Documentación de atajos y comandos Emacs

Carlos

November 24, 2025

Contents

1 Introducción

Este archivo documenta:

- Los atajos de teclado **básicos** de Emacs (los más usados para movimiento, cortar/pegar, ventanas, archivos, etc.).
- Todos los atajos **personalizados** definidos en mi configuración:
  - `core-base.el`
  - `core-files.el`
  - `core-ui.el`
  - `core-dev.el`
  - `core-notes.el`
  - `lang-org.el`
  - `lang-writing.el`
  - `lang-prog.el`
  - `lang-web.el`
  - `lang-web-extras.el`
- Las funciones `my/...` que agregan comportamiento nuevo, con su código fuente en bloques `emacs-lisp`.

Puedo abrir este archivo dentro de Emacs y usarlo como manual rápido.

2 Convenciones de teclas

`C-` tecla Control

`M-` tecla Meta (Alt o `ESC`)

`S-` tecla Shift

Ejemplos:

- `C-x C-f` → Control + x, luego Control + f
- `M-x` → Meta (Alt) + x
- `<f6>` → tecla de función F6

3 Movimiento, edición, cortar y pegar (por defecto)

3.1 Movimiento del cursor

Tecla	Acción
C-f	Carácter adelante
C-b	Carácter atrás
C-n	Línea siguiente
C-p	Línea anterior
M-f	Palabra adelante
M-b	Palabra atrás
C-a	Inicio de línea
C-e	Fin de línea
M-a	Inicio de oración
M-e	Fin de oración
M-<	Principio del buffer
M->	Final del buffer
C-v	Página abajo
M-v	Página arriba

3.2 Selección, cortar, copiar, pegar

Tecla	Acción
C-SPC	Empezar a marcar región
C-g	Cancelar región/comando
C-w	Cortar (kill) región
M-w	Copiar región
C-y	Pegar (yank)
M-y	Navegar por el kill-ring

3.3 Deshacer

Tecla	Acción
C-/	Deshacer
C-_	Deshacer

4 Archivos y buffers

4.1 Archivos

Tecla	Acción
C-x C-f	Abrir archivo
C-x C-s	Guardar archivo
C-x C-w	Guardar como...
C-x C-v	Reemplazar buffer por otro

4.2 Buffers

Tecla	Acción
C-x b	Cambiar de buffer
C-x C-b	Lista de buffers
C-x k	Cerrar (kill) buffer actual

5 Ventanas (splits)

Tecla	Acción
C-x 1	Dejar solo la ventana actual
C-x 2	Dividir horizontalmente (una arriba/otra)
C-x 3	Dividir verticalmente (izquierda/derecha)
C-x 0	Cerrar ventana actual
C-x o	Ir a la otra ventana

6 Búsqueda y reemplazo

Tecla	Acción
C-s	Búsqueda incremental adelante
C-r	Búsqueda incremental atrás
M-%	Buscar y reemplazar (query-replace)
C-M-%	Buscar y reemplazar con regexp

7 Ayuda en Emacs

Tecla	Acción
C-h k	Describir qué hace una tecla
C-h f	Describir función
C-h v	Describir variable
C-h b	Ver todos los atajos activos en el buffer
C-h m	Ayuda de los modos activos
C-h t	Tutorial interactivo de Emacs

8 Dired (gestor de archivos)

En `Dired` se agregan atajos personalizados, pero primero los básicos:

Tecla	Acción
RET	Abrir archivo / entrar directorio
^	Subir al directorio padre
+	Crear directorio
m	Marcar archivo
u	Desmarcar archivo
d	Marcar para borrar
x	Ejecutar borrados marcados
g	Refrescar listado

Los atajos extra definidos en mi configuración están documentados más abajo.

9 Org mode básico

Tecla	Acción
TAB	Expandir/colapsar bloque
S-TAB	Ciclar visibilidad global
M-RET	Nuevo ítem al mismo nivel
M-<up>	Subir encabezado
M-<down>	Bajar encabezado
M-<left>	Promover encabezado
M-<right>	Degradar encabezado

TODOs y agenda:

Tecla	Acción
C-c C-t	Cambiar TODO/DONE
C-c .	Insertar fecha
C-c C-s	SCHEDULED
C-c C-d	DEADLINE

10 Atajos personalizados globales

Los siguientes atajos se definen en mis módulos `core-*` y `lang-*`.

10.1 Terminal integrada

Tecla	Comando	Descripción
C-c t	`vterm`	Abrir terminal `vterm`

10.2 Layout de desarrollo

Tecla	Comando	Descripción
<f9>	`my/dev-layout`	Layout con Treemacs a la izquierda, código y vterm

10.3 Org-roam (notas diarias)

Tecla	Comando	Descripción
C-c n d	`org-roam-dailies-goto-today`	Ir a la nota diaria de hoy
C-c n j	`org-roam-dailies-capture-today`	Capturar entrada en el diario

10.4 Org agenda y capture

Tecla	Comando	Descripción
C-c a	`org-agenda`	Agenda de Org
C-c c	`org-capture`	Capturas rápidas de Org

10.5 Compilar y ejecutar C/C++

Tecla	Comando	Descripción
C-c r	`my/compile-and-run`	Compilar y ejecutar archivo actual

10.6 Diccionarios de ortografía

Tecla	Comando	Descripción
C-c d s	`my/dict-spanish`	Diccionario español (es_{AR})
C-c d e	`my/dict-english`	Diccionario inglés (en_{US})

10.7 Exportar a DOCX

Tecla	Comando	Descripción
C-c e w	`my/export-buffer-to-docx`	Exportar buffer actual a DOCX usando pandoc

11 Atajos personalizados en Dired

En `core-files.el` se agregan atajos para crear directorios y archivos directamente desde Dired:

Modo	Tecla	Comando	Descripción
Dired	<f6>	`my/dired-create-empty-file`	Crear archivo vacío en Dired
Dired	<f7>	`my/dired-create-directory`	Crear nuevo directorio en Dired

12 Atajos personalizados en LaTeX

En `lang-writing.el` (AUCTeX / LaTeX):

Modo	Tecla	Comando	Descripción
LaTeX-mode	C-c e	`my/latex-equation`	Insertar entorno `equation`
LaTeX-mode	C-c f	`my/latex-figure`	Insertar entorno `figure`
LaTeX-mode	C-c t	`my/latex-table`	Insertar entorno `table`
LaTeX-mode	C-c i	`my/latex-itemize`	Insertar entorno `itemize`

13 Funciones `my/...` y código fuente

A continuación, el código fuente de todas las funciones `my/...` definidas en los módulos. Sirve como referencia y también como ejemplo para futuras modificaciones.

13.1 `core-dev.el` :: layout de desarrollo

```
(defun my/dev-layout ()
  "Abrir layout con Treemacs a la izquierda, código arriba y vterm abajo."
  (interactive)
  (delete-other-windows)
  ;; Panel izquierdo: Treemacs
  (treemacs)
  ;; Nos movemos a la ventana de la derecha para código
  (select-window (next-window))
  ;; Partimos la derecha en dos (arriba código, abajo terminal)
  (split-window-below)
  ;; Ventana superior derecha: se queda para el buffer actual
  (other-window 1)
  ;; Ventana inferior derecha: vterm
  (vterm)
  ;; Volver a la ventana de código
  (other-window -1))
```

13.2 `core-files.el` :: utilidades para Dired

```
(defun my/dired-create-directory ()
  "Crear un nuevo directorio en el Dired actual, con un prompt claro."
  (interactive)
  (let* ((dir (dired-current-directory))
         (name (read-string (format "Nombre del nuevo directorio en %s: " dir))))
    (when (and name (not (string-empty-p name)))
      (let ((full (expand-file-name name dir)))
        (make-directory full t)
        (revert-buffer)
        (message "Directorio creado: %s" full)))))

(defun my/dired-create-empty-file ()
  "Crear un nuevo archivo vacío en el Dired actual, con un prompt claro."
```

```
  (interactive)
  (let* ((dir (dired-current-directory))
         (name (read-string (format "Nombre del nuevo archivo en %s: " dir))))
    (when (and name (not (string-empty-p name)))
      (let ((full (expand-file-name name dir)))
        (with-temp-buffer
          (write-file full))
        (revert-buffer)
        (message "Archivo creado: %s" full)))))

(with-eval-after-load 'dired
  (define-key dired-mode-map (kbd "<f6>") #'my/dired-create-empty-file)
  (define-key dired-mode-map (kbd "<f7>") #'my/dired-create-directory))
```

13.3 lang-prog.el :: programación (C/C++)

```
(defun my/c-enabled ()
  "Config básica para C/C++: estilo, sangría, etc."
  (c-set-style "linux")
  (setq c-basic-offset 4
        indent-tabs-mode nil))

(add-hook 'c-mode-hook #'my/c-enabled)
(add-hook 'c++-mode-hook #'my/c-enabled)

(defun my/compile-and-run ()
  "Compilar y ejecutar el archivo C/C++ actual en una ventana vterm.
Compila con g++ -std=c++20 -O2 y ejecuta el binario resultante."
  (interactive)
  (unless buffer-file-name
    (user-error "El buffer no está asociado a un archivo"))
  (save-buffer)
  (let* ((src buffer-file-name)
         (exe (file-name-sans-extension src))
         (cmd (format "g++ -std=c++20 -O2 -o %s %s && %s\n" exe src exe)))
    (unless (executable-find "g++")
      (user-error "g++ no está instalado"))
    (let ((vterm-buf (get-buffer "*vterm-compilacion*")))
      (if vterm-buf
          (pop-to-buffer vterm-buf)
```

```
      (setq vterm-buf (vterm "*vterm-compilacion*")))
    (vterm-send-string cmd)
    (vterm-send-return))))
```

13.4 `lang-web-extras.el` :: helpers para Vue / Alpine, etc.

```
(defun my/vue-insert-sfc ()
  "Insertar skeleton básico de Single File Component Vue 3."
  (interactive)
  (insert "<template>\n  <div class=\"\">\n  </div>\n</template>\n\n")
  (insert "<script setup>\n\n</script>\n\n")
  (insert "<style scoped>\n\n</style>\n")
  (message "Vue SFC insertado."))

(defun my/alpine-insert-component ()
  "Insertar un snippet básico para un componente con Alpine.js."
  (interactive)
  (insert "<div x-data=\"{ open: false }\">\n")
  (insert "  <button @click=\"open = !open\">Toggle</button>\n")
  (insert "  <div x-show=\"open\">\n")
  (insert "    Contenido...\n")
  (insert "  </div>\n")
  (insert "</div>\n")
  (message "Snippet Alpine.js insertado."))

(defun my/tailwind-insert-container ()
  "Insertar un contenedor básico con clases de Tailwind CSS."
  (interactive)
  (insert "<div class=\"max-w-4xl mx-auto px-4 sm:px-6 lg:px-8\">\n")
  (insert "  <!-- Contenido -->\n")
  (insert "</div>\n")
  (message "Contenedor Tailwind CSS insertado."))
```

13.5 `lang-writing.el` :: escritura, Markdown, LaTeX, ortografía, exportación

Incluye helpers para:

- Cambiar diccionarios (español/inglés).
- Helpers de Markdown (negrita, itálica, código, enlaces, etc.).

- Helpers de LaTeX (entornos `equation`, `figure`, `table`, `itemize`).
- Exportar el archivo actual a DOCX con `pandoc`.

Código completo:

```
(defun my/dict-spanish ()
  "Cambiar diccionario a español (es_AR)."
  (interactive)
  (ispell-change-dictionary "es_AR")
  (message "Diccionario cambiado a español (es_AR)"))

(defun my/dict-english ()
  "Cambiar diccionario a inglés (en_US)."
  (interactive)
  (ispell-change-dictionary "en_US")
  (message "Dictionary changed to English (en_US)"))

(global-set-key (kbd "C-c d s") #'my/dict-spanish)
(global-set-key (kbd "C-c d e") #'my/dict-english)

;; Helpers Markdown
(defun my/markdown-wrap (left right)
  "Envolver la región activa (o palabra actual) entre LEFT y RIGHT."
  (if (use-region-p)
      (let ((beg (region-beginning))
            (end (region-end)))
        (goto-char end)
        (insert right)
        (goto-char beg)
        (insert left))
    (let ((bounds (bounds-of-thing-at-point 'word)))
      (when bounds
        (goto-char (cdr bounds))
        (insert right)
        (goto-char (car bounds))
        (insert left)))))

(defun my/markdown-bold ()
  "Negrita Markdown para región o palabra actual."
  (interactive)
```

```
  (my/markdown-wrap "**" "**"))

(defun my/markdown-italic ()
  "Itálica Markdown para región o palabra actual."
  (interactive)
  (my/markdown-wrap "*" "*"))

(defun my/markdown-inline-code ()
  "Código en línea Markdown para región o palabra actual."
  (interactive)
  (my/markdown-wrap "`" "`"))

(defun my/markdown-link ()
  "Insertar un enlace Markdown. Si hay región, usarla como texto."
  (interactive)
  (let* ((url (read-string "URL: "))
         (text (if (use-region-p)
                   (buffer-substring-no-properties (region-beginning)
                                                   (region-end))
                 (read-string "Texto del enlace: "))))
    (when (use-region-p)
      (delete-region (region-beginning) (region-end)))
    (insert (format "[%s](%s)" text url))))

(defun my/markdown-setup-keys ()
  "Configurar atajos para helpers de Markdown."
  (local-set-key (kbd "C-c m b") #'my/markdown-bold)
  (local-set-key (kbd "C-c m i") #'my/markdown-italic)
  (local-set-key (kbd "C-c m c") #'my/markdown-inline-code)
  (local-set-key (kbd "C-c m l") #'my/markdown-link))

(add-hook 'markdown-mode-hook #'my/markdown-setup-keys)

;; Helpers LaTeX
(defun my/latex-insert-environment (name)
  "Insertar un entorno LaTeX \\begin{NAME} ... \\end{NAME}, con punto de inserción en m
  (interactive "sNombre del entorno: ")
  (insert (format "\\begin{%s}\n" name))
  (save-excursion
    (insert (format "\n\\end{%s}\n" name))))
```

```
(defun my/latex-equation ()
  "Insertar entorno equation."
  (interactive)
  (my/latex-insert-environment "equation"))

(defun my/latex-itemize ()
  "Insertar entorno itemize."
  (interactive)
  (my/latex-insert-environment "itemize"))

(defun my/latex-figure ()
  "Insertar entorno figure."
  (interactive)
  (my/latex-insert-environment "figure"))

(defun my/latex-table ()
  "Insertar entorno table."
  (interactive)
  (my/latex-insert-environment "table"))

(with-eval-after-load 'latex
  (define-key LaTeX-mode-map (kbd "C-c e") #'my/latex-equation)
  (define-key LaTeX-mode-map (kbd "C-c f") #'my/latex-figure)
  (define-key LaTeX-mode-map (kbd "C-c t") #'my/latex-table)
  (define-key LaTeX-mode-map (kbd "C-c i") #'my/latex-itemize))

;; Exportar a DOCX con pandoc
(defun my/export-buffer-to-docx ()
  "Exportar el archivo actual a DOCX usando pandoc.
Funciona bien para Org y Markdown."
  (interactive)
  (unless buffer-file-name
    (user-error "El buffer no está asociado a un archivo"))
  (unless (executable-find "pandoc")
    (user-error "pandoc no está instalado"))
  (save-buffer)
  (let* ((in  (buffer-file-name))
         (out (concat (file-name-sans-extension in) ".docx")))
    (call-process "pandoc" nil "*Pandoc Output*" t in "-o" out)
```

```
    (message "Exportado a %s" out)))

(global-set-key (kbd "C-c e w") #'my/export-buffer-to-docx)
```