

通配符

通配符是系统级别的 而正则表达式需要相关工具和语言的支持: `egrep`, `awk`, `vi`, `perl`。

当您键入 `ls .txt` 命令并按 *Enter* 后, 寻找哪些文件同 `.txt` 模式相匹配的任务不是由 `ls` 命令, 而是由 `shell` 自己完成。这需要对命令行是如何被 `shell` 解析的作进一步解释。当您键入:

通配符语法: *

* 将与零个或多个字符匹配。这就是说“什么都可以”。例子:

```
/etc/g* 与 /etc 中以 g 开头的所有文件匹配。  
/tmp/my*.txt 与 /tmp 中以 my 开头, 并且以 .txt 结尾的所有文件匹配。
```

```
$ ls *.txt  
  readme.txt  recipes.txt
```

该命令首先被分割成一系列单词(本例中的 `ls` 和 `.txt`)。当 *shell* 在某个单词中发现了, 它会将整个单词当作通配模式解析, 并用所有相匹配的文件替换它。因此, 该命令在执行前就变为 `ls readme.txt recipe.txt`, 而这将得到期望的结果。其余通配符有:

通配符语法: ?

? 与任何单个字符匹配。例子:

```
myfile? 与文件名为 myfile 后跟单个字符的任何文件匹配。  
/tmp/notes?txt 将与 /tmp/notes.txt 和 /tmp/notes_txt 都匹配, 如果它们存在。
```

通配符语法: [] [!]

该通配符与 ? 相似, 但允许指定得更确切。要使用该通配符, 把您想要匹配的所有字符放在 [] 内。结果的表达式将与 [] 中任一字符相匹配。您也可以用 - 来指定范围, 甚至还可以组合范围。

[...]: 同方括号中的任意一个字符相匹配。这些字符可以用字符范围(比如 1-9)或者离散值或同时使用两者表示。例如: `[a-zA-Z0-9]` 同所有 a 到 z 之间的字符和 B、E、5、6、7 相匹配。

[!...]: 与所有不在方括号中的某个字符匹配。例如 `[!a-z]` 同某个非小写字母相匹配[5];

例子：

`myfile[12]` 将与 `myfile1` 和 `myfile2` 匹配。只要当前目录中至少有一个这样的文件存在，该通配符就可以进行扩展。

`ls /etc/[0-9]*` 将列出 `/etc` 中以数字开头的所有文件。

`ls /tmp/[A-Za-z]*` 将列出 `/tmp` 中以大写字母或小写字母开头的所有文件。

`rm myfile[!9]` 将删除除 `myfile9` 之外的名为 `myfile` 加一个字符的所有文件。

`{c1,c2}`: 同 `c1` 或者 `c2` 相匹配。其中 `c1` 和 `c2` 也是通配符。因此，您可以使用 `{[0-9]*,[acr]}`。

正则表达式

基本语法

- 选择

| 垂直分隔符代表选择。例如“`gray|grey`”可以匹配`grey`或`gray`。

- 数量限定

+ 加号代表前面的字符必须至少出现一次。（1次、或多次）。例如，“`goo+gle`”可以匹配`google`、`gooogle`、`goooogle`等

? 问号代表前面的字符最多只可以出现一次。（0次、或1次）。例如，“`colou?r`”可以匹配`color`或者`colour`

* 星号代表前面的字符可以不出现，也可以出现一次或者多次。（0次、或1次、或多次）。例如，“`0*42`”可以匹配`42`、`042`、`0042`、`00042`等。

- 匹配

圆括号可以用来定义操作符的范围和优先度。例如，“`gr(ale)y`”等价于“`gray|grey`”，“`(grand)?father`”匹配`father`和`grandfather`。

上述这些构造子都可以自由组合，因此，“`H(ae?lb)ndel`”和“`H(alaelb)ndel`”是相同的。

更多表达式

- (pattern)

匹配pattern并获取这一匹配的子字符串。该子字符串用于向后引用。所获取的匹配可以从产生的Matches集合得到

- (?:pattern)

匹配pattern但不获取匹配的子字符串，也就是说这是一个非获取匹配

- (?=pattern)

正向肯定预查，在任何匹配pattern的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如，“Windows(?=95I98INTI2000)”能匹配“Windows2000”中的“Windows”，但不能匹配“Windows3.1”中的“Windows”

- (?!pattern)

正向否定预查，在任何不匹配pattern的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如“Windows(?!95I98INTI2000)”能匹配“Windows3.1”中的“Windows”，但不能匹配“Windows2000”中的“Windows”

- (?<=pattern)

反向肯定预查，与正向肯定预查类似，只是方向相反。例如，“(?<=95I98INTI2000)Windows”能匹配“2000Windows”中的“Windows”，但不能匹配“3.1Windows”中的“Windows”

- (?<!pattern)

反向否定预查，与正向否定预查类似，只是方向相反。例如“(?!<95I98INTI2000)Windows”能匹配“3.1Windows”中的“Windows”，但不能匹配“2000Windows”中的“Windows”

[More](#)