My literate GNU Emacs config

Marius Peter
23 Jul, 2020 (Thu)

Contents

List of Figures

List of Tables

Abstract

GNU Emacs is most often used as a text editor. The utmost level of customisation is afforded by enabling the user to rewrite *any* part of the source code and observe the editor's modified behaviour in real time. Since its inception in 1984, GNU Emacs has grown to be much more than a full-featured, high-productivity text editor—new *modes* have been written to interact with hundreds of file formats, including `.txt`, `.pdf`, `.jpg`, `.csv`, and `.zip` just to name a few. This configuration file itself was written in *Org mode*, a collection of functions enabling the harmonious mixing of code and comments in view of publication: this is the endgame of *literate programming*.

(insert "")

README

The `README.org` was exported from the first section of my literate configuration file, `blendoit-init.org`.

Document structure

Blending Linux and Windows

The GNU Emacs cabal is attempting to create a complete OS out of a text editor.

Microsoft has a notorious *embrace, extend, extinguish* approach when it comes to rival technologies. Both are simultaneously possible.

1 TODO First-time setup

Spacemacs-like dialog for default settings.

```
;; Prompt enterprise or personal install. Create file in .emacs.d/ on Linux,
;; AppData/ on Windows. Ask user for details and preferred bindings.
```

2 Early setup

2.1 Garbage collection

First, we increase the RAM threshold beyond which the garbage collector is activated.

```
(setq gc-cons-threshold 100000000)
```

2.2 Emacs client

Makes opening emacs faster for following instances.

```
; (setq initial-buffer-choice (lambda () (get-buffer "*dashboard*")))
```

2.3 Refresh configuration

Add an `after-save-hook` that tangles and loads this literary configuration.

2.4 Custom file

Load settings created automatically by GNU Emacs Custom. (For example, any clickable option/toggle is saved here.) Useful for fooling around with M-x customize-group <package>.

```
(setq custom-file "~/.emacs.d/init-custom.el")
(load custom-file)
```

2.5 Profiling — start

We start the profiler now , and will interrupt it in section 7.1. We will then present profiling report in section 7.2.

```
; (profiler-start)
```

2.6 Customization shortcuts

We begin by defining a user shortcut to this very file:

```
(defun find-init-blendoit ()
  "Jump to this very file."
   (interactive)
   (find-file "~/.emacs.d/blendoit/blendoit-init.org"))

(global-set-key (kbd "C-c c") 'find-init-blendoit)
```

Now, different shortcuts for other customization actions:

```
(global-set-key (kbd "C-c v") 'customize-variable)
(global-set-key (kbd "C-c f") 'customize-face)
```

2.7 Backups

Backups are so important that they should be described right after the shortcut to this file.

```
(setq backup-directory-alist `((".*" . ,temporary-file-directory))
auto-save-file-name-transforms `((".*" ,temporary-file-directory t))
    backup-by-copying t    ; Don't delink hardlinks
    version-control t      ; Use version numbers on backups
    delete-old-versions t  ; Automatically delete excess backups
    kept-new-versions 20   ; how many of the newest versions to keep
    kept-old-versions 5    ; and how many of the old
    )
```

2.8 Secrets

```
(setq user-full-name "Marius Peter"
      user-mail-address "blendoit@gmail.com")

(setq user-full-name "Marius Peter"
      user-mail-address "blendoit@gmail.com")
```

2.9 File system paths

In this subsection, we tell Emacs about relevant paths to resources.

On my Windows machine, I add the path to Portable Git.[1]

```
(if (string-equal system-type "windows-nt")
    (add-to-list 'exec-path "C:/Users/marius.peter/PortableGit/bin/"))
```

3 Global key bindings

The following bindings strive to further enhance CUA[2] mode.

3.1 Navigation

```
  (global-set-key (kbd "C-‘") 'delete-other-windows)
  (global-set-key (kbd "C-s") 'save-buffer)
  (global-set-key (kbd "C-b") 'ibuffer-sidebar-toggle-sidebar)
  (global-set-key (kbd "C-o") 'menu-find-file-existing)
  (global-set-key (kbd "C-r") 'counsel-recentf)
;  (global-set-key (kbd "C-n") 'make-frame) ; 7aram!
```

The following bindings lead to more natural exit behaviors.

```
  (defun delete-window-or-previous-buffer ()
    "Delete window; if sole window, previous buffer."
     (interactive)
     (if (> (length (window-list)) 2)
       (delete-window)
       (previous-buffer)))

(global-set-key (kbd "C-w") 'delete-window-or-previous-buffer)
(global-set-key (kbd "C-q") 'save-buffers-kill-terminal)
```

[1] Download from `https://git-scm.com/download/win`

[2] Common User Access.

3.2 TODO Mouse zoom

Set conditional if in MS Windows or GNU/Linux.

Zoom in/out of selected buffer using Alt + mouse wheel. Bindings are slightly different in MS Windows.

```
; GNU/Linux
; (global-set-key [M-mouse-4] 'text-scale-increase)
; (global-set-key [M-mouse-5] 'text-scale-decrease)

; MS Windows
(global-set-key [M-wheel-up] 'text-scale-increase)
(global-set-key [M-wheel-down] 'text-scale-decrease)
```

4 Packages

Packages are collections of `.el` files providing added functionality to Emacs.

4.1 Package archives

List of package archives.

```
(require 'package)
(add-to-list 'package-archives '("melpa" . "https://melpa.org/packages/") t)
(add-to-list 'package-archives '("org" . "https://orgmode.org/elpa/") t)
(package-initialize)
```

4.2 `use-package`

Ensure `use-package` is installed, as well as all packages described in this configuration file.

```
(unless (package-installed-p 'use-package)
  (package-refresh-contents)
  (package-install 'use-package)
  (eval-when-compile (require 'use-package)))
(setq use-package-always-ensure t)
```

4.3 TODO Convenient package update

One-function rollup of upgradeable package tagging, download and lazy install.

4.4 `ivy`

Auto completion.

```
(use-package ivy
 :config
 (setq ivy-use-virtual-buffers t
       ivy-count-format "%d/%d "
       enable-recursive-minibuffers t))
(ivy-mode t)
```

4.4.1 counsel

Wonderful counsellor!

```
(use-package counsel
 :bind ("M-x" . counsel-M-x)
 :config (counsel-mode t))

(global-set-key (kbd "C-f") 'counsel-grep-or-swiper)
```

4.4.2 swiper

```
(use-package swiper
 :bind (("C-f" . counsel-grep-or-swiper)))
```

4.5 evil-mode

Forgive me, for I have sinned.

```
(use-package evil)
;; (evil-mode 1)
```

4.6 org-mode

Phew, I can finally introduce Org mode! I am so **excited**.

Org mode replaces aword processor, a presentation creator, and a spreadsheet editor. IMHO, the spreadsheet ability captures more than 80% use cases wherein one wishes to include a table in a text document destined for physical publication. (It is clear that Excel spreadsheets are *not* destined for physical publication—simply attempt to print an Excel spreadsheet with the default settings.) In my opinion, Org mode matches all *useful* features of the Microsoft Office suite 1-to-1.

What follows are customizations designed to make Org mode behave more like Microsoft Word. The end goal is, once again, to draw as many new users to Emacs as possible!

4.6.1 Basic customization

Org base directory is in user home on GNU/Linux, or in `AppData` in MS Windows.

```
(setq org-directory "~/org")
```

First, we hide markup symbols for **bold**, *italic*, underlined and ~~strikethrough~~ text, and ensure our document appears indented upon loading:[3]

```
(setq org-hide-emphasis-markers t)
(setq org-startup-indented t)
```

Then, we customize Org headings to emulate WYSIWYG[4] behavior normally found in Word:

```
(font-lock-add-keywords
 'org-mode
 '(("^ *\\([-]\\) "
  (0 (prog1 () (compose-region (match-beginning 1) (match-end 1) "•"))))))
```

- Look at
- This beautifully indented
  - List…
  - Of lists!
- (Rendered with pretty bullets in Emacs)

4.6.2 Org-bullets

The following prettifies Org mode heading bullets:

4.6.3 Invisible edits

```
(setq org-catch-invisible-edits t)
```

4.6.4 Agenda

The agenda displays a chronological list of headings across all agenda files for which the heading or body contain a matching `org-time-stamp`.[5]

```
(global-set-key (kbd "C-c a") 'org-agenda-list)
```

4.6.5 Timestamps

More literary timestamps are exported to LaTeX using the following custom format:

```
(setq org-time-stamp-custom-formats
  '("%d %b, %Y (%a)" . "%d %b, %Y (%a), at %H:%M"))
```

[3] It *appears* indented, but the underlying plaintext file does not contain tab characters!

[4] What You See Is What You Get (input and output are identical), as opposed to What You See Is What You Mean (the input contains instructions that can modify the output).

[5] An `org-time-stamp` can be inserted with `C-c .` (period)

4.6.6 LaTeX export

The following makes `CLOSED` items appear green in LaTeX. Very stylish, much flair!

```
(setq org-latex-inactive-timestamp-format
  "\\textcolor{ForestGreen!60}{\\textit{%s}}")
```

4.6.7 Publish

In the following *alist* (association list), we describe the projects publishable via `org-publish`. We separate the publishing of `.org` files and attachments, because an online tutorial recommended we do so.

```
(require 'ox-publish)
(setq org-publish-project-alist
'(
  ("Safran-VIP-html"
   :base-directory "~/org/WORK/Safran/programs/B787/VIP/doc/org/"
   :base-extension "org"
   :publishing-directory "~/org/WORK/Safran/programs/B787/VIP/doc/wiki/"
   :recursive t
   :publishing-function org-html-publish-to-html
   :auto-preamble t
   :auto-sitemap t
   :sitemap-title "" )

  ("Safran-VIP-static"
   :base-directory "~/org/WORK/Safran/programs/B787/VIP/doc/org/"
   :base-extension "css\\|js\\|png\\|jpg\\|gif\\|pdf\\|mp3\\|mp4\\|ogg\\|swf"
   :publishing-directory "~/org/WORK/Safran/programs/B787/VIP/doc/wiki/"
   :recursive t
   :publishing-function org-publish-attachment )

  ("Safran-VIP-all"
   :components ("Safran-VIP-html" "Safran-VIP-static"))

  ("Safran-MA700-html"
   :base-directory "~/org/WORK/Safran/programs/MA700/doc/org/"
   :base-extension "org"
   :publishing-directory "~/org/WORK/Safran/programs/MA700/doc/wiki/"
   :recursive t
   :publishing-function org-html-publish-to-html
   :auto-preamble t
   :auto-sitemap t
   :sitemap-title "" )
```

```
  ("Safran-MA700-static"
   :base-directory "~/org/WORK/Safran/programs/MA700/doc/org/"
   :base-extension "css\\|js\\|png\\|jpg\\|gif\\|pdf\\|mp3\\|mp4\\|ogg\\|swf"
   :publishing-directory "~/org/WORK/Safran/programs/MA700/doc/wiki/"
   :recursive t
   :publishing-function org-publish-attachment )

  ("Safran-MA700-all"
   :components ("Safran-MA700-html" "Safran-MA700-static"))))

  (add-to-list 'org-latex-packages-alist '("table" "xcolor"
   t ("pdflatex")))
  (add-to-list 'org-latex-packages-alist '("AUTO" "babel"
   t ("pdflatex")))
  (add-to-list 'org-latex-packages-alist '("AUTO" "polyglossia"
   t ("xelatex" "lualatex")))
```

4.6.8 Export

This creates a shorter binding for the most common Org export: Org → LaTeX → PDF.

```
(defun blendoit-org-quick-export ()
  "Org export to PDF and open.
  This basically reimplements `C-c C-e l o'."
  (interactive)
    (org-latex-export-to-pdf)
    (org-open-file
      (concat (substring buffer-file-truename 0 -3) "pdf")))

(global-set-key (kbd "C-c e") 'blendoit-org-quick-export)
```

4.7 undo-tree

```
(global-undo-tree-mode)
```

4.8 dumb-jump

```
(use-package dumb-jump)
(add-hook 'xref-backend-functions #'dumb-jump-xref-activate)
```

4.9 gnuplot

```
(use-package gnuplot)
```

4.10 Ledger

```
(use-package ledger-mode
  :bind
   ("C-c r" . ledger-report)
   ("C-c C" . ledger-mode-clean-buffer))
```

4.11 ibuffer-sidebar

```
(use-package ibuffer-sidebar)
(ibuffer-sidebar-show-sidebar)

;     :bind ("mouse-1" . ibuffer-mouse-visit-buffer)
;     :bind ("mouse-3" . ibuffer-mouse-toggle-mark))

;     (add-hook 'ibuffer-sidebar-mode-hook
;       (lambda ()
;        (local-unset-key (quote mouse-1))
;        (local-unset-key (quote mouse-2))
;        (local-set-key (quote mouse-1) (quote ibuffer-mouse-visit-buffer))
;        (local-set-key (quote mouse-2) (quote ibuffer-mouse-toggle-mark))))
```

4.12 Which-key

```
(use-package which-key
:init
 (which-key-mode)
;; :config
;;  (setq which-key-idle-delay 1000)
;;  (setq which-key-idle-secondary-delay 0.05)
;;  (setq which-key-show-early-on-C-h t)
)
```

4.13 Company

```
;  (add-hook 'after-init-hook 'global-company-mode)
```

4.14 Flycheck

```
(use-package flycheck
 :init (global-flycheck-mode))
```

4.15 CSV

```
(use-package csv-mode)
```

4.16 JSON

Oí, Jason!

```
(use-package json-mode)
```

4.17 Magit

```
(use-package magit
 :bind ("C-c g" . magit-status))
```

4.18 PDF-tools

```
(use-package pdf-tools)
;; (pdf-tools-install)
```

4.19 Dashboard

We replace the standard welcome screen with our own.

```
(setq inhibit-startup-message t)
(use-package dashboard
  :config
    (dashboard-setup-startup-hook)
    (setq dashboard-startup-banner "~/.emacs.d/blendoit/img/Safran_logo.svg")
    (setq dashboard-items '((recents  . 5)
    (projects . 5)))
    (setq dashboard-banner-logo-title "A modern professional text editor."))
```

4.20 rainbow

This highlights hexadecimal numbers which look like colors, in that same color.

```
(use-package rainbow-mode
  :ensure t
  :init
    (add-hook 'prog-mode-hook 'rainbow-mode))
```

4.21 Projectile

This enables us to better manage our `.git` projects.

```
(use-package projectile
 :bind ("C-c p" . 'projectile-command-map)
 :init (projectile-mode 1)
       (setq projectile-completion-system 'ivy))
```

4.22 Better parentheses

```
(show-paren-mode 1)

  (use-package smartparens
    :config
    (add-hook 'prog-mode-hook #'smartparens-mode))

  (use-package rainbow-delimiters
    :config
    (add-hook 'prog-mode-hook #'rainbow-delimiters-mode))
```

5 Cosmetics

5.1 Faces & cursors

In order to imitate other modern text editors, we'll resort to a blinking bar cursor. We choose red, the most captivating color, because the cursor is arguably the region on our screen:

1. most often looked at;
2. most often searched when lost.

The default cursor already blinks.

5.1.1 Default cursor

In files containing only fixed-pitch fonts (i.e. files containing only code), the cursor becomes a high-visibility box.

```
(setq-default cursor-type (quote box))
```

5.1.2 Mixed pitch in Org mode

Fixed-pitch and variable-pitch fonts will be used intelligently in all hooked modes.

```
(use-package mixed-pitch
 :hook ((org-mode . mixed-pitch-mode)
 (Info-mode . mixed-pitch-mode)))
```

5.2 Initial frame

These settings affect the first and subsequent frames spawned by Emacs in GNU/Linux. Frame transparency increases when focus is lost.

```
(if (and (display-graphic-p) (string-equal system-type "gnu/linux"))
    ((set-frame-parameter (selected-frame) 'alpha '(85 . 50))
     (add-to-list 'default-frame-alist '(alpha . (85 . 50)))))
```

5.3 Theme

```
(setq custom-theme-directory "~/.emacs.d/blendoit/themes/")
(load-theme 'blendoit-light)
```

5.4 All the icons

```
(use-package all-the-icons)
```

6 Editing preferences

These customizations enhance editor usability.

A line of text is considered 'filled' when it reaches 79 characters in length.

```
(setq-default fill-column 79)
```

We replace the longer `yes-or-no-p` questions with more convenient `y-or-n-p`.

```
(defalias 'yes-or-no-p 'y-or-n-p)
```

Disable minibuffer scroll bar.

```
(set-window-scroll-bars (minibuffer-window) nil nil)
; (set-window-scroll-bars (ibuffer-sidebar-window) nil nil)
```

6.1 Clean up menus

Originally, I wished to inhibit certain entries in the GUI menus. Not worth the effort at this time.

`menu-bar-mode` is inhibited if on Linux.

```
(setq menu-bar-mode t)
```

6.2 Coding standards

This is just a better default.

```
(setq c-default-style "linux"
      c-basic-offset 4)
```

6.3 Dividers

This ensures users can resize windows using the GUI.

```
(menu-bar-bottom-and-right-window-divider)
```

6.4 Tabs

6.5 auto fill

Automatically break lines longer than `fill-column`.

```
(add-hook 'org-mode-hook 'turn-on-auto-fill)
```

6.6 Recent files

```
(recentf-mode 1)
(setq recentf-max-menu-items 25)
(setq recentf-max-saved-items 25)
(run-at-time nil (* 5 60) 'recentf-save-list)
```

7 Late setup

7.1 Profiling — stop

```
;; (profiler-stop)
```

7.2 Profiling — report

```
;; (profiler-report)
```

8 Conclusion

In this configuration file, we described a series of customization steps taken to make Emacs more palatable to modern IDE users.