# Emacs 实践笔记

aborn

2018-08-01 09:24

## Contents

# 1 基本数据类型

## 1.1 Lisp 的数据类型

Lisp 的对象至少属于一种数据类型。Emacs 里最基础的数据类型称之为原始类型 ( primitive type ) , 这些原始类型包括整型、浮点、cons、符号 (symbol)、字符串、数组、哈希表 ( hash-table )、subr、二进制编码函数 (byte-code function)，再加上一些特殊的类型，如 buffer。同时，每种原始类型都有一个对应的函数去校验对象是否属于其类型。

## 1.2 符号类型 (Symbols)

符号类型是一种有唯一标识的命名对象。它常用于变量及函数名。判断一个对象是否为符号类型用 **symbolp object** 方法。

### 1.2.1 符号类型的组成

每个符号类型由四部分组成，每部分称之为单元，每个单元指向其他对象。

1. 名字 即符号标识，获取符号标识名的函数为 (symbol-name symbol)
2. 变量值 当标识对象作为变量时的值
3. 函数 标识函数定义，函数单元可保存另一个标识对象、或者 keymap、或者一个键盘宏
4. 属性列表 标识对象的属性列表 (plist)，获取属性列表函数为 (symbol-plist symbol) 注意，其中 **名字**为字符串类型，不可改变，其他三个组成部分可被赋值为任意 lisp 对象。其值为属性列表 (plist)。一个以冒号开头的符号类型称之为 keyword symbol，它常用于常量类型。

### 1.2.2 定义符号类型

定义符号类型对象是一种特殊的 lisp 表达式，它表示将标识类型用于特殊用途。

1. defvar 和 devconst 它们是一种特殊表达式 ( Special Forms )，它定义一个标识作为全局变量。实际应用中往往使用 *setq*，它可以将任意变量值绑定到标识对象。
2. defun 用于定义函数，它的作用是创建一个 lambda 表达式，并将其存储在标识对象的函数单元里。
3. defmacro 定义标识符为宏，创建一个宏对象并前对象保存在函数单元里。

### 1.2.3 符号类型操作函数

常见的与标识类型相关的函数有 **make-symbol** 和 **intern**

1. make-symbol

   ```
   make-symbol name
   ```

   这个函数返回一个新的标识对象，它的名字是 **name** (必须为字符串)

2. intern

   ```
   intern name &optional obarray
   ```

   这个函数返回一个被绑定的名字为 **name** 的标识对象。如果标识符不在变量 **obarray** 对应的对象数组 (obarray ) 里，创建一个新的，并加入到对象对象数组里。当无 obarray 参数时，采用全局的对象数组 obarray。

### 1.2.4 标识符属性

标识符属性记录了标识符的额外信息，下面的函数是对标签符属性进行操作：

1. get symbol property 获取标识符属性为 property 的属性值，属性不存在返回 nil

2. put symbol property value 设置标识符属性 property 的值为 value, 如果之前存在相同的属性名，其值将被覆盖。这个函数返回 value。下面是一些例子：

   ```
   (put 'fly 'verb 'transitive)              ;;  'transitive
   (put 'fly 'noun '(a buzzing little bug)) ;;  (a buzzing little bug)
   (get 'fly 'verb)       ;;   transitive
   (symbol-plist 'fly)   ;;  (verb transitive noun (a buzzing little bug))
   ```

3. 标准标识符属性 下面列的一些标准标识符属性用于 emacs 的特殊用途

   (a) :advertised-binding 用于函数的 key 的绑定

   (b) interactive-form 用于交互函数，不要手工设置它，通过 **interactive** 特殊表达式来设置它

   (c) disabled 如果不为 nil，对应的函数不能作为命令

   (d) theme-face 用于主题设置

## 1.3 列表

列表是由零个或者多个元素组成的序列，列表中的每个元素都可由任意的对象组成。

### 1.3.1 关联列表 alist (Association Lists)

关联列表是一种特殊的列表，它的每个元素都是一个点对构成，如下示例:

```
(setq alist-of-colors
   '((rose . red) (lily . white) (buttercup . yellow)))
```

关联列表可以用来记录 key-value 这样的 map 结构；对每个元素做 car 操作拿到 key，做 cdr 操作即拿到相关系的 value。

1. 关联列表操作

   - (assoc key alist) 获取列表第一个 key 所关联的值；下面是一个例子：

   ```
   ELISP> (assoc 'rose alist-of-colors)
   (rose . red)
   ```

   注意：这里用得比较是 equal 函数，如想用 eq 函数，请采用 (assq key alist) 这个函数

   - (rassoc value alist) 获取列表第一个 value 为 **value** 所关联的值；
   - (assoc-default key alist) 获取列表中第一个 key 为 **key** 的 value；

   ```
   ELISP> (assoc-default 'rose alist-of-colors)
   red
   ```

### 1.3.2 属性列表 plist (Property Lists)

属性列表是由成对元素 ( paired elements ) 组成的列表，每个元素对关联着一个属性的名及其对应属性值。下面是一个例子：

```
(pine cones numbers (1 2 3) color "blue")
```

这里 pine 关联其值为 cons，numbers 关联其值为 (1 2 3)，一般每个元素对的关联值是由 symbol 类型组成的。

1. 属性列表的操作

- (plist-get plist property) 返回属性列表中属性名为 property 的属性值:

```
ELISP> (setq pl '(pine cones numbers (1 2 3) color "blue"))
(pine cones numbers
      (1 2 3)
      color "blue")
ELISP> (plist-get pl 'pine)
cones
ELISP> (plist-get pl 'numbers)
(1 2 3)
```

- (plist-member plist property) 如果属性列表 plist 中含有属性 property，则返回 non-nil。
- (plist-put plist property value) 保存属性 property 及值 value 的属性对

```
(setq my-plist '(bar t foo 4))              ;; => (bar t foo 4)
(setq my-plist (plist-put my-plist 'foo 69))    ;; => (bar t foo 69)
(setq my-plist (plist-put my-plist 'quux '(a)))  ;; => (bar t foo 69 quux (a))
```

### 1.3.3 对列表进行排序

对列表进行排序可以采用 sort 这个函数 **(sort list predicate)** 。不过这个函数是有副作用的，这个函数调用后会改变原有 list 的结构。第三个参数 predicate 传入的是一个比较函数，它接收两个参数。如果是想递增排序，当第一个参数小于第二个参数时返回 non-nil，否则返回 nil。注意这个 sort 函数对 list 的排序，始终保持 car 部分不变。下面是一个例子：

```
ELISP> (setq nums '(1 3 2 6 5 4 0))
(1 3 2 6 5 4 0)
ELISP> (sort nums '<)
(0 1 2 3 4 5 6)
ELISP> nums
(1 2 3 4 5 6)
```

注意这里的 nums 排序后，的 car 与原来 list 的 car 是一样的。所以一般采用重新赋值的方式 **(setq nums (sort nums '<))**

# 2 求值

## 2.1 求值

Lisp 解释器会对表达式进行求值操作, 也可以手工调用求值方法 **eval** 。Lisp 解释器通常先读取 Lisp 表达式，然后对表达式进行求值。其实，读取和求值是两个相互独立的过程，它们也可以进行单独操作。

## 2.2 表达式类型

表达式是一种用于求值的 lisp 对象, Emacs 有三种不同的求值表达式类型：标识符 (Symbols)、列表和其他类型。下面从其他类型开始介绍。

### 2.2.1 自解释表达式

自解释类型，其意思很明确是自己对自己求值，例如 25 自解释成 25，自符串"foo" 自解释成"foo"。

### 2.2.2 标识符类型

当标识符类型被求值，它将被当成变量使用，求值的结果就是变量的值。如果变量没有值，Lisp 解释器会抛出一个错误提示。

```
(setq a 123)   ;;  123
(eval 'a)      ;;  123
a              ;;  123
```

### 2.2.3 自动载入 (Autoloading)

自动载入的特性允许函数或者宏还没有载入到 Emacs 中前使用它们。

## 2.3 引用

引用 (quote) 是一种特殊表达式，它返回它的参数且不对其进行求值。它提供了一种在程序里包含标识符常量和列表却不需要对其求值的使用方式。

```
(quote object)
```

它返回 object，但不对 object 进行求值操作。它提供了一种简写方式，即“”, **'object** 。

### 2.4 反引号

反引号 (backquote ‘) 可用于列表，它与引用唯一的区别的，它允许对列表中部分元素进行求值。采用逗号 (,) 来标识那些元素需要进行求值，下面是一些例子：

```
`(a list of ,(+ 2 3) elements)  ;;  (a list of 5 elements)
`(1 2 (3 ,(+ 4 5)))             ;;  (1 2 (3 9))
```

## 3 控制结构

### 3.1 控制结构

Lisp 程序由一系列表达式结成，Lisp 解释器解释并执行这些表达式。在执行这些表达式过程中用到了控制结构，Lisp 里的控制结构都是特殊表达式 ( Special Forms )。最简单的控制结构是顺序执行，也是符合人的书写和线性习惯。其他控制结构有：条件语句、迭代。

### 3.2 顺序结构

顺序结构是最简单的控件结构，如果想自己定义顺序结构，可以采用 **progn** 这个特殊表达式:

```
(progn a b c ...)
```

它的执行结构是最后一句的结果。与之类似有另一个特殊表达式 **(prog1 form1 forms...)** 它也是顺序执行，不过它的返回值是 form1 的返回值。同时还有一个特殊表达式 **(prog2 form1 form2 forms...)** 它效果也是一样，不过它返回的是 form2 的值。

### 3.3 条件语句

ELisp 提供四种条件语句：if、when、unless 和 cond

#### 3.3.1 if

if 语句跟其他语言的 if 语言类似，它的结构如下：

```
(if condition then-form else-forms...)
```

这里有一点要引起注意的是当 condition 为 nil, 并且没有给定 else-forms 时，if 返回的是 nil。

### 3.3.2 when

when 是 if 的变体，是当没有 else-forms 的特殊情况：

```
(when condition then-forms. . .)
```

### 3.3.3 unless

unless 也是 if 的一个变体，是当没有 then-form 的特殊情况：

```
(unless condition forms...)
```

### 3.3.4 cond

cond 是一种选择条件语句，每一个条件语句必须是一个列表，其中列表的头 (car clause) 是条件，列表的其他部分是执行语句。cond 的执行过程是按顺序执行，对每个条件语句 clause，先对条件部分进行求值，如果条件的执行结果不是 nil，说明条件满足，则接下来执行条件语句的主体部分，最后返回主体部分的执行结果作为 cond 的结果，其他部分的条件语句则被忽略。

```
(cond ((numberp x) x)
      ((stringp x) x)
      ((bufferp x)
       (setq temporary-hack x) ; multiple body-forms
       (buffer-name x))        ; in one clause
      ((symbolp x) (symbol-value x)))
```

有时候当前面所有的条件语句都没有“命中”时，可以采用 t 进行默认处理，下面是一个例子：

```
(setq a 5)
(cond ((eq a 'hack) 'foo)
      (t "default"))           ;;   "default"
```

## 3.4 迭代语句

迭代在程序语言里表示重复执行某段代码，举例来说，如果你想对 list 的每个元素重复执行相同的计算，这就是一个迭代过程。

### 3.4.1 while

while 的定义如下：

```
(while condition forms...)
```

while 首先对 condition 进行求值操作，如果结果不是 nil，则执行 forms 里语句；接下来再次对 condition 进行求值，如果不是 nil，则执行 forms 里的语句；这个过程不断重复直到 condition 的求值为 nil。

```
(setq num 0) ;;  0
(while (< num 4)
  (princ (format "Iteration %d." num))
  (setq num (1+ num)))
```

### 3.4.2 dolist

dolist 的定义如下:

```
dolist (var list [result]) body...
```

dolist 对 list 里的每个元素执行 body 里的语操作，这里绑定 list 里的每个元素到 var 作为局部变量。最后返回 result，当 result 省略时，返回 nil。下面是一个例子：

```
(defun reverse (list)
   (let (value)
    (dolist (elt list value)
      (setq value (cons elt value)))))
```

### 3.4.3 dotimes

dotimes 的定义如下：

```
dotimes (var count [result]) body...
```

它的作用与 dolist 很类似，它从 0(包含) 到 count(不包含) 执行 body 语句，将当前的值绑定到 var，返回 result 作为结果。下面是一个例子：

```
(dotimes (i 100)
 (insert "I will not obey absurd orders\n"))
```

# 4 变量

## 4.1 变量

变量在程序中是一种标识符，其指向某个值，这是一个很通用的编辑概念，不需要做过多解析。在 Lisp 中，每个变量都通过符号类型来表达。变量名就是对应的符号类型名，变量值是保存在符号类型的值单元 (value cell) 里。注意在前一章节里，我们介绍到符号类型既可用于变量名也可用于函数名，它们是相互独立不冲突的。

## 4.2 全局变量

全局变量在任意时刻都只有一个值，并且这个变量可使用于整个 lisp 运行环境。我们经常用 **setq** 这个特殊表达式将一个值绑定到具体某个符号变量中，如下：

```
(setq x '(a b))
```

这里 setq 是一个特殊表达式，所以它不会对第一个参数 x 进行求值，它会对第二个参数进行求值，然后将求得的值绑定到第一个参数对应的变量中。

### 4.2.1 不变量

在 Emacs Lisp 中，某些符号类型的求值是其本身。最常见的如 nil、t，以及以: 开头的符号类型（这些符号类型称之为关键字 keywords）。这些特殊的变量不能再进行绑定，同时其值也无法进行修改。

```
(keywordp object)
```

用来判断一个对象是否为关键字类型 (keywords)，即以: 开头的符号类型。

### 4.2.2 定义全局变量

变量的定义主要有三个目的: 首先，它提示阅读代码人的，该符号变量用于一种特殊用途（这里用于变量）; 其次，它提供给 Lisp 系统，有时候还会赋于初始值和文档；最后，它为编程工具提供信息，如 etags，提示它去哪里找到变量定义。定义全局变量采用 **defvar** 关键字，它定义一个符号类型为变量。

```
defvar symbol [value [doc-string]]
```

还有一种方式，采用 **defconst** 关键字

```
defconst symbol value [doc-string]
```

## 4.3 变量是无效的

当一个符号类型对应的值单元没有被赋值 (unassigned) 时，称对应的变量为无效的 (void)。对于个无效变量进行求值，会抛出 **void-variable error**。注意变量为无效的 (void) 与变量值为 nil 本质上是不一样的，nil 为一种对象，它可以赋值给变量。

```
(makunbound symbol)
```

makunbound 清空符号类型里值单元，使得一个变量成为无效的 (void)。它返回符号类型。

```
(boundp variable)
```

boundp 当 variable 不是无效的 (nil) 时返回 t，否则返回 nil。

### 4.4 局部变量

局部变量一般只用于一段程序，最常用的声明方式是采用 **let** 关键字。它的定义格式如下：

```
let (bindings. . . ) forms. . .
```

这里的 let 是一个特殊表达式 (Special Forms)，它按顺序绑定局部变量。下面是一个例子:

```
(let ((y 1)
      (z y))
   (list y z))  ;;   (1 2)
```

还有一种局部变量，即函数的调用参数，因为这些参数只用于函数调用阶段。

### 4.5 Buffer 本地变量 (Buffer-Local Variables)

Buffer 本地变量，从字面意思可以看出，这种类型的变量只应用于 Buffer 中。这种机制可以满足对于同一个变量在不同的 Buffer 中的值不一样。

### 4.6 文件本地变量 (File-Local Variables)

## 5 函数

### 5.1 什么是函数？

函数是有传入参数的可计算的单元。每个函数的计算结果为函数返回值。大部分计算机语言里，每个函数有其自己函数名。从严格意义来说，lisp 函数是没有名字的。lisp 函数其本质是一个对象，该对象可关联到一个标识符 ( 本书把 Symbol 翻译成标识符 ) , 这个标识符就是函数名。

### 5.2 定义函数

定义一个函数的语法如下：

```
defun name args [doc] [declare] [interactive] body. . .
```

### 5.2.1 检查一个函数是否定义

检查一个变量是否绑定到函数，**fboundp symbol** , 还有一个函数 **(functionp OBJECT)**

```
(fboundp 'info)                          ; t
(fboundp 'setq)                          ; t
(fboundp 'xyz)                           ; nil
(functionp (lambda () (message "Anonymous Functions")))   ; t
(fboundp (lambda () (message "Anonymous Functions")))    ; *** Eval error ***
```

### 5.2.2 函数参数

有些参数是可选的，当用户没有传时，设置一个默认值，下面是一个例子：

```
(defun cookbook/fun-option-parameter (a &optional b &rest e)
  (when (null b)
    (message "paramete b is not provided")
    (setq b "ddd"))    ;; set to default value
  (message "a=%s, b=%s" a b))
```

函数 cookbook/fun-option-parameter 中，a 为必传参数，b 为可选择参数，e 为其余参数，当实际传入的参数大于 2 时，其他参数将组成一个 list 绑定到 e 上。

## 5.3 函数调用

最通用的函数的调用方式是对 list 进行求值，如对 list **(concat "a" "b")** 进行求值，相当于用参数"a" 和"b" 调用函数 concat。这种方式用在你清楚程序上下文中调用哪个函数、传递哪个参数。但有时候你需要在程序运行时才决定调用哪个函数。针对这种情况，Emacs Lisp 提供了另外两种方式 **funcall** 和 **apply** 。其中 apply 一般用在运行时行决定传递多少个参数的情况。

### 5.3.1 funcall

funcall 它的语法如下：

```
funcall function &rest arguments
```

这里 funcall 本身是一个函数，因此 funcall 在调用前，它的所有参数都将事先做求值运算，对 funcall 来说它不知道具体的求值过程。同时请注意第一个参数 **function** 必须为一个 Lisp 函数或者原生函数，不能为特殊表达式

(Special Forms) 和宏，但可以为匿名函数 ( lambda 表达式 )。下面为一个例子：

```
(setq f 'list)        ;;  list
(funcall f 'x 'y 'z)  ;;  (x y z)
```

### 5.3.2 apply

apply 的定义如下：

```
apply function &rest arguments
```

apply 与 funcall 作用一样，唯独有一点不一样：它的 arguments 是一个对象列表，每个对象作为单独的参数传入，如下例子:

```
(setq f 'list)         ;;  list
(apply f 'x 'y 'z)     ;; Wrong type argument: listp, z
(apply '+ 1 2 '(3 4))  ;;  10
```

### 5.3.3 映射函数 (Mapping Functions)

映射函数操作是指对一个列表或者集合逐个执行指定函数，这节介绍几个常的映射函数：mapcar, mapc, 和 mapconcat。

```
mapcar function sequence
```

这个函数功能有与 javascript 里的 array.map 操作类型，对 **sequence** 里的每个元素执行 function 操作，返回操作结果列表。这个函数应用非常广泛，以下几个应用举例：

```
(mapcar 'car '((a b) (c d) (e f)))   ;;  (a c e)
(mapcar '1+ [1 2 3])                 ;;  (2 3 4)
(mapcar 'string "abc")               ;;  ("a" "b" "c")
```

**mapc** 与 **mapcar** 调用方式一样，唯一不同的点是它始终返回的是 **sequence** 。

```
mapconcat function sequence separator
```

**mapconcat** 对 **sequence** 里的每个元素调用 **function** 最后将结果拼接成一个字符串作为返回值，采用 separator 作为拼接符。

## 5.4 匿名函数

在 elisp 里有三种方式可以定义匿名函数：**lambda** 宏、**function** 特殊表达式、#' 可读语法。

### 5.4.1 lambda 宏

它的定义如下：

```
lambda args [doc] [interactive] body. . .
```

这个宏返回一个匿名函数，实际上这个宏是自引用 (self-quoting)。

```
(lambda (x) (* x x))  ;;  (lambda (x) (* x x))
```

下面是另一个例子：

```
(lambda (x)
    "Return the hyperbolic cosine of X."
    (* 0.5 (+ (exp x) (exp (- x)))))
```

上面的表达式被计算成一个函数对象。

### 5.4.2 function 特殊表达式

定义如下：

```
function function-object
```

这是一个特殊表达式 (Special Forms)，表示对 **function-object** 不作求值操作。其实在实际使用中我们往往采用它的简写 #'，因此下面三个是等价的：

```
(lambda (x) (* x x))
(function (lambda (x) (* x x)))
#'(lambda (x) (* x x))
```

## 5.5 获取函数单元内容

当我们把一个标识符 (Symbol) 定义为函数，其本质是将函数对象存储在标签符号对应的函数单元 ( 标识符还有一个变量单元用于存储变量 ) , 下面是介绍函数单元处理方法:

### 5.5.1 symbol-function

定义如下：

```
symbol-function symbol
```

这个函数返回标识符 symbol 对应的函数对象，它不校验返回的函数是否为合法的函数。如果 symbol 的函数单元为空，返回 nil。

### 5.5.2 fboundp

用于判断 symbol 对应的函数单元是否为 nil

```
fboundp symbol
```

当 symbol 在函数单元有一个对象时返回 t，否则返回 nil。

## 5.6 特殊表达式 (Special Forms) 和宏

有些与函数看起来很像的类型，它们也接受参数，同时计算出结果。但在 Elisp 里，他们不被当成函数，下面给出简单介绍：

### 5.6.1 内建函数 (primitive)

是用 C 语言写的，可被调用的函数；

### 5.6.2 special form

一种类型的内建函数，如 if, and 和 while

# 6 Buffer

## 6.1 Buffer

## 6.2 创建 Buffer

有两个函数可用于创建 Buffer，它们是 get-buffer-create 和 generate-new-buffer

# 7 文件

## 7.1 文件及访问

文件是操作系统永久保存数据的单元，为了编辑文件，我们必要告诉 Emacs 去读取一个文件，并将文件的内容保存在一个 Buffer 里，这样 Buffer 与文件就关联在一起。下面介绍与文件访问相关的函数，由于历史原因这些函数的命令都是以 **find-** 开头的，不是以 **visit-** 开头。

### 7.1.1 打开文件

如果想在 buffer 里打开一个文件, 其命令是 **find-file** (C-x C-f)。当文件已经在 buffer 中存在时，这个命令返回文件对应的 buffer。如果当前没有 buffer 对应文件，则，创建一个 buffer，并将其文件内容读到 buffer 中，并返回这个 buffer。字义如下:

```
(find-file filename &optional wildcards)
```

这个函数有一个对应的 hook 变量，叫 **find-file-hook** 它的值是一个函数列表。这些函数在文件被打开后依次执行。

### 7.1.2 文件保存

文件被载入到 buffer 后，我们可以对其进行修改；修改完了后，将内容保存回文件，其对应的函数为：

```
(save-buffer &optional backup-option)
```

文件保存对应有两个 hook 变量，为：**before-save-hook** 和 **after-save-hook** 分别表示保存前的 hook 函数列表和保存后的 hook 函数列表。与之类似的还有一个函数 **write-file**

```
(write-file  filename &optional confirm)
```

这个函数的功能是将当前 buffer 的内容写入到 filename 对应的文件中，并将当前 buffer 与这个文件进行关联

### 7.1.3 读取文件内容 (Reading from Files)

将文件内容复制到 buffer，可以使用 **insert-file-contents** 函数，注意在 Lisp 代码里不要使用 **insert-file** 命令，因为它会设置 mark 标识。

```
(insert-file-contents filename &optional visit beg end replace)
```

这个函数在当前 buffer 的位置插入文件 **filename** 的内容，它返回一个它包含一个文件名和数据长度信息的列表。如果文件不存在 (或不可读)，则会抛出错误异常! 当这个函数执行后会调用 **after-insert-file-functions** 列表里的函数。一般情况下，在这个列表里的函数其中有一个是用来检测文件内容的编码。与这个函数类似的一个函数为 (insert-file-contents-literally filename &optional visit beg end [Function] replace)，它们唯一的区别的后者不内容进行格式化、不对字符做转换。如果参数 visit 不是 nil 时，执行这个函数后会将当前 buffer 设置为未修改 ( unmodified ) 状态。

### 7.1.4 往文件里写内容 (Writing to Files)

将 buffer 里的内容 ( 或者部分内容 ) 直接写入到一个文件，可以采用 **append-to-file** 和 **write-region** 函数。注意这里不要写入正在访问的文件，否则会出现异常情况：

```
(append-to-file start end filename)
```

这个函数的作用是将当前 buffer 里的部分内容 ( 从 start 到 end 部分内容 ) 追加到文件 **filename** 的后面。如果是在 lisp 中使用，这个函数完全等价于 (write-region start end filename t)。

```
(write-region start end filename &optional append visit lockname mustbenew)
```

这个函数的作用与 append-to-file 类似，不过其参数更多。

1. 当 start 为 nil 时，这个函数写入的是当前 buffer 所有内容，这时 end 参数没有用;
2. 当 start 为 string 时，这个函数写入的内容是 string 的内容，这时 end 参数失效;
3. 当 append 不是 nil 时，表示往现有文件里进行追加，当 append 是一个数字时，表示从当前文件开始到 append 的位置开始写入。
4. 当 mustbenew 不为 nil 时，当覆盖已有文件时，会询问用户，并获得用户确定后再操作。

```
(with-temp-file file body)
```

**with-temp-file** 是一个宏操作，它将创建一个临时 buffer 作为当前 buffer，在这个 buffer 里对 body 进行求值，最后将这个 buffer 的内容写入到文件 **file** 里。当整个 body 执行完成后，Emacs 将会把这个临时 buffer 关闭，恢复到执行 with-temp-file 之前的当前 buffer。它将 body 的最后执行结果作为 with-temp-file 的返回结果。

## 7.2 文件锁

当多个用户同时修改一个文件里，这时候需要文件锁。Emacs 里的文件锁是保存在同一目录下的一个文件，它有一个特殊的名字.

```
(file-locked-p filename)
```

**file-locked-p** 这个函数用来检查文件是否被锁。当文件没有被锁，则返回 nil ; 如果被 Emacs 进程锁了，则返回 t，当被其他 job 锁了，则返回使用都信息。

```
(lock-buffer &optional filename)
```

如果当前 buffer 被修改过，这个函数锁定当前 buffer 所关联的文件。与之相对应的操作有解锁，可以使用 (unlock-buffer) 这个函数。

```
(ask-user-about-lock file other-user)
```

当一个用户修改正在被另一个用户锁定的文件时，询问用户。该函数的返回值 ( 即用户的选择 )，决定 Emacs 接下来该如何执行。

## 7.3 文件基本信息函数

下面介绍一些与文件基本信息相关的函数

### 7.3.1 文件是否存在

判断一个文件是否存在采用 **file-exists-p** 这个函数:

```
(file-exists-p filename)
```

与之类似的有: **file-readable-p** 、**file-executable-p** 、**file-writable-p** 、**file-directory-p file-symlink-p** 这几个函数。

### 7.3.2 文件新旧比较

```
(file-newer-than-file-p filename1 filename2)
```

当 filename1 比 filename2 新时，该函数返回 t。如果 filename1 不存在，则返回 nil。如果 filename1 存在，但 filename2 不存在，则返回 t。

### 7.3.3 文件模式

```
(file-modes filename)
```

这个函数返回文件的属性，跟 linux 里的 chmod 命令相对应，它返回的是一个整数：它包含了文件的读、写和可执行权限。

```
(file-modes "~/junk/diffs")   ;;  492 ; Decimal integer.
```

### 7.3.4 文件属性

这小节介绍与文件属性有关的一些函数，如文件的所属人、所属组、文件大小、文件的最新读取和修改时间等。

```
(file-attributes filename &optional id-format)
```

这个函数返回文件对应的属性列表，下面是一个调用示例：

```
(file-attributes "~/tree.txt")
;; 返回如下
(nil 1 501 20
     (23331 5030 438781 943000)
     (23331 4821 822935 764000)
     (23331 4821 822935 764000)
     10496 "-rw-r--r--" t 8602715307 16777220)
```

属性列表按顺序说明如下：

1. t 表示目录，字符串表示符号链接，nil 为文本文件；
2. 这个文件有多少名字与之关联，一般为 1，当有符号链接时不一样；
3. 文件的 UID;
4. 文件的 GID;
5. 文件最近 accessTime，有 4 个元素的列表 (sec-high sec-low microsec picosec);
6. 文件最后修改时间；
7. 文件状态最后被修改时间；主要是用 chmod 来改变文件模式;
8. 文件大小，单位 byte;
9. 文件模式；
10. 未使用值，主要用来做向下兼容；
11. 文件的 inode 编码；
12. 设备的文件系统码；

## 7.4 文件操作

### 7.4.1 文件复制和重命名

文件重命名函数为 **rename-file**

```
(rename-file filename newname &optional ok-if-already-exists)
```

复制文件函数为 **copy-file**

```
(copy-file oldname newname &optional ok-if-exists time
 preserve-uid-gid preserve-extended-attributes)
```

这个函数的作用是复制老的文件 **oldname** 到新的文件 **newname**, 这里有一点要注意的是如果新的文件名 **newname** 为目录，则复制老的文件到这个目录 ( 文件名保持为 **oldname** 不变 )。当参数 time 不为 nil 时，则新文件保持与老文件相同的最后修改时间属性信息。

### 7.4.2 文件删除

文件删除的函数为 **delete-file**

```
(delete-file filename &optional trash)
```

这里有一点要注意的是如果文件 filename 为符号链接，这个函数只删除符号链接，不删除原目标文件。

### 7.4.3 设置文件属性

设置文件属性函数为 **set-file-modes**

```
(set-file-modes filename mode)
```

这里的 mode 必须为整数，下面是一个例子：

```
(set-file-modes "a.txt" #o644)
```

如果想获取默认的文件权限属性，可使用 (default-file-modes) 来获取，它返回的是一个整数。

## 7.5 文件查找

### 7.5.1 locate-file

```
(locate-file filename path &optional suffixes predicate)
```

**locate-file** 这个函数用来查找在 path 目录下文件名为 filename 的文件，如果找到则返回绝对文件名。注意第二个参数 path 必需为目录列表，像 **exec-path** 对应的列表一样。下面是一个例子：

```
(locate-file "03_file.org" '("/Users/aborn/github/emacs-cookbook/chapters/"))
;; "/Users/aborn/github/emacs-cookbook/chapters/03_file.org"
(locate-file "03_file" '("/Users/aborn/github/emacs-cookbook/chapters/") '(".tex" ".org
;; "/Users/aborn/github/emacs-cookbook/chapters/03_file.tex"
```

可选参数 suffixes 为后缀列表，查找所有后缀，以第一个查到为准。注意，这里的文件查找只会查找 path 目录，不会查找其子目录。

### 7.5.2 executable-find

```
(executable-find program)
```

executable-find 用于查找可执行文件，查找所有 **exec-path** 目录下的可执行文件（以及查找所有后缀为 exec-suffixes 列表里的可执行文件），下面是一个例子:

```
(executable-find "emacs")
;; "/usr/local/bin/emacs"
```

## 7.6 文件与目录

判断文件是否在一个目录下，怎么做?

```
(file-in-directory-p  file dir)
```

如果 file 是一个在目录 dir 或者 dir 子目录下的文件，则返回 t。如果 file 与 dir 处于同一目录，也返回 t。如果想列出一个目录下的所有文件，那就要用到 **directory-files** 这个函数，其定义如下：

```
(directory-files directory &optional full-name match-regexp nosort)
```

这个函数按字母顺序返回目录 directory 下的所有文件。参数 full-name 不为 nil 时，则返回每个文件的绝对路径，否则返回相对路径。match-regexp 如果不是 nil，该函数返回只与 match-regexp 相匹配的文件列表。nosort 如果不为 nil，则不按字母排序。

### 7.6.1 创建、复制和删除目录

对目录的创建、复制和删除都有相关的处理函数，下面一一介绍：

```
(make-directory dirname &optional parents)
```

**make-directory** 创建一个目录名为 dirname 的目录

## 7.7 文件名与文件路径

我们知道大部分操作系统，文件名由两部分组成：文件名和路径，任何一个文件都在某个具体路径下。下面介绍一些与之相关的函数操作。

### 7.7.1 文件名、路径、文件后缀

```
(file-name-directory filename)
```

**file-name-directory** 返回的文件名里的目录部分，如果文件名里没有包含目录部分，则返回 nil。与这个函数对应的一个函数为 **file-name-nondirectory**，它返回非目录部分。如果想获取文件的后缀，采用如下函数：

```
(file-name-extension filename &optional period)
```

这里有一点要引起注意，如果一个文件名以点号 (.) 开始，如.emacs，file-name-extension 返回的后缀不是.emacs，而是 nil。

```
(file-name-sans-versions filename &optional keep-backup-version)
```

**file-name-sans-versions** 这个函数返回不包含任何版本、备份号等信息的”纯”文件名，下面是一些例子：

```
(file-name-sans-versions "~rms/foo.~1~")   ;;    "~rms/foo"
(file-name-sans-versions "~rms/foo~")      ;;    "~rms/foo"
(file-name-sans-versions "~rms/foo")       ;;    "~rms/foo"
```

### 7.7.2 文件路径

**expand-file-name** 这个函数将文件名转成绝对文件名：

```
(expand-file-name filename &optional directory)
```

如果 directory 参数存在，将 filename 作为其相对路径，否则使用 **default-directory** 变量。这个函数在写 elisp 代码时经常用到，下面是一些例子:

```
(expand-file-name "foo")
;;   "/xcssun/users/rms/lewis/foo"
(expand-file-name "../foo")
;;   "/xcssun/users/rms/foo"
(expand-file-name "foo" "/usr/spool/")   "/usr/spool/foo"
```

与 expand-file-name 相似的函数还有 **file-truename** 这个函数

```
(file-truename filename)
```

下面是一些例子

```
(file-truename "~/tree.txt")        ;;  "/Users/aborn/tree.txt"
(file-truename "../tree.txt")       ;; "/Users/tree.txt"
(file-truename "../../tree.txt")    ;; "/tree.txt"
```

特别说明：**default-directory** 是一个 Buffer 的本地变量 ( Buffer-Local Variable )，仅对当前 Buffer 有效，且它是一个绝对路径 (但可以以 ~ 开头)。有时候，有程序里路径是含有环境变量的 (Bash 里是以 $ 开头的)，如想将这些环境变量转成其相应的值，则可使用 **substitute-in-file-name** 这个函数:

```
(substitute-in-file-name filename)
```

如下例子：

```
(substitute-in-file-name "$HOME/bin")
;; "/Users/aborn/bin"
```

### 7.7.3 目录文件列表

目录是一种特殊的文件，它可以包含其他文件或文件夹。获取目录下所有文件列表 ( 像 ls 命令一样 )，可以使用 **directory-files** 这个函数：

```
(directory-files directory &optional full-name match-regexp nosort)
```

这个函数返回 directory 下所有文件列表 (包括目录)，默认是按字母顺序排列。参数 full-name 不是 nil 时，返回的是包含路径的绝对文件名，默认是返回相对文件名。参数 match-regexp 不是 nil 时，只返回与 match-regexp 正则表达式相匹配的文件名。参数 nosort 不为 nil 时，这个函数不对文件列表进行排序，可用于对文件顺序不关系的场景 ( 这时可最快获取返回结果 )。这个函数只返回当前目录下的所有文件，如果想递归获取所有的文件列表，可采用 **directory-files-recursively** 函数：

```
(directory-files-recursively directory regexp &optional include-directories)
```

这个函数递归的搜索在目录 directory 及其子目录下所有文件名与 regexp 相匹配的文件，返回文件的绝对路径列表。默认情况下返回的文件名是深度优先排序，也就是说子目录的文件名排序在其父目录之前，处于同一级目录的文件是按字母排序。当 include-directories 不为 nil 时，目录文件也包括其搜索结果中。

## 7.8 文件格式转换

Emacs 将文件从磁盘载入到 buffer 中，或者将 buffer 中内容写入到磁盘中，需要经过许多步骤。如 **insert-file-contents** 读取文件内容到 buffer，*write-region* 写入一个 buffer 到文件。

### 7.8.1 整体流程

对于 **insert-file-contents** 过程：

1. 初始化，从文件中插入字节到 buffer；
2. 解码，根据文件编码进行解码操作；
3. 按 **format-alist** 进行的格式化列表对其进行格式化；
4. 调用所有在 **after-insert-file-functions** 列表中的函数。

对于 **write-region** 过程：

1. 初始化，调用 **write-region-annotate-functions** 列表中的函数；
2. 按 **format-alist** 定义的格式化列表对其进行格式化处理；
3. 按适当编码格式对其进行编码成字节；
4. 用字节修改其文件。

### 7.8.2 格式转换

从以上整体流程我们可以看得出，其中格式化都使用 **format-alist** 进行处理，这个列表的每一项定义了一种格式转换，它的定义如下：

```
(name doc-string regexp from-fn to-fn modify mode-fn preserve)
```

下面介绍每个参数含义：

# 8 多线程

## 8.1 多线程

Emacs 从 26.1 版本开始引入了多线程 。它提供了一种简单 ( 但功能有限 ) 多线程操作。跟其他编程语言一样，在同一个 Emacs 实例里所有的线程的内存是共享的。每个线程有其自己运行 Buffer(Current Buffer) 和对应的数据 (Match Data)。注意：下面的文档都是参考 Emacs 的草案手册。

## 8.2 基本的线程相关函数

下面介绍线程操作相关的基本函数。

### 8.2.1 创建线程

我们可以通过 **make-thread** 函数来创建线程并执行对应的 task。它的语法如下:

```
(make-thread function &optional name)
```

创建一个名为 name 的线程，该线程执行 function 函数，当函数执行结束后，退出该线程。新线程的 Current Buffer 继承当前 Buffer，这个函数返回一个线程对象。可以通过 **(threadp object)** 来判断一个对象是否为线程对象。

### 8.2.2 thread-join

thread-join，它阻塞当前执行直到线程执行完成，如果线程已经退出，它立刻返回。

```
(thread-join thread)
```

### 8.2.3 thread-yield

执行下一个可执行的线程。

### 8.2.4 获取线程名

可以通过 (thread-name thread) 函数来获取线程名。

### 8.2.5 线程状态

判断一个线程是否还在执行 (alive)，可以用 (thread-alive-p thread)。

### 8.2.6 当前线程

(current-thread) 返回当前线程。

### 8.2.7 所有线程列表

获取当前所有正在运行中的线程 (all-threads)。

## 8.3 互斥锁 (Mutexes)

互斥 是一种排它锁 (exclusive lock)，在任何时刻，最多只允许一个线程持有互斥锁。也就是说，当一个线程试图获取一个已经被其他线程持有的互斥锁时，它会引发阻塞，直到该互斥锁被释放为止。

### 8.3.1 创建一个互斥锁

创建一个互斥锁对象，采用 **make-mutes** 函数，该函数返回一个互斥锁对象，其名字为 name。

```
(make-mutex &optional name)
```

判断一个对象是否为互斥锁使用 (mutexp object)。

### 8.3.2 获取/释放互斥锁

( `mutex-unlock mutex` )

这个操作会引发阻塞, 直到当前线程获取互斥锁为止。与之相对的有 (mutex-unlock mutex) 释放互斥锁操作。

### 8.3.3 with-mutex

```
(with-mutex mutex body)
```

这是一个宏操作，它首先获取一个互斥锁，然后执行 body 里的行为，最后释放互斥锁。

## 8.4 条件变量 (Condition Variables)

条件变量 提供线程阻塞直到某个事件发生的机制。线程可以等待一个条件变量，直到别的线程触发这个条件才唤醒。条件变量在某些情况下往往与互斥机制相关联。下面是一个例子：

```
(with-mutex mutex
  (while (not global-variable)
    (condition-wait cond-var)))
```

这里互斥锁保证了原子性。

```
(with-mutex mutex
  (setq global-variable (some-computation))
  (condition-notify cond-var))
```

### 8.4.1 创建条件变量

创建条件变量的函数如下：

```
(make-condition-variable mutex &optional name)
```

创建一个与互斥锁 mutex 的条件变量，其名字为 name。判断一个对象是否为条件变量使用 (condition-variable-p object)

### 8.4.2 条件等待

```
(condition-wait cond)
```

等待另一个线程去触发条件 **cond**（它是一个条件变量）。这个函数也会阻塞主流程直到条件被触发为止。condition-wait 在等待时会释放与之关联的互斥锁，允许其他线程去获取这个互斥锁从而触发条件变量。

### 8.4.3 条件通知

```
(condition-notify cond &optional all)
```

通知 **cond** 条件变量。一般情况下，一个等待线程被 condition-notify 被唤醒，当 all 不是 nil 时，所有等待 cond 的线程都将收到唤醒通知。

### 8.4.4 其他函数

1. (condition-name cond) 返回条件变量名
2. (condition-mutex cond) 返回与条件变量相关联的互斥锁

# 9 org 实践

## 9.1 org 模式简介

Emacs 的 org-mode 可用于记笔记、管理自己的待办事项 (TODO lists)，同时，也可用于管理项目。它是一个高效的纯文本编辑系统。

## 9.2 文档结构

Org 是基于 Outline-mode，并提供灵活的命令编辑结构化的文档。其文档结构语法跟 markdown 很类似。

### 9.2.1 目录结构

Org 的目录结构在每行最左边以星号标记，星号越多，标题层级越深。下面是一些例子：

```
\* 一级目录
\** 二级目录
\*** 三级目录
\* 另一个一级目录
```

### 9.2.2 显示与隐藏

目录结构下的内容可以隐藏起来，通常用采用 **TAB** 和 *S-TAB* 这两个命令来切换。

### 9.2.3 列表

Org 提供三种类型的列表：有序列表、无序列表和描述列表

1. 有序列表以'1.' 或者'1)'
2. 无序列表以'-', '+' 或者'*'
3. 描述列表

### 9.2.4 块结构

在 Org 文档中，加入代码块这种类型的块结构，都是采用 begin…end 这种模式，下面是一个例子：

```
\#+BEGIN_EXAMPLE
\#+END_EXAMPLE
```

## 9.3 表格

## 9.4 超链接

Org 模式提供了比较好用的超链接方式，可以链接到普通网页、文件、email 等。

### 9.4.1 链接格式

Org 模式支持两种链接，即，内部链接和外部链接。它们有相同的格式：

`[[链接][描述]]` 或 当只有链接没有描述 `[[链接]]`

一旦链接编辑完成，在 org 模式下，只显示 **描述**部分，而不会显示整体 ( 后一种是只显示链接 )。为了编辑链接和描述，需要通过快捷键 **C-c C-l** 来完成 (注意：编辑结束后按 Enter 完成修改操作)。

1. 内部链接 内部链接是指向当前文件的链接, 它的链接格式:

   `[[#链接ID]]`

   其中 **链接 ID** 是文档中唯一的标识 ID

2. 外部链接 Org 支持的外部链接有很多中形式，如文件、网页、新闻组、电子邮件信息、BBDB 数据条目等。它们以一个短的标识字符串打头，紧接着是一个冒号，冒号后面没有空格字符。

### 9.4.2 链接处理相关命令

Emacs org 提供了很多链接处理相关的函数

- org-store-link 保存的一个链接到当前位置，以备后面插入使用，原始绑定的快捷键为 **C-c l**
- org-insert-link 插入链接，绑定的快捷键为 C-c C-l，如果光标正在一个链接上，那么这个命令

的行为是编辑这个链接及其描述。

- org-open-at-point 打开当前位置的链接。它将在浏览器中打开这个链接，快捷键为 **C-c C-o**

其实使用是的 **browse-url-at-point**

## 9.5 待办事项

Org 模式用来管理自己的 TODO list 非常方便

## 9.6 日程表 (Agenda View)

我们可以用 Org 来按排自己的行程

### 9.6.1 日程文件 (Agenda files)

变量 org-agenda-files 保存了一个文件列表, 这些文件用来记录日程, 下面是一些操作函数: C-c [ 将当前文件加入到 agenda 文件列表最前页面 org-agenda-file-to-front C-c ] 将当前文件从 agenda 文件列表中删除 org-remove-file

### 9.6.2 分发按键

默认采用 **C-c a** , 接下的默认的命令有 :

- a 创建一个日程
- t/T 创建一个 TODO items
- L 对当前文件生成 timeline

### 9.6.3 内建 Agenda 视图

### 9.6.4 计划 Schedule

用 org 来安排日程

- org-schedule 将当前 TODO 添加计划时间

## 9.7 Org 快速记录

有时候, 突然想到一些待办事项, 或者一些突发的灵感。这时, 我们想用 emacs 快速记录它, Org-Capture 提供这个好用的功能。它的前身是 org-remember.el ( 注 : 从 org 8.0 开始, org-remember 被 org-capture ) 替代。

### 9.7.1 如何使用 org-capture?

快速记录的命令为 **M-x org-capture** , 默认绑定的快捷键为 C-c c 。当这个命令被调用后, 你可以使用自己定义好的 模板 快速创建记录。一旦完成内容的输入, 按下 C-c C-c (org-capture-finalize), 来完成。然后, 你就能继续做你当下的事。如果想跳转到刚刚创建的记录的 buffer, 用 C-u C-c C-c 来完成。如果想中途中止输入, 只要按下 C-c C-k (org-capture-kill)。

### 9.7.2 org 条目复制与移动

有时候，我们想将当前的某条目转移到其他文件或者其他项目里。这时，我们会用到 org-copy 和 org-refile 这两个命令。它们对应的快捷键分别是 C-c M-w 及 C-c C-w 。这里有一个问题是，目标文件如何配置？目录文件的配置由一个变量决定，**org-refile-targets** ，我自己的配置如下：

```
(setq org-refile-targets
      '((nil :maxlevel . 3)        ;; 当前文件的最大层级
(aborn-gtd-files :maxlevel . 3)))
```

注意：我这时将文件放在 aborn-gtd-files 文件列表里。

### 9.7.3 记录模板

记录的模板为一个列表变量，org-capture-templates，列表的每条记录由如下几段组成：

```
("t" "Todo" entry (file+headline (expand-file-name org-default-notes-file org-directory
         "* TODO %?\n  创建于:%T  %i\n")
```

1. 快捷键 如例子中的那样，”t” 表示对应按键 t 这个快捷键。它能帮助我们快速地选中哪条模板进行快速记录。
2. 描述 接下来是一段简单的描述
3. 类型 第三段表示类型，有五种类型：entry item checkitem table-line plain
   - entry 普通的 Org 结点，保证目标文件为 org-mode 文件，插入的时候将作为目录结点的子结点

   ( 如果没有，将做为顶级结点 )；
   - item 与 entry 类似，不同点在于它的目标文件可以为简单的纯文本文件；
   - checkitem 复选条目；
   - table-line 在目标文件中的第一个 table 中插入新行；
   - plain 纯文本记录
4. 目标文件 第四个字段配置目标文件
5. 模板 第五个字段表示模板，模板参数 含义如下：

- %t 只有日期的时间戳
- %T 日期 + 时间的时间戳
- %u,%U 如上，只不过它们是 inactive 的
- %i 初始化文本，当前上下文将作为初始化文本

6. 属性 properties 最后一个字段表示属性列表，支持以下属性配置：

   - :prepend 一般一个记录条目插入在目标文件的最后，这个属性可以将条目插入在最前
   - :immediate-finish 立刻完成，没有交互
   - :clock-in 对这个条目设置闹钟
   - :kill-buffer 如果目标文件没有相应的访问 buffer, 插入后，自动关闭 buffer

### 9.8 Org 的导出功能

Org 文件支持导出多种格式的目标文件，如 ASCII 文件、HTML 文件 (用于发布为 Web)、PDF 文档等。

#### 9.8.1 导出的 Dispatcher

任何导出命令都有一个前缀按键，我们称之为 Dispatcher，为 **C-c C-e**

### 9.9 org-capture.el

Org 8.0 以后版本采用 org-capture.el 取代原有的 org-remember.el

## 10 书签

### 10.1 emacs 的书签功能

emacs 的书签用于记录你在文件中的阅读位置。它有点类似寄存器，跟寄存器一样，因为它也能记录位置位置。但同寄存器有两点不一样：1. 它有比较长的名字; 2. 当 emacs 关闭的时候，它会自动持久化到磁盘。

#### 10.1.1 设置一个书签

当我们阅读一个很长的文档，没能一口气读完时。我们希望记住当前文档的最后阅读的位置，以便下次再用 emacs 阅读的时候能快速地定位到。那么，我们设置一个书签，通过 **bookmark-set** 对应快捷键为 **C-x r m**

### 10.1.2 列出保存的书签

**bookmark-bmenu-list** 对应快捷键为 **C-x r l** , 它将打开一个 *Bookmark List* 的 buffer 同时列出所有保存的书签。

1. 书签列表 *Bookmark List* 在 *Bookmark List* 这个 buffer 里，有以下快捷键可以使用：
   - a 显示当前书签的标注信息;
   - A 在另一个 buffer 中显示所有书签的所有标注信息;
   - d 标记书签，以便用来删除 (x – 执行删除);
   - e 编辑当前书签的标注信息;
   - m 标记书签，以便用于进一步显示和其他操作 (v – 访问这个书签);
   - o 选中当前书签，并显示在另一个 window 中;
   - C-o 在另一个 window 中切换到当前这个书签;
   - r 重命名当前书签;
   - w 将当前书签的位置显示在 minibuffer 里。

### 10.1.3 跳转到一个书签

使用 **bookmark-jump** 函数，可以跳转到一个特定的书签，它绑定的快捷键为 **C-x r b** 。如果你的 emacs 中安装了helm 这个插件，你也可以使用 **helm-bookmarks** 这个命令来快速查找书签，并跳转到书签位置。

1. helm-bookmarks 通过 helm-bookmarks 命令来查找并跳转书签如下图：

2. 修改默认排序 书签查找和跳转的时候，默认的书签排序是按字母排序的。如果想将最近访问的书签放在最前面，将下面代码添加到你的 emacs 配置文件中。

```
(defadvice bookmark-jump (after bookmark-jump activate)
  (let ((latest (bookmark-get-bookmark bookmark)))
    (setq bookmark-alist (delq latest bookmark-alist))
    (add-to-list 'bookmark-alist latest)))
```

### 10.1.4 删除一个书签

删除一个书签对应的命令为 **bookmark-delete** 。

### 10.1.5 保存书签

最新版本 emacs ( 老版本的书签保存在 ~**/.emacs.bmk** )，在退出的时候会自动保存书签。如果想手动保存书签的话，可以采用 **bookmark-save** 这个函数命令。默认的情况，emacs 会将书签保存在 **bookmark-default-file** 变量对应的文件中。在我的机器中，对应的文件如下：

```
ELISP> bookmark-default-file
"/Users/aborn/.emacs.d/.cache/bookmarks"
ELISP>
```

### 10.1.6 其他设置

有一个变量 **bookmark-save-flag** 。如果这个变量的值为一个数值，它表示修改 ( 或新增 ) 多少次书签后，emacs 会自动保存书签到磁盘。当这个变量的值被设置为 1 时，每次对 bookmark 的改动，emacs 就会自动保存内容到磁盘相应位置 ( 这样可以防止 emacs 突然 crash 时 bookmark 的丢失 )。如果这个值设置为 nil，表示 emacs 不会主动保存 bookmark，除非用户手动调用 **M-x bookmark-save** 。

### 10.1.7 bookmark+

bookmark+ 是对 bookmark 的一个扩展的包。它有更多的功能：

1. 原始的 bookmark 只能对文件位置记录,bookmark+ 对孤立的 buffer(没有关联文件的 buffer) 也能保存书签;
2. 支持对书签进行打 tag;
3. 对文档的某个区域保存为书签，而不仅仅是某个位置;
4. 记录了每个书签的访问次数，及最后一次的访问时间，可以基于它们排序;
5. 多个书签可以有相同的名字;
6. 可以对函数、变量等加书签。

更多功能请参考: `https://www.emacswiki.org/emacs/BookmarkPlus#Bookmark%2b`

# 11 dired 实践

## 11.1 dired 文件管理

dired 的全称为 Directory Edit, 即目录编辑, 是一个非常老的模式。是 Emacs 下的一个文件管理神器! 进入当前文件的 dired 文件管理，*M-x dired* 。

### 11.1.1 常用命令

1. 光标移动命令
    - **n** 下移
    - **p** 上移
2. 文件操作
    - **C** 拷贝文件，dired-recursive-copies 变量决定了拷贝的类型, 一般为 top
    - **D** 删除文件，类似的有一个 dired-recursive-deletes 变量可以控制递归删除
    - **R** 重命名或者移动文件
    - **D** 删除文件或者目录
    - **+** 创建目录
    - **Z** gzip 压缩文件
    - **w** 复制文件名 (C-u 则复制相对于 dired 当前目录的相对目录)
    - **A** 对文件进行正则表达式搜索，会在第一个匹配的地方停下，然后使用 M-, 搜索下一个匹配。
3. 其他命令
    - **RET** 打开文件或者目录
    - **g** 刷新当前 dired buffer
    - **k** 隐藏不想显示出来的文件
    - **q** 退出

### 11.1.2 标记与操作

dired 可以对多个文件进行标记，然后进行批量操作。一个典型的是采用 **d** 对当前文件打上删除标记，然后使用 **x** 命令来删除所有标记的文件.

1. 标记操作命令

   - m 以星标记当前文件
   - * * 标记所有可执行文件
   - * @ 标记所有符号链接
   - * / 标记所有目录 (不包括. 和.. )
   - * s 标记所有文件 (不包括. 和.. )
   - * . 标记具有给定扩展名的文件
   - % m REGEXP <RET> 或 * % REGEXP <RET> 标记所有匹配到给定的正则表达式的文件。
   - % g REGEXP <RET> 标记所有文件内容匹配到给定的正则表达式的文件。

2. 其他标记相关命令

   - u 去除当前行的标记
   - U 去除所有标记

#### 11.1.3 批量执行 Shell 命令

在 dired 模式下，可以对标记的文件批量执行 shell 命令 ( 如果没有标记文件，则对当前文件执行 shell )，运行命令 **dired-do-shell-command** (绑定的快捷键为 **!** )，相应的它有一个对应的异步操作的命令 **dired-do-async-shell-command** (绑定的快捷键为 **&** )。

#### 11.1.4 dired 的扩展

1. diredful diredful 可使得不同的文件显示不同的颜色, 是一个非常好的扩展

2. dired-icon dired-icon 根据文件类型显示相应 icon

# 12 magit 实践

## 12.1 magit 模式简介

magit 是 emacs 下版本管理的强大武器

### 12.2 常用命令

- **magit-dispatch-popup** 命令分发器，在 spacemacs 里绑定到 **M-m g m**
- **magit-diff** 相当于 git diff, 当进入 diff-buffer 后按 *g* 更新之
- **magit-status** 相当于 git status, 进入 status-buffer 后按 s 添加文件或文件夹到本地仓库
- **magit-checkout** 切换分支
- **magit-branch-and-checkout** 从当前分支切一个新的分支

### 12.3 分支操作

常用的分支操作如下：

- (magit-branch-delete) **b k** 删除一个或多个（本地）分支
- (magit-branch-rename) **b r** 对当前 Branch 进行重命名
- (magit-get-current-branch) 获取当前分支名

## 13 包管理

### 13.1 Emacs 的 Package-Mode

当通过 *M-x list-package* 命令打开一个 *Package* 的 Buffer, 它有如下命令:

1. i 标识安装（u 取消标识）
2. x 执行安装操作
3. d 标识删除（x 执行删除操作）
4. U 标识要更新的 package
5. ~ 标识所有废弃包
6. M-x package-autoremove 删除那些无用的旧包

### 13.2 包列表

1. elisp-slime-nav 写 elisp 代码时，可用于跳转到函数的定义

# 14 Emacs 26 特性

## 14.1 本文档说明

这是 Emacs 26 版本发布说明的中文翻译版本：GNU Emacs NEWS – history of user-visible changes. Copyright (C) 2016-2018 Free Software Foundation, Inc. See the end of the file for license conditions. Please send Emacs bug reports to bug-gnu-emacs@gnu.org. If possible, use M-x report-emacs-bug. This file is about changes in Emacs version 26. See file HISTORY for a list of GNU Emacs versions and release dates. See files NEWS.25, NEWS.24, …, NEWS.18, and NEWS.1-17 for changes in older Emacs versions. You can narrow news to a specific version by calling 'view-emacs-news' with a prefix argument or by typing C-u C-h C-n.

## 14.2 Emacs 26.2 安装变化

### 14.2.1 使用'–with-xwidgets' 参数构建 Emacs 需要依赖 WebKit2

构建支持 xwidgets 版本的 Emacs，需要先安装 webkit2gtk-4.0 包，要求版本在 2.12 及以上版本。这个改变其实从 Emacs 26.1 开始的，只不过没有发布文档里提到。

## 14.3 Emacs 26.2 中特殊模式和包的变化

### 14.3.1 Gnus

Mailutils movemail will now be used if found at runtime. **mail-source-movemail-program** 这个变量的默认值已经改为 **movemail** 。这样做的目地是确保使用 GNU 的邮件工作包里的移动邮件程序，如果发现它在 exec-path 目录里 ( 即使在构建时没有找到 )。可以通过定制 **mail-source-movemail-program** 变量的值来切换到其他程序。

### 14.3.2 Shell 模式

Shell 模式下的 Buffer 的 **scroll-conservatively** 变量默认值变为 101。这样使得在 Shell 模式下，当有新的输出添加到屏幕时，更好地模拟在文本终端下的滚动行为。重新将 **scroll-conservatively** 变量的值设置为 0 或者任何其他值，就可回退出老版本的行为 (可以在 shell-mode-hook 的函数列表中设置)。这个改变也是从 Emacs 26.1 就有，不过没有在其文档中提到。

### 14.3.3 VC

优化了 VC 对 Mercurial 的支持。Emacs 为了更快的操作速度，尽量避免唤起 **hg** 。

1. 新 vc-hg 选项 新增'vc-hg-parse-hg-data-structures' 用来控制 vc-hg 是否解析直接 Mercurial 数据结构，或者采用执行 hg 来替代。默认值为 t (老版本值为 nil)。新增'vc-hg-symbolic-revision-styles' 用来控制在 mode line 模式下版本的展示样式。新增'vc-hg-use-file-version-for-mode-line-version' 用来控制将版本展示在 mode line 里是那些已经访问过的文件还是那整体工作仓库的拷贝。

2. Mercurial 版本在 mode-line 的显示变化 老版本，mode-line 显示的是本地修改的版本数值 (1, 2, 3, ...)。从 26.1 版本开始，默认显示做了变化, 当前显示的是全局的修改版本值, 格式为修改的 hash 值。如果想恢复到之前的显示，只需要修改变量 **vc-hg-symbolic-revision-styles** 的值为 **("{rev}")** 。

## 14.4 Emacs 26.1 安装变化

### 14.4.1 默认情况下编译 Emacs 需要 libgnutls 这个库

采用 **configure –with-gnutls=no** 这个选项已经废弃。

### 14.4.2 GnuTLS 版本需要 2.12.2 或者更高版本

### 14.4.3 The new option 'configure –with-mailutils' causes Emacs to rely on

GNU Mailutils to retrieve email. It is recommended, and is the default if GNU Mailutils is installed. When –with-mailutils is not in effect, the Emacs build procedure by default continues to build and install a limited 'movemail' substitute that retrieves POP3 email only via insecure channels. To avoid this problem, use either –with-mailutils or –without-pop when configuring; –without-pop is the default on platforms other than native MS-Windows.

### 14.4.4 The new option 'configure –enable-gcc-warnings=warn-only' causes

GCC to issue warnings without stopping the build. This behavior is now the default in developer builds. As before, use '–disable-gcc-warnings' to suppress GCC's warnings, and '–enable-gcc-warnings' to stop the build if GCC issues warnings.

#### 14.4.5 When GCC warnings are enabled, '–enable-check-lisp-object-type' is

now enabled by default when configuring.

#### 14.4.6 The Emacs server now has socket-launching support. This allows

socket based activation, where an external process like systemd can invoke the Emacs server process upon a socket connection event and hand the socket over to Emacs. Emacs uses this socket to service emacsclient commands. This new functionality can be disabled with the configure option '–disable-libsystemd'.

#### 14.4.7 A systemd user unit file is provided. Use it in the standard way:

'systemctl –user enable emacs'. (If your Emacs is installed in a non-standard location, you may need to copy the emacs.service file to eg ~/.config/systemd/user/)

#### 14.4.8 New configure option '–disable-build-details' attempts to build an

Emacs that is more likely to be reproducible; that is, if you build and install Emacs twice, the second Emacs is a copy of the first. Deterministic builds omit the build date from the output of the 'emacs-version' and 'erc-cmd-SV' functions, and the leave the following variables nil: 'emacs-build-system', 'emacs-build-time', 'erc-emacs-build-time'.

#### 14.4.9 Emacs can now be built with support for Little CMS.

If the lcms2 library is installed, Emacs will enable features built on top of that library. The new configure option '–without-lcms2' can be used to build without lcms2 support even if it is installed. Emacs linked to Little CMS exposes color management functions in Lisp: the color metrics 'lcms-cie-de2000' and 'lcms-cam02-ucs', as well as functions for conversion to and from CIE CAM02 and CAM02-UCS.

#### 14.4.10 The configure option '–with-gameuser' now defaults to 'no',

as this appears to be the most common configuration in practice. When it is 'no', the shared game directory and the auxiliary program update-game-score are no longer needed and are not installed.

#### 14.4.11 Emacs 不再对 IRIX 系统做支持

我们希望这部分 Emacs 用户不会因此受到影响，因为 SGI 从 2013 年 12 份开始已经停止了对 IRIX 的支持。

### 14.5 Emacs 26.1 启动的变化

#### 14.5.1 新增'–fg-daemon' 选项

这个选项跟'–daemon' 类似，只有一点不一样的那就是它是运行在前台并且不是 fork 的。这个选项的目的是用于现在启动系统，如 systemd，它管理许多传统的常住行为 (daemon behavior)。同时，这个'–bg-daemon' 选项是'–daemon' 的别名。

#### 14.5.2 新增'–module-assertions' 选项

When given this option, Emacs will perform expensive correctness checks when dealing with dynamic modules. This is intended for module authors that wish to verify that their module conforms to the module requirements. The option makes Emacs abort if a module-related assertion triggers.

#### 14.5.3 Emacs now supports 24-bit colors on capable text terminals.

Terminal is automatically initialized to use 24-bit colors if the required capabilities are found in terminfo. See the FAQ node "(efaq) Colors on a TTY" for more information.

#### 14.5.4 Emacs 在启动的时候遵守 X 资源"scrollBar"

这个效果与工具条 ( tool bar ) 里的"toolBar" 资源类似。

## 14.6 Emacs 26.1 的变化

### 14.6.1 'buffer-offer-save' 添加新值

'buffer-offer-save' 添加新的选项'always'。当配置为这个值时命令'save-some-buffers' 将将这个 buffer 作为保存项 (offer this buffer for saving)。

### 14.6.2 Security vulnerability related to Enriched Text mode is removed.

1. Enriched Text mode does not evaluate Lisp in 'display' properties. This feature allows saving 'display' properties as part of text. Emacs 'display' properties support evaluation of arbitrary Lisp forms as part of processing the property for display, so displaying Enriched Text could be vulnerable to executing arbitrary malicious Lisp code included in the text (e.g., sent as part of an email message). Therefore, execution of arbitrary Lisp forms in 'display' properties decoded by Enriched Text mode is now disabled by default. Customize the new option 'enriched-allow-eval-in-display-props' to a non-nil value to allow Lisp evaluation in decoded 'display' properties. This vulnerability was introduced in Emacs 21.1. To work around that in Emacs versions before 25.3, append the following to your ~/.emacs init file: (eval-after-load "enriched" '(defun enriched-decode-display-prop (start end &optional param) (list start end)))

### 14.6.3 Functions in 'write-contents-functions' can fully short-circuit the

'save-buffer' process. Previously, saving a buffer that was not visiting a file would always prompt for a file name. Now it only does so if 'write-contents-functions' is nil (or all its functions return nil).

### 14.6.4 New variable 'executable-prefix-env' for inserting magic signatures.

This variable affects the format of the interpreter magic number inserted by 'executable-set-magic'. If non-nil, the magic number now takes the form "#!/usr/bin/env interpreter", otherwise the value determined by 'executable-prefix', which is by default "#!/path/to/interpreter". By default, 'executable-prefix-env' is nil, so the default behavior is not changed.

### 14.6.5 The variable 'emacs-version' no longer includes the build number.

This is now stored separately in a new variable, 'emacs-build-number'.

### 14.6.6 Emacs now provides a limited form of concurrency with Lisp threads.

Concurrency in Emacs Lisp is "mostly cooperative", meaning that Emacs will only switch execution between threads at well-defined times: when Emacs waits for input, during blocking operations related to threads (such as mutex locking), or when the current thread explicitly yields. Global variables are shared among all threads, but a 'let' binding is thread-local. Each thread also has its own current buffer and its own match data. See the chapter "(elisp) Threads" in the ELisp manual for full documentation of these facilities.

### 14.6.7 The new user variable 'electric-quote-chars' provides a list

of curved quotes for 'electric-quote-mode', allowing user to choose the types of quotes to be used.

### 14.6.8 The new user option 'electric-quote-context-sensitive' makes

'electric-quote-mode' context sensitive. If it is non-nil, you can type an ASCII apostrophe to insert an opening or closing quote, depending on context. Emacs will replace the apostrophe by an opening quote character at the beginning of the buffer, the beginning of a line, after a whitespace character, and after an opening parenthesis; and it will replace the apostrophe by a closing quote character in all other cases.

### 14.6.9 The new variable 'electric-quote-inhibit-functions' controls when

to disable electric quoting based on context. Major modes can add functions to this list; Emacs will temporarily disable 'electric-quote-mode' whenever any of the functions returns non-nil. This can be used by major modes that derive from 'text-mode' but allow inline code segments, such as 'markdown-mode'.

#### 14.6.10 The new user variable 'dired-omit-case-fold' allows the user to

customize the case-sensitivity of dired-omit-mode. It defaults to the same sensitivity as that of the filesystem for the corresponding dired buffer.

#### 14.6.11 Emacs now uses double buffering to reduce flicker when editing and

resizing graphical Emacs frames on the X Window System. This support requires the DOUBLE-BUFFER extension, which major X servers have supported for many years. If your system has this extension, but an Emacs built with double buffering misbehaves on some displays you use, you can disable the feature by adding '(inhibit-double-buffering . t) to default-frame-alist. Or inject this parameter into the selected frame by evaluating this form: (modify-frame-parameters nil '((inhibit-double-buffering . t)))

#### 14.6.12 The customization group 'wp', whose label was "text", is now

deprecated. Use the new group 'text', which inherits from 'wp', instead.

#### 14.6.13 The new function 'call-shell-region' executes a command in an

inferior shell with the buffer region as input.

#### 14.6.14 The new user option 'shell-command-dont-erase-buffer' controls

if the output buffer is erased between shell commands; if non-nil, the output buffer is not erased; this variable also controls where to set the point in the output buffer: beginning of the output, end of the buffer or save the point. When 'shell-command-dont-erase-buffer' is nil, the default value, the behavior of 'shell-command', 'shell-command-on-region' and 'async-shell-command' is as usual.

#### 14.6.15 The new user option 'async-shell-command-display-buffer' controls

whether the output buffer of an asynchronous command is shown immediately, or only when there is output.

#### 14.6.16 New user option 'mouse-select-region-move-to-beginning'.

This option controls the position of point when double-clicking mouse-1 on the end of a parenthetical grouping or string-delimiter: the default value nil keeps point at the end of the region, setting it to non-nil moves point to the beginning of the region.

#### 14.6.17 New user option 'mouse-drag-and-drop-region'.

This option allows you to drag the entire region of text to another place or another buffer. Its behavior is customizable via the new options 'mouse-drag-and-drop-region-cut-when-buffers-differ', 'mouse-drag-and-drop-region-show-tooltip', and 'mouse-drag-and-drop-region-show-cursor'.

#### 14.6.18 The new user option 'confirm-kill-processes' allows the user to

skip a confirmation prompt for killing subprocesses when exiting Emacs. When set to t (the default), Emacs will prompt for confirmation before killing subprocesses on exit, which is the same behavior as before.

#### 14.6.19 'find-library-name' will now fall back on looking at 'load-history'

to try to locate libraries that have been loaded with an explicit path outside 'load-path'.

#### 14.6.20 Faces in 'minibuffer-prompt-properties' no longer overwrite properties

in the text in functions like 'read-from-minibuffer', but instead are added to the end of the face list. This allows users to say things like '(read-from-minibuffer (propertize "Enter something: " 'face 'bold))'.

#### 14.6.21 The new variable 'extended-command-suggest-shorter' has been added

to control whether to suggest shorter 'M-x' commands or not.

#### 14.6.22 icomplete now respects 'completion-ignored-extensions'.

#### 14.6.23 Non-breaking hyphens are now displayed with the 'nobreak-hyphen'

face instead of the 'escape-glyph' face.

#### 14.6.24 Approximations to quotes are now displayed with the new 'homoglyph'

face instead of the 'escape-glyph' face.

#### 14.6.25 New face 'header-line-highlight'.

This face is the header-line analogue of 'mode-line-highlight'; it should be the preferred mouse-face for mouse-sensitive elements in the header line.

#### 14.6.26 'C-x h' ('mark-whole-buffer') will now avoid marking the prompt

part of minibuffers.

#### 14.6.27 'fill-paragraph' no longer marks the buffer as changed unless it

actually changed something.

#### 14.6.28 The locale language name 'ca' is now mapped to the language

environment 'Catalan', which has been added.

#### 14.6.29 'align-regexp' has a separate history for its interactive argument.

'align-regexp' no longer shares its history with all other history-less functions that use 'read-string'.

#### 14.6.30 The networking code has been reworked so that it's more

asynchronous than it was (when specifying :nowait t in 'make-network-process'). How asynchronous it is varies based on the capabilities of the

system, but on a typical GNU/Linux system the DNS resolution, the connection, and (for TLS streams) the TLS negotiation are all done without blocking the main Emacs thread. To get asynchronous TLS, the TLS boot parameters have to be passed in (see the manual for details). Certain process oriented functions (like 'process-datagram-address') will block until socket setup has been performed. The recommended way to deal with asynchronous sockets is to avoid interacting with them until they have changed status to "run". This is most easily done from a process sentinel.

#### 14.6.31 'make-network-process' and 'open-network-stream' sometimes allowed

:service to be an integer string (e.g., :service "993") and sometimes required an integer (e.g., :service 993). This difference has been eliminated, and integer strings work everywhere.

#### 14.6.32 It is possible to disable attempted recovery on fatal signals.

Two new variables support disabling attempts to recover from stack overflow and to avoid automatic auto-save when Emacs is delivered a fatal signal. 'attempt-stack-overflow-recovery', if set to nil, will disable attempts to recover from C stack overflows; Emacs will then crash as with any other fatal signal. 'attempt-orderly-shutdown-on-fatal-signal', if set to nil, will disable attempts to auto-save the session and shut down in an orderly fashion when Emacs receives a fatal signal; instead, Emacs will terminate immediately. Both variables are non-nil by default. These variables are for users who would like to avoid the small probability of data corruption due to techniques Emacs uses to recover in these situations.

#### 14.6.33 File local and directory local variables are now initialized each

time the major mode is set, not just when the file is first visited. These local variables will thus not vanish on setting a major mode.

#### 14.6.34 A second dir-local file (.dir-locals-2.el) is now accepted.

See the doc string of 'dir-locals-file' for more information.

#### 14.6.35 Connection-local variables can be used to specify local variables

with a value depending on the connected remote server. For details, see the node "(elisp) Connection Local Variables" in the ELisp manual.

#### 14.6.36 International domain names (IDNA) are now encoded via the new

puny.el library, so that one can visit Web sites with non-ASCII URLs.

#### 14.6.37 The new 'list-timers' command lists all active timers in a buffer,

where you can cancel them with the 'c' command.

#### 14.6.38 'switch-to-buffer-preserve-window-point' now defaults to t.

Applications that call 'switch-to-buffer' and want to show the buffer at the position of its point should use 'pop-to-buffer-same-window' in lieu of 'switch-to-buffer'.

#### 14.6.39 The new variable 'debugger-stack-frame-as-list' allows displaying

all call stack frames in a Lisp backtrace buffer as lists. Both debug.el and edebug.el have been updated to heed to this variable.

#### 14.6.40 Values in call stack frames are now displayed using 'cl-prin1'.

The old behavior of using 'prin1' can be restored by customizing the new option 'debugger-print-function'.

#### 14.6.41 NUL bytes in text copied to the system clipboard are now replaced with "\0".

#### 14.6.42 The new variable 'x-ctrl-keysym' has been added to the existing

roster of X keysyms. It can be used in combination with another variable of this kind to swap modifiers in Emacs.

#### 14.6.43 New input methods: 'cyrillic-tuvan', 'polish-prefix', 'uzbek-cyrillic'.

#### 14.6.44 The 'dutch' input method no longer attempts to support Turkish too.

Also, it no longer converts 'IJ' and 'ij' to the compatibility characters U+0132 LATIN CAPITAL LIGATURE IJ and U+0133 LATIN SMALL LIGATURE IJ.

#### 14.6.45 File name quoting by adding the prefix "/:" is now possible for the

local part of a remote file name. Thus, if you have a directory named *"~" on the remote host "foo", you can prevent it from being substituted by a home directory by writing it as "/foo*:/~/file".

#### 14.6.46 The new variable 'maximum-scroll-margin' allows having effective

settings of 'scroll-margin' up to half the window size, instead of always restricting the margin to a quarter of the window.

#### 14.6.47 Emacs can scroll horizontally using mouse, touchpad, and trackbar.

You can enable this by customizing 'mouse-wheel-tilt-scroll'. If you want to reverse the direction of the scroll, customize 'mouse-wheel-flip-direction'.

#### 14.6.48 The default GnuTLS priority string now includes %DUMBFW.

This is to avoid bad behavior in some firewalls, which causes the connection to be closed by the remote host.

#### 14.6.49 Emacsclient changes

1. Emacsclient has a new option '-u' / '–suppress-output'. This option suppresses display of return values from the server process.

2. Emacsclient has a new option '-T' / '–tramp'. This helps with using a local Emacs session as the server for a remote emacsclient. With appropriate setup, one can now set the EDITOR environment variable on a remote machine to emacsclient, and use the local Emacs to edit

remote files via Tramp. See the node "(emacs) emacsclient Options" in the user manual for the details.

3. Emacsclient now accepts command-line options in ALTERNATE$_{\text{EDITOR}}$ and '–alternate-editor'. For example, ALTERNATE$_{\text{EDITOR}}$="emacs -Q -nw". Arguments may be quoted "like this", so that for example an absolute path containing a space may be specified; quote escaping is not supported.

#### 14.6.50 New user option 'dig-program-options' and extended functionality

for DNS-querying functions 'nslookup-host', 'dns-lookup-host', and 'run-dig'. Each function now accepts an optional name server argument interactively (with a prefix argument) and non-interactively.

#### 14.6.51 'describe-key-briefly' now ignores mouse movement events.

#### 14.6.52 The new variable 'eval-expression-print-maximum-character' prevents

large integers from being displayed as characters by 'M-:' and similar commands.

#### 14.6.53 Two new commands for finding the source code of Emacs Lisp

libraries: 'find-library-other-window' and 'find-library-other-frame'.

#### 14.6.54 The new variable 'display-raw-bytes-as-hex' allows you to change

the display of raw bytes from octal to hex.

#### 14.6.55 You can now provide explicit field numbers in format specifiers.

For example, '(format "%2$s %1$s %2$s" "X" "Y")' produces "Y X Y".

#### 14.6.56 Emacs 支持在 Buffer 里显示行号 (可选项)

这个显示行号的原生支持与之前的 **linum-mode** 提供的功能一样，但它更快。并且，不会因为行号的显示而占用边距。通过配置 Buffer 本地变量 display-line-numbers 去开启这个行号显示项。或者还可以通过打开 **display-line-numbers-mode** 这个 minor 模式也能达到相同效果。还有一种方式，那就是打开全局的 **global-display-line-numbers-mode** 模式。当这些模式打开后，配置 **display-line-numbers-type** 的值会与 **display-line-numbers** 的值相同。行号不会显示在所有的 minibuffer 的窗口以及提示窗口，因为他们不是用于此处。Lisp 程序可以通过设置 **display-line-numbers-disable** 的文本属性或者屏幕当前行第一个字符的属性来关闭此功能。做成可配置的包目的是为了更好地做显示控制。Lisp 程序想知道多少屏幕宽度被行号所占用，可以通过 **line-number-display-width** 这个函数来获取。**linum-mode** 和其他类似的包可以废弃，不再使用。Emacs 官方推荐使用新的原生方法来显示行号。

#### 14.6.57 添加'arabic-shaper-ZWNJ-handling' 用户选项用来处理 Arabic 文本渲染中的 ZWNJ 问题

### 14.7 Emacs 26.1 中编辑操作的变化

#### 14.7.1 新加变量'column-number-indicator-zero-based'.

Traditionally, in Column Number mode, the displayed column number counts from zero starting at the left margin of the window. This behavior is now controlled by 'column-number-indicator-zero-based'. If you would prefer for the displayed column number to count from one, you may set this variable to nil. (Behind the scenes, there is now a new mode line construct, '%C', which operates exactly as '%c' does except that it counts from one.)

#### 14.7.2 New single-line horizontal scrolling mode.

The 'auto-hscroll-mode' variable can now have a new special value, 'current-line', which causes only the line where the cursor is displayed to be horizontally scrolled when lines are truncated on display and point moves outside the left or right window margin.

#### 14.7.3 New mode line constructs '%o' and '%q', and user option

'mode-line-percent-position'. '%o' displays the "degree of travel" of the window through the buffer. Unlike the default '%p', this percentage approaches

100% as the window approaches the end of the buffer. '%q' displays the percentage offsets of both the start and the end of the window, e.g. "5-17%". The new option 'mode-line-percent-position' makes it easier to switch between '%p', '%P', and these new constructs.

#### 14.7.4 Two new user options 'list-matching-lines-jump-to-current-line' and

'list-matching-lines-current-line-face' to show the current line highlighted in **Occur** buffer.

#### 14.7.5 The 'occur' command can now operate on the region.

#### 14.7.6 New bindings for 'query-replace-map'.

'undo', undo the last replacement; bound to 'u'. 'undo-all', undo all replacements; bound to 'U'.

#### 14.7.7 'delete-trailing-whitespace' deletes whitespace after form feed.

In modes where form feed was treated as a whitespace character, 'delete-trailing-whitespace' would keep lines containing it unchanged. It now deletes whitespace after the last form feed thus behaving the same as in modes where the character is not whitespace.

#### 14.7.8 Emacs no longer prompts about editing a changed file when the file's

content is unchanged. Instead of only checking the modification time, Emacs now also checks the file's actual content before prompting the user.

#### 14.7.9 Various casing improvements.

1. 'upcase', 'upcase-region' et al. convert title case characters (such as ) into their upper case form (such as ).

2. 'capitalize', 'upcase-initials' et al. make use of title-case forms of initial characters (correctly producing for example  ungla instead of incorrect  ungla).

3. Characters which turn into multiple ones when cased are correctly handled. For example, fi ligature is converted to FI when upper cased.

4. Greek small sigma is correctly handled when at the end of the word. Strings such as ΟΣΟΣ are now correctly converted to Ο when capitalized instead of incorrect Ο (compare lowercase sigma at the end of the word).

#### 14.7.10 Emacs can now auto-save buffers to visited files in a more robust

manner via the new mode 'auto-save-visited-mode'. Unlike 'auto-save-visited-file-name', this mode uses the normal saving procedure and therefore obeys saving hooks. 'auto-save-visited-file-name' is now obsolete.

#### 14.7.11 New behavior of 'mark-defun'.

Prefix argument selects that many (or that many more) defuns. Negative prefix arg flips the direction of selection. Also, 'mark-defun' between defuns correctly selects N following defuns (or -N previous for negative arguments). Finally, comments preceding the defun are selected unless they are separated from the defun by a blank line.

#### 14.7.12 New command 'replace-buffer-contents'.

This command replaces the contents of the accessible portion of the current buffer with the contents of the accessible portion of a different buffer while keeping point, mark, markers, and text properties as intact as possible.

#### 14.7.13 New commands 'apropos-local-variable' and 'apropos-local-value'.

These are buffer-local versions of 'apropos-variable' and 'apropos-value', respectively. They show buffer-local variables whose names and values, respectively, match a given pattern.

#### 14.7.14 More user control of reordering bidirectional text for display.

The two new variables, 'bidi-paragraph-start-re' and 'bidi-paragraph-separate-re', allow customization of what exactly are paragraphs, for the purposes of bidirectional display.

#### 14.7.15 New variable 'x-wait-for-event-timeout'.

This controls how long Emacs will wait for updates to the graphical state to take effect (making a frame visible, for example).

### 14.8 Emacs 26.1 中特殊模式和包的变化

#### 14.8.1 Emacs 26.1 采用 Org 9.1.6 版本.

具体变化可查看 ORG-NEWS 文件

#### 14.8.2 新函数 cl-generic-p

#### 14.8.3 Dired 的变化

1. You can answer 'all' in 'dired-do-delete' to delete recursively all remaining directories without more prompts.

2. Dired supports wildcards in the directory part of the file names.

3. You can now use ''¿' in 'dired-do-shell-command'. It gets replaced by the current file name, like ' ? '.

4. A new option 'dired-always-read-filesystem' defaulting to nil. If non-nil, buffers visiting files are reverted before they are searched; for instance, in 'dired-mark-files-containing-regexp' a non-nil value of this option means the file is revisited in a temporary buffer; this temporary buffer is the actual buffer searched: the original buffer visiting the file is not modified.

5. Users can now customize mouse clicks in Dired in a more flexible way. The new command 'dired-mouse-find-file' can be bound to a mouse click and used to visit files/directories in Dired in the selected window. The new command 'dired-mouse-find-file-other-frame' similarly visits files/directories in another frame. You can write your own commands that invoke 'dired-mouse-find-file' with non-default optional arguments, to tailor the effects of mouse clicks on file names in Dired buffers.

6. In wdired, when editing files to contain slash characters, the resulting directories are automatically created. Whether to do this is controlled by the 'wdired-create-parent-directories' variable.

7. 'W' is now bound to 'browse-url-of-dired-file', and is useful for viewing HTML files and the like.

8. New variable 'dired-clean-confirm-killing-deleted-buffers' controls whether Dired asks to kill buffers visiting deleted files and directories. The default is t, so Dired asks for confirmation, to keep previous behavior.

#### 14.8.4 html2text is now marked obsolete.

#### 14.8.5 smerge-refine-regions can refine regions in separate buffers.

#### 14.8.6 Info menu and index completion uses substring completion by default.

This can be customized via the 'info-menu' category in 'completion-category-overrides'.

#### 14.8.7 The ancestor buffer is shown by default in 3-way merges.

A new option 'ediff-show-ancestor' and a new toggle 'ediff-toggle-show-ancestor'.

#### 14.8.8 TEX: Add luatex and xetex as alternatives to pdftex

#### 14.8.9 Electric-Buffer-menu

1. Key 'U' is bound to 'Buffer-menu-unmark-all' and key 'M-DEL' is bound to 'Buffer-menu-unmark-all-buffers'.

#### 14.8.10 hideshow mode got four key bindings that are analogous to outline

mode bindings: 'C-c @ C-a', 'C-c @ C-t', 'C-c @ C-d', and 'C-c @ C-e'.

#### 14.8.11 bs

1. Two new commands 'bs-unmark-all', bound to 'U', and 'bs-unmark-previous', bound to <backspace>.

#### 14.8.12 Buffer-menu

1. Two new commands 'Buffer-menu-unmark-all', bound to 'U' and 'Buffer-menu-unmark-all-buffers', bound to 'M-DEL'.

#### 14.8.13 Checkdoc

1. 'checkdoc-arguments-in-order-flag' now defaults to nil.

#### 14.8.14 Gnus

1. The ~/.newsrc file will now only be saved if the native select method is an NNTP select method.

2. A new command for sorting articles by readedness marks has been added: 'C-c C-s C-m C-m'.

3. In 'message-citation-line-format' the '%Z' format is now the time zone name instead of the numeric form. The '%z' format continues to be the numeric form. The new behavior is compatible with 'format-time-string'.

#### 14.8.15 Ibuffer

1. New command 'ibuffer-jump'.

2. New filter commands 'ibuffer-filter-by-basename', 'ibuffer-filter-by-file-extension', 'ibuffer-filter-by-directory', 'ibuffer-filter-by-starred-name', 'ibuffer-filter-by-modified' and 'ibuffer-filter-by-visiting-file'; bound respectively to '*b', '*.', '//', '/*', '/i' and '/v'.

3. Two new commands 'ibuffer-filter-chosen-by-completion' and 'ibuffer-and-filter', the second bound to '/&'.

4. The commands 'ibuffer-pop-filter', 'ibuffer-pop-filter-group', 'ibuffer-or-filter' and 'ibuffer-filter-disable' have the alternative bindings '/<up>', '/S-<up>', '/|' and '/DEL', respectively.

5. The data format specifying filters has been extended to allow explicit logical 'and', and a more flexible form for logical 'not'. See 'ibuffer-filtering-qualifiers' doc string for full details.

6. A new command 'ibuffer-copy-buffername-as-kill'; bound to 'B'.

7. New command 'ibuffer-change-marks'; bound to '* c'.

8. A new command 'ibuffer-mark-by-locked' to mark all locked buffers; bound to '% L'.

9. A new option 'ibuffer-locked-char' to indicate locked buffers; Ibuffer shows a new column displaying 'ibuffer-locked-char' for locked buffers.

10. A new command 'ibuffer-unmark-all-marks' to unmark all buffers without asking confirmation; bound to 'U'; 'ibuffer-do-replace-regexp' bound to 'r'.

11. A new command 'ibuffer-mark-by-content-regexp' to mark buffers whose content matches a regexp; bound to '% g'.

12. Two new options 'ibuffer-never-search-content-name' and 'ibuffer-never-search-content-mode' used by 'ibuffer-mark-by-content-regexp'.

#### 14.8.16 Browse-URL

1. Support for opening links to man pages in Man or WoMan mode.

#### 14.8.17 Comint

1. New user option 'comint-move-point-for-matching-input' to control where to place point after 'C-c M-r' and 'C-c M-s'.

2. New user option 'comint-terminfo-terminal'. This option allows control of the value of the TERM environment variable Emacs puts into the environment of the Comint mode and its derivatives, such as Shell mode and Compilation Shell minor-mode. The default is "dumb", for compatibility with previous behavior.

#### 14.8.18 Compilation mode

1. Messages from CMake are now recognized.

2. The number of errors, warnings, and informational messages is now displayed in the mode line. These are updated as compilation proceeds.

#### 14.8.19 Grep

1. Grep commands will now use GNU grep's '–null' option if available, which allows distinguishing the filename from contents if they contain colons. This can be controlled by the new custom option 'grep-use-null-filename-separator'.

2. The grep/rgrep/lgrep functions will now ask about saving files before running. This is controlled by the 'grep-save-buffers' variable.

#### 14.8.20 Edebug

1. Edebug can be prevented from pausing 1 second after reaching a breakpoint (e.g. with "f" and "o") by customizing the new option 'edebug-sit-on-break'.

2. New customizable option 'edebug-max-depth'. This allows you to enlarge the maximum recursion depth when instrumenting code.

3. 'edebug-prin1-to-string' now aliases 'cl-prin1-to-string'. This means edebug output is affected by variables 'cl-print-readably' and 'cl-print-compiled'. To completely restore the previous printing behavior, use (fset 'edebug-prin1-to-string #'prin1-to-string)

#### 14.8.21 Eshell

1. 'eshell-input-filter's value is now a named function 'eshell-input-filter-default', and has a new custom option 'eshell-input-filter-initial-space' to ignore adding commands prefixed with blank space to eshell history.

#### 14.8.22 EUDC

1. Backward compatibility support for BBDB versions less than 3 (i.e., BBDB 2.x) is deprecated and will likely be removed in the next major release of Emacs. Users of BBDB 2.x should plan to upgrade to BBDB 3.x.

#### 14.8.23 eww

1. New 'M-RET' command for opening a link at point in a new eww buffer.

2. A new 's' command for switching to another eww buffer via the minibuffer.

3. The 'o' command ('shr-save-contents') has moved to 'O' to avoid collision with the 'o' command from 'image-map'.

4. A new command 'C' ('eww-toggle-colors') can be used to toggle whether to use the HTML-specified colors or not. The user can also customize the 'shr-use-colors' variable.

5. Images that are being loaded are now marked with gray "placeholder" images of the size specified by the HTML. They are then replaced by the real images asynchronously, which will also now respect width/height HTML specs (unless they specify widths/heights bigger than the current window).

6. The 'w' command on links is now 'shr-maybe-probe-and-copy-url'. 'shr-copy-url' now only copies the url at point; users who wish to avoid accidentally accessing remote links may rebind 'w' and 'u' in 'eww-link-keymap' to it.

#### 14.8.24 Ido

1. The commands 'find-alternate-file-other-window', 'dired-other-window', 'dired-other-frame', and 'display-buffer-other-window' are now remapped to Ido equivalents if Ido mode is active.

#### 14.8.25 Images

1. Images are automatically scaled before displaying based on the 'image-scaling-factor' variable (if Emacs supports scaling the images in question).

2. It's now possible to specify aspect-ratio preserving combinations of :width/:max-height and :height/:max-width keywords. In either case, the "max" keywords win. (Previously some combinations would, depending on the aspect ratio of the image, just be ignored and in other instances this would lead to the aspect ratio not being preserved.)

3. Images inserted with 'insert-image' and related functions get a keymap put into the text properties (or overlays) that span the image. This keymap binds keystrokes for manipulating size and rotation, as well as saving the image to a file. These commands are also available in 'image-mode'.

4. A new library for creating and manipulating SVG images has been added. See the "(elisp) SVG Images" section in the ELisp reference manual for details.

5. New setf-able function to access and set image parameters is provided: 'image-property'.

6. New commands 'image-scroll-left' and 'image-scroll-right' for 'image-mode' that complement 'image-scroll-up' and 'image-scroll-down': they have the same prefix arg behavior and stop at image boundaries.

#### 14.8.26 Image-Dired

1. Now provides a minor mode 'image-dired-minor-mode' which replaces the function 'image-dired-setup-dired-keybindings'.

2. Thumbnail generation is now asynchronous. The number of concurrent processes is limited by the variable 'image-dired-queue-active-limit'.

3. 'image-dired-thumbnail-storage' has a new option 'standard-large' for generating 256x256 thumbnails according to the Thumbnail Managing Standard.

4. Inherits movement keys from 'image-mode' for viewing full images. This includes the usual char, line, and page movement commands.

5. All the -options types have been changed to argument lists instead of shell command strings. This change affects 'image-dired-cmd-create-thumbnail-options', 'image-dired-cmd-create-temp-image-options', 'image-dired-cmd-rotate-thumbnail-options', 'image-dired-cmd-rotate-original-options', 'image-dired-cmd-write-exif-data-options', 'image-dired-cmd-read-exif-data-options', and introduces 'image-dired-cmd-pngnq-options', 'image-dired-cmd-pngcrush-options', 'image-dired-cmd-create-standard-thumbnail-options'.

6. Recognizes more tools by default, including pngnq-s9 and OptiPNG.

7. 'find-file' and related commands now work on thumbnails and displayed images, providing a default argument of the original file name via an addition to 'file-name-at-point-functions'.

#### 14.8.27 The default 'Info-default-directory-list' no longer checks some obsolete

directory suffixes (gnu, gnu/lib, gnu/lib/emacs, emacs, lib, lib/emacs) when searching for info directories.

#### 14.8.28 The commands that add ChangeLog entries now prefer a VCS root directory

for the ChangeLog file, if none already exists. Customize 'change-log-directory-files' to nil for the old behavior.

#### 14.8.29 Support for non-string values of 'time-stamp-format' has been removed.

#### 14.8.30 Message

1. 'message-use-idna' now defaults to t (because Emacs comes with built-in IDNA support now).

2. When sending HTML messages with embedded images, and you have exiftool installed, and you rotate images with EXIF data (i.e., JPEGs), the rotational information will be inserted into the outgoing image in the message. (The original image will not have its orientation affected.)

3. The 'message-valid-fqdn-regexp' variable has been removed, since there are now top-level domains added all the time. Message will no longer warn about sending emails to top-level domains it hasn't heard about.

4. 'message-beginning-of-line' (bound to 'C-a') understands folded headers. In 'visual-line-mode' it will look for the true beginning of a header while in non-'visual-line-mode' it will move the point to the indented header's value.

#### 14.8.31 Package

1. The new variable 'package-gnupghome-dir' has been added to control where the GnuPG home directory (used for signature verification) is located and whether GnuPG's option '–homedir' is used or not.

2. Deleting a package no longer respects 'delete-by-moving-to-trash'.

#### 14.8.32 Python

1. The new variable 'python-indent-def-block-scale' has been added. It controls the depth of indentation of arguments inside multi-line function signatures.

#### 14.8.33 Tramp

1. The method part of remote file names is mandatory now. A valid remote file name starts with "/method:host:" or "/method:user@host:".

2. The new pseudo method "-" is a marker for the default method. "/-::" is the shortest remote file name then.

3. The command 'tramp-change-syntax' allows you to choose an alternative remote file name syntax.

4. New connection method "sg", which supports editing files under a different group ID.

5. New connection method "doas" for OpenBSD hosts.

6. New connection method "gdrive", which allows access to Google Drive onsite repositories.

7. Gateway methods in Tramp have been removed. Instead, the Tramp manual documents how to configure ssh and PuTTY accordingly.

8. Setting the "ENV" environment variable in 'tramp-remote-process-environment' enables reading of shell initialization files.

9. Tramp is able now to send SIGINT to remote asynchronous processes.

10. Variable 'tramp-completion-mode' is obsoleted.

#### 14.8.34 'auto-revert-use-notify' is set back to t in 'global-auto-revert-mode'.

#### 14.8.35 JS mode

1. JS mode now sets 'comment-multi-line' to t.

2. New variable 'js-indent-align-list-continuation', when set to nil, will not align continuations of bracketed lists, but will indent them by the fixed width 'js-indent-level'.

#### 14.8.36 CSS mode

1. Support for completing attribute values, at-rules, bang-rules, HTML tags, classes and IDs using the 'completion-at-point' command. Completion candidates for HTML classes and IDs are retrieved from open HTML mode buffers.

2. CSS mode now binds 'C-h S' to a function that will show information about a CSS construct (an at-rule, property, pseudo-class, pseudo-element, with the default being guessed from context). By default the information is looked up on the Mozilla Developer Network, but this can be customized using 'css-lookup-url-format'.

3. CSS colors are fontified using the color they represent as the background. For instance, #ff0000 would be fontified with a red background.

#### 14.8.37 Emacs now supports character name escape sequences in character and

string literals. The syntax variants 'character name' and 'U+code' are supported.

#### 14.8.38 Prog mode has some support for multi-mode indentation.

This allows better indentation support in modes that support multiple programming languages in the same buffer, like literate programming environments or ANTLR programs with embedded Python code. A major mode can provide indentation context for a sub-mode. To support this, modes should use 'prog-first-column' instead of a literal zero and avoid calling 'widen' in their indentation functions. See the node "(elisp) Mode-Specific Indent" in the ELisp manual for more details.

#### 14.8.39 ERC

1. New variable 'erc-default-port-tls' used to connect to TLS IRC servers.

#### 14.8.40 URL

1. The new function 'url-cookie-delete-cookie' can be used to programmatically delete all cookies, or cookies from a specific domain.

2. 'url-retrieve-synchronously' now takes an optional timeout parameter.

3. The URL package now supports HTTPS over proxies supporting CONNECT.

4. 'url-user-agent' now defaults to 'default', and the User-Agent string is computed dynamically based on 'url-privacy-level'.

#### 14.8.41 VC and related modes

1. 'vc-dir-mode' now binds 'vc-log-outgoing' to 'O'; and has various branch-related commands on a keymap bound to 'B'.

2. 'vc-region-history' is now bound to 'C-x v h', replacing the older 'vc-insert-headers' binding.

3. New user option 'vc-git-print-log-follow' to follow renames in Git logs for a single file.

#### 14.8.42 CC mode

1. Opening a .h file will turn C or C++ mode depending on language used. This is done with the help of the 'c-or-c++-mode' function, which analyzes buffer contents to infer whether it's a C or C++ source file.

#### 14.8.43 New option 'cpp-message-min-time-interval' to allow user control

of progress messages in cpp.el.

#### 14.8.44 New DNS mode command 'dns-mode-ipv6-to-nibbles' to convert IPv6 addresses

to a format suitable for reverse lookup zone files.

#### 14.8.45 Ispell

1. Enchant is now supported as a spell-checker. Enchant is a meta-spell-checker that uses providers such as Hunspell to do the actual checking. With it, users can use spell-checkers not directly supported by

Emacs, such as Voikko, Hspell and AppleSpell, more easily share personal word-lists with other programs, and configure different spelling-checkers for different languages. (Version 2.1.0 or later of Enchant is required.)

#### 14.8.46 Flymake

1. Flymake has been completely redesigned Flymake now annotates arbitrary buffer regions, not just lines. It supports arbitrary diagnostic types, not just errors and warnings (see variable 'flymake-diagnostic-types-alist'). It also supports multiple simultaneous backends, meaning that you can check your buffer from different perspectives (see variable 'flymake-diagnostic-functions'). Backends for Emacs Lisp mode are provided. The old Flymake behavior is preserved in the so-called "legacy backend", which has been updated to benefit from the new UI features.

#### 14.8.47 Term

1. **term-char-mode** 模式下的 buffer 默认为只读模式 这个 buffer 默认设置为只读模式主要防止其他除了程序过滤器之外的任何对其进行修改的行为；并且光标的移动也在有所限制，主要用为了保持每次命令执行后光标都处理"当前"位置。这也是为了在交互式命令中使得 emacs 更加方便获取其状态。可以通过 **term-char-mode-buffer-read-only** 来控制 buffer 是否为只读, Emacs 26 以后这个值默认设置不为 nil, 如果想关闭只读，设置其值为 nil 就行。同理可以通过另一个变量，即: **term-char-mode-point-at-process-mark** 来控制光标是否可上下移动, 其默认值不是 nil。如果想让光标能上下移动，设置这个值为 nil。

#### 14.8.48 Xref

1. When an **xref** buffer is needed, 'TAB' quits and jumps to an xref. A new command 'xref-quit-and-goto-xref', bound to 'TAB' in **xref** buffers, quits the window before jumping to the destination. In many situations, the intended window configuration is restored, just as if the **xref** buffer hadn't been necessary in the first place.

## 14.9 Emacs 26.1 新的模式和包

### 14.9.1 新的 Elisp 数据结构库 radix-tree

### 14.9.2 New library 'xdg' with utilities for some XDG standards and specs.

### 14.9.3 HTML

1. A new submode of 'html-mode', 'mhtml-mode', is now the default mode for *.html files. This mode handles indentation, fontification, and commenting for embedded JavaScript and CSS.

### 14.9.4 New mode 'conf-toml-mode' is a sub-mode of 'conf-mode', specialized

for editing TOML files.

### 14.9.5 New mode 'conf-desktop-mode' is a sub-mode of 'conf-unix-mode',

specialized for editing freedesktop.org desktop entries.

### 14.9.6 New minor mode 'pixel-scroll-mode' provides smooth pixel-level scrolling.

### 14.9.7 New major mode 'less-css-mode' (a minor variant of 'css-mode') for

editing Less files.

### 14.9.8 New package 'auth-source-pass' integrates 'auth-source' with the

password manager password-store (`http://passwordstore.org`).

## 14.10 Emacs 26.1 中不兼容的 Lisp 部分说明

### 14.10.1 'password-data' is now a hash-table so that 'password-read' can use

any object for the 'key' argument.

#### 14.10.2 Command 'dired-mark-extension' now automatically prepends a '.' to the

extension when not present. The new command 'dired-mark-suffix' behaves similarly but it doesn't prepend a '.'.

#### 14.10.3 Certain cond/pcase/cl-case forms are now compiled using a faster jump

table implementation. This uses a new bytecode op 'switch', which isn't compatible with previous Emacs versions. This functionality can be disabled by setting 'byte-compile-cond-use-jump-table' to nil.

#### 14.10.4 If 'comment-auto-fill-only-comments' is non-nil, 'auto-fill-function'

is now called only if either no comment syntax is defined for the current buffer or the self-insertion takes place within a comment.

#### 14.10.5 ucs-names 为哈希表 (hash table)

#### 14.10.6 if-let 和 when-let 现在支持绑定列表 (如 Scheme Request for Implementation 2 中的实现一样).

#### 14.10.7 term-mod 的一些变量

C-up, C-down, C-left 和 C-right 快捷键保持跟在 xterm 下一样。这样在 readline 模式下，表现得像 forward-word 一样

#### 14.10.8 Customizable variable 'query-replace-from-to-separator'

now doesn't propertize the string value of the separator. Instead, text properties are added by 'query-replace-read-from'. Additionally, the new nil value restores pre-24.5 behavior of not providing replacement pairs via the history.

#### 14.10.9 下面一些老的函数、变量、和样式 (face) 被移除

1. make-variable-frame-local 不再是 frame-local.
2. 在 subr.el 库里下列被移除: window-dot, set-window-dot, read-input, show-buffer, eval-current-buffer, string-to-int.
3. icomplete-prospects-length

4. 所有默认值为 FOO 的 default-FOO 变量，使用 default-value 和 setq-default 分别去获取和修改 FOO 所有被移除的变量如下：default-mode-line-format, default-header-line-format, default-line-spacing, default-abbrev-mode, default-ctl-arrow, default-truncate-lines, default-left-margin, default-tab-width, default-case-fold-search, default-left-margin-width, default-right-margin-width, default-left-fringe-width, default-right-fringe-width, default-fringes-outside-margins, default-scroll-bar-width, default-vertical-scroll-bar, default-indicate-empty-lines, default-indicate-buffer-boundaries, default-fringe-indicator-alist, default-fringe-cursor-alist, default-scroll-up-aggressively, default-scroll-down-aggressively, default-fill-column, default-cursor-type, default-cursor-in-non-selected-windows, default-buffer-file-coding-system, default-major-mode, and default-enable-multibyte-characters.

5. 许多在 22.1 版本被废弃的样式

#### 14.10.10 变量 text-quoting-style 改为可定制选项

它控制是否以及怎样在消息和帮助界面显示 ASCII 的引用符。从 Emacs 25 开始它的值和语义没有变化。在特定的情况下，当这个变量的值为 grave 时，所有的引用显示按原样输出。

#### 14.10.11 函数 check-declare-file 和 check-declare-directory 的变化

它们将输出更加紧凑的诊断输出，辅助函数 check-declare-errmsg 已经被移除。

#### 14.10.12 正则表达集合 [:blank:] 的变化

当前可匹配统一编码的空格（如在 UTS 技术标准 18 里定义的那样），如果你仅仅想匹配空格和 tab 符，请使用 [ \t]。

#### 14.10.13 min 和 max 函数不在对结果进行截断 (round)

老版本的 Emacs，当参数中含有浮点类型时，这些函数返回一个浮点的值（有时这些数值是不正确的），例如，在 64 位的机器里 (max 1e16 10000000000000001) 返回的是第二个参数，而不是第一个。

#### 14.10.14 变量 old-style-backquotes 的变化

这个变量已经声明为内部使用，同时重命名为 lread–old-style-backquotes。任何用户代码不应用使用这个变量

### 14.10.15 default-file-name-coding-system 默认指定的编码系统不处理 CRLF

例如, 默认由原来的 utf-8 改为 utf-8-unix。在这个变化之前, Emacs 有时会对文件名中含有这些控制字符做错误处理

### 14.10.16 'file-attributes', 'file-symlink-p' and 'make-symbolic-link' no

longer quietly mutate the target of a local symbolic link, so that Emacs can access and copy them reliably regardless of their contents. The following changes are involved.

1. 'file-attributes' and 'file-symlink-p' no longer prepend "/:" to symbolic links whose targets begin with "/" and contain ":". For example, if a symbolic link "x" has a target "*y:z:", '(file-symlink-p "x")' now returns "/y:z:" rather than* ":/y:z:".

2. 'make-symbolic-link' no longer looks for file name handlers of target when creating a symbolic link. For example, '(make-symbolic-link "/y:z:" "x")' now creates a symbolic link to "/y:z:" instead of failing.

3. 'make-symbolic-link' removes the remote part of a link target if target and newname have the same remote part. For example, '(make-symbolic-link "/x:y:a" "/x:y:b")' creates a link with the literal string "a"; and '(make-symbolic-link "/x:y:a" "/x:z:b")' creates a link with the literal string "/x:y:a" instead of failing.

4. 'make-symbolic-link' now expands a link target with leading "~" only when the optional third arg is an integer, as when invoked interactively. For example, '(make-symbolic-link "~y" "x")' now creates a link with target the literal string "~y"; to get the old behavior, use '(make-symbolic-link (expand-file-name "~y") "x")'. To avoid this expansion in interactive use, you can now prefix the link target with *":". For example, '(make-symbolic-link* ":~y" "x" 1)' now creates a link to literal "~y".

#### 14.10.17 file-truename 在一个符号链接指定为远程文件语法时返回一个引用的文件名

#### 14.10.18 模块函数的实现有轻微的变化

作为特殊考虑，函数 internal–module-call 已经被删除，对此有依赖的系统可能会导致异常

#### 14.10.19 write-region 的 LOCKNAME 参数会被传递给文件名处理器

#### 14.10.20 构建最新的 GTK+ 的变化

当构建最新版本的 GTK+ 时, Emacs 使用 $\text{gtk}_{\text{window}}\text{move}$ 来移除 frames，同时忽略变量 x-gtk-use-window-move 的值，这个变量已经被废弃。

#### 14.10.21 一些文件创建或者重命名的变化

这些函数在处理目标参数为目录类型时有一些变化，如，当在 GNU 或者类 POSIX 体系里其参数以“/”结尾。当这些函数的目地参数 D 为已经存在的目录时，其目的是在这个目录里操作。举例 (rename-file ”e” ”f/”) 结果为’f/e’. Although this formerly happened sometimes even when D was not a directory name, as in (rename-file ”e” ”f”) where ’f’ happened to be a directory, the old behavior often contradicted the documentation and had inherent races that led to security holes. A call like (rename-file C D) that used the old, undocumented behavior can be written as (rename-file C (file-name-as-directory D)), a formulation portable to both older and newer versions of Emacs. Affected functions include ’add-name-to-file’, ’copy-directory’, ’copy-file’, ’format-write-file’, ’gnus-copy-file’, ’make-symbolic-link’, ’rename-file’, ’thumbs-rename-images’, and ’write-file’.

#### 14.10.22 overlays-at 返回的结果列表按优先级降序排列

老版本的文档说明里指出当函数的第二个参数不为 nil 里，返回的结果为降序，但实际的代码返回的却是升序的结果。这个版本修复了这个问题，返回的结果正如文档中说明的那样按降序。

#### 14.10.23 format 的优化

format 避免了在大多数据情况下分配一个新的字符串。format 在之前的文档中返回一个新分配的字符串, 但这文档说明不是很准确，由于 **(eq x (format x))** 在 x 是一个空字符串时，返回的是 t。format 文档中不再提及

返回一个新分配的字符串，当前的实现如在文档中提及的那样避免了大部分情况下字符串的无用拷贝，如 **(format "foo")** 和 **(format "%s" "foo")** 。

#### 14.10.24 函数 eldoc-message 可接收一个参数

在对多参数的程序调用之前，应该对其进行 format 操作。尽管不鼓励这样做，对 ElDoc 的支持，通过设置 eldoc-documentation-function 函数，而不是直接调用 eldoc-message。

#### 14.10.25 对 &rest 或者 &optional 不正确的使用作为一种错误类型处理

例如 &optional 后面没有一个变量，或者对 &optional 多次使用：

```
(defun foo (&optional &rest x))
(defun bar (&optional &optional x))
```

老版本里，Emacs 只是忽略额外的关键字，或者在某种情况下给出一个不正确的结果。

#### 14.10.26 pinentry.el 库被移除

这个库以及对 GnuPG 和 pinentry 的修改目的是使得 Emacs 可采用 GnuPG2.0 来实现密码的输入。然而，这次修改变动只针对 GnuPG 大于 2.1（包含）版本做了实现，对 GnuPG2.0 版本不兼容。对于 GnuPG2.1 及后续版本，不再需要 pinentry.el。因此这个库成了无用库，被我们移除了。GnuPG 2.0 不再被上游项目支持。为了兼容这个变化，你需要设置 epa-pinentry-mode 为 loopback，或者为默认值 nil。从你的 gpg-agent.conf 配置文件里删除 allow-emacs-pinentry（一般在 ~/.gnupg 目录)。注意之前提到的，通过 Minibuffer 输入密码的安全性比其他图形界面的要差。但是，现在看来是差别不大：read-password 函数阻止了通过日志的方式泄漏密码的可能。Emacs 仍没有通过使用安全内存来保护密码机制，但这种攻击在当前计算机系统（指没有实现内存交互）是不可能。

## 14.11 Emacs 26.1 中 Lisp 的变化

### 14.11.1 函数 assoc 接受一个可选的第三位参数 TESTFN，当这个参数不为空时，其用于比较而不是相等

### 14.11.2 在 alist-get、map-elt 和 map-put 中新增可选参数 TESTFN，当不为 nil 时，这个参数指定一个具体的比较函数，而不是用默认的 assq 和 eql

### 14.11.3 新增函数 seq-set-equal-p 用于检查 SEQUENCE1 和 SEQUENCE2 是否含有相同元素（不考虑顺序）

### 14.11.4 新增函数 mapbacktrace，用于将一个函数应用于当前椎栈的所有 frames 中

### 14.11.5 新增函数 file-name-case-insensitive-p 测试给定的一个文件是否在一个不区分大小写的文件系统里

### 14.11.6 为 file-attributes 添加多个寄存器返回值

他们是 file-attribute-type, file-attribute-link-number, file-attribute-user-id, file-attribute-group-id, file-attribute-access-time, file-attribute-modification-time, file-attribute-status-change-time, file-attribute-size, file-attribute-modes, file-attribute-inode-number, file-attribute-device-number 和 file-attribute-collect

### 14.11.7 新增函数 buffer-hash 用于快速计算一个 Buffer 内容的非连续 (non-consing) 的哈希值

### 14.11.8 interrupt-process 的变化

interrupt-process 采用列表 interrupt-process-functions 来实现传递 SIGINT 信号决定哪个函数会被调用。这可使 Tramp 实现向远程异步进程发送 SIGINT 信号。老的实现方式被迁移到 internal-default-interrupt-process。

#### 14.11.9 函数 read-multiple-choice 弹出多选择提问窗，提供一种手工方式来显示帮助信息

#### 14.11.10 comment-indent-function 的值可返回一个 cons 用于指定格式化范围

#### 14.11.11 函数 make-temp-file 新增加一个可选参数 TEXT

#### 14.11.12 新增函数 define-symbol-prop

#### 14.11.13 新增函数 secure-hash-algorithms，其返回 secure-hash 支持的算法列表，详情请参考 Emacs 手册

#### 14.11.14 Emacs 开放了 GnuTLS 的加密接口 (API)

对应的函数为 gnutls-macs 和 gnutls-hash-mac; gnutls-digests 和 gnutls-hash-digest; gnutls-ciphers 和 gnutls-symmetric-encrypt 和 gnutls-symmetric-decrypt

#### 14.11.15 函数 gnutls-available-p 返回在 Emacs 中支持的 GnuTLS 库的功能列表

#### 14.11.16 Emacs 可通过新函数 make-record、record 和 recordp 来实现用户定义类型的记录。

记录用于描述 cl-defstruct 和 defclass 的实例。例如当你的代码里定义了新的记录类型，使用 package-naming 去命名这些类型，从而且避免类型冲突。

**14.11.17** save-some-buffers 采用 save-some-buffers-default-predicate 决定哪些 buffer 会被确定询问, 当 PRED 参数为 nil 时, save-some-buffers-default-predicate 也为 nil, 表示询问所有正在访问的 Buffer。

**14.11.18** string-(to|as|make)-(uni|multi)byte 被声明为废弃

**14.11.19** 新变量 while-no-input-ignore-events 用于设置哪些特定的 while-no-input 事件可以被忽略，它的值是一个 symbol 的列表

**14.11.20** 新增函数 undo-amalgamate-change-group 去除两个状态 undo 之间的边界

**14.11.21** 新变量 definition-prefixes 是有相应文件定义的哈希表的映射前缀，可用于获取那些还没加载的定义 (如 C-h f)。

**14.11.22** 新变量 syntax-ppss-table 用于在 syntax-ppss 控制 syntax-table

**14.11.23** define-derived-mode 可指定一个:after-hook 形式，当新的 mode 的 hook 被执行后它将被求值，可用于在 mode 启动时, mode-hook 引起配置发生变化的情况

**14.11.24** 自动加载产生的文件名不包含时间戳,可通过设置变量 autoload-timestamps 的值不是 nil 产生时间戳

**14.11.25** gnutls-boot 接受一个参数:complete-negotiation，表示即使在非阻塞的 socket 链接里协商应该完成

**14.11.26** 新增变量 flyspell-sort-corrections-function,它是 flyspell.el 库里的变量，用于排序修正

**14.11.27** 新增命令 fortune-message 用于在 echo 区域显示名言警句

**14.11.28** 新增函数 func-arity 返回任意函数的参数列表信息，用于替代之前的 subr-arity

**14.11.29** 新增函数 region-bounds 用于交互情境，主要用于提供区域边界 ( 超过一个的长方形区域 )，只需要传递一个参数，取代之前的两个参数 (region-beginning 和 region-end)

**14.11.30** parse-partial-sexp 函数的返回列表新增一种元素，元素 10。当最后一个被扫描的字符可能是两个结构字符的第一个字符时, 如注释符，这个元素的值就是最一个字符的语法值

**14.11.31** parse-partial-sexp 函数返回列表的第 9 个元素作为固定返回元素，可用于 Lisp 编程。它的值是括号所含位置的列表，从最外层开始。



**14.11.32** read-color 在将颜色作为背景色时将显示颜色名

**14.11.33** 函数 redirect-debugging-output 可在非 GNU/Linux 平台上运行

**14.11.34** 新增函数 string-version-lessp 比较两个字符串, 这里会把不连续的数字解析成数值,然后进行比较例如"foo2.png" 比"foo12.png"

#### 14.11.42 函数 sxhash 被重命名为 sxhash-equal，为了兼容 sxhash 成为 sxhash-equal 别名

#### 14.11.43 make-hash-table 默认的刷新阈值从 0.8 改成 0.8125，避免舍入的小问题

#### 14.11.44 新增函数 add-variable-watcher 用于当一个变量的值发生变化时的函数调用。用于实现新的调试命令 debug-on-variable-change

#### 14.11.45 新变量 print-escape-control-characters 改变 prin1 和 print 输出控制字符为反斜杠

#### 14.11.46 时间转换函数支持时间区间规划参数

其可为 OFFSET 或者列表 (OFFSET ABBR) 整型的 OFFSET 是相对于 UT(Universal Time) 时间的东向偏移 ( 单位秒 )。字符串 ABBR 是时间区间的简写。受影响的函数有 current-time-string, current-time-zone, decode-time,format-time-string 和 set-time-zone-rule.

#### 14.11.47 format-time-string 函数支持%q，用于日历的季度

#### 14.11.48 新的内置函数 mapcan，避免不必要的链接 (consing) 和 gc 操作

#### 14.11.49 car 和 cdr，以及 cXXXr 和 cXXXXr 的结构已经集成为 Elisp 一部分

#### 14.11.50 gensym 已经集成进 Elisp 一部分

#### 14.11.51 基础列表函数，如 length 和 member，做了相应优化成列表循环

#### 14.11.52 新增加函数 make-nearby-temp-file 和 temporary-file-directory，用于在远端或者所在目录创建临时文件

#### 14.11.53 在 GNU 平台操作一个本地文件时，同时另一个进程改变文件系统时，file-attributes 函数不再进行死循环；如果不是 GNU 的平台遇到同样情况，file-attributes 尝试去检测死循环，同时返回 nil

#### 14.11.54 新函数 file-local-name 可用于指定远端进程的参数

#### 14.11.55 新增加函数 file-name-quote、file-name-unquote 和 file-name-quoted-p

用于对文件名加前缀"/:" 进行引用或者取消引用

#### 14.11.56 新增错误类型 file-missing 作为 file-error 子类型

如果操作不存在的文件，将采用这种错误类型取代之前的 file-error 这种错误类型

#### 14.11.57 delete-directory 函数的变化

当递归操作进行时，同时有其他进程在执行 delete-directory 前删除了目录或者文件时，不再抛错误异常。

#### 14.11.58 新的错误类型 user-search-failed，与 search-failed 类似，但像 user-error 一样禁止高度机制

#### 14.11.59 函数 line-number-at-pos 的变化

函数 line-number-at-pos 支持传递第二个可选参数 absolute。如果这个参数为 nil, 默认将返回包含收缩 ( potential narrowing ) 的行号。如果这个参数不为 nil，它将不考虑收缩情况，返回绝对的行号。

#### 14.11.60 函数 color-distance 的调整

函数 color-distance 在新版本支持第二个可选参数 **metric** 。当这个参数不为 nil 时，它将接收两个参数 ( 两个颜色值 ) 返回一个距离值。

#### 14.11.61 Frame 和 Window 操作

1. 调整 frame 大小方式的改变 请使用 **window-size-change-functions** 代替老的 **window-configuration-change-hook** 。
2. Frame 大小是否被调整 判断 Frame 是否被调整可使用新函数 **frame-size-changed-p** 。这里的大小是否被调整过是从上一次调用 **window-size-change-functions** 算起。
3. 函数 frame-geometry 也会返回 frame 的外边界宽度
4. 新增部分 frame 相关参数及老参数一些语义变化
   (a) 'z-group' 将 frame 置于其他之上或之下
   (b) 'min-width' 和'min-height' 用于指定 frame 的绝对最小范围
   (c) 'parent-frame' 使得一个 frame 成为另一个 frame 的子 frame。详情请参数 Elisp 手册的 Elisp 子 frames 部分
   (d) 'delete-before' 在删除一个 frame 前触发删除另一个
   (e) 'mouse-wheel-frame' 指定另一个 frame 的窗口可滚动
   (f) 'no-other-frame' 新增'next-frame' 和'previous-frame' 跳过当前 frame
   (g) 'skip-taskbar' 从任务条 (taskbar) 中删除 frame 的图标, 同时, 可通过'Alt-<TAB>' 跳过这个 frame
   (h) 'no-focus-on-map' 避免一个规划过的 frame 获得输入选中
   (i) 'no-accept-focus' 表示 frame 不希望通过鼠标获得输入选中
   (j) 'undecorated' 从 frame 中移除窗口管理样式
   (k) 'override-redirect' 让窗口管理忽视当前 frame
   (l) 'width' and 'height' 支持按比例指定和按像素值
   (m) 'left' and 'top' 支持按比例指定
   (n) 'keep-ratio' 当父 frame 被调整时, 保持子 frame 的大小和位置不变
   (o) no-special-glyphs 在 frame 中将覆盖截断和连续符号的显示

(p) auto-hide-function 和 minibuffer-exit 分别用于处理自动隐藏 frame 和从 minibuffer 中退出

(q) fit-frame-to-buffer-margins 和 fit-frame-to-buffer-sizes 用于处理 frame 对其 buffer 的大小适应

(r) drag-internal-border, drag-with-header-line, drag-with-mode-line, snap-width, top-visible and bottom-visible 允许通过拖拽来调整 frame 的大小

(s) minibuffer 当初始值被指定为 nil 时，设置为默认的 minibuffer 窗口

5. 新增函数 frame-list-z-order 返回按 z(栈) 顺序的 frame 列表

6. 函数'x-focus-frame' 不激活它的 frame 变成可选项

7. 变量 focus-follows-mouse 增加第三个有意义的值 auto-raise 这个值用于当鼠标进入 frame 时，标明窗口管理自动聚焦这个 frame。

8. 新增函数 frame-restack，用于调整 frame 的层级关系

9. 新增 internal-border 用于指定 frame 的外部边界背景

10. select-window 函数 NORECORD 参数增加 **mark-for-redisplay** 选项 这个选项类似其他不是 nil 的值，但它标记窗口（WINDOW）重新显示

11. 正式支持对边窗 (side windows) 显示操作函数 display-buffer-in-side-window 可将 buffer 显示在边空上。更从详情请参考 ELisp 手册里的边窗口部分。

12. 支持原子窗口 更从详情请参考 ELisp 手册里的原子窗口部分

13. 新增 display-buffer 列表项 window-parameters 分配给要展示 buffer 的窗口参数

14. 新增函数 display-buffer-reuse-mode-window 这个函数是一个交互函数，适用在 display-buffer-alist。例如，当打开一个帮助手册，为了避免再次创建一个已经存在的窗口，可用这个机制完成。

```
(add-to-list 'display-buffer-alist
    '("\\`\\*Man .*\\*\\'" .
     (display-buffer-reuse-mode-window
 (inhibit-same-window . nil)
     (mode . Man-mode))))
```

15. 新窗口参数 no-delete-other-windows 防止窗口被 delete-other-windows 命令删除掉

16. 新窗口参数 mode-line-format 和'header-line-format 这两个参数允许 buffer 本地 (buffer-local) 修改进行覆盖。

17. 新增命令 window-swap-states 这个命令用于交换两个打开的 windows 的状态 (选中和被选中)，这个方法常用于当前只有两个窗口时，交换两个窗口的内容。

18. 新增 window-pixel-width-before-size-change 和 window-pixel-height-before-size-change 函数 用于当 window-size-change-functions 运行里，检测哪个窗口大小发生了变化

19. 新增函数 window-lines-pixel-dimensions 用于返回一个窗口文本行的像素大小

20. 新增函数 window-largest-empty-rectangle 用于返回在窗口里没有文本区域最大矩形大小

21. 函数'mouse-autoselect-window' 语义微调 详情参考 Elisp 手册里的"(elisp) Mouse Window Auto-selection" 函数。

22. 'select-frame-by-name' 在当前的显示器里无法找到一个匹配的 frame，可返回一个在其他显示器上的 frame

#### 14.11.62 'tcl-auto-fill-mode' 已经声明废弃

这个函数的功能可通过设置 **comment-auto-fill-only-comments** 来实现。

#### 14.11.63 新增 pcase 模式'rx' 用于匹配 rx 风格的正则表达式

详情请参考'rx–pcase-macroexpander' 函数文档。

#### 14.11.64 新增区域二次选择函数

新增加函数'secondary-selection-to-region' 和'secondary-selection-from-region'，满足用户再次设置区间的开始和结束。

#### 14.11.65 新增函数'lgstring-remove-glyph'

这个函数于用修改底层布局引擎 ( 如 m17n-flt, uniscribe ) 返回的 gstring。

## 14.12 Emacs 26.1 在一些特殊模式和包中的变化

### 14.12.1 在 Windows7& 以上系统对快捷键的捕获性能更好

新的键盘钩子代码 (hooking code) 更加准捕获系统快捷键如'Win-*' 和'Alt-TAB', 通过这种方式, Emacs 比系统更早地获取按键事件。这使得'w32-register-hot-key' 函数功能违背了所有 Windows7 开始的原意功能。Windows NT 及之后版本可以注册任务快捷键的组合。( 在 Windows 9X, 之前的限制, 在 Emacs 的手册里已经说明了依然需要申请 )。

### 14.12.2 'convert-standard-filename' 函数的变化

在 MS-Windows 系统中'convert-standard-filename' 函数不再将斜线 (/) 替换成反斜线 (\)。*老版本在 MS-Windows 系统中这个函数将文件名里的斜线 (/)* 强制转成反斜线 (\)。因为不再这样做, 如果你 Lisp 代码里使用了 **convert-standard-filename** 这个函数去做这个变化, 你需要自己手工处理一下这块的代码。如按照如下方式 :

```
(let ((start 0))
    (while (string-match "/" file-name start)
    (aset file-name (match-beginning 0) ?\\)
    (setq start (match-end 0))))
```

### 14.12.3 MS-Windows 平台下的 GUI 会话像 Posix 平台一样将被当作 SIGINT

Ctrl-C (SIGINT) 信号在 MS-Windows 平台 GUI 的效果与 Posix 平台一样 : Eamcs 保存会话然后退出。譬如, 如果你在 Windows shell 里启动 emacs.exe 程序, 然后在 shell 窗口里按下 Ctrl-C, 也会产生同样效果。

### 14.12.4 Windows XP 及以后系统'signal-process' 支持 SIGTRAP

在 Windows 系统下对'kill' 的仿真映射到 SIGTRAP 信号去调用'DebugBreakProcess' 接口。这会导致检索程序终止执行, 然后把控制权交给调试者 (debugger)。如果没有调试者绑定到搜索程序, 这个调用将被忽略。这跟 POSIX 系统的默认行为形成对比, POSIX 系统是会终止检索程序的执行 ( 终止前会做一次 core dump )。

### 14.12.5 macOS 系统里的一些小优化

1. 修复在 macOS 系统里'set-mouse-position' 和'set-mouse-absolute-pixel-position' 命令不生效问题 ;

2. 在 macOS 系统，从命令行中也可打开 Emacs GUI 程序；
3. macOS 10.9+ 'ns-appearance' 和'ns-transparent-titlebar' 改变 frame 的显示；
4. macOS 10.8+ 系统里使用'ns-use-thin-smoothing' 开启瘦字段更加柔和；
5. Darwin 系统'process-attributes' 返回更多信息；
6. macOS 10.7+ 鼠标方向键和触摸板的滚动效果表现更加与原生系统相同，新增变量'ns-mwheel-line-height' 、'ns-use-mwheel-acceleration' 和'ns-use-mwheel-momentum' 用来定制化这些行为。

### 14.13 GNU Emacs 协议声明原文