Blendoit's 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*.

1 Preliminary setup

1.1 Garbage collection

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

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

1.2 Server start

Makes opening emacs faster for following instances.

```
(server-start)
```

1.3 Profiling — start

We start the profiler now , and will interrupt it in section 6. We will then present profiling report in 7.

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

1.4 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)
```

1.5 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/init-blendoit.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)
```

1.6 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
    )
```

1.7 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 Global key bindings

The following bindings strive to further enhance CUA[1] mode.

2.1 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)
```

The following bindings lead to more natural exit behaviors.

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

[1] Common User Access.

2.2 Mouse zoom

Zoom in/out of selected buffer using Alt + mouse wheel.

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

3 Packages

3.1 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)
```

3.2 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)
```

3.3 Org

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!

3.3.1 Basic customization

First, we hide markup symbols for **bold**, *italic*, underlined and ~~strikethrough~~ text, and ensure our document appears indented upon loading:[2]

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

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

Then, we customize Org headings to emulate WYSIWYG[3] behavior normally found in Word:

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

The following prettifies Org mode heading bullets:

```
(use-package org-bullets
  :hook
  (org-mode . org-bullets-mode)
)


(let* ((variable-tuple
(cond ((x-list-fonts "Liberation Sans") '(:font "Liberation Sans"))
      ((x-family-fonts "Sans Serif")    '(:family "Sans Serif"))
      (nil (warn "Cannot find a Sans Serif Font.  Install Source Sans Pro."))))
       (base-font-color     (face-foreground 'default nil 'default))
       (headline           `(:inherit default :weight bold)))

  (custom-theme-set-faces
   'user
   `(org-level-8 ((t (,@headline ,@variable-tuple))))
   `(org-level-7 ((t (,@headline ,@variable-tuple))))
   `(org-level-6 ((t (,@headline ,@variable-tuple))))
   `(org-level-5 ((t (,@headline ,@variable-tuple))))
   `(org-level-4 ((t (,@headline ,@variable-tuple :height 1.1))))
   `(org-level-3 ((t (,@headline ,@variable-tuple :height 1.25))))
   `(org-level-2 ((t (,@headline ,@variable-tuple :height 1.5))))
   `(org-level-1 ((t (,@headline ,@variable-tuple :height 1.75))))
```

[3] 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).

```
   '(org-document-title ((t (,@headline ,@variable-tuple :height 2.0 :underline nil))))))

;(custom-theme-set-faces
; 'user
; '(variable-pitch ((t (:family "Liberation Sans"))))
; '(fixed-pitch ((t ( :family "Hack")))))
```

3.3.2 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`.[4]

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

3.3.3 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"))
```

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

```
  ("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")))
```

3.3.4 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)
```

3.4 gnuplot

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

3.5 company

3.6 Ledger

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

3.7 TODO ibuffer-sidebar

```
(ibuffer-sidebar-show-sidebar)

    (use-package ibuffer-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))))
```

3.8 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)
)
```

3.9 Ivy

Auto completion.

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

3.9.1 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)
```

3.9.2 Swiper

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

3.10 Company

```
;; (use-package company
;;  :config (add-hook 'after-init-hook 'global-company-mode))
```

3.11 Flycheck

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

3.12 Magit

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

3.13 PDF-tools

```
(use-package pdf-tools
 :config (pdf-loader-install))
```

3.14 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/logo-blendux_small.png")
    (setq dashboard-items '((recents  . 5)
    (projects . 5)))
    (setq dashboard-banner-logo-title "The editor for the 2nd millenium."))
```

3.15 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 Cosmetics

4.1 Faces & cursors

In order to imitate other modern text editors, we'll resort to a blinking bar cursor.

4.1.1 Default cursor

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

4.1.2 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)))
```

4.2 Initial frame

These settings affect the first and subsequent frames spawned by Emacs.

```
(if (display-graphic-p)
    (progn
      (setq initial-frame-alist
    '(
      (tool-bar-lines . 1)
      (width . 80) ; chars
      (height . 50) ; lines
      (alpha . (90 . 50))))
      (setq default-frame-alist
    '(
      (tool-bar-lines . 1)
      (width . 80)
      (height . 50)
      (alpha . (90 . 50))))))
```

4.3 Scrollbars

```
(set-window-scroll-bars (minibuffer-window) nil nil)
```

4.4 Theme

```
;;  (use-package zenburn-theme
;;   :config
;;    (load-theme 'zenburn))

; (load-theme 'wombat)
```

4.5 All the icons

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

5 Editing preferences

These customizations enhance editor usability.

```
(setq-default fill-column 79)
(defalias 'yes-or-no-p 'y-or-n-p)
```

This is just a better default.

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

5.1 Recent files

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

5.2 Better parentheses

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

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

6 Profiling — stop

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

7 Profiling — report

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