**学习正则表达式**

<http://www.zeuux.org/science/learning-regex.cn.html>

### 1 常用的正则表达式元字符

|  |  |  |  |
| --- | --- | --- | --- |
| . | 匹配任意字符 | \* | 修饰匹配符为0次任意次 |
| ^ | 匹配输入行的开始位置 | $ | 匹配输入行的结束位置 |
| + | 修饰匹配符至少1次 | ? | 修饰匹配符0次或1次 |
| [] | 用来自定义匹配“多种字符”表达式 | {} | 修饰[匹配次数](http://www.zeuux.org/science/learning-regex.cn.html#matchmulti)的符号 |
| \s | 空格 | \S | 非“空格”字符 |
| \d | 匹配任何一个0-9数字 | \D | 匹配一个非0-9数字 |
| \w | 匹配一个 数字、字母或下划线 | \W | 匹配一个非 数字、字母和下划线 |
| \*? | 非贪婪修饰匹配符为0次任意次 | +? | 非贪婪修饰匹配符至少一次 |
| ?? | 非贪婪修饰匹配符0次或1次 | () | 把一个表达式做为一个匹配整体 |
| ｜ | 两边的表达式为“或”的关系 |  |  |

### 2 转义字符

在正则表达式中对一些特殊符号和特殊含义的字符的使用需要用“\”进行转义操作，需要转义的字符除了包括以上列出的“元字符”外，还有以下这些具有具体含义的字符：

|  |  |  |  |
| --- | --- | --- | --- |
| \t | 横向制表符 | \n | 换行符 |
| \r | 回车符 | \\ | 转义字符本身 |

### 3 修饰符

用在正则表达式结尾。例如 /asdf/i,“i”即为“修饰符”，意思是不区分大小写的匹配“asdf”。所有修饰符的含义如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| g | 全局匹配 | i | 忽略大小写 |
| m | 多行匹配 | s | 把整个匹配串当作一行处理 |
| x | 允许注释和空格的出现 | U | 非贪婪匹配 |

### 4 正则表达式兼容性对比

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 元字符 | GNU grep | [POSIX BRE](http://www.zeuux.org/science/learning-regex.cn.html#PBRE) | [POSIX ERE](http://www.zeuux.org/science/learning-regex.cn.html#PERE) | GNU Emacs | Perl | Python | T c l |
| . | 任意字符 | 除\0外的任意字符 | | 除\n外的任意字符 | | | 任意字符 |
| [...] | [括号表达式](http://www.zeuux.org/science/learning-regex.cn.html#bracketexpression) | | | [括号表达式](http://www.zeuux.org/science/learning-regex.cn.html#bracketexpression) | [括号表达式](http://www.zeuux.org/science/learning-regex.cn.html#bracketexpression) | | [括号表达式](http://www.zeuux.org/science/learning-regex.cn.html#bracketexpression) | | |
| \(*re*\) | [子表达式](http://www.zeuux.org/science/learning-regex.cn.html#grouping) | | 未支持 | [分组](http://www.zeuux.org/science/learning-regex.cn.html#grouping) | 未支持 | | |  |  |
| *re*\{...\} | [匹配多次](http://www.zeuux.org/science/learning-regex.cn.html#matchmulti) | | 未支持 | [匹配多次](http://www.zeuux.org/science/learning-regex.cn.html#matchmulti) | 未支持 | | |  |  |
| (*re*) | 未支持 | | [子表达式](http://www.zeuux.org/science/learning-regex.cn.html#grouping) | 未支持 | [分组](http://www.zeuux.org/science/learning-regex.cn.html#grouping) | | |  |  |
| *re*{...} | [匹配多次](http://www.zeuux.org/science/learning-regex.cn.html#matchmulti) | 未支持 | [匹配多次](http://www.zeuux.org/science/learning-regex.cn.html#matchmulti) | 未支持 | [匹配多次](http://www.zeuux.org/science/learning-regex.cn.html#matchmulti) | | |  |  |
| *re*{...}? | 未支持 | | | | 非贪婪的[匹配多次](http://www.zeuux.org/science/learning-regex.cn.html#matchmulti) | | |  |  |
| \*digit* | [后续引用](http://www.zeuux.org/science/learning-regex.cn.html#grouping) | | | | | | |  |  |
| ^ | 匹配输入字符的开始位置 | | | | | | |  |  |
| $ | 匹配输入字符的结束位置 | | | | | | |  |  |
| *re*? | 未支持 | | *re*出现0次或1次 | | | | |  |  |
| *re*\* | *re*出现0次或多次 | | | | | | |  |  |
| *re*+ | 未支持 | | *re*出现一次或多次 | | | | |  |  |
| *re*\? | *re*匹配 0 或 1 次 | 未支持 | | | | | |  |  |
| *re*\+ | *re*匹配 1 或 多次 | 未支持 | | | | | |  |  |
| *l*|*r* | 未支持 | | 满足 *l* 或*r* 的匹配 | 未支持 | 满足 *l* 或 *r* 的匹配 | | |  |  |
| *l*\|*r* | *l* 或 *r*匹配 | 未支持 | | *l* 或 *r* 匹配 | 未支持 | | |  |  |
| \*? | 未支持 | | | 非贪婪修饰匹配符为0次或任意次 | | | |  |  |
| +? | 未支持 | | | 非贪婪修饰匹配符至少一次 | | | |  |  |
| ?? | 未支持 | | | 非贪婪修饰匹配符0次或1次 | | | |  |  |
| \A | 未支持 | | | | 匹配字符的结束位置 | | |  |  |
| \b | 单词边缘字符 | 未支持 | | 单词的边缘字符 | | | 未支持 |  |  |
| \B | 非 单词的边缘字符 | 未支持 | | 非 单词的边缘字符 | | | 与\同义 |  |  |
| \C | 未支持 | | | | Any octet | 未支持 | |  |  |
| \d | 未支持 | | | | 匹配任何一个0-9数字 | | |  |  |
| \D | 未支持 | | | | 匹配一个非0-9数字字符 | | |  |  |
| \G | 未支持 | | | | At pos() | 未支持 | |  |  |
| \m | 未支持 | | | | | | 单词的开始 |  |  |
| \M | 未支持 | | | | | | 单词的结束 |  |  |
| \p *property*  \p{ *property*} | 未支持 | | | | Unicode property | 未支持 | |  |  |
| \P*property*  \P{*property*} | 未支持 | | | | Not unicode property | 未支持 | |  |  |
| \s | 未支持 | | | | 空格 | | |  |  |
| \S | 未支持 | | | | 非空格字符 | | |  |  |
| \w | 匹配一个 数字或字母 | 未支持 | | 同 \sw 一致 | 匹配一个 数字、字母和下划线 | | |  |  |
| \W | 配一个非 数字或字母 | 未支持 | | 同 \Sw 一致 | 配一个非 数字、字母和下划线 | | |  |  |
| \Z | 未支持 | | | | 匹配字符或行的结束位置 | 匹配字符的结束位置 | |  |  |
| \z | 未支持 | | | | 匹配的单个字符串结尾 | 未支持 | |  |  |
| \< | 单词开始位置 | 未支持 | | 单词结束位置 | 未支持 | | |  |  |
| \> | 单词结束位置 | 未支持 | | 单词结束位置 | 未支持 | | |  |  |
| (?#*text*) | 未支持 | | | | 注释，忽略 | | |  |  |
| (?*modifiers*) | 未支持 | | | | 内置修饰符 | | |  |  |
| (?:*re*) | 未支持 | | | | 括号内的内容不计入$1，$2等变量 | [Shy grouping](http://www.zeuux.org/science/learning-regex.cn.html#grouping) | |  |  |
| (?=*re*) | 未支持 | | | | Lookahead | | |  |  |
| (?!*re*) | 未支持 | | | | Negative lookahead | | |  |  |
| (?{ *code* })  (??{ *code* }) | 未支持 | | | | 嵌入的Perl程序 | 未支持 | |  |  |
| (?>*re*) | 未支持 | | | | 独立表达式 | 未支持 | |  |  |
| (?(*cond*)*re*)  (?(*cond*)*re*|*re*) | 未支持 | | | | 条件选择 | 未支持 | |  |  |
| 元字符 | GNU grep | [POSIX BRE](http://www.zeuux.org/science/learning-regex.cn.html#PBRE) | [POSIX ERE](http://www.zeuux.org/science/learning-regex.cn.html#PERE) | GNU Emacs | Perl | Python | Tcl |  |  |

#### 4.1 说明

BRE: POSIX "basic regular expressions"。传统的Unix grep（除去grep -E使用）即遵循这个标准。

ERE: POSIX "Extended regular expressions"。传统Unix的egrep和grep -E均使用此标准。

grep: 传统Unix的grep是使用BREs;grep -E使用时使用EREs；grep -F 不使用任何正则表达式。但是现在的grep一般使用GNU grep，这个是在ERES的基础上加入了一些扩展。

sed: 传统Unix的sed是使用BREs;FreeBSD的sed -E使用EREs；GNU sed -r使用和GNU grep相似的正则表达式标准。

awk: 使用EREs，但是加入了\\,\a,\b,\f,\n,\r,\t,\v等转义字符。

regex(3): 根据的不同的参数可以使用BREs和EREs。

PCRE: 使用Perl的正则表达式。

Perl: Perl对正则表达式的支持最好，并且速度也很快。详细的用法介绍见 [Perl RE](http://perldoc.perl.org/perlre.html)。

Python: Python使用正则表达式需要使用引用re module，关于其更多介绍见[re module](http://docs.python.org/lib/module-re.html)。

#### 4.2 括号表达式

括号表达式是POSIX自定义的一套表达式规范。有以下规范要遵守：

0. 如果在“[”后紧接着“^”，那么整个括号表达式中的取“反”匹配。

也就是说，如果括号表达式以“^”开头，那么在被匹配字符中任何一个在“括号表达式”中的字符都不能出现。

1. 如果在“[”后是除“^”以外的字符，那么只要被匹配字符串中有一个在“括号表达式”中，那么即匹配成功。

2. 符号类型。类似于[:class:]，表示一个已定义的符号集合。已知的部分符号集合如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [:alnum:] | 所有的数字和字母（A－Z，a-z，0－9） | [:alpha:] | 所有的字母（A－Z，a－z） | [:cntrl:] | 任何一个控制字符（\x00 - \x7F） |
| [:graph:] | 任何一个可以显示的ASCII字符，不包括空格（\x21 - \x7E） | [:lower:] | 任何一个小写字母（a-z） | [:upper:] | 任何一个大写字母（A-Z） |
| [:digit:] | 任何一个数字（0－9） | [:blank:] | 空格和制表符（横向和纵向） | [:space:] | 任何一个空白字符（\x09 - \x0D，\x20） |

3. 表示数字或字母的范围。例如，[3-6]表示从3到6的任意一个数字；[2-5]表示从A-H的任意一个字母。注意，如果“－”不是出现在在字符或字母中间，那么被为普通字符。

#### 4.3 匹配多次

关于“匹配次数”的语法的使用有两种，一种是使用\{\}；一种是{}，但是无论如何含义都是一样的：

* RE{N}:是精确的匹配RE N次
* RE{N,}:会匹配RE N次或多于N次
* RE{N,M}:会匹配RE在N次和M次之间
* RE{,M}:匹配RE小于或等于M次

#### 4.4 子表达式、分组、Shy grouping、后序引用

使用小括号“(”“ )”括起来的表达式为“子表达式”或“分组”。他有两个做用：A，他把括号中的内容作为一个整体匹配；B，其后的修饰符（+、?、\*等）对括号的整体有效；C，并且这个其中匹配到的内容能通过“\digit”（后续引用）在正在表达式内部或着外部引用

"Shy grouping"：高优先级匹配，而非“捕捉”的整体

### 5 参考资料

[wikipedia regular expression](http://myspacesurfer.hee-ro.net/index.php?hl=f5&q=uggc%3A%2F%2Fra.jvxvcrqvn.bet%2Fjvxv%2FErthyne_rkcerffvba)   
[POSIX Basic and Extended Regular Expressions](http://www.regular-expressions.info/posix.html)   
[Perl RE](http://perldoc.perl.org/perlre.html)   
[Python RE](http://docs.python.org/lib/module-re.html)