```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; A few things before you read my .emacs:
;; -I've never learnt emacs-Lisp, so my programming style is awful.
;;  Those are mainly adapted examples, and I let you read this file
;;  because I enjoyed reading other's configuration files.
;; -Don't learn all the shortcuts in one day! They were added and learnt over a long time.
;; -Comments invited!
;; -You'll need AucTeX & Tuareg to use it out of the box. Under Debian or Ubuntu, simply run:
;;    apt-get install auctex tuareg-mode
;; -I also use the following packages (you can ask me details about them), but including
;;  all here would require too much installation before being able to use this .emacs:
;;  mmm multiple majoe modes in one buffer
;;     (like OCaml code samples inside \begin{verbatim} ... \end{verbatim} in LaTeX-mode)
;;  doxymacs doxygen configuration file
;;  rs-ucs-coding-system (I work with people under Windows, who use windows1252 encoding)
;;  WhizzyTeX "real" time display of the result of compiling the .tex file you're editing.
;; -sources: friends, the emacs manual, http://paulp.nerim.net/tex/4P.emacs.el,
;;    my account in cscs.umich.edu, www.emacsfr.org, http://diwww.epfl.ch/~fleuret


;; custom-set-variables
;; "custom" erases the comments when modifying a variable,
;;  but works better than hand-set variables
(custom-set-variables
  ;; custom-set-variables was added by Custom -- don't edit or cut/paste it!
  ;; Your init file should contain only one such instance.
 '(case-fold-search t)
 '(comment-column 0 t)
 '(current-language-environment "UTF-8")
 '(default-input-method "rfc1345")
 '(fill-column 80)
 '(find-tag-marker-ring-length 100)
 '(global-font-lock-mode t nil (font-lock))
 '(save-place t nil (saveplace))
 '(show-paren-mode t nil (paren))
 '(tab-width 2)
 '(transient-mark-mode t)
 '(tuareg-interactive-scroll-to-bottom-on-output t))
;; comment-column 0 t ;when adding a comment at end of line, start as left as possible
;; fill-column 80 ;width of the text
;; save-place t nil (saveplace) ;restore the position of the cursor when reopening a file

(setq
 case-fold-search t ;searches and matches should ignore case
 column-number-mode t ;display the column number
 confirm-kill-emacs nil ;don't confirm when exiting emacs
 kill-ring-max 20 ;number of remembered cutted texts
 show-paren-style 'mixed ;my favorite style of highlighting matching parentheses
 standard-indent 2 ;don't go to fast towards right
 transient-mark-mode t;show the region
 scroll-preserve-screen-position t ;don't move the cursor when scrolling
; doxymacs: emacs mode for doxygen, a source code documentation generator
 european-calendar-style t
 calendar-week-start-day 1
 calendar-day-name-array ["Dimanche" "Lundi" "Mardi" "Mercredi" "Jeudi" "Vendredi" "Samedi"]
 calendar-month-name-array
   ["Janvier" "Février" "Mars" "Avril" "Mai" "Juin"
    "Juillet" "Août" "Septembre" "Octobre" "Novembre" "Décembre"]
 frame-title-format "%b (%f)" ;the title-bar displays the filename of the current buffer
 font-lock-maximum-decoration t ;as colored as possible
 visible-bell t ;stop beeping
 show-paren-delay 0;Show the matching immediately
 default-indicate-empty-lines t;show me empty lines at the end of the buffer
 inhibit-startup-message t;no startup message
 next-line-add-newlines t ; for use with comment-and-go-down, see below
 kill-whole-line t;take the CR when killing a line
 x-stretch-cursor t;when on a TAB, the cursor has the TAB length
;;  require-final-newline t ;adds a newline at the end of the file beeing saved
   ; if it doesn't already have one
 compile-command '"make"
)

(show-paren-mode t) ;highlight matching parentheses
(global-font-lock-mode t) ;colorize code
(prefer-coding-system 'utf-8) ;when guessing encoding of a file, first try utf-8
(fset 'yes-or-no-p 'y-or-n-p); Make all "yes or no" prompts show "y or n" instead
(tool-bar-mode 0) ;no buttons, thanks
;; (menu-bar-mode 0) ;no menu either

; Add to the "File" menu a list of recently opened files.
(require 'recentf nil t)
(when (fboundp 'recentf-mode) (recentf-mode 1))
(global-set-key "\C-o" 'recentf-open-files) ;displays this menu in a buffer

;show white spaces (in "pale turquoise", see "custom-set-faces" below) at the end of lines
```

```
(when (= emacs-major-version 21)
  (setq-default show-trailing-whitespace t))

(custom-set-faces
  ;; custom-set-faces was added by Custom -- don't edit or cut/paste it!
  ;; Your init file should contain only one such instance.
 '(default ((t (:stipple nil :background "#ffffff" :foreground "#000000" :inverse-video nil :b
ox nil :strike-through nil :overline nil :underline nil :slant normal :weight normal :height 6
5 :width condensed :family "misc-fixed"))))
 '(trailing-whitespace ((((class color) (background light)) (:foreground unspecified :strike-t
hrough "chartreuse2")))))

;; geometry of the opened window, I have a big screen, please adapt to yours.
(setq default-frame-alist
      '((top . 000) (left . 1000) (width . 150) (height . 90)))

;; Time stamps
;; write
;;   Time-stamp: <>
;; in your file and save it to see time stamps at work.
;; Other time stamp format: "%a %b %e, %Y %l:%M %p"
(add-hook 'write-file-hooks 'time-stamp)
(setq time-stamp-active t)
(setq time-stamp-format "%:a %02d %:b %:y, %02H:%02M:%02S, %u")
(setq european-calendar-style 't)
;; first day of the week is monday instead of sunday:
(setq calendar--week--start--day 1)

;; mouse wheel: try it with S and C
(defun up-slightly () (interactive) (scroll-up 5))
(defun down-slightly () (interactive) (scroll-down 5))
(global-set-key [mouse-4] 'down-slightly)
(global-set-key [mouse-5] 'up-slightly)

(defun up-one () (interactive) (scroll-up 1))
(defun down-one () (interactive) (scroll-down 1))
(global-set-key [S-mouse-4] 'down-one)
(global-set-key [S-mouse-5] 'up-one)

(defun up-a-lot () (interactive) (scroll-up))
(defun down-a-lot () (interactive) (scroll-down))
(global-set-key [C-mouse-4] 'down-a-lot)
(global-set-key [C-mouse-5] 'up-a-lot)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;customization of major modes
;; Makefile: switch to makefile-mode whenever opening a file whose name contains "Makefile"
(setq auto-mode-alist (append (list
  '("\\.mpl$" . maplev-mode)
  '("\\.cfg$" . python-mode)
  '("\\.crm$" . crm114-mode)
  '("/etc/apache2/.*\\.\\(conf$\\|load$\\)" . apache-mode)
  '("\\.r$" . R-mode)
  '("\\.ahk$" . ahk-mode)
  '("Makefile" . makefile-mode)
  ) auto-mode-alist))

;; switch to python-mode whenever opening a file whose name ends with ".py"
(autoload 'python-mode "python-mode" "" t)
(add-to-list 'auto-mode-alist '("\\.py$" . python-mode))

;; nyquist scripts for Audacity
(add-to-list 'auto-mode-alist '("\\.ny$" . lisp-mode))
;; configuration files for Unison (a file synchronizer, useful for backups)
(add-to-list 'auto-mode-alist '("\\.prf$" . shell-script-mode))

(add-hook 'metapost-mode-hook 'turn-on-auto-fill)

(add-hook 'emacs-lisp-mode-hook '(lambda () (setq comment-column 0)))
;this mode seems to undo my global setting of comment-column, so let's redo it

(add-hook 'text-mode-hook '(lambda ()
;;   (turn-on-auto-fill) ;automatically wrap long lines (if uncommented)
  (text-mode-hook-identify))) ;asked by the man, I didn't tried to know why

(when (locate-library "javascript")
  (autoload 'javascript-mode "javascript" nil t)
  (add-to-list 'auto-mode-alist '("\.js$" . javascript-mode))
)

(defun gnuplot-eval-line-and-next-line () (interactive)
  (gnuplot-send-line-to-gnuplot); C-cC-l also work
  (next-line 1))
(add-to-list 'auto-mode-alist '("\\.gplt$" . gnuplot-mode))
(add-hook 'gnuplot-mode-hook '(lambda ()
```

```
  (global-set-key [f11] 'gnuplot-send-region-to-gnuplot)
  (global-set-key [f12] 'gnuplot-eval-line-and-next-line)
  ))

(defun ess-eval-line-and-next-line () (interactive)
;;   (ess-eval-line ess-eval-visibly-p); C-cC-j also work
  (ess-eval-line nil); C-cC-j also work
  (next-line 1))
(add-hook 'ess-mode-hook '(lambda ()
  (global-set-key [f11] 'ess-eval-region)
  (global-set-key [f12] 'ess-eval-line-and-next-line)
  (text-mode-hook-identify))) ;asked by the man, I didn't tried to know why

;; use html-helper-mode for file ending with ".html". .htm is set by default
(add-to-list 'auto-mode-alist '("\\.html$" . html-helper-mode))
(setq html-script-toggle-key [f5]) ;default is f4, but I already use it for kill-this-buffer
;; customize the auto-updated text "last update on ..." that html-helper-mode inserts:
(load-library "calendar.elc")
(defun html-helper-custom-insert-timestamp ()
  "Custom timestamp insertion function."
  (insert "Derni&egrave;re modification : "
          (calendar-date-string (calendar-current-date))))
(setq html-helper-timestamp-hook 'html-helper-custom-insert-timestamp)


;;;;;;;;;;
;; LaTeX
;; (add-to-list 'auto-mode-alist '("\\.tex$" . latex-mode))
(add-hook 'tex-mode-hook (define-key global-map [(control return)] 'switch-to-buffer))
(add-hook 'bibtex-mode-hook 'turn-on-auto-fill)
;; AUCTeX is a good mode for editing LaTeX code, see the AUCTeX manual
(require 'tex-site nil t)
(setq TeX-default-mode 'LaTeX-mode); never open a file in TeX-mode, use LaTeX
(setq
;;   LaTeX-math-abbrev-prefix "!";I have an azerty keyboard, the default "'" is unconvenient
  LaTeX-math-list '(;most useful for me, not necessarily for you
    (?i "infty" nil) (?v "vee" nil)
    (?q "quad"  nil) (?Q "qquad"  nil)
    ))
(add-hook 'LaTeX-mode-hook '(lambda ()
;;   (turn-on-auto-fill)
  (turn-on-reftex) ;see the reftex man. Useful for labels and navigating among \section
  (LaTeX-math-mode)
  (setq
    TeX-open-quote  "«";text inserted when you type a double quote : "
    TeX-close-quote "»";idem, but at the end of a word. I'm french...
    )
  (setq LaTeX-section-hook;I don't want to put useless labels everywhere
  '(LaTeX-section-heading
    LaTeX-section-title
    LaTeX-section-section))
  (defun TeX-insert-dollar () "custom redefined insert-dollar" (interactive)
    (insert "$$")      ;in LaTeX mode, typing "$" automatically insert  "$$"
    (backward-char 1)) ;and go between them: no more matching problems!
  (setq ispell-tex-skip-alists
    (list
     (append
      (car ispell-tex-skip-alists) ;tell ispell to ignore content of this:
      '(("\\\\cite"            ispell-tex-arg-end)
        ("\\\\nocite"          ispell-tex-arg-end)
        ("\\\\includegraphics" ispell-tex-arg-end)
        ("\\\\bibliography"    ispell-tex-arg-end)
        ("\\\\ref"             ispell-tex-arg-end)
        ("\\\\web"             ispell-tex-arg-end) ;personal
        ("\\\\code"            ispell-tex-arg-end) ;personal
        ("\\\\label"           ispell-tex-arg-end)))
     (cadr ispell-tex-skip-alists)))
;;   ;; optionnaly add support for customized (not my fault!) versions of \section:
;;   (setq reftex-section-levels (append
;;      '(("partie" . -1) ("Partie" . -1) ("souspartie" . -2) ("question" . -3))
;;      reftex-section-levels))
;;   (setq LaTeX-section-list (append
;;      '(("partie" 1) ("Partie" 1) ("souspartie" 2) ("question" 3))
;;      LaTeX-section-list))
))
;; (setq font-latex-match-title-1-keywords '("partie" "Partie" ))
;; (setq font-latex-match-title-2-keywords '("souspartie" "question"))


;;;;;;;;;;
;;OCaml
;;ad : OCaml is great ! see caml.inria.fr
(add-hook 'tuareg-mode-hook '(lambda ()
  (setq
   tuareg-in-indent 0 ; no extra indentation after 'in' keywords, should be set everywhere
```

```
   ;then comes personal taste indenting:
   tuareg-with-indent 0 tuareg-function-indent 0 tuareg-parse-indent 0
   comment-column 0)
  (global-set-key [f10] 'caml-types-show-type); requires caml-types
  (global-set-key [f11] 'tuareg-eval-region); C-cC-r
  (global-set-key [f12] 'tuareg-eval-phrase); C-cC-e
  ))
;;OCaml: automatically launch the tuareg mode when opening a file ending with .ml, .mli etc.
(add-to-list 'auto-mode-alist '("\\.ml\\w?" . tuareg-mode))
(autoload 'tuareg-mode "tuareg" "Major mode for editing Caml code" t)
(autoload 'camldebug "camldebug" "Run the Caml debugger" t)
(autoload 'caml-types-show-type "caml-types" "Show type of expression / pattern at point." t)

(if (featurep 'sym-lock)
    (setq tuareg-sym-lock-keywords
    '(("<-" 0 1 172 nil) ("->" 0 1 174 nil)
      ;; (":=" 0 1 220 nil)
      ("<=" 0 1 163 nil) (">=" 0 1 179 nil)
      ("<>" 0 1 185 nil) ("==" 0 1 186 nil)
      ("||" 0 1 218 nil) ("&&" 0 1 217 nil)
      ("[^*]\\(\\*\\ nil)\\." 1 8 180 nil)
      ("\\(/\\ nil)\\." 1 3 184 nil)
      ;; (":>" 0 1 202 nil)
      ;; (";;" 0 1 191 nil)
      ("\\<_\\>" 0 3 188 nil) ("\\<sqrt\\>" 0 3 214 nil)
      ("\\<unit\\>" 0 3 198 nil) ("\\<fun\\>" 0 3 108 nil)
      ("\\<or\\>" 0 3 218 nil) ("\\<not\\>" 0 3 216 nil))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; personal shortcuts
(global-set-key [C-tab] 'other-window)
;What's the difference between define-key and global-set-key?

;;I find "C-xb RET" to complicated in comparison to how often it is used:
(define-key global-map [(control return)] 'iswitchb-buffer)
; nice way to change buffer. Use C-enter then C-s and C-r then enter to change the buffer.
; More help with C-h f iswitchb.
; C-c to toggle case-sensitivity
(require 'iswitchb)
(iswitchb-default-keybindings)

(defun recenter-zero () (interactive) (recenter 0))
(global-set-key [M-up] 'recenter-zero)
;scroll down so that the line under the cursor comes at the top of the window

;; C-home deletes indentation of the current line, then merges it with the preceeding line.
;; Useful when programming.
;; Default key-binding is M-^
(define-key global-map [(control home)] 'delete-indentation)

(defun rlr-copy-line-as-kill (&optional arg)
  "Copy current line to kill ring. With arg, copies that many lines."
  (interactive "nLines to copy?")
  (save-excursion
    (beginning-of-line)
    (copy-region-as-kill (point)
       (progn (forward-line arg) (point)))))
(global-set-key [C-w] 'rlr-copy-line-as-kill) ;use [S-del] to cut the region


(global-set-key [f3] 'bury-buffer) ;put the current buffer at the end of the buffer list
(global-set-key [f4] 'kill-this-buffer) ;C-x k
(global-set-key [f5] 'font-lock-fontify-buffer) ;"reload colors"
(global-set-key "\C-xg" 'goto-line) ;C-x g allows one to reach a line with its number

;;automatically close brackets, quotes, etc when typing
(setq skeleton-pair t)
(setq skeleton-pair-on-word t) ; apply skeleton trick even in front of a word.
(global-set-key "[" 'skeleton-pair-insert-maybe)
(global-set-key "{" 'skeleton-pair-insert-maybe)
(global-set-key "(" 'skeleton-pair-insert-maybe)
(global-set-key "\"" 'skeleton-pair-insert-maybe)
;; and ensure those bindings are kept for c and c++ modes
(add-hook 'c-mode-hook '(lambda ()
 (define-key c-mode-base-map "{" 'skeleton-pair-insert-maybe)
 (define-key c-mode-base-map "(" 'skeleton-pair-insert-maybe)
))
(add-hook 'c++-mode-hook '(lambda ()
 (define-key c-mode-base-map "{" 'skeleton-pair-insert-maybe)
 (define-key c-mode-base-map "(" 'skeleton-pair-insert-maybe)
))

;for efficient commenting
(defun comment-and-go-down ()
  "Comments the current line and goes to the next one" (interactive)
```

```
  (condition-case nil (comment-region (point-at-bol) (point-at-eol)) (error nil))
  (next-line 1))
(defun uncomment-and-go-up ()
  "Uncomments the current line and goes to the previous one" (interactive)
  (condition-case nil (uncomment-region (point-at-bol) (point-at-eol)) (error nil))
  (next-line -1))
;shift-down comments the current line and goes down
(define-key global-map [(shift down)] 'comment-and-go-down)
;shift-up uncomments the current line and goes up
(define-key global-map [(shift up)] 'uncomment-and-go-up)

(unless (fboundp 'next-buffer)
  (defun next-buffer ()
    "Switch to the next buffer in cyclic order."
    (interactive)
    (let ((buffer (current-buffer)))
      (switch-to-buffer (other-buffer buffer))
      (bury-buffer buffer))))
(unless (fboundp 'prev-buffer)
  (defun prev-buffer ()
    "Switch to the previous buffer in cyclic order."
    (interactive)
    (let ((list (nreverse (buffer-list)))
          found)
      (while (and (not found) list)
        (let ((buffer (car list)))
          (if (and (not (get-buffer-window buffer))
                   (not (string-match "\\` " (buffer-name buffer))))
              (setq found buffer)))
        (setq list (cdr list)))
      (switch-to-buffer found))))
(define-key global-map [?\C-x right] 'next-buffer)
(define-key global-map [?\C-x left] 'prev-buffer)

(require 'ps-print nil t)
(setq
 ps-print-color-p t;print with colors
 ps-paper-type 'a4
 ps-print-header nil;no header (try removing this line to see the header)
)

(defun ps-print-in-file (filename) "Print the buffer in a colored postscript file"
  (interactive "FPS file: ")
  (ps-print-buffer-with-faces filename))

;; Simplified version of Steve Wilmarth's "match-parent" function to
;; go to the corresponding closing / opening parenthesis
(defun match-paren (arg)
  "Go to the matching parenthesis"
  (interactive "p")
  (cond ((looking-at "\\s\(") (forward-list 1) (backward-char 1))
        ((looking-at "\\s\)") (forward-char 1) (backward-list 1))
        (t (condition-case nil
               (progn (re-search-forward "\\s\(") (backward-char 1)) (error nil)))
        ))

(define-key global-map [M-right] 'match-paren)

(defun downcase-html-tags () (interactive)
  (save-excursion
    (beginning-of-buffer)
    (while (re-search-forward "</?\\([a-zA-Z]+\\)" nil t)
      (downcase-region (match-beginning 1) (match-end 1)))
    )
  )


;; automatic text, at least adapt author!
(defun start-latex () "Adds all that stuff to start a new LaTeX document" (interactive)
  (goto-char (point-min))
  (insert "\\documentclass[a4paper,french]{article}
\\title{}
\\author{Jean-Baptiste Rouquier}
\\date{}

\\usepackage[french]{babel}    %pour la typographie et le texte automatique en français
\\usepackage{indentfirst}      %pour les paragraphes à la française
\\usepackage[latin1]{inputenc} %pour les accents
\\usepackage[T1]{fontenc}      %pour la césure des mots accentués
%\\usepackage{aeguill}          %pour un pdf propre à l'écran
\\usepackage[pdftex]{graphicx}

\\begin{document}
\\maketitle
```

```
")
  (goto-char (point-max))
  (insert "
\\end{document}
")
  (goto-char (point-min))
  (next-line 2)
  (backward-char 2)
  (latex-mode)
)



(defun add-gpl () "Adds the GPL statements at the beginning of the file" (interactive)
  (let ((comment-style 'box)
        (gpl
         (concat
"This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation\; either version 2 of the
License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY\; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.
"
                 (if (boundp 'user-full-name)
                     (concat "\nWritten and (c) by " user-full-name "\n")
                     "")
;;                   (if (boundp 'user-mail-address) (concat
;;                                                    "Contact <"
;;                                                    user-mail-address
;;                                                    "> for comment & bug reports\n")
;;                       "")
                 )))

    (goto-char (point-min))
    (insert gpl)
    (comment-region (point-min) (+ (point-min) (length gpl))))
)
```