Contents

1 packages

1.1 melpa

1.1.1 exec-path-from-shell

```
(use-package exec-path-from-shell
  :ensure t
  :config (exec-path-from-shell-initialize))
```

1.1.2 ispell

```
(use-package ispell
  :ensure t)
```

1.1.3 popup-kill-ring

```
(use-package popup-kill-ring
  :ensure t
  :bind ("M-y" . popup-kill-ring))
```

1.1.4 sudo-edit

```
(use-package sudo-edit
  :ensure t)
```

1.1.5 hungry-delete

```
(use-package hungry-delete
  :ensure t
  :config (global-hungry-delete-mode))
```

1.1.6 which-key

```
(use-package which-key
  :ensure t
  :init
  (which-key-mode))
```

1.1.7 avy

```
(use-package avy
  :ensure t
  :bind
  ("C-c C-s" . avy-goto-char))
```

1.1.8 rainbow

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

1.2 local

1.2.1 odin-mode

```
(load-file "~/.emacs.d/packages/odin-mode.el")
```

1.2.2 glsl-mode

```
(load-file "~/.emacs.d/packages/glsl-mode.el")
```

2 major mode settings

2.1 org

2.1.1 language support

1. setup

   ```
   (setq org-confirm-babel-evaluate nil)
   (org-babel-do-load-languages
    'org-babel-load-languages
    '((python . t)
      (haskell . t)))

   (setq org-babel-python-command "python3")
   ```

2. templates

   ```
   (add-to-list 'org-structure-template-alist
                '("el" . "src emacs-lisp"))
   (add-to-list 'org-structure-template-alist
                '("py" . "src python"))
   (add-to-list 'org-structure-template-alist
                '("hs" . "src haskell"))
   (require 'org-tempo)
   ```

2.1.2 set default buffer to org mode

```
(setq initial-major-mode 'org-mode)
(setq initial-scratch-message nil)
```

2.1.3 config

```
(setq org-src-window-setup 'current-window)

(with-eval-after-load 'org
  (setq org-startup-indented t)
  (add-hook 'org-mode-hook #'visual-line-mode))
```

```
(setq org-latex-tables-centered nil)

(setq user-full-name "Nick Celestin Zizic")
```

2.1.4 fancy bullets

```
(use-package org-bullets
  :ensure t
  :config
  (add-hook 'org-mode-hook (lambda () (org-bullets-mode))))
```

2.1.5 LaTex stuff

```
(setq org-latex-with-hyperref nil)
(setq org-latex-pdf-process
  '("xelatex -interaction nonstopmode %f"
    "xelatex -interaction nonstopmode %f")) ;; for multiple passes

(unless (boundp 'org-export-latex-classes)
  (setq org-export-latex-classes nil))

(with-eval-after-load "ox-latex"
  (add-to-list 'org-latex-classes
               '("article"
                 "\\documentclass[a4paper, 12pt]{article}
\\usepackage[T1]{fontenc}
\\usepackage{fontspec}
\\usepackage{graphicx}
\\usepackage{lipsum}
\\usepackage[left=1in,top=1in,right=1in,nohead,nofoot]{geometry}
\\usepackage[compact]{titlesec}
\\defaultfontfeatures{Mapping=tex-text}
\\setromanfont{Gentium}
\\setromanfont [BoldFont={Gentium Basic Bold},
                ItalicFont={Gentium Basic Italic}]{Gentium Basic}
\\setsansfont{Charis SIL}
\\setmonofont[Scale=0.8]{DejaVu Sans Mono}
\\geometry{a4paper, textwidth=8.5in, textheight=10in,
            marginparsep=7pt, marginparwidth=1in}
\\pagestyle{fancy}
\\linespread{1.5}
      [NO-DEFAULT-PACKAGES]
      [NO-PACKAGES]"
                 ("\\section{%s}" . "\\section*{%s}")
                 ("\\subsection{%s}" . "\\subsection*{%s}")
                 ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
                 ("\\paragraph{%s}" . "\\paragraph*{%s}")
                 ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))

               ;; beamer class, for presentations
               '("presentation"
                 "\\documentclass\[presentation\]\{beamer\}"
                 ("\\section\{%s\}" . "\\section*\{%s\}")
                 ("\\subsection\{%s\}" . "\\subsection*\{%s\}")
                 ("\\subsubsection\{%s\}" . "\\subsubsection*\{%s\}"))))
(setq org-latex-title-command (concat
                               "\\begin{titlepage}\n"
                               "\\vspace{1in}\n"
                               "{\\normalsize %a \\par}\n"
                               "{\\LARGE \\today \\par}\n"
                               "\\centering"
                               "{\\LARGE %t \\par}\n"
                               "\\end{titlepage}\n"))
```

2.2 ido

2.2.1 enable ido

```
(setq ido-enable-flex-matching nil)
(setq ido-create-new-buffer 'always)
(setq ido-everywhere t)
(ido-mode 1)

;; vertical autocomplete

(use-package ido-vertical-mode
  :ensure t
  :init
  (ido-vertical-mode 1))
```

```
(setq ido-vertical-define-keys 'C-n-and-C-p-only)
```

2.2.2 smex

```
(use-package smex
  :ensure t
  :init (smex-initialize)
  :bind ("M-x" . smex))
```

2.3 shell

```
(define-key shell-mode-map (kbd "C-p") 'comint-previous-input)
(define-key shell-mode-map (kbd "C-n") 'comint-next-input)
```

3 minor mode settings

3.1 display settings

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

(column-number-mode)
(global-subword-mode 1)

(when window-system (global-hl-line-mode t))
(when window-system (global-prettify-symbols-mode t))
```

3.2 programming modes

3.2.1 haskell

```
(use-package haskell-mode
  :ensure t)
```

3.2.2 rust

```
(use-package rustic
  :ensure t
  :bind (:map rustic-mode-map
              ("M-j" . lsp-ui-imenu)
```

```
            ("M-?" . lsp-find-references)
            ("C-c C-c l" . flycheck-list-errors)
            ("C-c C-c a" . lsp-execute-code-action)
            ("C-c C-c r" . lsp-rename)
            ("C-c C-c q" . lsp-workspace-restart)
            ("C-c C-c Q" . lsp-workspace-shutdown)
            ("C-c C-c s" . lsp-rust-analyzer-status))
  :config
  (setq rustic-format-on-save nil)
  (add-hook 'rustic-mode-hook 'rk/rustic-mode-hook))

(defun rk/rustic-mode-hook ()
  ;; so that run C-c C-c C-r works without having to confirm
  (setq-local buffer-save-without-query t))
```

4 other settings

4.1 spacing and tabs

```
(setq-default tab-width 2)
(setq-default indent-tabs-mode nil)
```

4.2 follow symlinks

```
(setq vc-follow-symlinks t)
```

4.3 electric pairs

```
(setq electric-pair-pairs '(
          (?\( . ?\))
          (?\[ . ?\])
          (?\{ . ?\})
          (?\" . ?\")
          ))
(electric-pair-mode t)
```

4.4 higlight matching pairs

```
(require 'paren)
(setq show-paren-style 'parenthesis)
```

```
(show-paren-mode 1)
```

4.5 minor settings

```
(setq inhibit-startup-message t)
(setq scroll-conservatively 100)
(setq ring-bell-function 'ignore)
(setq make-backup-files nil)
(setq auto-save-default nil)
(setq-default indent-tabs-mode nil)
```

4.6 reload buffer without confirmation

```
(setq revert-without-query '(".+"))
```

5 buffer settings

5.1 enable ibuffer

```
(global-set-key (kbd "C-x C-b") 'ibuffer)
(setq ibuffer-expert t)
```

5.2 mode specific rebinds

```
;;  (define-key ibuffer-mode-map (kbd "C-k") nil)
```

5.3 switching buffers

```
(global-set-key (kbd "C-x b") 'ido-switch-buffer)
```

5.4 kill current buffer

```
(global-set-key (kbd "C-c k") 'kill-current-buffer)
```

6 unsetting and resetting movement keys

```
(global-unset-key (kbd "C-n"))
(global-unset-key (kbd "C-p"))
(global-unset-key (kbd "C-b"))
(global-unset-key (kbd "C-f"))
```

```
(global-unset-key (kbd "C-h"))
(global-unset-key (kbd "C-j"))
(global-unset-key (kbd "C-k"))
(global-unset-key (kbd "C-l"))

(global-set-key (kbd "C-h") 'backward-char)
(global-set-key (kbd "C-j") 'next-line)
(global-set-key (kbd "C-k") 'previous-line)
(global-set-key (kbd "C-l") 'forward-char)

(global-set-key (kbd "C-p") 'eval-print-last-sexp)

(bind-key "C-d" 'kill-whole-line)

(unbind-key "C-k" org-mode-map)
(unbind-key "C-j" lisp-interaction-mode-map)
(define-key org-mode-map (kbd "C-j") nil)

(global-set-key (kbd "C-'") 'recenter-top-bottom)
```

7 convenience functions

7.1 reload-config

```
(defun config-reload ()
  (interactive)
  (org-babel-load-file (expand-file-name "~/.emacs.d/config.org")))
(global-set-key (kbd "C-c r") 'config-reload)
```

7.2 edit-config

```
(defun config-edit ()
  (interactive)
  (find-file "~/.emacs.d/config.org"))
(global-set-key (kbd "C-c e") 'config-edit)
```

7.3 kill-whole-word

```
(defun kill-whole-word ()
  (interactive)
  (backward-word)
  (kill-word 1))

(global-set-key (kbd "C-c C-w") 'kill-whole-word)
```

7.4 kill-current-buffer

```
(global-set-key (kbd "C-x k") 'kill-current-buffer)
```

7.5 copy-whole-line

```
(defun copy-whole-line ()
  (interactive)
  (save-excursion
    (kill-new
     (buffer-substring
(point-at-bol)
(point-at-eol)))))
(global-set-key (kbd "C-c y") 'copy-whole-line)
```

8 auto completion

```
(use-package company
  :ensure t
  :init
  (add-hook 'after-init-hook 'global-company-mode))
```

9 mode line

9.1 disable minor modes on modeline

```
(use-package diminish
  :ensure t
  :init
  (diminish 'hungry-delete-mode)
  (diminish 'beacon-mode)
  (diminish 'which-key-mode)
  (diminish 'company-mode)
  (diminish 'subword-mode)
  (diminish 'rainbow-mode))
```

10 fonts

10.1 fira

```
(add-to-list 'default-frame-alist '(font . "Fira Code-12"))

(use-package fira-code-mode
  :ensure t
  :custom (fira-code-mode-disabled-ligatures '("[]" "x"))
  :hook prog-mode)
```