegExp.txt
good morning teacher
gold sunshine looks beautiful
golden time flies
god bless me
you fell glad
wrong word gooood
wrong word g10d
wrong word g12d
wrong word g13d
php have a gd module
```

文件中有一些拼写错误的单词，发现把glod中的o写成数字了，如何找出?

[root@sunday-test ~]# grep 'gl[0-9]d' RegExp.txt

```
[root@sunday-test ~]# grep 'gl[0-9]d' RegExp.txt
wrong word gl0d
wrong word gl2d
wrong word gl3d
```

搜索文件中包含域名www.helloworld.com的行

```
[root@sunday-test ~]# grep 'www.helloworld.com' RegExp.txt
www.helloworld.com
www@helloworld@com
[root@sunday-test ~]# grep 'www\.helloworld\.com' RegExp.txt
www.helloworld.com
```

搜索以字母g开头包含两个以上o的单词

```
[root@sunday-test ~]# grep 'go\{2,\}' RegExp.txt
good morning teacher
wrong word gooood
```

搜索以字母g开头，中间正好包含4个o的单词

```
[root@sunday-test ~]# grep 'go\{4\}' RegExp.txt
wrong word gooood
[root@sunday-test ~]#
```

特殊的POSIX字符，示例如下：

grep支持一类特殊的POSIX字符，列举如下：

#[:alnum:]	文字数字字符
#[:alpha:]	文字字符
#[:digit:]	数字字符
#[:graph:]	非空字符（非空格、控制字符）
#[:lower:]	小写字符
#[:cntrl:]	控制字符
#[:print:]	非空字符（包括空格）
#[:punct:]	标点符号
#[:space:]	所有空白字符（新行，空格，制表符）
#[:upper:]	大写字符
#[:xdigit:]	十六进制数字（0-9，a-f，A-F）

搜索以大写字母开头的行：

```
[root@sunday-test ~]# grep ^[[:upper:]]  RegExp.txt
Upper case
[root@sunday-test ~]#
```

搜索以数字开头的行：

```
[root@sunday-test ~]# grep ^[[:digit:]]  RegExp.txt
100% means pure
[root@sunday-test ~]#
```

使用扩展的正则表达式，搜索g和d之间至少有一个o的行

“+”代表匹配前面的字符1次以上（含1次）

```
[root@sunday-test ~]# egrep 'go+d' RegExp.txt
good morning teacher
god bless me
wrong word gooood
[root@sunday-test ~]#
```

搜索g和d之间只有0个或1个o的行（0次或1次）

#“？”代表匹配前面的字符0次或1次

```
[root@sunday-test ~]# egrep 'go?d' RegExp.txt
god bless me
php have a gd module
[root@sunday-test ~]#
```

搜索有glad或gold的行

```
[root@sunday-test ~]# egrep 'glad|gold' RegExp.txt
gold sunshine looks beautiful
golden time flies
you fell glad
[root@sunday-test ~]#
```

或者这样写：

[root@sunday-test ~]# egrep 'g(la|ol)d' RegExp.txt

附录：常用grep命令选项

1. -A NUM，--after-context=NUM 除了列出符合行之外，并且列出后NUM行。

如： $ grep –A 1 panda file（从file中搜寻有panda样式的行，并显示该行的后1行)

2. -B NUM，--before-context=NUM 与 -A NUM 相对，但这此参数是显示除符合行之外并显示在它之前的NUM行。如: (从file中搜寻有panda样式的行，并显示该行的前1行)

$ grep -B 1 panda file

3、 -C [NUM], -NUM, --context[=NUM] 列出符合行之外并列出上下各NUM行，默认值是2。

如: (列出file中除包含panda样式的行外并列出其上下2行)(若要改变默认值，直接改变NUM即可)

$ grep -C[NUM] panda file

4、 -c, --count 不显示符合样式行，只显示符合的总行数。若再加上-v,--invert-match，参数显示不符合的总行数

5、-i，--ignore-case 忽略大小写差别

6、-n，--line-number 在匹配的行前面打印行号

7、-v，--revert-match 反检索，只显示不匹配的行

8、-s 不显示不存在或无匹配文本的错误信息
如：执行命令grep "root" /etc/password，因为password文件不存在，所以在屏幕上输出错误信息，若使用grep命令-s开关，可屏蔽错误信息