Emacs Configuration

July 21, 2023

```
;;; -*- lexical-binding: t; -*-

(setq user-full-name "Lucas V. R."
     user-mail-address "redacted")
```

1 Disables and unpins

```
;; -*- no-byte-compile: t; -*-
;;; $DOOMDIR/packages.el
(package! flycheck-popup-tip :disable t)
(package! writegood-mode :disable t)
(package! hl-line :disable t)
(package! revealjs :disable t)
(package! org-re-reveal :disable t)
(package! company :disable t)
(unpin! doom-themes)
(unpin! vertico)
(unpin! treemacs)
(unpin! evil-tex)
(unpin! all-the-icons)
```

2 My library

2.1 Non-interactive

2.1.1 Advices

Para desabilitar as mensagens chatas:

```
(defun advice--inhibit-message (f &rest r) (let ((
    ↪ inhibit-message t)) (apply f r)))
```

2.1.2 List of strings predicate

```
(defun string-list-p (x) (and (listp x) (--all? (stringp it) x
    ↪ )))
```

2.1.3 Macros

1. **TODO** Defhook

2. Boolean variable toggle (`setq-toggle`)

   ```
   (defmacro setq-toggle (s)
     `(setq ,s (not ,s)))
   ```

2.2 Interactive

2.2.1 Remove all advice from a symbol

This was taken from StackExchange long ago.

```
(defun advice-unadvice (sym)
  "Remove all advices from symbol SYM."
  (interactive "aFunction symbol: ")
  (advice-mapc (lambda (advice _props) (advice-remove sym
      ↪ advice)) sym))
```

3 Editor

3.1 Completion, search and insertion

3.1.1 In-buffer completion and insertion

1. AAS (auto expanding snippets)

   ```
   (package! aas :recipe (:host github :repo "ymarco/
       ↪ auto-activating-snippets"))
   ```

2. Abbrev

```
(add-hook! 'text-mode-hook
          (abbrev-mode +1))

(setq abbrev-file-name (concat doom-private-dir "
   ↪ abbrev_defs"))
```

3. Cape (completion-at-point extensions)

   (a) Installation

   ```
   (package! cape)
   ```

   (b) Configuration

   ```
   (use-package cape
     :after corfu
     ;; Bind dedicated completion commands
     :bind (("C-c c p" . completion-at-point) ;; capf
            ("C-c c t" . complete-tag) ;; etags
            ("C-c c d" . cape-dabbrev) ;; or
               ↪ dabbrev-completion
            ("C-c c f" . cape-file)
            ("C-c c k" . cape-keyword)
            ("C-c c s" . cape-symbol)
            ("C-c c a" . cape-abbrev)
            ("C-c c i" . cape-ispell)
            ("C-c c l" . cape-line)
            ("C-c c w" . cape-dict)
            ("C-c c \\" . cape-tex)
            ("C-c c &" . cape-sgml)
            ("C-c c r" . cape-rfc1345))
     :init
     ;; Add `completion-at-point-functions', used by `
        ↪ completion-at-point'.
     (setq cape-dabbrev-check-other-buffers nil
           cape-dabbrev-min-length 3
           dabbrev-case-fold-search t)
     (add-to-list 'completion-at-point-functions #'
        ↪ cape-file)
     (add-to-list 'completion-at-point-functions #'
        ↪ cape-tex)
   ```

```
(add-to-list 'completion-at-point-functions #'
   ↪ cape-keyword))
  ;;(add-to-list 'completion-at-point-functions #'
     ↪ cape-sgml)
  ;;(add-to-list 'completion-at-point-functions #'
     ↪ cape-rfc1345)
  ;;(add-to-list 'completion-at-point-functions #'
     ↪ cape-abbrev)
  ;;(add-to-list 'completion-at-point-functions #'
     ↪ cape-ispell)
  ;;(add-to-list 'completion-at-point-functions #'
     ↪ cape-dict)
  ;;(add-to-list 'completion-at-point-functions #'
     ↪ cape-symbol)
  ;;(add-to-list 'completion-at-point-functions #'
     ↪ cape-line)
```

4. Corfu

   (a) Installation

```
(package! corfu :recipe (:host github :repo "minad/
   ↪ corfu" :files ("*.el" "extensions/*.el")))
(package! popon :recipe (:type git :repo "https://
   ↪ codeberg.org/akib/emacs-popon.git"))
(package! corfu-terminal :recipe (:type git :repo "
   ↪ https://codeberg.org/akib/emacs-corfu-terminal.
   ↪ git"))
```

   (b) Configuration

```
(use-package corfu
  :bind (:map corfu-map
        ("\\" . corfu-quit)
        ("TAB" . corfu-next)
        ("S-TAB" . corfu-previous)
        ("<tab>" . corfu-next)
        ("<backtab>" . corfu-previous)
        ("M-s" . corfu-insert-separator))
  :hook (doom-first-input . global-corfu-mode)
  ;; Optional customizations
  :custom
  (corfu-cycle t) ;; Enable cycling for `corfu-next/
      ↪ previous'
  (corfu-auto t)
  (corfu-auto-delay 0)
  (corfu-auto-prefix 3)
  (corfu-preselect 'prompt)
  ;; (corfu-commit-predicate nil) ;; Do not commit
      ↪ selected candidates on next input
  ;; (corfu-quit-at-boundary t) ;; Automatically quit
      ↪  at word boundary
  (corfu-quit-no-match 'separator)) ;; Automatically
      ↪ quit if there is no match
  ;; (corfu-echo-documentation nil) ;; Do not show
      ↪ documentation in the echo area
  ;; You may want to enable Corfu only for certain
      ↪ modes.
  ;; :hook ((prog-mode . corfu-mode)
  ;; (shell-mode . corfu-mode)
  ;; (eshell-mode . corfu-mode))

;; Dabbrev works with Corfu
(use-package dabbrev
  :custom (dabbrev-search-all-buffers nil)
  ;; Swap M-/ and C-M-/
  :bind (("M-/" . dabbrev-completion)
         ("C-M-/" . dabbrev-expand)))

;; A few more useful configurations...
(use-package emacs
  :init
  ;; TAB cycle if there are only few candidates
  (setq completion-cycle-threshold 3)

  ;; Emacs 28: Hide commands in M-x which do not work
      ↪  in the current mode.
  ;; Corfu commands are hidden, since they are not
      ↪ supposed to be used via M-x.
  (setq read-extended-command-predicate
          #'command-completion-default-include-p)

      ;; Enable indentation+completion using the TAB key.
      ;; `completion-at-point' is often bound to M-TAB.
      (setq tab-always-indent 'complete))
```

5. Color picker (Zenity)

```
(package! zenity-color-picker)

(map! :leader :n "e c" #'zenity-cp-color-at-point-dwim)
```

6. Yasnippet

   Fixes the issue where when a placeholder appeared on an empty line, no indentation was applied. For instance, this:

```
1. bla
   quote|
```

   would be expanded to this:

```
 1. bla
    ,#+begin_quote
|
    ,#+end_quote
```

   quite annoing...

```
(setq yas-also-indent-empty-lines t)
```

   (a) Movimentos entre campos

   Uma dessas coisas que se imagina, por que não fizeram assim?

```
(defadvice! my-yas--maybe-move-to-active-field (
    ↪ snippet)
  "Try to move to SNIPPET's active (or first) field
    ↪ and return it if found."

  :override #'yas--maybe-move-to-active-field

  (let ((target-field (or (yas--snippet-active-field
    ↪ snippet)
```

```
                              (car (yas--snippet-fields
                                ↪ snippet)))))
            (when target-field
              (yas--move-to-field snippet target-field)
              (goto-char (yas--field-end target-field))
              target-field)))
```

(b) Interaction with Corfu

```
(defadvice! yas-expand-filter-corfu-a (&rest r)
  :before-while #'yas-maybe-expand-abbrev-key-filter
  (not (and (frame-live-p corfu--frame) (
      ↪ frame-visible-p corfu--frame))))
```

3.1.2 GUI enhacements

1. Kind-icon

```
(package! kind-icon)

(use-package kind-icon
  :after corfu
  :custom
  (kind-icon-default-face 'corfu-default) ; to compute
      ↪ blended backgrounds correctly
  :config
  (setq kind-icon-default-style '(:padding 0 :stroke 0 :
      ↪ margin 0 :radius 0 :height 0.8 :scale 1.0))
  (add-hook! 'doom-load-theme-hook #'kind-icon-reset-cache
      ↪ )
  (add-to-list 'corfu-margin-formatters #'
      ↪ kind-icon-margin-formatter))
```

3.1.3 Global search and completion

1. Consult

   (a) Installation

```
(unpin! consult)
```

   (b) Configuration

```
(use-package consult
  :bind (:map doom-leader-map
              ("," . consult-buffer)))

(after! consult
  (map! :n "M-y" #'consult-yank-replace
        :leader "h I" #'consult-info
                "r r" #'consult-register
                "r s" #'consult-register-store
                "r l" #'consult-register-load))
```

2. Orderless

```
(use-package corfu
  :config
  (setq completion-styles '(basic-limited orderless basic)
   ↪ ))
```

Também quero que tenha inicialismos (por exemplo, `hmlm -> hide-mode-line-mode`):

```
(setq orderless-matching-styles
      '(orderless-initialism
        orderless-literal
        orderless-regexp))
```

(a) Orderless fast dispatch

```
(defun basic-limited-all-completions (string table
   ↪ pred point)
  (when (length< string 4)
    (completion-emacs21-all-completions string table
       ↪ pred point)))

(defun basic-limited-try-completion (string table
   ↪ pred point)
  (when (length< string 4)
    (completion-emacs21-try-completion string table
       ↪ pred point)))

(add-to-list 'completion-styles-alist
                '(basic-limited
                  basic-limited-try-completion
                  basic-limited-all-completions
                  "Limited basic completion."))
```

3. Register interaction with Evil

   ```
   (after! (consult evil)
     (defadvice! evil-paste--pretend-to-be-yank-a (&rest _r)
       :after #'evil-paste-after
       :after #'evil-paste-before
       (setq this-command 'yank
             yank-undo-function (lambda (_ _) (evil-undo-pop))
                 ↪ )))
   ```

4. Vertico

   (a) Configuration

   ```
   (use-package vertico
     :bind (:map vertico-map
            ("M-k" . vertico-next)
            ("M-j" . vertico-previous))
     :config
     ;; (vertico-reverse-mode +1)
     (setq vertico-resize nil
           vertico-count 8))
   ```

3.2 Help and error system

3.2.1 Eldoc-box

```
(package! eldoc-box)
```

3.2.2 Jinx (spell)

```
(package! jinx)
```

```
(use-package jinx
  :config
  (setq jinx-languages "pt_BR en_US")
  (dolist (hook '(text-mode-hook conf-mode-hook))
    (add-hook hook #'jinx-mode))
  (define-key evil-visual-state-map "z=" 'jinx-correct)
  (define-key evil-normal-state-map "z=" 'jinx-correct))
```

3.2.3 Which-key

1. Posframe ARCHIVE

   ```
   (package! which-key-posframe)
   ```

   ```
   (use-package which-key-posframe
     :hook (which-key-mode . which-key-posframe-mode)
     :config
     (add-hook 'doom-after-reload-hook #'posframe-delete-all)
     (setq which-key-posframe-poshandler #'
         ↪ posframe-poshandler-frame-bottom-center))
   ```

3.3 ORGANIZE Text editing

3.3.1 Evil

1. Variables

   ```
   (setq evil-shift-round nil
         evil-cross-lines t
         evil-move-cursor-back nil
         evil-want-fine-undo t
         evil-snipe-spillover-scope 'visible
         evil-respect-visual-line-mode t

         ;; Substitui vários matches por linha no evil-ex
         evil-ex-substitute-global t)
   ```

   This is very important. The newline at the end of a line is a character too!

   ```
   (setq evil-move-beyond-eol t)
   ```

2. Mouse bindings for multicursor (`evil-mc`)

   Toggle multicursors at mouse pointer with `C-<mouse-1>`.

```
(defun evil-mc/toggle-cursor-on-click (event)
  "Add a cursor where you click, or remove a fake cursor
      ↪ that is
already there."
  (interactive "e")
  (mouse-minibuffer-check event)
  (require 'evil-mc)
  ;; Use event-end in case called from mouse-drag-region.
  ;; If EVENT is a click, event-end and event-start give
      ↪ same value.
  (let ((position (event-end event)))
    (if (not (windowp (posn-window position)))
        (error "Position not in text area of window"))
    (select-window (posn-window position))
    (let ((pt (posn-point position)))
      (if (numberp pt)
          ;; is there a fake cursor with the actual *point*
              ↪  right where we are?
          (unless (evil-mc-undo-cursor-at-pos pt)
            (save-excursion
              (goto-char pt)
              (evil-mc-make-cursor-here)))))))

(after! evil
  (map! "C-<down-mouse-1>" nil)
  (map! "C-<mouse-1>" #'evil-mc/toggle-cursor-on-click))
```

3. Text objects

   (a) Org headlines

```
(defun evil-org--parse-headline ()
  (save-excursion
    (end-of-line)
    (outline-previous-heading)
    (skip-chars-forward "* \t")
    (let* ((todo-start (point))
           (todo-end1 (and org-todo-regexp
                              (let (case-fold-search) (
                                ↪ looking-at (concat
                                ↪ org-todo-regexp " "
                                ↪ )))
                              (goto-char (1- (match-end
                                ↪ 0)))))
          (todo-end2 (when todo-end1 (
            ↪ skip-chars-forward " \t") (point)))
          (priority-start (point))
          (priority-end (when (looking-at "\\[#.\\][ \
            ↪ t]*") (goto-char (match-end 0))))
          (_ (and (let (case-fold-search) (looking-at
            ↪ org-comment-string))
                              (goto-char (match-end 0)))
                                ↪ )
          (title-start (point))
          (tags-start (when (re-search-forward "[ \t
            ↪ ]+\\(:[[:alnum:]_@#%:]+:\\)[ \t]*$"
                                        (
                                          ↪ line-end-position
                                          ↪ )
                                          ↪ '
                                          ↪ move
                                          ↪ )
                            (goto-char (match-beginning
                              ↪ 0))
                            (match-beginning 1)))
          (title-end (point)))
        (list todo-start todo-end1 todo-end2
          ↪ priority-start
              priority-end title-start title-end
              tags-start (line-end-position)))))

(after! evil
  (evil-define-text-object evil-org-headline (count &
      ↪ optional beg end type)
    "Select the current org heading" :jump t
    (save-excursion
      (end-of-line)
      (outline-previous-heading)
    (list (line-beginning-position) (
      ↪ line-end-position))))

(evil-define-text-object evil-org-headline-title (c
    ↪  &rest _)
  "Select the title text in the current org heading"
      ↪  :jump t
  (let ((parse (evil-org--parse-headline)))
    (list (nth 5 parse) (nth 6 parse))))

(evil-define-text-object evil-org-headline-todo (c
    ↪ &rest _)
  "Select the todo entry in the current org heading"
      ↪  :jump t
  (let ((parse (evil-org--parse-headline)))
    (list (nth 0 parse) (nth 2 parse))))

(evil-define-text-object
    ↪ evil-org-headline-inner-todo (c &rest _)
  "Select the inner todo entry in the current org
      ↪ heading" :jump t
  (let ((parse (evil-org--parse-headline)))
    (list (nth 0 parse) (nth 1 parse))))

(evil-define-text-object evil-org-headline-priority
    ↪  (c &rest _)
  "Select the priority entry in the current org
      ↪ heading" :jump t
  (let ((parse (evil-org--parse-headline)))
    (list (nth 3 parse) (nth 4 parse))))

(evil-define-text-object evil-org-headline-tags (c
    ↪ &rest _)
  "Select the tags in the current org heading" :jump
      ↪  t
  (let ((parse (evil-org--parse-headline)))
    (list (nth 6 parse) (nth 8 parse))))

(evil-define-text-object
    ↪ evil-org-headline-inner-priority (c &rest r)
    "Select the inner part of priority in the current
      ↪ org heading" :jump t
    (let ((parse (evil-org--parse-headline)))
      (when (nth 4 parse)
        (let ((p (+ 2 (nth 3 parse)))) (list p (1+ p))
          ↪ ))))

  (evil-define-text-object
    ↪ evil-org-headline-inner-tags (c &rest _)
    "Select the inner part of tags in the current org
      ↪ heading" :jump t
    (let ((parse (evil-org--parse-headline)))
      (when (nth 7 parse)
        (list (1+ (nth 7 parse)) (1- (nth 8 parse)))))
          ↪ )

  (map! :map 'evil-inner-text-objects-map
        "h h" #'evil-org-headline-title
        "h t" #'evil-org-headline-inner-todo
        "h p" #'evil-org-headline-inner-priority
        "h a" #'evil-org-headline-inner-tags)

  (map! :map 'evil-outer-text-objects-map
        "h h" #'evil-org-headline
        "h t" #'evil-org-headline-todo
        "h p" #'evil-org-headline-priority
        "h a" #'evil-org-headline-tags))
```

(b) CameL case

```
(after! evil
  (evil-define-text-object evil-prog-camelcase (c &
    ↪ rest _)
    "Select a camelCase \"word\". For instance, if
      ↪ cursor is at | in
camelCase|dWord, then it selects \"Cased\"." :jump t
    (let ((case-fold-search nil))
       (if-let* ((_ (looking-at-p "[[:lower:]]"))
                 (begin (save-excursion (
                    ↪ re-search-backward "[[:upper
                    ↪ :]]\\|[^[:alpha:]].")
```

```
                                (match-end 0)))
                (end (save-excursion (re-search-forward
                   ↪  "[^[:lower:]]"))))
           (list (1- begin) (1- end))
         (if-let* ((_ (looking-at-p "[[:upper:]]"))
                   (end (save-excursion (
                      ↪ re-search-forward "[[:alpha
                      ↪ :]][[:upper:]]\\|[^[:alpha:]]"
                      ↪ )
                                (match-end 0))))
             (list (point) (1- end))))))

  (map! :mode 'prog-mode
        :map 'evil-inner-text-objects-map
        "l" #'evil-prog-camelcase)
```

3.3.2 Scroll

```
(setq mouse-wheel-scroll-amount '(3 ((shift) . 6)) ;; one line
   ↪  at a time
      mouse-wheel-progressive-speed nil ;; don't accelerate
         ↪ scrolling
      scroll-margin 0
      scroll-step 1) ;; keyboard scroll one line at a time

(when (fboundp 'pixel-scroll-precision-mode)
  (pixel-scroll-precision-mode +1)
  (setq pixel-scroll-precision-interpolate-mice nil))
```

3.3.3 Variables

```
(setq-default fill-column 80)

(setq amalgamating-undo-limit 3)

(setq tab-always-indent t)

;; (setq company-idle-delay 0.1
;; company-minimum-prefix-length 1)
```

```
(setq mouse-drag-and-drop-region t
      mouse-drag-and-drop-region-cut-when-buffers-differ t
      mouse-drag-and-drop-region-show-tooltip nil)
```

```
(setq default-input-method "TeX")
```

Deixa o `text-scale-mode` mais devagar.

```
(setq text-scale-mode-step 1.05)
```

3.3.4 Shrink whitespace bindings

```
(setq doom-leader-alt-key "M-SPC")
(map! :i "C-SPC" #'cycle-spacing)
```

3.3.5 Highlighting (Tree-sitter)

```
(global-tree-sitter-mode)
(add-hook 'tree-sitter-after-on-hook #'tree-sitter-hl-mode)
```

3.3.6 Saving

Desabilita a mensagem de salvamento.

```
(advice-add 'save-buffer :around #'advice--inhibit-message)
```

3.3.7 Blink cursor

```
(blink-cursor-mode -1)
```

3.3.8 Enable perl-like regex search

```
(pcre-mode +1)
```

3.3.9 Popups

```
(setq +popup-defaults
     '(:side bottom
       :height 0.3
       :width 130
       :quit t
       :select ignore
       :ttl 5))

(setq +popup-default-alist
     '((window-height . 0.3)
       (reusable-frames . visible)))
```

3.3.10 Others

```
(remove-hook! '(org-mode-hook text-mode-hook) #'flyspell-mode)
(remove-hook! 'org-mode-hook #'org-cdlatex-mode)

(setq vterm-shell "zsh"
     delete-by-moving-to-trash t
     mouse-autoselect-window nil)
```

3.4 Formatting

3.4.1 Apheleia

```
(package! apheleia)

(use-package apheleia
  :config
  (push '(fourmolu . ("fourmolu" "--stdin-input-file" (or (
     ↪ buffer-file-name) (buffer-name)))) apheleia-formatters
     ↪ )
  (setf (alist-get 'latexindent apheleia-formatters) '("
     ↪ latexindent" "-y=defaultIndent:'  '" "--logfile=/dev/
     ↪ null"))
  (setf (alist-get 'haskell-mode apheleia-mode-alist) '
     ↪ fourmolu))
```

3.4.2 WS butler

```
(after! ws-butler
  (setq ws-butler-global-exempt-modes
        '(special-mode
          comint-mode
          term-mode
          eshell-mode
          diff-mode
          markdown-mode
          org-mode
          latex-mode)))
```

3.5 Selection

3.5.1 Expand-region bindings

```
(use-package expand-region
  :config
  (map! :n "C-a" #'er/expand-region
          "C-S-a" #'er/contract-region))
```

3.6 Overall UI

3.6.1 Doom dashboard

Small dashboard changes:

```
;; Disables "benchmark" echo message
(remove-hook 'window-setup-hook #'doom-display-benchmark-h)
```

1. Banner

   ```
   (setq +doom-dashboard-functions '(
       ↪ doom-dashboard-widget-shortmenu
                               doom-dashboard-widget-loaded
                                   ↪ ))
   ```

3.6.2 ORGANIZE Faces

1. Fonts

   Note: the twemoji font is the CBDT/CBLC variant from Fedora, and in AUR it is named `ttf-twemoji`. The SVG-in-OTF variant will not work!

   If it works, you will see a twemoji smile:

   ```
   (setq doom-font (font-spec :family "Victor Mono" :size 19
       ↪  :weight 'medium)
         doom-variable-pitch-font (font-spec :family "IBM
             ↪ Plex Sans" :size 19 :weight 'normal)
         doom-serif-font (font-spec :family "IBM Plex Mono" :
             ↪ weight 'light))
         ;; doom-unicode-font (font-spec :family "JuliaMono"
             ↪ :weight 'normal))

   ;; Colocamos uma ordem de prioridade para tentar ter
       ↪ todos os unicodes e emojis.
   (setq use-default-font-for-symbols t)
   (defun my/adjust-fonts ()
     ;; (set-fontset-font t 'unicode (font-spec :family "
         ↪ Concrete Math"))
     (set-fontset-font t 'unicode (font-spec :family "Julia
         ↪ Mono") nil 'append)
     (set-fontset-font t 'emoji "Twemoji" nil 'prepend))

   (add-hook! 'after-setting-font-hook #'my/adjust-fonts)
   ```

2. Comments and keywords

   Deixamos os comentários itálicas, e os `keywords` oblíquos.

   ```
   (custom-set-faces!
     '(font-lock-comment-face :slant italic :weight normal)
     '(font-lock-keyword-face :slant italic :weight normal))
   ```

3. Child frames

   ```
   (custom-set-faces!
     `(child-frame-border :inherit default))
   ```

4. Echo area

```
(defun customize-echo ()
  (with-current-buffer " *Echo Area 0*"
    (face-remap-add-relative 'default '(:family "Julia
        ↪ Mono"))
    (face-remap-add-relative 'default '(:height 140 :
        ↪ inherit shadow)))
  (with-current-buffer " *Echo Area 1*"
    (face-remap-add-relative 'default '(:family "Julia
        ↪ Mono"))
    (face-remap-add-relative 'default '(:height 140 :
        ↪ inherit shadow))))

;; (add-hook 'doom-load-theme-hook #'customize-echo 40)
```

5. Icons

   Adjusts the icon sizes so they are a bit smaller.

```
(setq all-the-icons-scale-factor 0.88)
```

6. Ligatures

```
(package! ligature :recipe (:host github :repo "mickeynp/
    ↪ ligature.el"))

(use-package ligature
  :config
  (ligature-set-ligatures
   't '("</" "</>" "/>" "~-" "-~" "~@" "<~" "<~>" "<~~" "
      ↪ ~>" "~~"
        "~~>" ">=" "<=" "<!--" "##" "###" "####" "|-" "-|"
            ↪ "|->"
        "<-|" ">-|" "|-<" "|=" "|=>" "<-" "<--" "-->" "->"
            ↪ "-<"
        ">->" ">>-" "<<-" "<->" "->>" "-<<" "<-<" "==>" "=>
            ↪ " "=/="
        "!==" "!=" "<==" ">>=" "=>>" ">=>" "<=>" "<=<" "<<=
            ↪ " "=<<"
        ".-" ".=" "=:=" "=!=" "==" "===" "::" ":=" ":>" ":<
            ↪ " ">:"
        "<|" "<|>" "|>" "<>" "<$" "<$>" "$>" "<+" "<+>" "+>
            ↪ "
        "?=" "/=" "/==" "/\\" "\\/" "__" "&&" "++" "+++"))
  ;; Enables ligature checks globally in all buffers. You
      ↪ can also do it
  ;; per mode with `ligature-mode'.
  (global-ligature-mode t))
```

7. Mixed-pitch

```
(defface my-mixed-pitch-face '((t . nil))
  "Face for `mixed-pitch-mode'")
(custom-set-faces!
  '(my-mixed-pitch-face :family "Alegreya Sans Scaled" :
      ↪ height 1.1))

(setq mixed-pitch-face 'my-mixed-pitch-face
      mixed-pitch-set-height nil)
```

No modeline pode aparecer um trecho com fonte `font-lock-string-` ↪ `face`. Como fica feio, vamos removê-lo.

Além disso, `org-drawer` não está na lista por padrão.

```
(after! mixed-pitch
  (setq mixed-pitch-fixed-pitch-faces
        (seq-difference
         (seq-union mixed-pitch-fixed-pitch-faces
                    '(org-drawer))
         '(font-lock-string-face diff-added diff-removed)))
            ↪ )
```

8. Yasnippet

```
(custom-set-faces!
  `(yas-field-highlight-face
    :inherit nil
    :background ,(doom-blend "#b315b3" (face-attribute '
        ↪ default :background) 0.2)
    :foreground "undefined"))
```

3.6.3 Mode-line

1. Faces

```
(custom-set-faces!
  '(mode-line :height 105 :family "Julia Mono")
  '(mode-line-inactive :height 105 :family "Julia Mono")
  '(doom-modeline-buffer-modified :underline t :inherit
      ↪ nil)
  '(doom-modeline-info :foreground "white"))
(setq! doom-modeline-height 22
       doom-modeline-bar-width 1)
```

2. Doom mode-line

```
(setq doom-modeline-irc nil
      doom-modeline-icon nil)

(after! doom-modeline
  (doom-modeline-def-segment buffer-name
    "Display the current buffer's name, without any other
        ↪ information."
    (concat
      (doom-modeline-spc)
      (doom-modeline--buffer-name)))

  (doom-modeline-def-segment pdf-icon
    "PDF icon from all-the-icons."
    (concat
      (doom-modeline-spc)
      (doom-modeline-icon 'octicon "file-pdf" nil nil
                          :face (if (doom-modeline--active)
                                    'all-the-icons-red
                                  'mode-line-inactive)
                          :v-adjust 0.02)))

  (defun doom-modeline-update-pdf-pages ()
    "Update PDF pages."
    (setq doom-modeline--pdf-pages
          (let ((current-page-str (number-to-string (eval
              ↪ `(pdf-view-current-page))))
        (total-page-str (number-to-string (
            ↪ pdf-cache-number-of-pages))))
    (concat
      (propertize
        (concat (make-string (- (length
            ↪ total-page-str) (length
            ↪ current-page-str)) 32)
              " P" current-page-str)
        'face 'mode-line)
      (propertize (concat "/" total-page-str) 'face
          ↪  'doom-modeline-buffer-minor-mode)))))

(doom-modeline-def-segment pdf-pages
  "Display PDF pages."
  (if (doom-modeline--active) doom-modeline--pdf-pages
    (propertize doom-modeline--pdf-pages 'face '
        ↪ mode-line-inactive)))

(doom-modeline-def-modeline 'pdf
  '(bar window-number pdf-pages pdf-icon buffer-name)
  '(misc-info matches major-mode process vcs)))
```

3.6.4 Tab bar

```
(map! :n "M-z" #'toggle-frame-tab-bar)
```

3.6.5 Treemacs

1. Esconder algumas coisas

   Roubado do tecosaur.

   ```
   (defcustom treemacs-file-ignore-extensions
     '("aux" "ptc" "fdb_latexmk" "fls" "synctex.gz" "toc" ;;
         ↪ LaTeX
       "glg" "glo" "gls" "glsdefs" "ist" "acn" "acr" "alg" ;;
           ↪  LaTeX - glossary
       "mw" ;; LaTeX - pgfplots
       "pdfa.xmpi") ;; LaTeX - pdfx
     "File extension which `treemacs-ignore-filter' will
         ↪ ensure are ignored"
   ```

```
  :safe #'string-list-p)

(defcustom treemacs-file-ignore-globs
  '("*/_minted-*" ;; LaTeX
     "*/.auctex-auto" "*/_region_.log" "*/_region_.tex") ;
         ↪ ; AucTeX
  "Globs which will are transformed to `
      ↪ treemacs-file-ignore-regexps'
which `treemacs-ignore-filter' will ensure are ignored"
  :safe #'string-list-p)
```

2. Make treemacs fringes appear

   They only appear if this variable is set to a value ≥ 7.

   ```
   (setq doom-themes-treemacs-bitmap-indicator-width 7)
   ```

3.6.6 Theme

```
(setq doom-theme 'ef-cherie)
```

1. ef-themes

   ```
   (package! ef-themes)
   ```

3.6.7 Window divisors

```
(setq window-divider-default-bottom-width 2 ; default is 1
      window-divider-default-right-width 2) ; default is 1
```

4 Living in Emacs

4.1 LSP

```
(unpin! lsp-mode)
```

```
(use-package lsp
  :custom
  (lsp-completion-provider :none)
  (lsp-lens-enable t)
```

```
  (lsp-enable-snippet nil)
  (lsp-use-plists "true")
  :init
  (defun my/lsp-mode-setup-completion ()
    (setf (alist-get 'styles (alist-get 'lsp-capf
        ↪ completion-category-defaults))
          '(orderless)))
  :hook
  (lsp-completion-mode . my/lsp-mode-setup-completion))
```

```
(map! :map 'lsp-mode-map :leader :n "c f" #'lsp-format-buffer)
```

4.1.1 LSP-UI

```
(setq lsp-ui-sideline-diagnostic-max-lines 10)
```

4.2 Terminal

4.2.1 Eat (Emulate A Terminal)

1. Installation

   ```
   (package! eat)
   ```

4.3 Calibre

```
(package! calibredb)
```

4.4 VC/Git/Magit

4.4.1 Magit Todos

```
(put 'magit-todos-exclude-globs 'safe-local-variable #'listp)
```

4.4.2 Git Auto Commit Mode

```
(package! git-auto-commit-mode)
```

```
(pushnew! safe-local-variable-values '(
    ↪ gac-automatically-push-p . t))
```

5 ORGANIZE Features

5.1 Benchmark init

```
(package! benchmark-init :recipe (:host github :repo "
    ↪ kekeimiku/benchmark-init-el"))
```

5.2 Citations and Citar

```
(use-package citar
  :custom
  (citar-file-open-functions '(("pdf" .
    ↪ citar-file-open-external)))
  (citar-bibliography '("/home/lucas/Zotero/bibs/all.bib"))
  (org-cite-csl-styles-dir "/home/lucas/Zotero/styles")
  (citar-symbols `((file ,(all-the-icons-faicon "file-o" :face
    ↪  'all-the-icons-green :v-adjust -0.1) . " ")
                   (note ,(all-the-icons-material "speaker_notes"
                   ↪  :face 'all-the-icons-blue :v-adjust
                   ↪ -0.3) . " ")
                   (link ,(all-the-icons-octicon "link" :face '
                   ↪ all-the-icons-orange :v-adjust 0.01) .
                   ↪ " ")))
  (citar-symbol-separator " "))
```

5.2.1 citar-org-roam templates

```
(setq citar-org-roam-note-title-template "${author editor}: ${
    ↪ title}")
```

5.2.2 Org-cite-csl-activate

```
(unpin! citar)
(package! oc-csl-activate :recipe (:host github :repo "
    ↪ andras-simonyi/org-cite-csl-activate"))
```

```
(use-package oc-csl-activate
  :after (org citar)
  :config
  (setq org-cite-activate-processor 'csl-activate
        org-cite-csl-activate-use-document-style t)
  (add-hook 'org-font-lock-hook (lambda (&rest _) (
     ↪ org-cite-csl-activate-render-all)))
  (add-hook! org-mode
    (cursor-sensor-mode +1)))
```

5.3 CRDT

```
(package! crdt)
```

5.4 Elfeed

```
(after! elfeed
    (setq elfeed-search-filter "@3-year-old #200"))
```

```
(package! elfeed-goodies :disable t)
```

5.4.1 Elfeed-tube

```
(package! elfeed-tube)
(package! elfeed-tube-mpv)
```

```
(use-package elfeed-tube
  :after elfeed
  :demand t
  :config
  ;; (setq elfeed-tube-auto-save-p nil) ; default value
  ;; (setq elfeed-tube-auto-fetch-p t) ; default value
  (elfeed-tube-setup)

  :bind (:map elfeed-show-mode-map
        ("F" . elfeed-tube-fetch)
        ([remap save-buffer] . elfeed-tube-save)
        :map elfeed-search-mode-map
        ("F" . elfeed-tube-fetch)
        ([remap save-buffer] . elfeed-tube-save)))
```

```
(use-package elfeed-tube-mpv
  :bind (:map elfeed-show-mode-map
              ("C-c C-f" . elfeed-tube-mpv-follow-mode)
              ("C-c C-w" . elfeed-tube-mpv-where)))
```

5.5 Esxml

```
(package! esxml)
```

5.6 Engrave Faces

```
(package! engrave-faces)
```

5.7 Google translate

```
(package! google-translate)
```

```
(use-package google-translate
  :commands (google-translate-version)
  :custom
  (google-translate-backend-method 'curl)
  :config
  (defun google-translate--search-tkk () "Search TKK." (list
    ↪ 430675 2721866130)))
```

5.8 **TODO** Olivetti

```
(package! olivetti)
```

```
(use-package olivetti
  :init
  (add-hook! olivetti-mode
    (if olivetti-mode
        (progn
          (remove-hook! lsp-mode #'lsp-ui-mode)
          (when (bound-and-true-p lsp-mode) (lsp-ui-mode -1)))
      (add-hook! lsp-mode #'lsp-ui-mode)
      (when (bound-and-true-p lsp-mode) (lsp-ui-mode +1))))
  :commands #'olivetti-mode
  :hook (org-mode . olivetti-mode)
  :config
  (map! :leader :desc "Centered mode" "t e" #'olivetti-mode)
  (map! :map 'olivetti-mode-map
      "C-c \\" nil
      "C-c |" nil)

  (setq-default olivetti-body-width 80
                olivetti-recall-visual-line-mode-entry-state nil
                    ↪ )

  (after! persp-mode
    (defvar persp--olivetti-buffers-backup nil)

    (defun persp--olivetti-deactivate (fow)
      (dolist (b (mapcar #'window-buffer
                         (window-list (selected-frame)
                                      'no-minibuf)))
        (with-current-buffer b
          (when (eq 'olivetti-split-window-sensibly
                    split-window-preferred-function)
            (push b persp--olivetti-buffers-backup)
            (setq-local split-window-preferred-function nil)
            (olivetti-reset-all-windows)))))

    (defun persp--olivetti-activate (fow)
      (dolist (b persp--olivetti-buffers-backup)
        (with-current-buffer b
          (setq-local split-window-preferred-function
                      'olivetti-split-window-sensibly)))
      (setq persp--olivetti-buffers-backup nil))

    (add-hook 'persp-before-deactivate-functions #'
        ↪ persp--olivetti-deactivate)
    (add-hook 'persp-activated-functions #'
        ↪ persp--olivetti-activate)))
```

5.9 PDF Tools

```
(unpin! pdf-tools)

(after! pdf-tools
  (defvar pdf-scroll-multiplier 2)

  (defun pdf-tools--scroll-mul (l)
    (mapcar (lambda (x) (* pdf-scroll-multiplier x)) l))

  (advice-add 'pdf-view-next-line-or-next-page :filter-args #'
    ↪ pdf-tools--scroll-mul)
  (advice-add 'pdf-view-previous-line-or-previous-page :
    ↪ filter-args #'pdf-tools--scroll-mul)
  (advice-add 'image-forward-hscroll :filter-args #'
    ↪ pdf-tools--scroll-mul)
  (advice-add 'image-backward-hscroll :filter-args #'
    ↪ pdf-tools--scroll-mul)

  (defun pdf-tools-center-page ()
    (interactive)
    (let* ((image (image-get-display-property))
           (edges (window-inside-edges))
           (win-width (- (nth 2 edges) (nth 0 edges)))
           (img-width (ceiling (car (image-display-size image)))
             ↪ ))
      (image-set-window-hscroll (max 0 (/ (- img-width
        ↪ win-width -1) 2)))))

  (advice-add 'pdf-view-shrink :after (lambda (_) (
    ↪ pdf-tools-center-page)))
  (advice-add 'pdf-view-enlarge :after (lambda (_) (
    ↪ pdf-tools-center-page)))

  (add-hook! '(doom-load-theme-hook ef-themes-post-load-hook)
    (setq pdf-view-midnight-colors (cons (face-attribute '
      ↪ default :foreground) (face-attribute 'default :
      ↪ background)))))
```

5.10 Perps-mode

The following advice disables the annoying, big and ugly messages when auto-saving.

```
(advice-add 'persp-parameters-to-savelist :around #'
    ↪ advice--inhibit-message)
```

5.11 Real auto-save

```
(package! real-auto-save)
```

```
(use-package real-auto-save
  :after doom-first-file-hook
  :commands (real-auto-save-mode))
```

```
(pushnew! safe-local-variable-values '(real-auto-save-interval
    ↪  . 0.5))
```

6 Minor modes

6.1 Focus

```
(package! focus :type 'local :recipe (:local-repo "lisp/lib/
    ↪ focus.el"))
```

```
(setq focus-fraction 0.7)
;; (custom-set-faces!
;; '(focus-unfocused :inherit custom-comment-tag :foreground "
    ↪ gray"))
```

6.2 Iedit

Desativa uma mensagem chata quando apertamos `M-d`

```
(setq iedit-toggle-key-default nil)
```

7 Languages

7.1 Dart (flutter)

```
(setq flutter-sdk-path "/opt/flutter")
```

7.2 F#

```
(setq inferior-fsharp-program "dotnet fsi --readline-")
```

7.3 Haskell

```
(setq lsp-haskell-server-path "haskell-language-server-wrapper
   ↪ "
      lsp-haskell-formatting-provider "fourmolu"
      lsp-haskell-plugin-eval-global-on t
      lsp-haskell-plugin-class-global-on nil
      lsp-haskell-plugin-ghcide-type-lenses-global-on nil
      lsp-haskell-plugin-ghcide-completions-config-auto-extend-on
         ↪  nil
      lsp-haskell-plugin-import-lens-code-lens-on nil
      lsp-haskell-plugin-import-lens-code-actions-on nil)
```

Work around bad doom-emacs use of :prelude.

```
(remove-hook 'haskell-mode-local-vars-hook #'lsp!)
```

7.4 Lisps

7.4.1 Parinfer

```
(use-package parinfer-rust-mode
  :when (bound-and-true-p module-file-suffix)
  :hook (emacs-lisp-mode . parinfer-rust-mode)
  :init
  (setq parinfer-rust-library
        (concat doom-data-dir "parinfer-rust/"
                (cond (IS-MAC "parinfer-rust-darwin.so")
                      (IS-LINUX "parinfer-rust-linux.so")
                      (IS-WINDOWS "parinfer-rust-windows.dll")
```

```
                  (IS-BSD "libparinfer_rust.so"))))
  :config
  (map! :map parinfer-rust-mode-map
        :localleader
        "P" #'parinfer-rust-switch-mode
        "p" #'parinfer-rust-toggle-disable))
```

- `paren` Mode gives you full control of parens, while Parinfer corrects indentation. You can still adjust indentation, but you won't be able to indent/dedent past certain boundaries set by parens on previous lines.
- `indent` Mode gives you full control of indentation, while Parinfer corrects or inserts close-parens where appropriate. Specifically, it only touches the groups of close-parens at the end of each line.
- `smart` Mode is like Indent Mode, but it tries to preserve the structure too.

NOTE TO SELF: `smart` and `indent` won't allow inserting unmached }

```
(setq parinfer-rust-preferred-mode "smart")
```

1. Hooks

   I don't want to run it in all lisps, just elisp. Had some issues with `kbd-mode`.

   ```
   (remove-hook 'lisp-mode-hook #'parinfer-rust-mode)
   (add-hook! 'kbd-mode-hook (parinfer-rust-mode -1))
   ```

7.4.2 kmonad

```
(package! kbd-mode :recipe (:host github :repo "kmonad/
    ↪ kbd-mode"))
```

```
(use-package kbd-mode)
```

7.5 Lean

```
(package! lean4-mode :recipe
  (:host github
   :repo "leanprover/lean4-mode"
   :files ("*.el" "data")))

(use-package lean4-mode
  :commands (lean4-mode))

(set-popup-rule! "^\\*Lean Goal\\*"
  :side 'right
  :ttl 10
  :quit 'current
  :width 50
  :select nil
  :modeline nil)
```

7.6 LaTeX

```
(package! latex-preview-pane :disable t)

(after! tex
  (setq TeX-save-query nil
        TeX-view-evince-keep-focus t
        TeX-indent-open-delimiters "["
        TeX-indent-close-delimiters "]"
        TeX-view-program-selection '((output-pdf "Zathura"))
        TeX-view-program-list
        '(("Sioyek-flatpak"
           ("flatpak run --file-forwarding com.github.ahrm.
               ↪ sioyek @@ %o @@"
            (mode-io-correlate " --forward-search-file \"%b\" --
                ↪ forward-search-line %n --inverse-search \"
                ↪ emacsclient -n +%2 %1\"")))))

        font-latex-fontify-script 'multi-level
        font-latex-fontify-script-max-level 3
        font-latex-script-display '((raise -0.4) . (raise 0.4))
            ↪ )
  (custom-set-faces!
    '(font-latex-subscript-face :height 0.8)
    '(font-latex-superscript-face :height 0.8)))

(add-hook! (LaTeX-mode latex-mode)
  (display-line-numbers-mode -1)
  (setq fill-nobreak-predicate nil
        fill-column 9999999999))
```

7.6.1 Do not auto fill my text when I wrap it inside an environment

```
(defadvice! latex-environment-do-not-justify (f &rest r)
  :around 'LaTeX-environment
  (let ((auto-fill-function nil))
    (apply f r)))
```

7.7 Org

7.7.1 Installation

```
(unpin! org)
```

7.7.2 Variáveis

```
(setq org-directory "~/dados/org"
      org-attach-id-dir "data/"
      org-fold-core-style 'overlays
      org-startup-folded nil
      org-startup-indented t
      org-support-shift-select t
      org-hide-emphasis-markers nil
      org-src-window-setup 'plain
      org-highlight-latex-and-related '(latex script)
      org-emphasis-regexp-components '("-[:space:]('\"{" "-[:
         ↪ space:].,:!?;'\")}\\[" "{}*[:space:]" "." 1)
      org-indent-indentation-per-level 2)
```

```
(after! org-src
  (pushnew! org-src-lang-modes
            '("html" . web)
            '("lean4" . lean4)))
```

7.7.3 Attach & Download

We can make `org-attach` work before the first headline:

```
(after! org-attach
  (setq org-attach-auto-tag nil
        org-attach-id-to-path-function-list
        '(org-attach-id-ts-folder-format
            ↪ org-attach-id-uuid-folder-format identity)))
```

```
(after! org-download
  (setq org-download-image-org-width 300))
```

1. Org-attach-extra

   ```
   (add-load-path! "/home/lucas/.doom.d/lisp/lib/
       ↪ org-attach-extra/")
   ```

   ```
   (use-package org-attach-extra
     :after org)
   ```

7.7.4 Appearance

1. Font lock

   (a) Hide fragment delimiters monkey

   ```
   (defadvice! my-org-do-latex-and-related (limit)
     "Highlight LaTeX snippets and environments,
       ↪ entities and sub/superscript.
   Stop at first highlighted object, if any.  Return t
       ↪ if some
   highlighting was done, nil otherwise."
     :override #'org-do-latex-and-related
     (when (org-string-nw-p org-latex-and-related-regexp
       ↪ )
   ```

```
(let ((latex-prefix-re (rx (or "$" "\\(" "\\[")))
      (blank-line-re (rx (and "\n" (zero-or-more (
          ↪ or " " "\t")) "\n"))))
  (catch 'found
   (while (and (< (point) limit)
               (re-search-forward
                  ↪ org-latex-and-related-regexp nil t
                  ↪ ))
     (cond
          ((>= (match-beginning 0) limit)
           (throw 'found nil))
      ((cl-some (lambda (f)
                  (memq f '(org-code org-verbatim
                        ↪ underline
                              org-special-keyword)))
            (save-excursion
              (goto-char (1+ (match-beginning 0)))
              (face-at-point nil t))))
      ;; Try to limit false positives. In this case
          ↪ , ignore
      ;; $$...$$, \(...\), and \[...\] LaTeX
          ↪ constructs if they
      ;; contain an empty line.
      ((save-excursion
         (goto-char (match-beginning 0))
         (and (looking-at-p latex-prefix-re)
              (save-match-data
                (re-search-forward blank-line-re (1-
                    ↪  (match-end 0)) t)))))
      (t
       (let* ((offset (if (memq (char-after (1+ (
           ↪ match-beginning 0)))
                                  '(?_ ?^))
                            1
                          0))
              (start (+ offset (match-beginning 0)))
              (end (match-end 0)))
         (if (memq 'native
               ↪ org-highlight-latex-and-related)
             (org-src-font-lock-fontify-block "latex
              ↪ " start end)
            (font-lock-prepend-text-property start
             ↪ end
                                         'face '
                                           ↪ org-latex-and-related
                                           ↪ ))
          ;; my code starts here
          (when (and org-hide-emphasis-markers (< (+
            ↪  start 4) end))
            (cond ((member (buffer-substring start
             ↪ (+ start 2)) '("$$" "\\("))
                   (add-text-properties start (+
                    ↪ start 2) '(invisible
                    ↪ org-link)))
                  ((string= (buffer-substring (1+
                   ↪ start) (+ start 2)) "$")
                   (add-text-properties (1+ start) (+
                    ↪  start 2) '(invisible
                    ↪ org-link))))
            (cond ((member (buffer-substring end (-
             ↪ end 2)) '("$$" "\\)"))
                   (add-text-properties end (- end 2)
                    ↪  '(invisible org-link)))
                  ((string= (buffer-substring (1- end
                   ↪ ) (- end 2)) "$")
                   (add-text-properties (1- end) (-
                    ↪ end 2) '(invisible org-link)
                    ↪ ))))
          ;; my code ends here
          (add-text-properties (+ offset (
            ↪ match-beginning 0)) (match-end 0)
                                 '(font-lock-multiline t)
                                   ↪ )
          (throw 'found t)))))
      nil))))
```

(b) Fragment fontification without `org-block`

Org reuses the `org-src-font-lock-fontify-block` function to fontify LaTeX fragments natively. But this function adds the very inappropiate face `org-block` to everything. Let's remove it

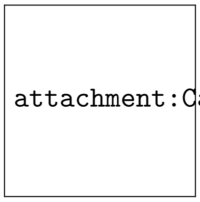
attachment:Captura de tela de 2022-04-21 12-12-49.png

Figure 1: Before

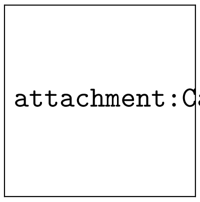
attachment:Captura de tela de 2022-04-21 12-13-01.png

Figure 2: After

when the native block is one of our fragments.

```
(defvar org--font-locking-latex-fragment nil)

(undefadvice! signal-font-locking-latex (orig-fun &
    ↪ rest args)
  :around #'org-do-latex-and-related
  (let ((org--font-locking-latex-fragment t))
    (apply orig-fun args)))

(undefadvice! do-not-org-block-my-latex-advice (_
    ↪ start end)
  :after #'org-src-font-lock-fontify-block
  (when org--font-locking-latex-fragment
    (alter-text-property start end 'face (lambda (l) (
        ↪ remove 'org-block l)))))
```

(c) Better alignment for `mixed-pitch`

```
(defun org-add-indent-face-to-prespace ()
  (setq
   org-font-lock-extra-keywords
   (append (delete
           '("^ *\\([-+]\\|\\(?:[0-9]+\\|[a-zA-Z]\\)[)
               ↪ .]\\)[ \t]" 1 'org-list-dt append)
```

```
 org-font-lock-extra-keywords)
;; Add org-indent face to all spaces at line
   ↪  starts
'(("^\\( +\\)"
   (1 'org-indent append))
  ;; Also fontify * bullets
  ("^ +\\(\\*\\)\\([ \t]\\)"
   (1 'org-list-dt append)
   (2 'org-indent append))
  ;; This is modified from user @psii
  ;; https://github.com/doomemacs/themes/
     ↪ pull/716
  ("^ *\\([-+]\\|\\(?:[0-9]+\\|[a-zA-Z]\\)[)
     ↪ .]\\)\\([ \t]\\)"
               (1 'org-list-dt append)
               (2 'org-indent append)))))

(add-hook 'org-font-lock-set-keywords-hook #'
    ↪ org-add-indent-face-to-prespace)
```

We can also make list bullets fixed-pitch, so they are even more aligned.

```
(after! mixed-pitch
  (add-to-list 'mixed-pitch-fixed-pitch-faces '
     ↪ org-list-dt))
```

(d) Fontify counter cookies

```
(defun org-fontify-counter-cookies ()
  (setq
   org-font-lock-extra-keywords
   (append org-font-lock-extra-keywords
           '(("^[ \t]*\\(?:[-+*]\\|\\(?:[0-9]+\\|[
               ↪ a-zA-Z]\\)[.)]\\)[ \t]+\\(\\[@\\(?:
               ↪ start:\\)?\\(?:[0-9]+\\|[a-zA-Z]\\)
               ↪ \\]\\)"
              (1 'org-property-value prepend))))))

(add-hook 'org-font-lock-set-keywords-hook #'
    ↪ org-fontify-counter-cookies)
```

2. Faces

   ```
   (custom-set-faces!
     '(font-latex-math-face :foreground unspecified)
     '(org-indent :inherit org-hide)
     '(org-headline-done :foreground unspecified)
     '(org-verse :inherit nil)
     '(font-latex-sedate-face :inherit nil))

   (custom-set-faces!
     '(org-link :weight normal))
   (custom-set-faces!
     `(outline-1 :weight bold :inherit nil)
     `(outline-2 :weight bold :inherit nil)
     `(outline-3 :weight bold :inherit nil)
     `(outline-4 :weight bold :inherit nil)
     `(outline-5 :weight bold :inherit nil)
     '(outline-6 :weight bold :inherit nil)
     '(outline-8 :weight bold :inherit nil)
     '(outline-9 :weight bold :inherit nil))
   ```

7.7.5 Bindings   bindings

```
(map! :mode 'org-mode
      :g "C-S-s" (cmd! (org-latex-export-to-pdf nil))
      :i "C-i" (cmd! (org-emphasize ?/))
      :i "C-b" (cmd! (org-emphasize ?*))
      :n "\\" #'org-edit-special
      :localleader "s p" #'org-paste-subtree)
```

1. Better emphasis toggle   advice

   If the point is before the emphasis marker, then `org-emphasize` should exit it.

   ```
   (defadvice! org-emphasize-emph-exit-ad (char)
     :before-until #'org-emphasize
     (when (eq (char-after) char)
       (forward-char)
       t))
   ```

2. `SPC s I` to open subtree with narrowing

   ```
   (map! :mode 'org-mode
         :map 'doom-leader-search-map
         "I" (cmd!
              (let ((this-buffer (current-buffer))
                    (pos (point)))
                (consult-imenu)
                (org-tree-to-indirect-buffer)
                (setq indirect-buffer (current-buffer))
                ;; with-current-buffer does not save pos for
                    ↪ some reason
                (switch-to-buffer this-buffer)
                (goto-char pos)
                (evil-scroll-line-to-center nil)
                (switch-to-buffer indirect-buffer))))
   ```

7.7.6 General config

```
(add-hook! org-mode
   <<org-mode-hook>>
   nil)

(set-popup-rule! "\*Org Src .+\*"
  :size 0.5)

(after! org
  (require 'org-src))
  ;; (add-to-list 'org-src-block-faces '("latex" (:inherit
     ↪ default :extend t)))
```

7.7.7 Linking

```
(setq +org-roam-link-to-org-use-id 'use-existing
      org-id-link-to-org-use-id 'use-existing)
```

7.7.8 Logging

```
(setq org-log-states-order-reversed t
      org-log-done 'note
      org-log-into-drawer t)
```

7.7.9 Minor

1. Org-appear

   ```
   (package! org-appear)

   (use-package org-appear
     :after org
     :init
     (defun org-appear-toggle ()
       (interactive)
       (if org-appear-mode
           (progn (setq org-hide-emphasis-markers nil)
                  (org-restart-font-lock)
                  (org-appear-mode -1))
         (setq org-hide-emphasis-markers t)
         (org-restart-font-lock)
         (org-appear-mode 1)))
     :config
     (setq org-appear-autolinks nil
           org-appear-inside-latex t
           org-appear-autosubmarkers t)
     (map! :mode 'org-mode :leader "t a" #'org-appear-toggle)
       ↪ )
   ```

2. Org-noter

   ```
   (setq org-noter-notes-search-path (list (expand-file-name
     ↪  "org-noter" org-roam-directory)))
   ```

3. Org-remark

   ```
   (package! org-remark)

   (use-package org-remark
     :after org
     :config)
   ```

4. Org-roam-timestamps   ARCHIVE

```
(package! org-roam-timestamps
  :recipe (:host github :repo "stites/org-roam-timestamps"
      ↪ )
  :pin "fbbe57a7d6283624e567bd1ee46ebea3d179a321")

(use-package! org-roam-timestamps
  :after org-roam
  :config (org-roam-timestamps-mode))
```

5. Org-superstar

```
(package! org-superstar)

(use-package org-superstar
  :after (org)
  :hook (org-mode . org-superstar-mode)
  :config
  (setq org-superstar-headline-bullets-list '(?  ?  ?  ?
      ↪  ?)))
        ;; org-superstar-headline-bullets-list '(""  ""  ""
            ↪   ""  ""  ""  ""  "")))
```

6. Org-transclusion   monkey

```
(package! org-transclusion)

(use-package! org-transclusion
  :after org
  :init
  (map!
   :map global-map "<f12>" #'org-transclusion-add
   :leader
   :prefix "n"
   :desc "Org Transclusion Mode" "t" #'
       ↪ org-transclusion-mode)
  (setq org-transclusion-exclude-elements '(
      ↪ property-drawer keyword)))

(after! org
  (defadvice! +org--recenter-after-follow-link-a (&rest
    ↪ _args)
    "Recenter after following a link, but only internal or
      ↪  file links."
    :after '(org-footnote-action
             org-follow-timestamp-link
             org-link-open-as-file
             org-link-search)
    (if-let* ((window (get-buffer-window)))
        (with-selected-window window
          (recenter)))))

(defun org-transclusion-add-better-id (link plist)
  "Return a list for Org-ID LINK object and PLIST.
Return nil if not found."
  (when (string= "id" (org-element-property :type link))
    ;; when type is id, the value of path is the id
    (let* ((both (split-string (org-element-property :path
      ↪  link) "::"))
           (id (cl-first both))
           (search (cl-second both))
           (mkr (ignore-errors (org-id-find id t)))
           (payload '(:tc-type "org-id")))
      (if mkr
          (append payload (
            ↪ org-transclusion-content-better-marker mkr
            ↪  search plist))
        (message
         (format "No transclusion done for this ID. Ensure
           ↪ it works at point %d, line %d"
                 (point) (org-current-line)))
        nil))))

(defun org-transclusion-content-better-marker (marker
    ↪ search plist)
  "Return a list of payload from MARKER and PLIST.
This function is intended to be used for Org-ID.  It
    ↪ delates the
work to
`org-transclusion-content-org-buffer-or-element'."
```

```
  (if (and marker (marker-buffer marker)
           (buffer-live-p (marker-buffer marker)))
      (progn
        (with-current-buffer (marker-buffer marker)
          (org-with-wide-buffer
           (goto-char marker)
           (when search
             (org-link-search search))
           (if (and (not search) (
               ↪ org-before-first-heading-p))
               (
                   ↪ org-transclusion-content-org-buffer-or-element
                   ↪
                nil plist)
             (
                 ↪ org-transclusion-content-org-buffer-or-element
                 ↪
              'only-element plist)))))
    (message "Nothing done. Cannot find marker for the ID.
        ↪ ")))

(defun org-transclusion-add-org-attach (link plist)
  "Return a list for attached file LINK object and PLIST.
Return nil if not found."
  (when (string= "attachment" (org-element-property :type
      ↪ link))
    (let* ((both (split-string (org-element-property :path
        ↪  link) "::"))
           (path (org-attach-expand (cl-first both)))
           (search (cl-second both))
           (link_ (org-element-put-property link :path path
               ↪ ))
           (link__ (org-element-put-property link_ :
               ↪ search-string search)))
      (or (org-transclusion-add-org-file link__ plist)
          (org-transclusion-add-other-file link__ plist))))
              ↪ )

(after! org-transclusion
  ;; (defadvice! org-transclusion-no-fringe ()
  ;; :override #'org-transclusion-propertize-source
  ;; nil)
  (setq! org-transclusion-add-functions
        '(org-transclusion-add-src-lines
          org-transclusion-add-better-id
          org-transclusion-add-org-attach
          org-transclusion-add-org-file
          org-transclusion-add-other-file)))
```

7.7.10 Gutter

O `git-gutter` não funciona bem com o `org-indent-mode`:

```
(push 'org-mode git-gutter:disabled-modes)
```

7.7.11 Hook

```
(setq-local auto-save-visited-interval 0.2
          display-line-numbers nil)
(setq line-spacing 5)
(add-to-list 'completion-at-point-functions #'cape-dabbrev)
```

7.7.12 Organon

```
(define-minor-mode organon-follow-mode
  "Set whether organon should follow your every move in Emacs.
    ↪ "
  :lighter " organon"
  :global t
  :group 'organon
  :init-value nil
  (if organon-follow-mode
      (progn
        (add-hook 'post-command-hook #'organon--update-position
          ↪ )
        (message "organon will now follow you around."))
    (remove-hook 'post-command-hook #'organon--update-position)
    (message "organon will now leave you alone.")))
```

```
(defvar organon--last-pos nil)
(defvar organon--conn nil)

(defun organon--connect ()
  (require 'websocket)
  (unless organon--conn
    (websocket-open
     "ws://127.0.0.1:9160"
     :on-open (lambda (ws) (message "organon: connected") (setq
         ↪  organon--conn ws))
     :on-close (lambda (ws) (message "organon: disconnected") (
         ↪ setq organon--conn nil)))))

(defun organon--get-info ()
  (list :id (org-entry-get nil "ID" t)
        :file (buffer-file-name)
        :anchor (or (org-entry-get nil "CUSTOM_ID")
                    (condition-case nil
                        (concat "h-" (nth 4 (
                            ↪ org-heading-components)))
                      (user-error nil)))))

(defun organon--update-position ()
  (when-let ((_ (eq major-mode 'org-mode))
             (cur-pos (organon--get-info))
             (_ (not (equal cur-pos organon--last-pos))))
    (setq organon--last-pos cur-pos)
    (send-to-organon)))

(defun send-to-organon ()
  (interactive)
  (organon--connect)
  (when organon--conn
    (let ((cur-info (organon--get-info)))
      (websocket-send-text organon--conn (json-encode cur-info)
          ↪ ))))
```

7.7.13 Exporting

```
(after! ox
  (add-to-list
   'org-export-smart-quotes-alist
   '("pt-br"
     (primary-opening :utf-8 """ :html "&ldquo;" :latex "``" :
         ↪ texinfo "``")
     (primary-closing :utf-8 """ :html "&rdquo;" :latex "''" :
         ↪ texinfo "''")
     (secondary-opening :utf-8 "‘" :html "&lsquo;" :latex "`" :
         ↪ texinfo "`")
     (secondary-closing :utf-8 "’" :html "&rsquo;" :latex "'" :
         ↪ texinfo "'")
     (apostrophe :utf-8 "’" :html "&rsquo;"))))
```

1. LaTeX

   ```
   (after! org
     ;; Note to self: NEVER change this!!! You have had
         ↪ problems with uncompilable documents when you
         ↪ changed the default preamble in the past!!
     (setq org-latex-packages-alist nil
           org-latex-preview-default-process 'dvisvgm)

     (after! ox-latex
       (setq org-latex-pdf-process '("latexmk -f -pdf -%latex
           ↪  -interaction=nonstopmode -output-directory=%o
           ↪ %f")
             org-latex-compilers '("tectonic" "pdflatex" "
                 ↪ xelatex" "lualatex")
             org-latex-compiler "xelatex"))

     (custom-reevaluate-setting '
         ↪ org-latex-preview-process-alist)

     ;; To avoid issues with pdfcrop
     (plist-put! org-format-latex-options
                 :background "Transparent"
                 :scale 2.0))
   ```

7.7.14 Org-roam

1. Variables

   (a) Common

   Shared with desktop and termux.

   ```
   (after! org-roam
     (setq org-id-method 'ts
           org-roam-completion-everywhere nil))
   ```

   (b) Desktop-specific

   Those aren't shared with termux.

   ```
   (after! org-roam
     (setq org-roam-directory "~/dados/notas"
           org-roam-node-display-template
           #("${doom-hierarchy} ${doom-type} ${doom-tags}
               ↪ " 18 30
             (face font-lock-comment-face)
             31 43
             (face font-lock-comment-face))))
   ```

2. Protocol

   ```
   (after! org-roam
     (require 'org-roam-protocol))
   ```

3. Switch workspace

   ```
   (defadvice! yeet/org-roam-in-own-workspace-a (&rest _)
     "Open all roam buffers in their own workspace."
     :before #'org-roam-node-find
     :before #'org-roam-node-random
     :before #'org-roam-buffer-display-dedicated
     :before #'org-roam-buffer-toggle
     (when (modulep! :ui workspaces)
       (+workspace-switch "notas" t))
     (when (functionp 'tabspaces-switch-or-create-workspace)
       (tabspaces-switch-or-create-workspace "notas")))
   ```

4. Capture Templates

(a) Roam

```
(after! org-roam
  (setq org-roam-capture-templates
        '(("d" "default" plain "%?"
           :target (file+head "%<%Y%m%d%H%M%S>.org" "#+
               ↪ title: ${title}
#+language: pt
")
           :unnarrowed t)
          ("m" "math" plain "%?"
           :target (file+head "math/%<%Y%m%d%H%M%S>.org
               ↪ " "#+title: ${title}
#+language: pt
")
           :unnarrowed t))))
```

(b) Roam-ref

```
(setq org-roam-capture-ref-templates
      '(("m" "math" plain "%?"
           :target (file+head "math/%<%Y%m%d%H%M%S>.org
               ↪ " "#+title: ${title}\n\n${body}")
           :unnarrowed t)
        ("fr" "Add to my future-read list" entry "* ${
            ↪ title}\n%?"
         :target (file+olp "to-read.org" ("${title}"))
         :empty-lines-before 1 nil nil)
        ("r" "ref" plain "%?" :target
         (file+head "${slug}.org" "#+title: ${title}")
         :unnarrowed t)))
```

5. Workarounds

```
(defadvice! inhibit-redisplay-on-roam-autosync (fn)
  "Inhibit redisplay when syncing roam database on saves."
  :around #'org-roam-db-autosync--try-update-on-save-h
  (let ((inhibit-redisplay t)) (funcall fn)))

(defadvice! org-roam-db-insert-link--remove-search-a (f
    ↪ link)
  :around #'org-roam-db-insert-link
(let ((newpath (car (split-string (org-element-property
   ↪ :path link) "::"))))
  (funcall f (org-element-put-property link :path
     ↪ newpath))))
```

6. md-roam

   ```
   (package! md-roam :recipe (:host github :repo "nobiot/
      ↪ md-roam"))
   ```

7.7.15 Org-roam-ui

```
(unpin! org-roam)
(package! org-roam-ui)

(use-package! websocket
    :after org-roam-ui)

(use-package! org-roam-ui
  :after org-roam
  :commands (org-roam-ui-mode)
  :config
  (setq org-roam-ui-sync-theme t
        org-roam-ui-follow t
        org-roam-ui-update-on-save t))
```

7.7.16 Org-protocol

```
(after! org-protocol
  (add-to-list 'org-protocol-protocol-alist
             '("org-file" :protocol "org-file"
               :function org-protocol-goto-org-file)))

(defun org-protocol-goto-org-file (info)
  (if-let ((id (plist-get info :id)))
      (org-id-goto id)
    (when-let ((file (plist-get info :file)))
      (org-open-file file)))
  nil)
```

1. Raise frame

   Requires this extension to be installed.

   ```
   (defun org-protocol/select-current-frame ()
     ;; gnome stuff
     ;; (shell-command-to-string
     ;; (format "gdbus call \
     ;; --session \
     ;; --dest org.gnome.Shell \
     ;; --object-path /org/gnome/Shell/Extensions/Windows \
     ;; --method org.gnome.Shell.Extensions.Windows.List \
     ;; | cut -c 3- | rev | cut -c4- | rev \
     ;; | jq '.[] | select(.pid == %s) .id'" (emacs-pid)))))
     ;; (dbus-call-method
     ;; :session
     ;; "org.gnome.Shell"
     ;; "/org/gnome/Shell/Extensions/Windows"
     ;; "org.gnome.Shell.Extensions.Windows"
     ;; "Activate"
     ;; wid))
     ;; this should be compatible with other WMs
     (select-frame-set-input-focus (selected-frame)))
   (add-hook 'org-capture-mode-hook 'org-protocol/
       ↪ select-current-frame)
   ```

7.7.17 Unfill instead of fill

With time, I came to the conclusion that having no line breaks inside paragraphs works better in most Org documents. So instead of `org-fill-` ↪ `paragraph`, I'll bind `M-q` to some sort of "`org-unfill-paragraph`".

```
(defun org-unfill-paragraph ()
  (interactive)
  (let ((fill-column most-positive-fixnum))
    (org-fill-paragraph)))

(map! :map 'org-mode-map "M-q" #'org-unfill-paragraph)
```

7.7.18 Vulpea

```
(package! vulpea)
```

```
(use-package! vulpea
  :hook ((org-roam-db-autosync-mode .
      ↪ vulpea-db-autosync-enable)))
```

7.7.19 Writing mode

```
(defvar-local org-writing-mode--previous-mixed-pitch nil)
(defvar-local org-writing-mode--previous-line-numbers nil)
(defvar-local org-writing-mode--previous-line-spacing 0)
(defvar-local org-writing-mode--frame nil)

(define-minor-mode org-writing-mode
  "Minor mode for writing in org."
  :init-value nil
  :lighter nil
  (require 'mixed-pitch)
  (unless (frame-live-p org-writing-mode--frame)
    (setq-local org-writing-mode--frame (selected-frame)))
  (if org-writing-mode
      (progn (setq-local
              org-writing-mode--previous-mixed-pitch
                  ↪ mixed-pitch-mode
              org-writing-mode--previous-line-numbers
                  ↪ display-line-numbers
              org-writing-mode--previous-line-spacing
                  ↪ line-spacing
              display-line-numbers nil
              line-spacing 5)
             (mixed-pitch-mode +1)
             ;; (set-frame-parameter org-writing-mode--frame '
                 ↪ internal-border-width 30)
             (evil-normal-state))
    (mixed-pitch-mode (if
        ↪ org-writing-mode--previous-mixed-pitch 1 -1))
    ;; (set-frame-parameter org-writing-mode--frame '
        ↪ internal-border-width 0)
    (setq display-line-numbers
        ↪ org-writing-mode--previous-line-numbers
        line-spacing org-writing-mode--previous-line-spacing))
           ↪ )

(map! :mode 'org-mode :leader "t z" #'org-writing-mode)
```

7.8 Web

```
(setq-default web-mode-code-indent-offset 2
              web-mode-markup-indent-offset 2)
```

7.9 Yuck mode

```
(package! yuck-mode)
```

8 Scientific writing (mostly mathematics)

8.1 Mamimo

```
(package! mamimo
  :type 'local
  :recipe (:local-repo "lisp/lib/mamimo"
           :files ("*.el" "snippets")))

(use-package mamimo
  :commands (mamimo-mode
             mamimo-yas-mode
             mamimo-smartkeys-mode)
  :config
  (add-hook! 'mamimo-mode-hook (evil-tex-mode +1)))
```

8.2 Abbrev

8.2.1 Language & math predicate

```
(defsubst abbrev/math-text-lang-p (lang)
  (and (mamimo-notmathp)
       (-any (lambda (k) (string= lang (cadr k)))
             (org-collect-keywords '("language")))))
```

```
(defun abbrev/math-text-pt-p () (abbrev/math-text-lang-p "pt")
   ↪ )
(defun abbrev/math-text-en-p () (abbrev/math-text-lang-p "en")
   ↪ )
```

8.2.2 Textual abbrevs

```
(setq abbrev/math-text-abbrevs-pt
  '(("pa" "podemos assumir")
    ("pd" "por definição")
    ("ie" "i.e.")
    ("tq" "tal que")
    ("spg" "sem perda de generalidade")
    ("qtp" "q.t.p.")
    ("sss" "se, e somente se,")
    ("li" "linearmente independentes")))

(setq abbrev/math-text-abbrevs-en
  '(("wlog" "without loss of generality")
    ("iff" "if and only if")
    ("ie" "i.e.")
    ("st" "such that")
    ("ae" "a.e.")
    ("bd" "by definition")
    ("li" "linearly independent")))
```

8.2.3 Variable abbrevs

```
(setq abbrev/var-abbrevs-pt '(b c d f g h i j k l m n p q r s
   ↪ t u v w x y z))
(setq abbrev/var-abbrevs-en '(b c d e f g h j k l m n o p q r
   ↪ s t u v w x y z))

(defun abbrev/compile-var-abbrevs (abbrevs)
  (mapcar (lambda (s) (list (symbol-name s) (format "\\(%s\\)"
     ↪  s) nil :system t))
        abbrevs))
```

8.2.4 Tables and mode-local tables

```
(setq abbrev/tables
  `((abbrev/math-text-pt-table
     ,(append
       abbrev/math-text-abbrevs-pt
       (abbrev/compile-var-abbrevs abbrev/var-abbrevs-pt))
     abbrev/math-text-pt-p)
    (abbrev/math-text-en-table
     ,(append
       abbrev/math-text-abbrevs-en
       (abbrev/compile-var-abbrevs abbrev/var-abbrevs-en))
     abbrev/math-text-en-p)))

(defun abbrev/setup ()
  (require 'abbrev)
  (setq-local local-abbrev-table nil)
  (pcase-dolist (`(,name ,defs ,cond) abbrev/tables)
   (define-abbrev-table name defs :enable-function cond)
   (push (symbol-value name) local-abbrev-table))
  (abbrev-mode +1))

(add-hook 'mamimo-mode-hook #'abbrev/setup)
```

9 ORGANIZE Bindings

9.1 Comandos familiares

Porque ninguém merece tantos atalhos diferentes.

```
(map! :g "C-s" 'save-buffer)
;; (map! :g "C-/" 'evilnc-comment-or-uncomment-lines)

(map! :i "C-v" 'yank)
(map! :i "C-z" 'evil-undo)
(map! :i "C-S-z" 'evil-redo)
(map! :i "C-x" 'evil-delete)
(map! :g "C-<backspace>" 'evil-delete-backward-word)
```

9.2 Linhas visuais

```
(map! :map evil-motion-state-map
     "j" 'evil-next-visual-line
     "k" 'evil-previous-visual-line
     "<down>" 'evil-next-visual-line
     "<up>" 'evil-previous-visual-line)
```

9.3 Hydras

Uma história antiga.

Não gosto do estilo do pop up

```
(setq hydra-is-helpful nil)
```

Tamanho da janela

```
(defhydra window-height-hydra (evil-window-map)
  "window height"
  ("=" evil-window-increase-height "")
  ("-" evil-window-decrease-height "")
  (">" evil-window-increase-width "")
  ("<" evil-window-decrease-width ""))

;; (defhydra workspace-hydra (doom-leader-workspace-map)
;; "workspace"
;; ("]" +workspace/switch-right "")
;; ("[" +workspace/switch-left "")
;; ("}" +workspace/swap-right "")
;; ("{" +workspace/swap-left ""))
```

9.4 Kitty (Terminal)

```
(map! :prefix-map ("\x80" . "kitty C map")
     :map 'key-translation-map
     "/" "C-/")

(map! :prefix-map ("\x81" . "kitty C-S map")
     :map 'key-translation-map
     "z" (kbd "C-S-z"))
```

9.5 Leader edit key

```
(map! :leader
      :prefix ("e" . "edit")
      :desc "New snipet" "s" #'+snippets/new
      :desc "New alias" "a" #'+snippets/new-alias)

(map! :i "C-M-x" ctl-x-map)
```

10 Workarounds

10.1 Doom

Those may become obsolete in the future.

```
(unpin! straight)
```

```
;; for some reason consult preview is not working in +default/
   ↪ p-s
(map! :leader "/" #'+vertico/project-search)
```