Rock Your Emacs

Ben Sturmfels
ben@sturm.com.au
www.sturm.com.au

LibrePlanet 2015, 22 March 2015

Section 1

Introduction

Hello from Free Software Melbourne

Overview

1. There's a Lisp in my Emacs (40 mins)
2. All of the things (20 mins)

Slides and code:
sturm.com.au/2015/talks/rock-your-emacs-libreplanet/

Section 2

There's a Lisp in my Emacs

The problem

From: parents@sturm.com.au
Subject: Hope you're ok.

Hi Ben,

We've left you lots of phone messages,
but you haven't called back. Are you ok?

Love Mum & Dad

Exercise 1: Controlling Emacs with Lisp

```
;; Exercise 1: Type this, then run
;; M-x eval-last-sexp (C-x C-e).
(compose-mail "parents@sturm.com.au" "I'm ok.")

;; What is it? What does it do?

;; What would that be in C, PHP, Python, Perl, Ruby etc?
```

Exercise 2: Make Emacs type some text

```
;; Exercise 2: Run M-x eval-expression (M-:),
;; type this, then RET.
(insert "Hi Mum & Dad, I'm ok.")

;; What happens if I misspell insert?

;; Try Eldoc and Show Paren minor modes.
;; M-x eldoc-mode, M-x show-paren-mode
```

Exercise 3: Re-use our code

```
;; Exercise 3: Make this into a reusable function
;; called "mail-parents" and evaluate it (C-x C-e).
(compose-mail "parents@sturm.com.au" "I'm ok.")
(insert "Hi Mum & Dad, I'm ok.\n\n")
(insert "Love " user-login-name)

;; Run M-x describe-function mail-parents.
;; It's not very useful. Can we fix it?

;; What's \n and what's user-login-name?

;; What happens if I restart Emacs?
```

Exercise 4: Adding to the user interface

```
;; Exercise 4: Make this function available as
;; the command M-x mail-parents.
(defun mail-parents ()
  "Compose a reply to worried parents."
  (compose-mail "parents@sturm.com.au" "I'm ok.")
  (insert "Hi Mum & Dad, I'm ok.\n\n")
  (insert "Love " user-login-name))

;; Commands? Functions? What's the difference?
```

Exercise 5: Bind our command to a key

```
;; Exercise 5: Bind our command to the key C-c p.
(global-set-key (kbd "C-c p") 'mail-parents)

;; Great, everyone's happy right?
```

More problems

```
From: parents@sturm.com.au
Subject: Re: Hope you're ok.

Hi Ben,

All your emails look the same. Is this
an auto-reply?

Love Mum & Dad
```

Exercise 6: Add some random love

```
;; Exercise 6: Write a function to return a
;; kiss or a hug using "random".
(defun kiss-or-hug ()
  "Return a kiss ‘x’ or a hug ‘o’."
  ;; Your code goes here.
)
```

Exercise 7: Pull it all together

```
;; Exercise 7: Modify mail-parents to add kisses/hugs.
(defun mail-parents ()
  "Compose a reply to worried parents."
  (interactive)
  (compose-mail "parents@sturm.com.au" "I'm ok.")
  (insert "Hi Mum & Dad, I'm ok.\n\n")
  (insert "Love " user-login-name))

(defun kiss-or-hug ()
  "Return a kiss x or a hug o."
  (if (equal (random 2) 0)
      "x"
    "o"))
```

Exercise 8: Position the cursor

```
;; Exercise 8: Put the cursor in the right place
;; to add our own words. Hint: M-x describe-key C-p
(defun mail-parents ()
  "Compose a reply to worried parents."
  (interactive)
  (compose-mail "parents@sturm.com.au" "I'm ok.")
  (insert "Hi Mum & Dad, I'm ok.\n\n")
  (insert "Love " user-login-name " ")
  (dotimes (i 10)
    (insert (kiss-or-hug))))
```

Exercise 9: Save it for later

```
;; Exercise 9: Copy everything into your .emacs file.
(defun mail-parents ()
  "Compose a reply to worried parents."
  (interactive)
  (compose-mail "parents@sturm.com.au" "I'm ok.")
  (insert "Hi Mum & Dad, I'm ok.\n\n")
  (insert "Love " user-login-name " ")
  (dotimes (i 10) (insert (kiss-or-hug)))
  (forward-line -1))
(defun kiss-or-hug ()
  "Return a kiss x or a hug o."
  (if (equal (random 2) 0) "x" "o"))
(global-set-key (kbd "C-c p") 'mail-parents)
```

Summary

- seen some Lisp, looks different, takes time
- we've written useful code to drive Emacs
- our changes are first-class parts of Emacs
- menus, toolbar and keys just interactive Lisp functions
- helped ourselves with amazing documentation system
- call your parents, they miss you!

Section 3

All of the things

Freedom to study and modify

When source is installed, look for the hyperlink in `M-x describe-function` and `M-x describe-variable`.

- eg. Lisp source for function `compose-mail`
- eg. C source for function `insert`

Install on Trisquel 7:

```
$ sudo apt-get install emacs24-el
$ apt-get source emacs24
```

Navigating help

Don't memorise, use the awesome help.
Type: `M-x help` (`C-h ?`)

- `M-x describe-function` (`C-h f`)
- `M-x describe-variable` (`C-h v`)
- `M-x info` (`C-h i`)
- `M-x apropos-documentation` (`C-h d`)
- `M-x describe-key` (`C-h k`)

The Emacs Lisp Reference Manual

Type: `M-x info m Elisp`

- easy to read, easy to search, useful examples
- useful keys `u` for "up", `l` for "last"
- starting to wish everything was in Info format

Install on Trisquel 7:

```
$sudo apt-get install emacs24-common-non-dfsg
```

Introduction to Emacs Lisp

Type: `M-x info m Emacs Lisp Intro`

- great first few chapters, gets a little heavy later
- skip to "Emacs Initialization" and "Debugging"
- you *don't* have to finish this before you can program Emacs

Changing global variables

Tweaking Emacs is often as easy as changing a setting. I use `customize` to find the variables, then manually set them in my `.emacs`.

```
;; Change some global variables.
(setq user-mail-address "ben@sturm.com.au"
      erc-nic "sturm"
      default-major-mode 'org-mode)

;; Buffer-local variables are special; use setq-default
;; to change the global default.
(setq-default fill-column 80)
```

Customizing Emacs with hooks

- predefined places for your extensions
- eg. starting Emacs Lisp Mode, runs all functions on `emacs-lisp-mode-hook`, opening a file runs `find-file-hook`
- hooks are everywhere, see `Standard Hooks` in manual

```
;; Load Eldoc and Show Paren modes when editing
;; Emacs Lisp.
(add-hook 'emacs-lisp-mode-hook
          (lambda ()
            (eldoc-mode)
            (show-paren-mode)))
```

Mode-specific key bindings

- `define-key`: you need to specify the keymap
- or use `local-set-key` in a mode hook

```
;; Shortcut key to insert current date in Python Mode.
(require 'python)
(define-key python-mode-map (kbd "C-c .") 'insert-date)

;; Or use a mode hook.
(add-hook 'python-mode-hook
          (lambda ()
            (local-set-key (kbd "C-c .") 'insert-date)))
```

Loading external code

In your `.emacs`, you can use `load-file`, `load`, or `require` to load in external files.

```
;; Load a file explicitly:
(load-file "~/.emacs.d/lisp/junk.el")

;; Or update load-path and don't provide path to file.
(add-to-list 'load-path "~/.emacs.d/lisp")
(load "junk")

;; Library code often uses provides/require to load named
;; features only when required and avoid reloading.
(require 'junk)
```

Major modes for editing Emacs Lisp

Emacs Lisp mode the default for editing `.el` files
(`emacs-lisp-mode`)

Inferior Emacs Lisp mode for shell/REPL interface (`ielm`)

```
*** Welcome to IELM ***  Type (describe-mode) for help.
ELISP> (+ 1 1)
2
```

Debugging with Edebug

Instrument function for debugging with `M-x edebug-defun` or use `Emacs-Lisp` menu.

- re-eval function to clear debugging
- common debugger features, inc. breakpoints
- see `Edebug` in manual

Writing great docs with Checkdoc

Inside a Lisp file open, type: `M-x checkdoc`

- interactive coaching to write thorough documentation
- nicely formatted header documentation

```
There is no first line summary! Add one? (y or n) y
Summary:
```

Unit test all the things

Type: `M-x info m ERT`

```
;; Run tests with M-x ert.
(ert-deftest test-addition ()
  (should (equal (+ 1 2) 3)))
```

Macros

- transform Lisp code
- you don't need them yet

Other references

Emacs Wiki, eg. ElispCookbook	http://www.emacswiki.org/emacs/ElispCookbook
Read a book Lisp book	*Land of Lisp* by Conrad Barski — a fun way to learn more about the spirit of Lisp
Screencasts	Emacs Bites and Emacs Rocks - Extending Emacs
IRC	#emacs on Freenode network

Common Lisp functions

Especially useful if you're reading a Common Lisp book.
Type: `M-x info m CL`

```
;; Use loop macro to sum integers 5 to 10.
(require 'cl-lib)
(cl-loop for i
         from 5
         to 10
         sum i)

;; Select only odd numbers from a list.
(cl-remove-if-not 'cl-oddp '(1 2 3 4))
```

A wonderful place

Emacs is a wonderful place to program Emacs Lisp!

Summary

- now you know what those paretheses mean
- we've looked at the remarkable ways Emacs is designed to be extended
- you have a .emacs file with code you've written
- the best way to learn is to start coding, happy hacking!

Ben Sturmfels
ben@sturm.com.au
www.sturm.com.au

Section 4

Homework

Homework

Write an function called `insert-heart` that inserts a heart symbol. Use your Character Map program to copy the symbol into Emacs. Make your function into an interactive command and bind it to `C-c h`. Don't forget to add documentation.
Useful functions: `insert`, `interactive`, `global-set-key`, `kbd`.

Homework (harder)

Write a function `insert-license` that inserts a license notice at the top of the current file. Write a function `check-for-license` that looks for such a notice in the current file and, if not found, offers to insert one. Make `check-for-license` run for every source code file you open.
Useful functions: `goto-char`, `point`, `comment-region`, `search-forward`, `y-or-n-p`, `add-hook`, `current-time` and `format-time-string`.
Useful variables: `prog-mode-hook` and `user-full-name`.

Section 5

Appendix

Lisp in other free software

Lisp dialects are used in other free software, including:

- Denemo and GNU Lilypond
- GnuCash
- GDB
- Gimp (uses Script-Fu Scheme)
- Guix
- Make

Useful functions

Not a complete list, just some ideas. See the reference manual.

Strings	`concat`, `substring`, `format`
Lists	`car`, `cdr`, `cons`, `push`, `pop`, `add-to-list`, `assoc`, `mapcar`, `mapc`, `reverse`
Interacting with the user	`message`, `read-string`, `read-file-name`
Comparing things	`eq`, `equal`, `and`, `or`, `not`
Loops	`while`, `dolist`, `dotimes`
Conditionals	`if`, `when/unless`, `cond`
Working with/in buffers	`point`, `mark`, `insert`, `save-excursion`, `with-current-buffer`, `save-restriction`, `forward-char`, `forward-line`, `kill-region`,
Searching	`search-forward`, `re-search-forward`

License