elisp的简单介绍

elisp的简介

- elisp是lisp的一个方言
- 特色是优美和晦涩，当然，更出名的是括号。。。
- emacs利用elisp作为上层抽象
- 操作emacs就是调用elisp函数

elisp表达式

- 原子
  - 符号，对象（数字或者字符串，包括函数）
  - 序对，表（包括空表），树
  - 以及他们的嵌套
- 求值
  - 第一个原子做动词，先求值第一个原子，得到一个对象
  - 根据第一个原子的特性决定正则序和应用序
  - 应用序的先对每个后续原子求值，再调用第一个原子对应的对象
  - 正则序直接交给第一个原子对应对象处理

一个例子：弹出console

```
(defun popup-term ()
  (interactive)
  (apply 'start-process "terminal" nil popup-terminal-command)
  )
```

换成C的说法：

```
int popup-term()
{
    interactive();
    start-process("terminal", NULL, popup-terminal-command);
    return 0;
}
```

其中popup-terminal-command是一个动态列表，照理C中不存在这东西。

似乎python更像一点

```
(defun popup-term ()
  (interactive)
  (apply 'start-process "terminal" nil popup-terminal-command)
  )
```

换成python：

```
def popup-term():
    interactive()
    start-process("terminal", None, *popup-terminal-command)
```

不给力呀

缺了两句:
(setq popup-terminal-command '("x-terminal-emulator"))
定义弹出窗口的程序 。
(global-set-key [(control c) (s)] 'popup-term)
定义绑定到什么键上。
现在，按下C-c s？

给力，再来一个例子

```
(defun dired-open-file (&optional arg)
  (interactive)
  (apply 'start-process "dired-open" nil
         (append (split-string
                  (read-shell-command
                   "command: " (dired-guess-cmd (dired-get-
filename))))
          (list (dired-get-filename)))))
(defun dired-copy-from (&optional arg)
  (interactive)
  (let ((source-path (read-file-name "filepath: ")))
    (copy-file source-path (file-name-nondirectory source-path))))
```

...再来？

```
(defun dired-rename-from (&optional arg)
  (interactive)
  (let ((source-path (read-file-name "filepath: ")))
    (rename-file source-path (file-name-nondirectory source-
path))))
(add-hook 'dired-mode-hook
          (lambda ()
            (define-key dired-mode-map "b" 'dired-open-file)
            (define-key dired-mode-map "c" 'dired-copy-from)
            (define-key dired-mode-map "r" 'dired-rename-from)
            (define-key dired-mode-map [(control c) (g)] 'dired-
etags-tables)
            ))
```

dired-open-file

```
(defun dired-open-file (&optional arg)
  (interactive)
  (apply 'start-process "dired-open" nil
         (append (split-string
                  (read-shell-command
                   "command: " (dired-guess-cmd (dired-get-
filename))))
          (list (dired-get-filename)))))
```

```
def dired-open-file (arg, *optional):
    interactive()
    default-cmd = dired-guess-cmd(dired-get-filename())
    cmd = read-shell-command("command:", default-cmd).split()
    cmd.append(dired-get-filename())
    start-process("dired-open", None, cmd)
```

dired-etags-tables

```
(defun dired-etags-tables ()
  (interactive)
  (let ((etags-path (expand-file-name (read-directory-name
"etags path:")))
       (python-command (expand-file-name "~/.emacs.
d/gen_etags.py")))
    (call-process "python" nil t nil python-command etags-path)))

def dired-etags-tables():
    interactive()
    etags-path = expand-file-name(read-directory-name("etags
path:"))
    python-command = expand-file-name("~/.emacs.
d/gen_etags.py")
    call-process("python", None, True, None, python-command
etags-path)
```

dired-mode-hook

```
(add-hook 'dired-mode-hook
          (lambda ()
            (define-key dired-mode-map "b" 'dired-open-file)
            (define-key dired-mode-map "c" 'dired-copy-from)
            (define-key dired-mode-map "r" 'dired-rename-from)
            (define-key dired-mode-map [(control c) (g)] 'dired-
etags-tables)
            ))
```

匿名函数, 加入加载钩子。
局部绑定。