正则表达式快速参考手册

胡志飞

<WisdomFusion[at]gmail[dot]com>

2012 年之旧文重拾于 2016 年 2 月 26 日
version 0.1.0

目　录

1 简介 **1**

2 基本语法 **2**

3 高级语法 **8**

4 举些例子吧！ **11**

5 正则表达式“流派” **12**

6 应用场景 **15**

6.1 正则表达式工具箱 15

6.2 应用案例 20

6.2.1 Perl 正则表达式王国 20

6.2.2 Python 20

6.2.3 PHP PCRE 20

6.2.4 sed & awk 20

6.2.5 grep 20

6.2.6 Swift 20

6.2.7 Java 20

6.2.8 .NET Framework 正则表达式 20

6.2.9 JavaScript 21

6.2.10 Adobe Dreamweaver 表格处理 21

6.2.11 VBA 中使用正则表达式 21

6.2.12 GREP for Adobe InDesign 22

6.2.13 神的编辑器 GNU Emacs 之正则神器 22

7 后记 **22**

1 简介 Introduction

文字处理无处不在无时不有，日常工作和学习大多数任务都和文字息息相关，编辑们写文章、整理资料，开发人员编码、处理用户提交的数据或请求接口数据，等等，这些都是以字符和字符串相关的任务，既然如些，掌握一个快速文字处理的方法就变得很有必要。

正则表达式，（Regular Expression，在代码中常简写为 regex、regexp 或 RE），计算机科学的一个概念。正则表达式使用字符来描述、匹配一系列符合某个句法规则的字符串。在很多文本编辑器里，正则表达式通常被用来检索、替换那些符合某个模式的文本。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。例如，在 Perl[①]中就内建了一个功能强大的正则表达式引擎。正则表达式这个概念最初是由 Unix 中的工具软件（例如 sed[②]和 grep[③]）普及开的。

需要注意的是，用什么工具，用什么编辑语言，正则表达式的语法有些差别，特性的支持也参差不齐，称之为正则表达式“流派”(第5部分详述)，所以要单独参考工具和编程语言本身的文档才行。本文档旨在给大家一个通用的、概括的正则表达式宏观印象，辅以实例和应用案例，同时针对个别常用但又不易理解的特性，给大家作详细说明和总结，抛砖引玉。

期望本文档能给大家一个快速的参考，快速的掌握正则表达式这个棒棒哒效率工具，让大家平时工作学习中更加得心应手！☺

说明

我对排版及专业出版知之甚微，只是在平时笔记和文档时，哪怕是自己写的太乱的话也不乐意翻看，印象笔记里的东东又过于零碎，故把旧文完善并整排[a]。然而，专业领域知识因涉猎过多而不精，难保周全和准确，但只要在自己知识圈内，我会劲力完成尽可能规范和可靠的文档呈现给大家，并不断完善更新，请批评指正，共同提高。

[a]本文档 2012 年编写，现整拾，并加以完善。目前觉得 Markdown, org-mode适用于文本文档的快速编写，而 Adobe InDesign 和 LaTeX 适合更专业的文档和书籍的图文混排及设计。

[①]Perl被称为“实用报表提取语言”（Practical Extraction and Report Language），正则表达式特性的推动者，文本处理非常方便。

[②]sed是一种 UNIX/Linux 平台下的轻量级流编辑器，日常一般用于处理文本文件。

[③]grep，global search regular expression and print out the line，是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。

2 基本语法 Basic Syntax

语法部分结合了自己的理解和对正则表达式应用的一些心得，分类有不当之处，请指正。因为是总结性的参考文档，所以这里使用“大表哥”形式展示，列出语法的同时，关键语法举了几个栗子加强理解。

特性	语法	描述	举个栗子
字符	除 [\^$.|?*+() 以外的任意字符	除了 [\^$.|?*+() 以外的任意字符，{和} 也是文字文本，除了下面说到的成对出现的量词语法，如 {n} 和 {m,n} 等。	a 匹配 about 中的 a
	字符转义	\t , \? , * , \+ , \. , \\| , \{ , \} , \\ , \[, \] , \(, \)	\+ 匹配 + ； \?\- 匹配 ?-
	\n , \r 和 \t	Windows 文件格式换行符是\r\n，UNIX 文件格式换行符是\n，\t 匹配水平制表符	
	\cA 到 \cZ , \ca 到 \cz	Ctrl+A 到 Ctrl+Z，与 ASCII 字符 \x01 到 \x1A 等价	
	\a , \e , \f , \v	依次为警报（\x07 ）、Esc 字符 (\x1B)、进纸符（\x0C ）和垂直制表符（\x0B ）	
	\Q ... \E	文字文本范围，被包含在 \Q 和 \E 之间的文字，都被视为普通文字，如 [\^$.|?*+(){} 也不再用转义了，这个最早是由 Perl 引入正则表达式的。	\Q+-*/\E 匹配的就是 +-*/
基本特性	.（点）	匹配除换行符之外的任意字符，有些正则表达式“流派”还支持点是否匹配换行符的开关。	. 匹配 about 中的任意一个字符

To be continued...

（续表）

特性	语法	描述	举个栗子
	|	管道，或的关系，匹配 | 的左侧或右侧的字符串	abc|def|xyz 匹配 abc 或 def 或 xyz
字符类	[...]	匹配字符类中列举的任意一个字符	[abc] 匹配 a 或 b 或 c [aeiou] 匹配任何一个英文元音字母 [.!?] 匹配 . 或 ! 或 ?
	[\^\]]	在字符类中，要匹配 ^-]\这几字符，得使用\转义	[\^\]] 匹配 ^ 或]
	[^...]	排除型字符类，^（脱字符，caret）紧跟 [之后，可以把字符类中列举的字符排除匹配范围，也就是所这个字符类将匹配任意一个不在列出字符范围内的字符	[^a-d] 匹配除了 a,b,c,d 之外的任意一个字符
	\d，\w，\s	\d 匹配数字，与 [0-9] 等价；\w 匹配任意一个字母或数字或下划线或汉字；\s 匹配任意一个空白符	[\d\s] 匹配一个数字或空白符
	\D，\W，\S	是 \d，\w 和 \s 的反义字符类。\D 匹配任意非数字的字符；\W 匹配任意不是字母、数字、下划线、汉字的字符；\S 匹配任意不是空白符的字符	\D 匹配任意非数字的字符
	[\b]	在字符类中，[\b] 为Backspace退格键字符	
POSIX	[:alnum:]	匹配所有大小写字母及数字	等价于 [0-9a-zA-Z]

To be continued...

（续表）

特性	语法	描述	举个栗子
	[:alpha:]	匹配所有大小写字母	等价于 [a-zA-Z]
	[:ascii:]	匹配所有 ASCII 字符，查看完整ASCII 字符列表	等价于 [\x01-\x7F]
	[:blank:]	匹配半角空格和制表符	等价于 [\t]
	[:cntrl:]	匹配所有 ASCII 0 到 31 之间的控制符	等价于 [\x01-\x1F]
	[:digit:]	匹配所有数字	等价于 [0-9]
	[:graph:]	匹配所有可打印的字符	
	[:lower:]	匹配所有小写字母	等价于 [a-z]
	[:print:]	匹配所有可打印字符和空格	
	[:punct:]	匹配所有标点符号	
	[:space:]	空白字符	等价于 [\t\n\r\f\v]
	[:upper:]	匹配所有大写字母	等价于 [A-Z]
	[:word:]	字母、数字和下划线	等价于 [a-zA-Z0-9_]
	[:xdigit:]	匹配所有十六进制字符	等价于 [0-9a-fA-F]

To be continued...

（续表）

特性	语法	描述	举个栗子
锚点	^	匹配字符串开始位置或行首位置	单行模式下 ^. 在 foo\nbar 中匹配 f；在多行模式下，同时还匹配换行后的 b
	$	匹配字符串结尾位置或行尾位置	.$ 在 foo\nbar 中匹配 r；在多行模式下，同时还匹配换行符前的 o
	\A	字符串开头位置（类似 ^，但不受处理多行选项的影响）	\Ae 在 example 这个字符串中匹配开头的 e
	\Z	字符串结尾位置或行尾位置（不受处理多行选项的影响）	e\Z 在 example 这个字符串中匹配结尾的 e
	\b	单词分界位置，单词开头或结尾	.\b 在字符串 abc 中匹配 c
	\B	匹配不是单词开头或结尾的位置	\B.\B 在字符串 abc 中匹配 b
	\<	单词开头	
	\>	单词结尾	

To be continued...

（续表）

特性	语法	描述	举个栗子
量词	?	前导字符重复零次或一次，贪婪的[①]：当正则表达式中包含能接受重复的限定符时，通常的行为是（在使整个表达式能得到匹配的前提下）匹配尽可能多的字符。	abc? 匹配 abc 或 ab，如果可能，优先匹配前者
	??	前导字符重复零次或一次，非贪婪[②]：当正则表达式中包含能接受重复的限定符时，通常的行为是（在使整个表达式能得到匹配的前提下）匹配尽可能少的字符。与贪婪相反。	abc?? 匹配 ab 或 abc
	*	前导字符重复零次或更多次，贪婪的	
	*?	前导字符重复零次或更多次，非贪婪	
	+	前导字符重复一次或更多次，贪婪的	
	+?	前导字符重复一次或更多次，非贪婪	
	{n}	前导字符重复 n 次	

To be continued...

[①]贪婪模式：当正则表达式中包含能接受重复的限定符时，通常的行为是（在使整个表达式能得到匹配的前提下）匹配尽可能多的字符。考虑这个表达式：a.*b，它将会匹配最长的以 a 开始，以 b 结束的字符串。如果用它来搜索 aabab 的话，它会匹配整个字符串 aabab。这被称为贪婪匹配。

[②]相反，非贪婪，即匹配尽可能少的字符，只要在量词后面加上一个问号?。如 a.*?b 匹配最短的，以 a 开始，以 b 结束的字符串。如果把它应用于 aabab 的话，它会匹配 aab（第一到第三个字符）和 ab（第四到第五个字符）。

（续表）

特性	语法	描述	举个栗子
	{n,m}	前导字符重复 n 到 m 次，其中 $n >= 0$，$m >= n$	
	{n,}	前导字符重复 n 次或更多次，其中 $n >= 0$	
	{,m}	前导字符最多重复 m 次，基中 $m >= 0$	
分组与反向引用	(regex)	匹配 regex，并捕获文本到自动命名的组里	(abc){3} 匹配 abcabcabc
	(?:regex)	匹配 regex，不捕获匹配的文本，也不给此分组分配组号	(?:abc){3} 匹配 abcabcabc，无分组
	\1 到 \9	反向引用，用于重复搜索前面某个分组匹配的文本。例如，\1 代表分组 1 匹配的文本。有些与此正则表达式流派支持多于 9 的分组。左括弧顺序即是分组序号的顺序。见图1。	(abc|def)=\1 匹配 abc=abc 或 def=def，而不是 abc=def 或 def=abc
	\10 到 \99	反向引用，分组 10 到 99	
	\g{1} 到 \g{99}	Perl 语法中，反向引用语法优化[①]	避免出现歧义，同时用负数分组还能倒序引用。
	\g{-1}, \g{-2}, etc.	倒数第 1 个分组，倒数第 2 个分组，...	
	(?<name>regex)	命令分组	命名分组的最大好处是反向引用时不用再怕弄错分组了。

To be continued...

[①]如果想实现类似 (.)\1 的效果，若紧跟的字符和分组号相同，如 (.)\111，这样正则表达式引擎就不知所措，如果用 \g{1}11 这种语法就不会有歧义，最重要的是这种语法或以用负数分组号倒序选用分组。

（续表）

特性	语法	描述	举个栗子
	\k<name>	反向引用命令分组，Perl 中也可以使用\g{name}。	
	\`	正则表达式匹配部分之前的字符串，Perl 语言中也作 ${^PREMATCH}	
	\&	正则表达式匹配的部分，Perl 语言中也作 ${^MATCH}	
	\’	正则表达式匹配部分之后的字符串，Perl 语言中也作 ${^POSTMATCH}	

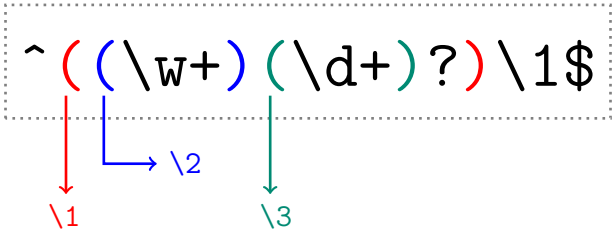

图 1: 正则表达式分组序号

3 高级语法 Advanced Syntax

之所以本文档中称这些正则为“高级语法”，一是有些不常用，二是有些语法不太好理解，故有此一说。

特性	语法	描述	举个栗子
模式修饰符	(?i)	打开忽略大小写模式，之后的模式不分大小写。模式修饰符有 i, s, m, x 四种，分别是忽略大小写（**I**gnoreCase）、单行模式（**S**ingleline）、多行模式（**M**ultiline）和注释模式	(?i)te(?-i)st 匹配 TEst，而不匹配 TEST
	(?-i)	关闭忽略大小写模式，之后的模式不分大小写	
	(?s)	打开单行模式，之后的模式不支持多行	默认情况下 . 是不匹配换行的，打开该模式后，待匹配的字符将为视为“一行”。
	(?-s)	关闭单行模式，之后的模式支持多行	
	(?m)	打开多行模式，之后的模式支持多行	
	(?-m)	^ 和 $ 匹配行首和行尾	
	(?x)	打开宽松和注释模式	打开该模式后，可以在正则表达式中插和空白和换行，使正则表达式可读性增强。
	(?-x)	关闭宽松和注释模式	
	(?i-sm)	打开 i 和 m 模式，关闭 s 模式	以上几种模式可以组合使用
	(?i-sm:regex)	在 (?i-sm:regex) 子模式内打开 i 和 m 模式，关闭 s 模式	
注释	(?#comment)	注释	

To be continued...

（续表）

特性	语法	描述	举个栗子
零宽断言	(?=Regex)	肯定顺序环视（Negative Lookahead） 子表达式 能 够匹配 右侧 文本，含以下三种些类正则被称“**环视**（Lookaround，Lookahead 和 Lookbehind 统称为 Lookaround）”，也称为“**零宽断言**”（Zero-Length Assertions）	\b\w+(?=ing\b)，匹配以 ing 结尾的单词的前面部分（除了 ing 以外的部分）
	(?!Regex)	否定顺序环视（Positive Lookahead） 子表达式 不能 匹配 右侧 文本	(?<=\bre)\w+\b 会匹配以 re 开头的单词的后半部分（除了 re 以外的部分） 又，(?<=\s)\d+(?=\s) 匹配以空白符间隔的数字（再次强调，不包括这些空白符）
	(?<=regex)	肯定逆序环视（Positive Lookbehind） 子表达式 能 够匹配 左侧 文本	\d{3}(?!\d) 匹配三位数字，而且这三位数字的后面不能是数字
	(?<!regex)	否定逆序环视（Negative Lookbehind） 子表达式 不能 匹配 左侧 文本	(?<![a-z])\d{7} 匹配前面不是小写字母的七位数字 又，(?<=<(\w+)>).(?=<\/\1>) 匹配不包含属性的简单 HTML 标签内里的内容

To be continued...

（续表）

特性	语法	描述	举个栗子
固化分组	(?>regex)	贪婪子表达式，也称“**固化分组**”，使用它可以加快匹配失败的速度，如 Subject 这个字符串，现用 ^\w: 对其进行匹配，正则表达式引擎发现 Subject 不匹配，就会试图匹配 Subjec，一直尝试到 S，发现都不匹配才得出无法匹配的结论。如果使用固化分组 ^(?>\w+):，它会直接试图使用 \w+ 去匹配 Subjec 字符串，而不会一一回溯，发现 \w+ 后面没有 :，立即报告失败。	如果字符串中没有第二个 x 的时候，x(?>\w+)x 要比 x\w+x 高效得多

4 举些例子吧！ Regex Examples

Email 地址

`^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$` 匹配形如 WisdomFusion@gmail.com 的邮箱地址。

日期

`^\d{4}\-(0?[1-9]|1[012])\-(0?[1-9]|[12][0-9]|3[01])$` 匹配形如 yyyy-mm-dd 格式的日期。

非零负整数

`^\-[1-9][0-9]*$` 匹配如 $-2, -1024, \ldots$之类的非零负整数。

匹配浮点数

`[-+]?([0-9]*\.[0-9]+|[0-9]+)`. 匹配如 3.1415926 的浮点数（带小数位的）。

去除重复行

查找`^(.*)(\r?\n\1)+$`，替的为`\1`。

匹配用户名

```
^[a-z0-9_-]{3,16}$
```

网址 URL

```
^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$
```

IPv4 地址

```
^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$
```

HTML 标记

```
<([a-z]+)([^<]+)*(?:>(.*)<\/\1>|\s+\/>)
```

UBB 代码清理

把`\[/?(?:font|size)([^\]]+)?\]` 替换为空。

汉字

```
[^u4E00-u9FA5]
```

5 正则表达式“流派” Regex Flavors

在标准制定之前，各家正则自成一派，不同工具不同编程语言各不相干。为了理清正则表达式的混乱局面，POSIX[①]把各种常见的流派分为两大类：*Basic Regular Expressions*(BREs) 和 *Extended Regular Expressions*(EREs)。POSIX 程序必须支持其中的任意一种标准。

[①]诞生于 1986 年的 POSIX 是 Portable Operating System Interface（可移植操作系统接口）的缩写，这是一系列标准，确保操作系统之间的移植性。

表3（POSIX 正则表达式流派概览）中列表出两大流派对正则表达式特性支持的情况，随着文档的完善和丰富，特性的总结将会更加全面。

正则表达式特性	BREs	EREs
点、^、$、[...]、[^...]	✓	✓
“任意数目”量词	*	*
+ 和? 量词		+ ?
区间量词	\{m,n\}	{m,n}
分组	\{…\}	
量词可否作用于括号	✓	✓
反向引用	\1 到\9	
多选结构		✓

表 3: POSIX 正则表达式流派概览

标准定了，流派也有了，那么各家支持的如何呢？表4（若干工具的正则表达式流派对比）作了简单的对比，可以看出，不同的工具和编程语言，虽没有统一语言，但相差不大，在具体使用某种工具或语言时需要单独了解和掌握。

特性	grep	egrep	GNU Emacs	Tcl	Perl	.NET	Java
*、^、$、[…]	✓	✓	✓	✓	✓	✓	✓
? + |	\? \+ |	? + |	? + |	? + |	? + |	? + |	? + |
分组	\(…\)	(…)	\(…\)	(…)	(…)	(…)	(…)
(?:…)					✓	✓	✓
单词分界符		\<\>	\<\>、\b\B	\n\M\y	\b\B	\b\B	\b\B
\w、\W		✓	✓	✓	✓	✓	✓
反向引用	✓	✓	✓	✓	✓	✓	✓

表 4: 若干工具的正则表达式流派对比

6 应用场景 Application Scenarios

6.1 正则表达式工具箱 Regex Toolbox

总有一款适合你，Windows 下的记事本太鸡肋，Word 处理方式主要是“通配符”而不是正则表达式。

JGsoft RegexBuddy

JGsoft 开发的一个强大的正则表达式测试工具，这款是正则测试界最最强大的工具了，**没有之**一，要墙裂向大家推介的哦！☺

图2中所示，①正则表达式区域，②替换字符区域，③暂存使用的正则表达式以备后用，④待测试的文字，⑤替换后的文字。没有标注的区域还有一大堆很贴心的正则表达式创建功能，比如不同语言的选择、模式的开关、正则表达式库等等，正如其名“正则表达式好兄弟”，我平时更喜欢“正则基友”这个简称，☺ 建议安装长期占有之。

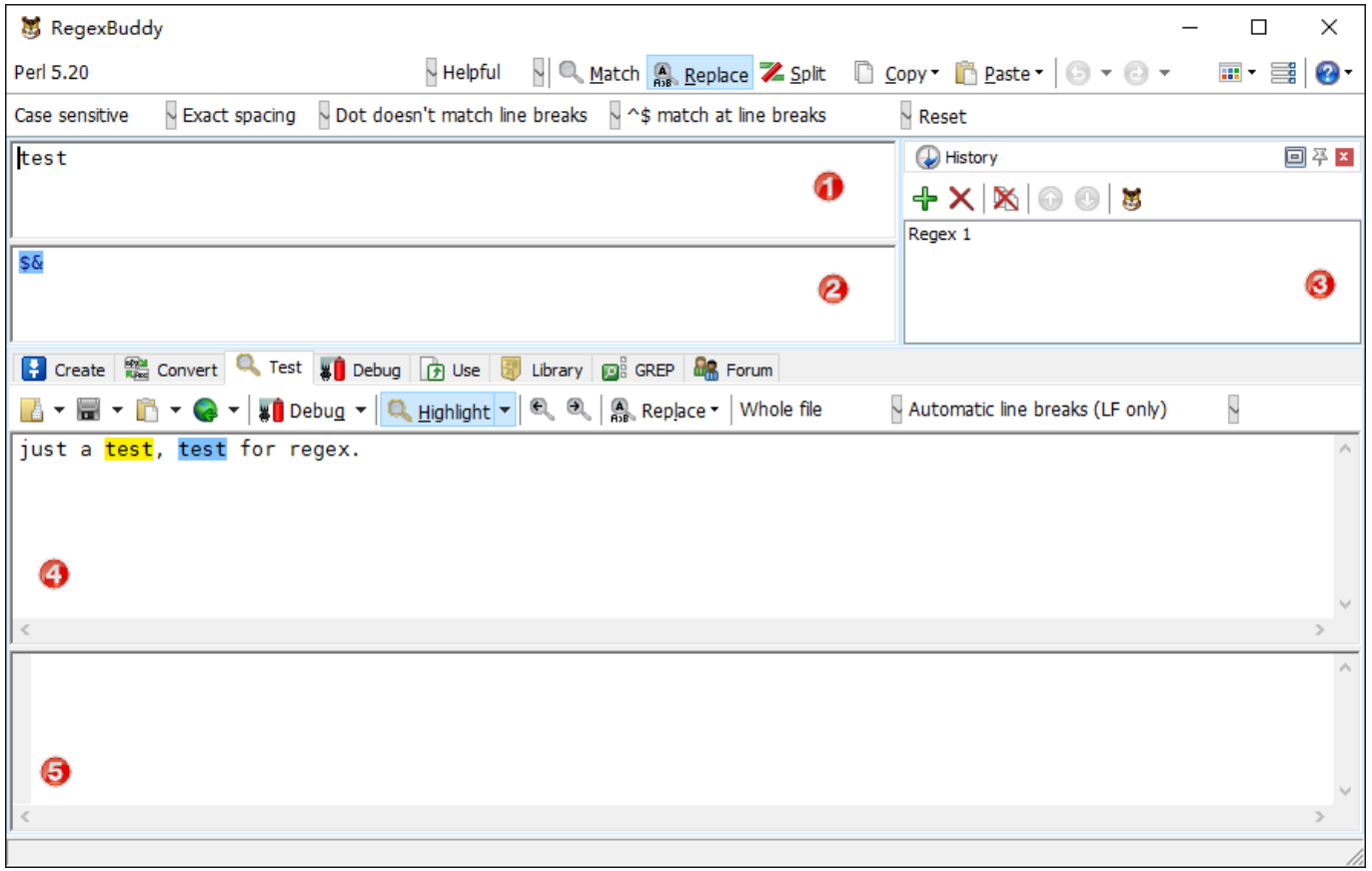

图 2: RegexBuddy 界面

JGsoft PowerGREP

PowerGREP RegexBuddy 的兄弟软件，同是 JGsoft 开发，是 grep 在 Windows 平台的实现和增强。

Debuggex

https://www.debuggex.com/

grep

grep

UltraEdit, Notepad++

UltraEdit, Notepad++

Vim

编辑器之神 Vim

GNU Emacs

GNU Emacs 中正则表达式异常强大，除了基本的正向正则查找C-M-s、反向正则查找C-M-r、正则替换 `M-x replace-regexp`、查询式正则替换 `M-x query-replace-regexp`、保留行 `M-x keep-lines`、删除行 `M-x flush-lines`、...等等，Emacs 正则表达式替换时还能直接执行 LISP[①] Form[②]。详见6.2.13。

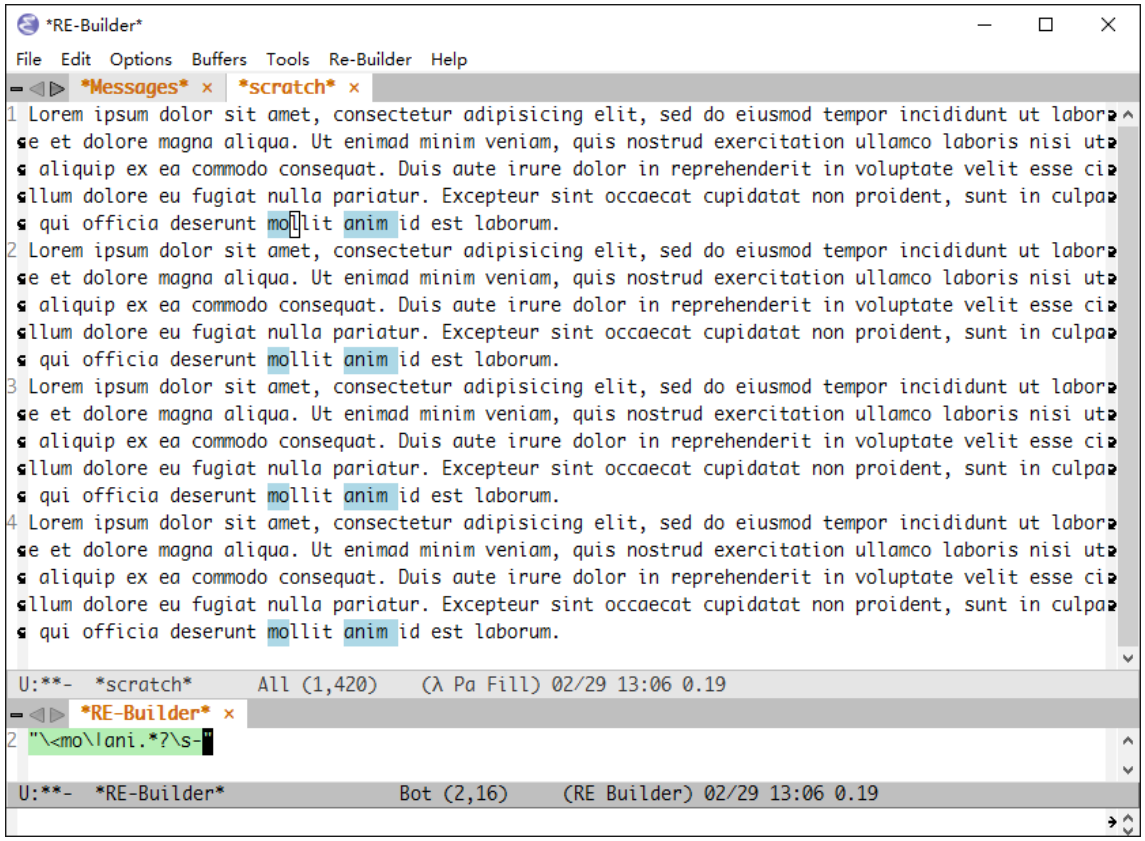

图 3: GNU Emacs 中 RE-Builder 模式

[①]出生自 1958 年，但目前仍很活跃的一门编程语言，尤其在 AI 领域中。

[②]简言之：可以直接执行 LISP 代码，利用强大的编程语言提供的函数方法去处理文本。

sed & awk

sed and awk

6.2 应用案例 Application Cases

6.2.1 Perl 正则表达式王国

6.2.2 Python

Python

6.2.3 PHP PCRE

PHP 中使用 PCRE[①]

6.2.4 sed & awk

sed
awk

6.2.5 grep

grep, egrep, fgrep

6.2.6 Swift

Apple Swift

6.2.7 Java

6.2.8 .NET Framework 正则表达式

.NET Framework 正则表达式

[①]PCRE, Perl Compatible Regular Expressions

6.2.9 JavaScript

6.2.10 Adobe Dreamweaver 表格处理

6.2.11 VBA 中使用正则表达式

VBA[①]是不直接支持正则表达式的,需要借助 VBScript RegExp Object,具体请参考Microsoft Beefs Up VBScript with Regular Expressions。

```
Sub IndentParaWithRegEx()
' PowerPoint VBA 批量给指定字符开头段落加动画
Dim oSld As Slide
Dim oShp As Shape
Dim i As Integer
' 正则相关变量
Dim regx As Object, oMatch As Object
strPattern = "^开头字符串"

Set regx = CreateObject("vbscript.regexp")
With regx
    .Global = True
    .IgnoreCase = True
    .Pattern = strPattern
End With

For Each oSld In ActivePresentation.Slides
    For Each oShp In oSld.Shapes
        If oShp.HasTextFrame Then
            If oShp.TextFrame2.HasText Then
                With oShp.TextFrame2.TextRange
```

[①]Visual Basic for Application, Microsoft Office 套件宏语言。

```
                    For i = 1 To .Paragraphs.Count
                        With .Paragraphs(i)
                            ' 可能会出现多个匹配项的
                            If (regx.Test(.Text) = True) Then
                                .ParagraphFormat.FirstLineIndent = 0
                            End If
                        End With
                    Next i 'para
                End With
            End If 'has text
        End If 'has textframe
    Next oShp
Next oSld
End Sub
```

6.2.12 GREP for Adobe InDesign

6.2.13 神的编辑器 GNU Emacs 之正则神器

7 后记

后记