Matt's Emacs Config: Windows-WSL

Matthew Sakkas

November 10, 2023

Using Org File to Configure Emacs

- I took this setup from minimal-emacs-setup-with-orgmode.
- When Emacs is started, it tries to load an init file from one of the following:
  - `~/.emacs`,
  - `~/.emacs.el`, or
  - `~/.emacs.d/init.el`.

Where Windows Finds Init File

- Here is a good explanation of where emacs searches for an init file: emacs in windows.
- By default in Windows emacs will look in the following places:
  - `%HOMEPATH%\AppData\Roaming\.emacs`
  - `%HOMEPATH%\AppData\Roaming\.emacs.d\init.el`
- But a better way is to create the environment variable named `HOME` (like this) and populate it with the directory where you want to put the init file.
- BTW, it doesn't matter where you install emacs. As long as you have a `HOME` environment variable, emacs knows where to look for its config files.

Steps to Setting Up Org File to Configure Emacs

1. Delete file ~/.emacs if you have one.
2. Create dir ~/.emacs.d/ if you don't have one.
3. Create file ~/.emacs.d/init.el if you don't have one.
4. Put the init.el (see below) into ~/.emacs.d/init.el.
5. Setup the ~/.emacs.d/emacs-config.org file.

Code for init.el

```
% This is the code for init.el
(require 'package)
(setq load-prefer-newer t
      package-enable-at-startup nil)

(setq package-archives '(("melpa" . "https://melpa.org/packages/")
             ("gnu"   . "https://elpa.gnu.org/packages/")))

(package-initialize)

;; Bootstrap `use-package'
(unless (package-installed-p 'use-package)
  (package-refresh-contents)
  (package-install 'use-package))

;; load org package and our emacs-config.org file
(require 'org)
(org-babel-load-file "~/.emacs.d/emacs-config.org")
```

Purpose of /.emacs.d/init.el

1. Allow emacs to download packages from the MELPA package server.

2. Install the `use-package` package to simplify package management.
3. Load the `org` package.
4. Use the `org-babel-load-file` function to load and execute the code inside `~/.emacs.d/emacs-config.org`.

Everything else is installed and configured in `emacs-config.org`.

Basic Settings

```
;suppress startup splash-screen
(setq inhibit-startup-screen t)

;disable the scroll bar
(toggle-scroll-bar -1)

;disable the toolbar
(tool-bar-mode -1)

;non-nil invokes Continuous mode
;reaching the page edge advances to next/previous page
(setq doc-view-continuous t)

;non-nil ignores case in file name completion
(setq read-file-name-completion-ignore-case t)

;globally enable visual line mode (i.e. buffer 'word-wrap')
(global-visual-line-mode 1)

;typed text replaces the active selection
(delete-selection-mode 1)

 (use-package paren
   :config
```

```
  ;toggle visualization of matching parens
  (show-paren-mode 1)
  ;set zero delay to showing a matching paren
  (setq show-paren-delay 0))

;go right into maximized screen
(toggle-frame-maximized)

;set default font to 16 pt (values are in .1 pt)
(set-face-attribute 'default nil :height 180)

;enable upcase-region (bound to C-x C-u)
(put 'upcase-region 'disabled nil)

;enable downcase-region (bound to C-x C-l)
(put 'downcase-region 'disabled nil)

;remap the function 'list-buffers' to the function 'buffer-menu'
; which shows a list of buffers in a window with focus - "T" toggles the
(global-set-key (kbd "C-x C-b") 'buffer-menu)
```

Dired

```
(use-package dired-x
  ;open multiple files by marking, then F to open
  :ensure nil)

;kills the Dired buffer (using "a" key),
;then visits the current line's file or directory
(put 'dired-find-alternate-file 'disabled nil)

;this function needs better commenting for purpose
(defun matt-dired-mode-setup ()
```

```
  "To be run as hook for ‘dired-mode’."
  (dired-hide-details-mode 1))

;In Dired ’(’ toggles dired-hide-details-mode.
(add-hook ’dired-mode-hook ’matt-dired-mode-setup)

;switches: all, ignore backups, human-readable, long (required)
(setq dired-listing-switches "-aBhl --group-directories-first")
```

Time Formatting (for Dired)

- By default emacs displays "recent" files with a higher-resolution timestamp than "older" files.
- This makes time sorting of files in dired messy.
- So, below, I set the time format uniform for both recent and older files.

```
(setq ls-lisp-format-time-list  ’("%Y.%m.%d %H:%M" "%Y.%m.%d %H:%M")
      ls-lisp-use-localized-time-format t)
```

Themes

- You can only choose one at a time!
- You can comment-out a code block by putting a space right after the # sign.

```
(load-theme ’tsdh-dark t)
```

Browse-Kill-Ring

- Install this with package manager, then delete the custom_variables from `init.el`.

```
(use-package browse-kill-ring
  :bind ("M-y" . browse-kill-ring)
  :config
  ;act like yank-pop
  (setq browse-kill-ring-replace-yank t))

(provide 'init-browse-kill-ring)
```

LaTeX

```
(use-package latex
  :defer t
  :ensure auctex
  :mode ("\\.tex\\'" . LaTeX-mode) ; this line fixed problem of auctex not
  :config
  (setq TeX-auto-save t)
  (setq TeX-parse-self t)
  (use-package preview)
  (add-hook 'LaTeX-mode-hook 'reftex-mode)
)

;; allow org-mode and elisp to run latex snippets
(org-babel-do-load-languages
 'org-babel-load-languages
 '((latex . t) (org . t) (emacs-lisp . t)))
```

aspell

```
(setq ispell-program-name "aspell")
```

pdftools

- For some reason `M-x list-packages` sometimes doesn't actually list the `pdf-tools` package. To fix that simply run: `M-x package-install RET pdf-tools RET`.
- I got the below, very nice, config from here: Alberto Alvarez.

```
(use-package pdf-tools
   :pin manual
   :config
   (pdf-tools-install)
   (setq-default pdf-view-display-size 'fit-width)
   (define-key pdf-view-mode-map (kbd "C-s") 'isearch-forward)
   :custom
   (pdf-annot-activate-created-annotations t "automatically annotate highl

(setq TeX-view-program-selection '((output-pdf "PDF Tools"))
      TeX-view-program-list '(("PDF Tools" TeX-pdf-tools-sync-view))
      TeX-source-correlate-start-server t)

(add-hook 'TeX-after-compilation-finished-functions
  #'TeX-revert-document-buffer)
```

org-pdftools

```
(use-package org-pdftools
  :hook (org-mode . org-pdftools-setup-link))
```

```
(add-to-list 'org-file-apps '("\\.pdf\\'" . emacs))
```

vertico - completion framework

- I got vertico idea and config from this System Crafters video.

```
(use-package vertico
  :ensure t
  :init
  (vertico-mode))
```

yasnippet

```
(use-package yasnippet
:config
(setq yas-snippet-dirs
'("/mnt/c/Users/matt/Dropbox/math_and_prog/emacs/yasnippet/snippets/"))
(setq yas-use-menu '("full"))
(yas-global-mode 1))
```

ace-window

```
(use-package ace-window
  :ensure t
  :init (setq aw-keys '(?a ?s ?d ?f ?g ?h ?j ?k ?l)) ; override default ke
  :bind (("C-x o" . ace-window))) ; override default key binding
```