

Plug Emacs Conf Presentation

Michael Olson

April 27, 2007

Contents

1	Beginnings	2
2	An example configuration	2

Presenter Michael Olson
HTML <http://mwolson.org/notes/PlugEmacsConfPresentation.html>
PDF <http://mwolson.org/notes/PlugEmacsConfPresentation.pdf>
Date 2007-02-09

1 Beginnings

See Plug Emacs Basics¹ for an introduction to what Emacs is, and what it is useful for.

2 An example configuration

Most of the presentation will involve going over a sample configuration file. To obtain this file, do the following.

1. Download `plug-emacs-conf-files.tar.gz`.
2. Extract this somewhere
3. Move the elisp directory to `~/elisp`.
4. `mv ~/.emacs ~/.emacs.old`
5. `mv ~/elisp/dotemacs ~/.emacs`

This renames your old `.emacs` file to `.emacs.old`, and installs the sample `.emacs` file to `~/.emacs`. Now on to the configuration file. Feel free to follow along on your lab machine if you like.

```
;; * About this file
;;
;; This is a sample Emacs configuration file that Michael Olson
;; created for the PLUG Emacs Configuration presentation on
;; 2007-02-09. It may be written to the ~/.emacs file.
;;
;; The presentation and the extra code used by this file may be found
;; at <http://www.mwolson.org/notes/PlugEmacsConfPresentation.html>.
;;
;; This should work with Emacs 21, XEmacs 21, and Emacs 22 (when it is
;; released). Lines that have two semicolons in front of them are
;; comments.
;;
;; * What you need to know about Emacs
;;
;; Emacs can make your life very easy. One example of this is in
;; TAB-completion. When Emacs prompts you for some value, you can
;; often hit TAB twice to see all of the available options. Then when
;; you start typing an option, hit TAB to try to complete the rest of
;; it. I use this constantly.

;;; Set up the environment

;; Make sure Emacs Lisp code in the ~/elisp directory can be found
```

¹<http://mwolson.org/notes/PlugEmacsBasics.pdf>

```

(add-to-list 'load-path "~/elisp")

;; * Installing more software for Emacs
;;
;; If you want to load other add-on programs for Emacs, do the
;; following, where PROG is the name of the program.
;;
;; 1. Unpack the program to ~/elisp/PROG
;;
;; 2. Add the line:
;;   (add-to-list 'load-path "~/elisp/PROG")
;;
;; 3. Add the line:
;;   (require 'PROG)
;;
;; Step 3 may need to be adjusted depending on the program -- read the
;; manual of the program in question for more specific details.
;;
;; 4. Add other lines that are suggested by the manual of the program,
;; if any.

;; As an example, I've enabled my "wtf" program, which allows you to
;; look up the definitions of acronyms by typing M-x wtf-is RET
;; <acronym> RET. Replace <acronym> with the term you want to look
;; up. If you move to the acronym, the prompt should be automatically
;; filled in with that acronym.
(require 'wtf)

;;; Miscellaneous extra features

;; Make the Delete and Backspace keys work as expected
(when (and (fboundp 'normal-erase-is-backspace-mode)
           window-system)
  (normal-erase-is-backspace-mode 1))

;; Load 'dired', a "directory editing" mode
(require 'dired)
(require 'wdired)
(condition-case nil
  (require 'dired-x)
  (error nil))
; ignore errors if this is not found

;; List directories before files in dired
(condition-case nil
  (require 'ls-lisp)
  (error nil))

;; Uncomment this if you don't like being prompted for recursive
;; deletion in dired.
;;
;; (setq dired-recursive-deletes 'always)

```

```

;; Load 'ido', which shows a different prompt when you're switching
;; between files
(condition-case nil
  (progn
    (require 'ido)
    (setq ido-everywhere nil
          ido-save-directory-list-file "~/ .emacs.d/.ido.last")
    (ido-mode 'buffer))
  (error nil))

;; Uncomment this to cause the Tab key to insert spaces instead of
;; tabs. The default is to insert tabs.
;;
;; (setq-default indent-tabs-mode nil)

;; When not using X, don't show the menu bar
(when (and (not window-system)
          (fboundp 'menu-bar-mode))
  (menu-bar-mode 0))

;; Find funtions and files at point
(when (fboundp 'ffap-bindings)
  (ffap-bindings))

;; Grand Unified Debugger (allows you to do M-x gdb)
(require 'gud)

;; Enable browsing of kill ring (that is, recently-killed/cut text)
;; when you hit M-y
(require 'browse-kill-ring)
(defadvice yank-pop (around kill-ring-browse-maybe (arg))
  "If last action was not a yank, run 'browse-kill-ring' instead."
  (if (not (eq last-command 'yank))
      (browse-kill-ring)
      ad-do-it))
(ad-activate 'yank-pop)

;; Load info before trying to deal with its mode map later on
(require 'info)

;; Delete the selection when hitting a key after selecting it, like
;; most other editors do
(if (featurep 'xemacs)
    (pending-delete-mode 1)
    (delete-selection-mode 1))

;; Uncomment this if you'd like your Emacs session to do amusing
;; things after 3 minutes of idle time. Hitting a key will stop the
;; madness :^) .
;;

```

```

;; (require 'zone)
;; (setq zone-idle (* 60 3))
;; (zone-when-idle zone-idle)

;;; Key customizations

;; Uncomment this to disable the Insert key, which I find annoying
;;
;; (global-set-key [insert] (lambda () (interactive)))

;; Make certain that Home and End do the right thing
(global-set-key [home] 'beginning-of-line)
(global-set-key [end] 'end-of-line)

;; Nasty hack to make Home and End work in Solaris
(when (string= (getenv "TERM") "dtterm")
  (global-set-key (kbd "ESC O") nil)
  (global-set-key (kbd "ESC O H") 'beginning-of-line)
  (global-set-key (kbd "ESC O F") 'end-of-line))

;; Make the <DEL> key scroll backwards in Info mode
(define-key Info-mode-map [delete] 'Info-scroll-down)

;; Prompt for a line and go to it when hitting C-x g
(global-set-key "\C-xg" 'goto-line)

;;; Customizations

;; These are options that can be changed by doing the following,
;; replacing <option> with the name of an option.
;;
;; M-x customize-option RET <option> RET
;;
;; This is called the "Customize" interface.
;;
;; * About Customize
;;
;; Everything that you can do in Customize, you could alternatively do
;; outside of Customize by editing your .emacs file. I like using
;; Customize better than changing the .emacs file for simple things,
;; but some people prefer to have absolute control over the appearance
;; of their .emacs file.
;;
;; * Tabs
;;
;; If you are editing files with tabs in them and things don't look
;; right (which is why you should never use tabs, IMNSHO), do:
;;
;; M-x customize-option tab-width RET
;;
;; and change the value to what you want the width of tabs to be. Be

```

```

;; sure to save the change for future sessions.
;;
;; * Customizing colors
;;
;; One excellent use for Customize is to change what colors are used
;; for syntax highlighting when you're editing code. We call these
;; different colors "faces".
;;
;; To change a particular face, move to it in a file with code and hit
;; M-x customize-face RET. At this point, you should see name of the
;; face at the bottom of the screen. If it looks right, hit RET,
;; otherwise change the name of the face.
;;
;; This will take you to a screen that lets you change various
;; attributes, such as color. To get a listing of available colors,
;; type M-x list-colors-display RET. You'll have to type in the color
;; name manually for the foreground and/or background.
;;
;; * Customization defaults in this file
;;
;; I've enabled quite a few things in the customizations listed below.
;; Here's a partial list of them.
;;
;; - Do the right thing when opening a .zip or .gz file.
;;
;; - Put backup data into the ~/.emacs.d/backup directory, instead of
;; putting backup files in the current directory. I find this to
;; be much cleaner.
;;
;; - When moving over parentheses or braces, highlight the other
;; matching parenthesis or brace.
;;
;; - Show the column number on the bottom of the screen, just after
;; the row number.
;;
;; - Colorize code when visiting a code file.
;;
;; - Enable doing M-x ibuffer RET to get a listing of open buffers.
;;
;; - Don't permit XEmacs to prompt you about migrating the contents
;; of ~/.emacs to the ~/.xemacs directory. Otherwise, further
;; changes made to ~/.emacs would have no effect, and that would be
;; confusing.
;;
;; - When editing a directory with M-x dired RET, show directories
;; before files and ignore case, much like most good file managers
;; do.
;;
;; - Make sure files end with a newline character, because otherwise
;; some commandline programs might potentially choke.
;;

```

```

;; - Remember the place you were in the last 20 files you've opened.
;;
;; - When highlighting text by hitting C-SPC, actually show what is
;;   being highlighted. Make the background of this region blue,
;;   like most text editors do.

(custom-set-variables
 '(allout-auto-activation t)
 '(apropos-do-all t)
 '(auto-compression-mode t nil (jka-compr))
 '(backup-directory-alist (quote (( "." . "~/ .emacs.d/backup"))))
 '(blink-cursor nil)
 '(blink-matching-paren-on-screen t)
 '(bookmark-default-file "~/ .emacs.d/.bookmark")
 '(column-number-mode t)
 '(dired-dwim-target t)
 '(dired-recursive-copies (quote always))
 '(eldoc-minor-mode-string " E" t)
 '(eldoc-mode t)
 '(global-font-lock-mode t nil (font-lock))
 '(highlight-nonselected-windows t)
 '(ibuffer-enable t)
 '(load-home-init-file t t)
 '(ls-lisp-dirs-first t)
 '(ls-lisp-ignore-case t)
 '(ls-lisp-use-insert-directory-program nil)
 '(mail-interactive t)
 '(mark-diary-entries-in-calendar t)
 '(post-jump-header nil)
 '(require-final-newline t)
 '(save-place t)
 '(save-place-file "~/ .emacs.d/.places")
 '(save-place-limit 20)
 '(session-save-file "~/ .emacs.d/.session")
 '(show-paren-mode t nil (paren))
 '(show-paren-style (quote parenthesis))
 '(text-mode-hook (quote (flyspell-mode text-mode-hook-identify)) t)
 '(transient-mark-mode t)
 '(visible-bell t))
(custom-set-faces
 '(region ((t (:background "blue" :foreground "white")))))

;; sample .emacs file ends here

```