# 什么是正则表达式?

**正则表达式**（Regular Expression、regex或regexp，缩写为RE），也译为正规表示法、常规表示法，是一种字符模式，用于在查找过程中匹配指定的字符。

许多程序设计语言都支持利用正则表达式进行**字符串操作**。在不同语言中，基本字符串匹配规则都是相同的，只是语法有差异。

# 匹配规则

常用匹配：

| 特别字符 | 描述 |
| --- | --- |
| . | 任意单个字符，除了换行符。 |
| * | 前导字符出现0次或连续多次 ab*能匹配“a”，“ab”以及“abb”，但是不匹配“cb”。 |
| .* | 任意长度的字符 ab.* ab123 abbb abab。 |
| ^ | 匹配输入字符串的开始位置，除非在方括号表达式中使用，当该符号在方括号表达式中使用时，表示不接受该方括号表达式中的字符集合。要匹配 ^ 字符本身，请使用 \^。 |
| $ | 匹配输入字符串的结束位置。 |
| ^$ | 匹配空行。 |
| [] | 匹配指定字符组内的任一单个字符 [abc]。 |
| [^] | 匹配不在指定字符组内的任一字符 [^abc]。 |
| ^[] | 匹配以指定字符组内的任一字符开头 ^[abc]。 |
| ^[^] | 匹配不以指定字符组内的任一字符开头 ^[^abc] 。 |
| < | 匹配单词的头。 |
| > | 匹配单词的尾。 |
| <> | 精确匹配单词。 |
| {n} | 匹配前导字符连续出现n次。 |
| {n,} | 匹配前导字符至少出现n次。 |
| {n,m} | 匹配前导字符出现n次与m次之间。 |
| (strings) | 保存被匹配的字符，将括号中的内容视为一个整体。 |

Perl内置匹配：

| 特别字符 | 描述 |
| --- | --- |
| \d | 匹配数字 [0-9]。 |
| \w | 匹配字母数字下划线[a-zA-Z0-9_]。 |

扩展类的正则表达式 grep -E 或则 egrep：

| 特别字符 | 描述 |
| --- | --- |
| + | 匹配一个或多个前导字符。 bo+ boo bo |
| ? | 匹配零个或一个前导字符。 bo? b bo |
| (a\|b) | 匹配a或b。其中a、b代表一组字符。 |

# Shell三剑客之grep工具

语法格式：

```
grep [选项] '关键字' 文件名
```

常用选项：

```
OPTIONS:
    -i: 不区分大小写
    -v: 查找不包含指定内容的行,反向选择
    -w: 按单词搜索
    -o: 打印匹配关键字
    -c: 统计匹配到的行数
    -n: 显示行号
    -R: 逐层遍历目录查找
    -e: 使用正则匹配
    -E:使用扩展正则匹配
    ^key:以关键字开头
    key$:以关键字结尾
    ^$:匹配空行
    --color=auto ：可以将找到的关键词部分加上颜色的显示
```

# cut工具

语法格式：

```
cut 选项 文件名

常用选项：
-c：以字符为单位进行分割，截取
-d：自定义分隔符，默认为制表符\t
-f：与-d一起使用，指定截取哪个区域
```

样例：

```
cut -d: -f1-3 passwd   以:冒号分割，截取第1-3列内容
cut -c3,4,6 passwd     截取文件中每行第3，4，6个字符
```

# sort工具

sort工具用于排序;它将文件的每一行作为一个单位，从首字符向后，依次按ASCII码值进行比较，最后将他们按升序输出。

语法格式：

```
sort 选项 文件名

常用选项：
-u ：去除重复行
-r ：降序排列，默认是升序
-o ：将排序结果输出到文件中,类似重定向符号>
-n ：以数字排序，默认是按字符排序
-t ：分隔符
-k ：指定列数
-b ：忽略前导空格
-R ：随机排序
```

样例：

```
sort -n -t: -k3 1.txt   按照用户的uid进行升序排列
sort -ur 2.txt          按照字符降序去重排序
```

# uniq工具

uniq用于去除连续的重复行。

```
uniq 选项 文件名

常用选项：
-i：忽略大小写
-c：统计重复行次数
-d：只显示重复行
```

样例：

```
uniq -i 3.txt     忽略大小写去重
uniq -ic 3.txt    统计重复次数
uniq -icd 3.txt   只显示重复行
```

## tee工具

tee工具是从标准输入读取并写入到标准输出和文件，即：双向覆盖重定向（屏幕输出|文本输入）。

```
hioier@yunpc:~$ echo "hello cat." | tee -a test.txt
hello cat.
hioier@yunpc:~$ cat test.txt
hello cat.
```

## paste工具

```
常用选项：
-d：自定义间隔符，默认是tab
-s：串行处理，非并行
```

```
hioier@yunpc:~$ cat 1.txt
aaa
bbb
ccc
hioier@yunpc:~$ cat 2.txt
111
222
hioier@yunpc:~$ paste 1.txt 2.txt
aaa     111
bbb     222
ccc
```

```
hioier@yunpc:~$ paste -d: 1.txt 2.txt
aaa:111
bbb:222
ccc:
```

```
hioier@yunpc:~$ paste -s 1.txt 2.txt
aaa     bbb     ccc
111     222
```

## diff工具

diff工具用于逐行比较文件的不同。

语法格式：

```
diff [选项] 文件1 文件2
```

常用选项：

| 选项 | 含义 |
| --- | --- |
| -b | 不检查空格 |
| -B | 不检查空白行 |
| -i | 不检查大小写 |
| -w | 忽略所有的空格 |
| --normal | 正常格式显示(默认) |
| -c | 上下文格式显示 |
| -u | 合并格式显示 |

```
hioier@yunpc:~$ diff -c 1.txt 2.txt
*** 1.txt       2023-01-04 22:08:53.332947213 +0800
--- 2.txt       2023-01-04 22:38:23.498886415 +0800
***************
*** 1,3 ****
! aaa
  bbb
! ccc
--- 1,6 ----
! aza
  bbb
! aza
! bbb
! bbb
! cccc
hioier@yunpc:~$ diff -u 1.txt 2.txt
--- 1.txt       2023-01-04 22:08:53.332947213 +0800
+++ 2.txt       2023-01-04 22:38:23.498886415 +0800
@@ -1,3 +1,6 @@
-aaa
+aza
 bbb
-ccc
+aza
+bbb
+bbb
+cccc
hioier@yunpc:~$
```

```
hioier@yunpc:~$ cat 1.txt
aaa
bbb
ccc
hioier@yunpc:~$
```

```
hioier@yunpc:~$ cat 2.txt
aza
bbb
aza
bbb
bbb
cccc
hioier@yunpc:~$
```

文件补丁：

```
hioier@yunpc:~$ diff -u 1.txt 2.txt > file.patch
hioier@yunpc:~$ cat file.patch
--- 1.txt       2023-01-04 22:08:53.332947213 +0800
+++ 2.txt       2023-01-04 22:38:23.498886415 +0800
@@ -1,3 +1,6 @@
-aaa
+aza
 bbb
-ccc
+aza
+bbb
+bbb
+cccc
hioier@yunpc:~$ patch 1.txt file.patch
patching file 1.txt
hioier@yunpc:~$ diff -u 1.txt 2.txt
```

比较目录：

```
hioier@yunpc:~$ diff AA/ BB/
diff AA/1.txt BB/1.txt
1,6c1
< aza
< bbb
< aza
< bbb
< bbb
< cccc
---
> 1.txt
hioier@yunpc:~$ diff -q AA/ BB/
Files AA/1.txt and BB/1.txt differ
```

## tr工具

tr用于字符转换，替换和删除；主要用于删除文件中控制字符或进行字符转换。

语法格式：

```
用法1：命令的执行结果交给tr处理，其中string1用于查询，string2用于转换处理
# command | tr  'string1'  'string2'

用法2：tr处理的内容来自文件，记住要使用"<"标准输入
# tr  'string1'  'string2' < filename

用法3：匹配string1进行相应操作，如删除操作
# tr options 'string1' < filename
```

常用选项：

```
-d 删除字符串1中所有输入字符。
-s 删除所有重复出现字符序列，只保留第一个；即将重复出现字符串压缩为一个字符串
tr -d '[:/]' < 1.txt        将:/删除，源文件保留，删除后的内容到stdout
head 1.txt | tr -d '[:/]'   管道读取内容
tr '[0-9]' '$' < 1.txt      将数字替换成$
tail 1.txt | tr -s '[a-z]' 匹配小写字母并将重复的压缩为一个
```