文件

aborn

2018-04-08

Contents

---

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。

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。

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

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 list, 它包含一个文件名和数据长度信息。如果文件不存在，则会抛出错误异常!

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。

1. 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。

2 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

1. 文件是否存在

```
file-exists-p  lename
```

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

2.1 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。

1. 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。

3 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。

3.1 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

4 文件名

下面介绍一些与文件名操作有关的函数

```
file-name-directory  lename
```

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。

4.1 文件名扩展

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"
```