

My Literate GNU Emacs Config

Marius Peter

23 Jul, 2020 (Thu)

Contents

1	Introduction	5
2	TODO First-time setup	5
2.1	File system paths	5
3	Early setup	5
3.1	Garbage collection	5
3.2	Profiling — start	6
3.3	Emacs client	6
3.4	Custom file	6
3.5	Customization shortcuts	6
3.6	Backups	6
3.7	Initial and default frames	7
3.7.1	GNU/Linux	7
3.8	Secrets	7
4	Global key bindings	7
4.1	Keyboard navigation	7
4.1.1	Saving a file	7
4.1.2	Opening a file	8
4.1.3	Opening a recently visited file	8
4.1.4	Locating a file	8
4.1.5	Closing window and quitting Emacs	8
4.2	Mouse zoom	8
5	Packages	8
5.1	Meta	9
5.1.1	Package archives	9
5.1.2	TODO Convenient package update	9
5.1.3	use-package	9
5.1.4	delight	9
5.2	org-mode	10
5.2.1	Basic customization	10
5.2.2	Invisible edits	10
5.2.3	Agenda	10
5.2.4	Timestamps	11
5.2.5	\LaTeX export	11
5.2.6	Publish	11
5.2.7	Export	12
5.3	TODO evil-mode	12
5.4	Spelling, completion, and snippets	12
5.4.1	flycheck	12
5.4.2	TODO flyspell	13

5.4.3	yasnippet	13
5.4.4	company	13
5.5	Utilities	13
5.5.1	magit	13
5.5.2	projectile	13
5.5.3	which-key	13
5.5.4	dumb-jump	14
5.5.5	undo-tree	14
5.5.6	ivy	14
5.6	File formats	14
5.6.1	csv-mode	14
5.6.2	pdf-tools	14
5.6.3	ledger	15
5.6.4	gnuplot	15
5.7	Cosmetics	15
5.7.1	dashboard	15
5.7.2	powerline	15
5.7.3	TODO Sidebar	15
5.7.4	Better parentheses	15
5.7.5	all-the-icons	15
5.7.6	rainbow-mode	16
6	Themes	16
6.1	My light and dark themes	16
6.1.1	Colors	16
6.1.2	Cursors	16
6.1.3	Faces	18
6.2	TODO minimal	18
7	Editing preferences	18
7.1	Editor	18
7.1.1	Coding standards	18
7.1.2	Recent files	18
7.2	Frame	19
7.2.1	Clean up menus	19
7.2.2	Dividers	19
7.3	Window	19
7.4	Buffer	19
7.4.1	Column filling	19
7.4.2	auto-fill	19
7.5	Minibuffer	19
7.6	pop-up-frames	19

8	Late setup	20
8.1	Profiling — stop	20
8.2	Profiling — report	20
9	Conclusion	20

List of Figures

List of Tables

1	Light and dark themes' colors	17
---	---	----

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*.

1 Introduction

The following sections were laid out very deliberately. When we start Emacs, Emacs Lisp source blocks contained in this document are evaluated sequentially. This means that our editing environment is bootstrapped at every Emacs startup! For instance, we only begin loading packages once we ensured `use-package` is working properly.

Customizing Emacs goes far, far beyond this document—feel free to experiment and discover.

- `C-h f` describe function
- `C-h v` describe variable
- `C-h k` describe key

These three commands will attempt to describe the element currently under our cursor, however you can start typing to search for another symbol.

2 TODO First-time setup

Spacemacs-like dialog for default settings.

2.1 File system paths

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

On my MS 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 Early setup

3.1 Garbage collection

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

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

¹Download from <https://git-scm.com/download/win>

3.2 Profiling — start

We start the profiler now , and will interrupt it in Section 8.1. We will then present profiling report in Section 8.2.

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

3.3 Emacs client

Makes opening emacs faster for following instances.

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

3.4 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>.

user-emacs-directory

```
(setq custom-file (concat user-emacs-directory "init-custom.el"))  
(load custom-file)
```

3.5 Customization shortcuts

We begin by defining a user shortcut to this very file. We load this as early as possible, this facilitates debugging.

```
(defun my/find-literate-config ()  
  "Jump to this very file."  
  (interactive)  
  (find-file my/literate-config))  
  
(global-set-key (kbd "C-c c") 'my/find-literate-config)
```

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)
```

3.6 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
)
```

3.7 Initial and default frames

We set the dimensions of the initial and default frames.

```
(add-to-list 'default-frame-alist '(width . 100))
(add-to-list 'default-frame-alist '(height . 50))

(add-to-list 'initial-frame-alist '(width . 100))
(add-to-list 'initial-frame-alist '(height . 50))
```

3.7.1 GNU/Linux

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

```
(when (and (display-graphic-p) (string-equal system-type "gnu/linux"))
  (set-frame-parameter (selected-frame) 'alpha '(90 . 50))
  (add-to-list 'default-frame-alist '(alpha . (90 . 50))))
```

3.8 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")
```

4 Global key bindings

The following bindings strive to further enhance CUA² mode.

4.1 Keyboard navigation

```
(global-set-key (kbd "C-'"') 'delete-other-windows)
```

4.1.1 Saving a file

```
(global-set-key (kbd "C-s") 'save-buffer)
```

²Common User Access.

4.1.2 Opening a file

Open a file with C-o.

```
(global-set-key (kbd "C-o") 'find-file)
```

4.1.3 Opening a recently visited file

```
(global-set-key (kbd "C-r") 'counsel-recentf)
```

4.1.4 Locating a file

Locate file.

```
(global-set-key (kbd "C-c l") 'counsel-locate)
```

4.1.5 Closing window and quitting Emacs

The following bindings lead to more natural exit behaviors.

```
(defun my/delete-window-or-previous-buffer ()  
  "Delete window; if sole window, previous buffer."  
  (interactive)  
  (if (> (length (window-list)) 1)  
      (delete-window)  
      (previous-buffer)))
```

```
(global-set-key (kbd "C-w") 'my/delete-window-or-previous-buffer)
```

```
(global-set-key (kbd "C-q") 'save-buffers-kill-terminal)
```

4.2 Mouse zoom

The typical binding on both GNU/Linux and MS Windows is adequate here: C-= to zoom in, C-- to zoom out.

It seems that starting with Emacs 27.1, Control + mousewheel works.

```
(global-set-key (kbd "C--") 'text-scale-decrease)
```

```
(global-set-key (kbd "C-=") 'text-scale-increase)
```

```
(global-set-key (kbd "C-+") 'text-scale-increase)
```

5 Packages

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

5.1 Meta

How do we bootstrap packages? First, let's figure out:

1. Where we get our packages from
2. How we upgrade packages
3. How we ensure our required packages are installed

5.1.1 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)
```

5.1.2 TODO Convenient package update

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

5.1.3 use-package

We 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)
(require 'use-package)
(require 'bind-key)
```

5.1.4 delight

With this package, we suppress the mention of certain minor modes in the mode line/powerline. We include it in Meta (cf. 5.1) because it concerns all other packages.

```
(use-package delight)
(delight '((org-indent-mode)
  (bufface-mode)))
```

5.2 org-mode

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

Org mode replaces a word 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!

5.2.1 Basic customization

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

```
(setq org-directory (concat user-emacs-directory "~/org"))
```

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

For the time being, I will in fact display emphasis markers, because hiding them corrupts tables.

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

Then, we customize Org headings to emulate WYSIWYG⁴ behavior normally found in Word:

5.2.2 Invisible edits

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

5.2.3 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)

(defun my/find-diary-file ()
  "Load 'org-agenda-diary-file'."
  (interactive)
  (find-file org-agenda-diary-file))

(global-set-key (kbd "C-c d") 'my/find-diary-file)
```

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

⁴What 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).

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

5.2.4 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"))
```

5.2.5 \LaTeX export

The following makes TODO items appear red and CLOSED items appear green in Org's \LaTeX exports. Very stylish, much flair!

```
(setq org-latex-active-timestamp-format
  "\\textcolor{SteelBlue}{\\texttt{%s}}")
(setq org-latex-inactive-timestamp-format
  "\\textcolor{ForestGreen}{\\texttt{%s}}")
```

5.2.6 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"))))

```

5.2.7 Export

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

```

(defun my/org-quick-export ()
  "Org export to PDF and open.
  This basically reimplements 'C-c C-e l o'."
  (interactive)
  (org-open-file (org-latex-export-to-pdf)))

(global-set-key (kbd "C-c e") 'my/org-quick-export)

```

5.3 TODO evil-mode

Forgive me, for I have sinned.

This is the 2nd most significant customization after org-mode. Enabling evil-mode completely changes editing keys. For more information on vi keybindings, visit <https://hea-www.harvard.edu/~fine/Tech/vi.html>.

```

(use-package evil)
; (setq evil-toggle-key "C-c d") ; devil...
; (evil-mode 1)

```

5.4 Spelling, completion, and snippets

The following customizations open the doors to vastly increased typing speed and accuracy.

5.4.1 flycheck

Syntax highlighting for Emacs.

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

5.4.2 TODO flyspell

```
(use-package flyspell
  :delight)
(add-hook 'text-mode-hook 'flyspell-mode)
```

5.4.3 yasnippet

```
(use-package yasnippet
  :delight)
(yas-global-mode 1)
```

5.4.4 company

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

5.5 Utilities

5.5.1 magit

Wonderful Git porcelain for Emacs. Enables the administration of a Git repository in a pain-free way.

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

5.5.2 projectile

This enables us to better manage our .git projects.

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

5.5.3 which-key

```
(use-package which-key
  :init (which-key-mode)
  :delight)
```

5.5.4 dumb-jump

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

5.5.5 undo-tree

```
(use-package undo-tree
  :delight)
(global-undo-tree-mode)
```

5.5.6 ivy

Auto completion.

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

1. counsel

Wonderful counsellor!

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

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

2. swiper

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

5.6 File formats

5.6.1 csv-mode

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

5.6.2 pdf-tools

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

5.6.3 ledger

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

5.6.4 gnuplot

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

5.7 Cosmetics

5.7.1 dashboard

We replace the standard welcome screen with our own.

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

5.7.2 powerline

```
(use-package powerline)
(powerline-default-theme)
```

5.7.3 TODO Sidebar

Get inspiration from `ibuffer-sidebar` and create a better sidebar.

```
;; (load-file)
```

5.7.4 Better parentheses

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

5.7.5 all-the-icons

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

5.7.6 rainbow-mode

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

```
(use-package rainbow-mode
  :init
  (add-hook 'prog-mode-hook 'rainbow-mode)
  :delight)
```

6 Themes

We load my custom theme.

```
(setq custom-theme-directory (concat user-emacs-directory "themes/"))
(load-theme 'blendoit-light)
; (load-theme 'blendoit-dark)
```

6.1 My light and dark themes

A highly legible unambiguous and thoughtful theme.

6.1.1 Colors

The default face is a black foreground on a white background, this matches MS Word. We are striving for a simple, intuitive color scheme.

Most of the visual cues derived from color are identical in both light and dark themes (Table 1).

6.1.2 Cursors

In order to imitate other modern text editors, we 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.

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

In files containing a mix of variable-pitch and fixed-pitch fonts, the cursor is a more MS Word-like bar.

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

Table 1: Light and dark themes' colors.

Color	blendoit-light	blendoit-dark
Black	default text	default background
Lighter shades	lesser headers	<i>n/a</i>
White	default background	default text
Darker shades	<i>n/a</i>	lesser headers
Red	negative	<i>same</i>
Tomato	timestamp 'TODO'	<i>same</i>
Green	positive	<i>same</i>
ForestGreen	timestamp 'DONE'	<i>same</i>
Blue	interactable content; links	<i>same</i>
SteelBlue	anything Org mode; anchor color	<i>same</i>
DeepSkyBlue	highlight	<i>same</i>
DodgerBlue	isearch	<i>same</i>
Purple		

6.1.3 Faces

- `default`: Hack
 - Legible, modern monospace font
 - Strict, sharp, uncompromising
- `fixed-pitch`: Hack
- `variable-pitch`: Liberation Sans
 - Libre alternative to Arial
- `org-block`: Hermit
 - Slightly wider than Hack
 - More opinionated shapes
 - Very legible parentheses

1. `variable-pitch-mode`

We use `variable-pitch-mode` for appropriate modes.

```
(add-hook 'org-mode-hook 'variable-pitch-mode)
(add-hook 'info-mode-hook 'variable-pitch-mode)
```

2. **TODO** Default font size

Make default font size larger on displays of which the resolution is greater than 1920x1080.

6.2 `TODO` minimal

7 Editing preferences

These customizations enhance editor usability.

7.1 Editor

7.1.1 Coding standards

This is just a better default. Don't @ me.

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

7.1.2 Recent files

The keybinding for opening a recently visited file is described in paragraph 4.1.3.

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

7.2 Frame

7.2.1 Clean up menus

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

```
(menu-bar-mode -1)
(tool-bar-mode -1)
```

7.2.2 Dividers

This ensures users can resize windows using the GUI.

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

7.3 Window

7.4 Buffer

Save cursor location in visited buffer after closing it or Emacs.

```
(save-place-mode 1)
```

7.4.1 Column filling

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

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

7.4.2 auto-fill

Automatically break lines longer than fill-column.

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

7.5 Minibuffer

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)
```

7.6 pop-up-frames

```
; (setq pop-up-frames (quote graphic-only))
```

8 Late setup

8.1 Profiling — stop

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

8.2 Profiling — report

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

9 Conclusion

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