EBook Gratis

APRENDIZAJE emacs

Free unaffiliated eBook created from
Stack Overflow contributors.

#emacs

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: emacs

It is an unofficial and free emacs ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official emacs.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con emacs

Observaciones

Emacs es un editor de texto cuya característica más destacada es la capacidad de los usuarios para personalizar mediante programación casi todos los aspectos de la misma. Esto se facilita a través de un dialecto especial del lenguaje de programación Lisp, llamado Emacs Lisp, creado específicamente para su uso en el editor de Emacs.

Hay una multitud de extensiones escritas en Emacs Lisp que se agregan a la funcionalidad de Emacs. Estas extensiones incluyen funciones de edición para lenguajes de programación específicos (similares a lo que podría proporcionar un IDE), clientes de correo electrónico e IRC, interfaces de Git, juegos como Tetris y 2048, y mucho más.

Muchos aspectos del editor de Emacs pueden usarse sin conocimientos de programación. Los usuarios que buscan personalizar Emacs mediante programación, sin embargo, encontrarán ciertas características del lenguaje Emacs Lisp, como el sistema de (auto) documentación, increíblemente útil y complaciente.

Referencias externas:

1. El sitio de Sacha Chua es un muy buen lugar para encontrar más recursos de aprendizaje en Emacs.

   a. Para aquellos que necesitan un atractivo más visual en el camino de aprendizaje de Emacs

   segundo. Para aquellos que deseen obtener los enlaces de clave fácilmente

2. Wikemacs se basa en mediawiki y, por lo tanto, tiene contenido estructurado, categorías navegables y demás . ¡Empieza a explorar !

Versiones

Versión	Fecha de lanzamiento
25.1	2016-09-17
24.5	2015-04-10
24.4	2014-10-20
24.3	2013-03-11
24.2	2012-08-27
24.1	2012-06-10

Versión	Fecha de lanzamiento
23.4	2012-01-29
23.3	2011-03-10
23.2	2010-05-08
23.1	2009-07-29
22.3	2008-09-05
22.2	2008-03-26
22.1	2007-06-02
21.4	2005-02-06
21.3	2003-03-24
21.2	2002-03-18
21.1	2001-10-28

Examples

Instalación o configuración

Instrucciones detalladas sobre cómo configurar o instalar emacs.

Las instrucciones oficiales están disponibles en el sitio web de GNU Emacs .

Sistemas debian

En sistemas con el administrador de paquetes de Debian (como Debian, Ubuntu y Mint), Emacs se puede instalar mediante el comando simple:

```
sudo apt-get install emacs
```

Para una liberación de vanguardia se puede usar el siguiente ppa:

```
sudo apt-add-repository ppa:ubuntu-elisp/ppa
sudo apt-get install emacs-snapshot
```

Construir para la fuente

Si su distribución basada en Debian no tiene la versión de emacs que desea, puede construirla

desde cero.

```
sudo apt-get build-dep emacs24 -y

cd /tmp/

wget http://alpha.gnu.org/gnu/emacs/pretest/emacs-25.0.93.tar.xz
tar -xvf emacs-25.0.93.tar.xz

cd emacs-25.0.93
./configure
make
sudo make install

rm -rf /tmp/emacs-25.0.93*
```

Sistemas redhat

En sistemas con el gestor de paquetes Redhat (como RHEL, CentOS y Fedora Core), Emacs se puede instalar mediante el comando simple:

```
sudo yum install emacs
```

Arco de linux

Emacs se puede instalar a través del comando simple:

```
sudo pacman -Syu emacs
```

Gentoo y Funtoo

En los sistemas que ejecutan Portage, Emacs se puede instalar mediante el comando simple:

```
sudo emerge emacs
```

GSRC (GNU Source Release Collection)

Funciona en cualquier sistema GNU / Linux para obtener la última versión de emacs sin usar el administrador de paquetes del sistema (que puede estar desactualizado) o descargar el archivo o el binario. Para instalar `gsrc` ver su documentación . Entonces:

```
cd gsrc
make -C gnu/emacs install
# add the binaries to your PATH
source ./setup.sh
```

Sistemas darwin

Homebrew

```
brew install emacs --with-cocoa # basic install
# additional flags of interest can be viewed by calling `brew info emacs`
brew linkapps emacs # to put a symlink in your Applications directory
```

MacPorts

```
sudo port install emacs
```

pkgsrc

```
sudo pkgin -y install emacs-24.5
```

Paquete de aplicaciones

Los paquetes de aplicaciones precompilados para las últimas versiones estables y de desarrollo se pueden descargar en https://emacsformacosx.com .

Windows

Gestor de paquetes chocolatey

Emacs se puede instalar con

```
choco install emacs
```

Gestor de paquetes Scoop

Se puede instalar desde extras.

```
scoop bucket add extras
scoop install emacs
```

Instaladores Binarios Oficiales

- estable
- último

(Tenga en cuenta que los binarios oficiales no vienen con algunas bibliotecas, por ejemplo, bibliotecas para formatos de imagen)

Otros instaladores binarios

- Emacs con AUCTeX y ESS precompilados
- GNU Emacs de 64 bits para MS Windows con optimización proporciona un instalador binario nativo y optimizado de 64 bits con código fuente no modificado de git master y versión de lanzamiento, con soporte para JPEG, GIF, PNG, TIFF, SVG, XML2 y GnuTLS caja.

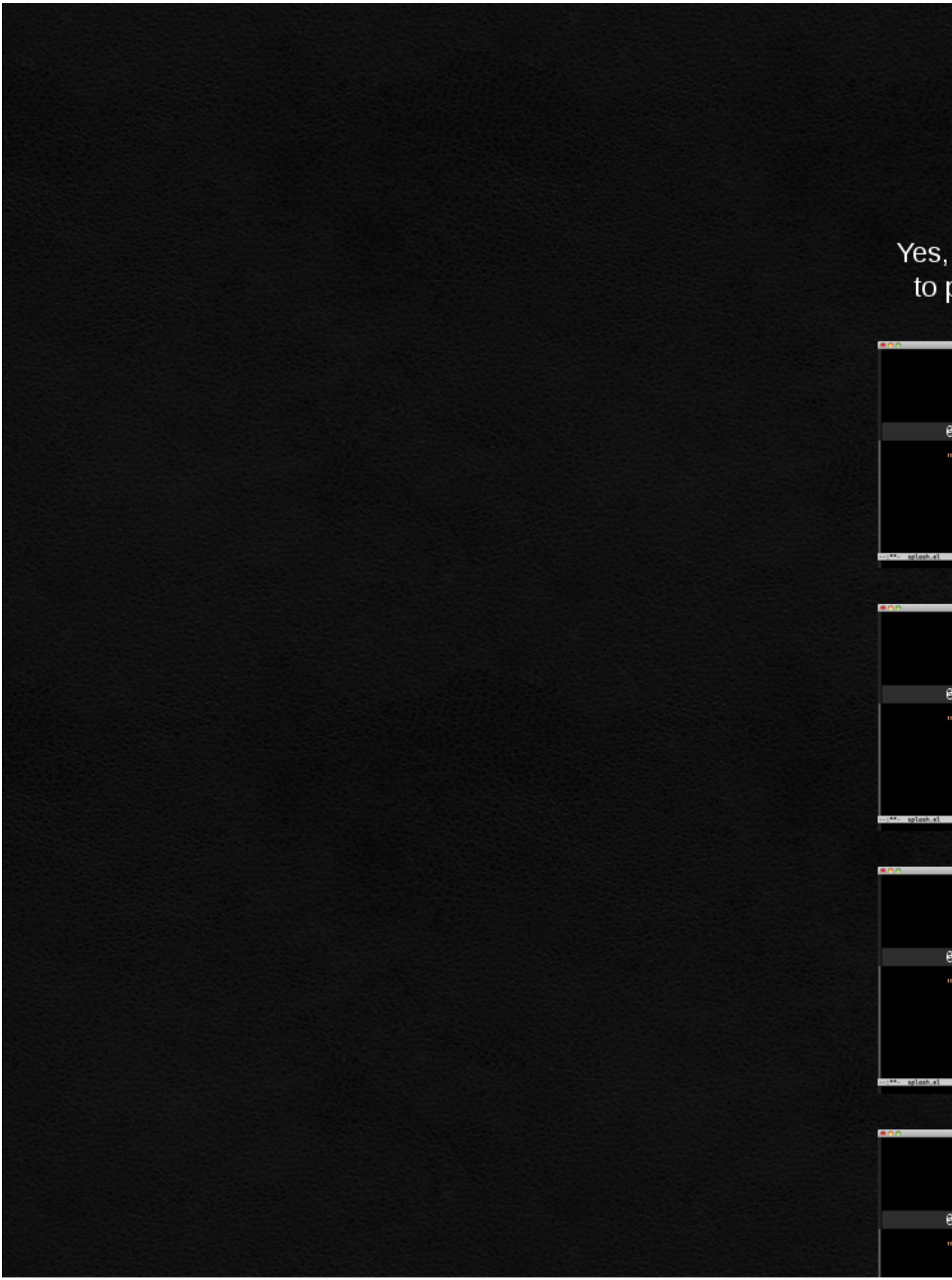
Tutorial interactivo de Emacs

Desde Emacs, escriba `Ch t` (Control-h, t) para obtener un excelente tutorial interactivo dentro de Emacs. El usuario aprende la navegación y edición básicas operando en el texto TUTORIAL en sí, mientras lee el tutorial. (Las modificaciones del tutorial se descartan cuando el tutorial está cerrado, por lo que cada vez que un usuario solicita el tutorial, es una versión predeterminada y limpia del tutorial.

Afortunadamente, lo primero en el tutorial es cómo entender las referencias `C-<chr>` y `M-<chr>` en el texto. Lo segundo es cómo avanzar y retroceder páginas en el texto.

Tutoriales de Emacs Rocks

Se pueden encontrar buenos tutoriales en video sobre Emacs en emacsrocks.com .


Yes,
to

https://riptutorial.com/es/emacs/topic/986/empezando-con-emacs

Capítulo 2: Administrar marcadores dentro de Emacs

Examples

Cómo marcar archivos de uso frecuente

Use los siguientes comandos para crear marcadores y acceder a ellos desde Emacs.

Digamos que está editando un archivo llamado `foobar.org` y supongamos que visita este archivo con frecuencia para editar / ver contenidos.

Sería conveniente acceder a este archivo con un par de pulsaciones de teclas en lugar de navegar a través de la estructura del archivo (Dired) y visitar el archivo.

Pasos:

1. Abra `foobar.org` por una vez navegando al archivo (*visite el archivo* en la jerga de Emacs)
2. Mientras el archivo está abierto, escriba `Cx r m`, esto le pedirá que proporcione el nombre del marcador para el archivo. Digamos `foobar` en este caso.
3. Cerrar el archivo (`Cx k` - kill buffer) - guardar si es necesario
4. Ahora para visitar el archivo, simplemente escriba `Cx r l` - esto llenará una lista que contendrá `foobar` .
5. Selecciona `foobar` y `foobar tecla` enter.
6. Use `Mx bookmark-delete` para eliminar cualquier marcador innecesario de un archivo.

Nota: Eliminar un marcador es análogo a eliminar un acceso directo en su escritorio de Windows. El archivo principal estará seguro en su ubicación y solo se eliminará la lista del archivo del menú de marcadores.

Lea Administrar marcadores dentro de Emacs en línea:
https://riptutorial.com/es/emacs/topic/5837/administrar-marcadores-dentro-de-emacs

Capítulo 3: Ayuda dentro de Emacs

Observaciones

Emacs se describe como un editor autodocumentado y proporciona mucha información sobre cómo usarlo dentro del propio editor. Entre los puntos de entrada a esta documentación hay un tutorial, información sobre qué funciones están disponibles relacionadas con un tema determinado, una información sobre los enlaces entre pulsaciones de teclas y funciones. Se accede a la documentación utilizando el prefijo `Ch` , *es decir* , `Ctrl h` , o `F1` , con una lista de otras opciones disponibles presionando `?`

Examples

Tutorial de Emacs

`Ch t` ejecuta la función `help-with-tutorial` , que abre un búfer que contiene un tutorial sobre la funcionalidad de edición básica de emacs, que incluye moverse en el texto y trabajar con archivos, búferes y ventanas.

Funciones disponibles y enlaces de teclas

Al presionar `Ch a` se ejecutará la función emacs `apropos-command` que hace que emacs pida las palabras (o una expresión regular) para buscar. A continuación, mostrará un búfer que contiene una lista de nombres y descripciones relacionadas con ese tema, incluidos los enlaces de teclas para cada una de las funciones disponibles mediante las pulsaciones de teclas.

Presionando `Ch m` (`describe-mode`) da un búfer que describe los modos mayor y menor en efecto, incluyendo listas de funciones disponibles y sus enlaces de teclas.

Al presionar `Ch b` (`describe-bindings`) se obtiene un búfer que enumera todos los enlaces de teclas actuales. La lista incluye los enlaces globales, así como los enlaces para los modos activo mayor y menor en el búfer actual.

Documentación de vinculación de teclas

`Ch k` ejecuta la función `describe-key` , que busca la función asignada a los golpes de tecla proporcionados, y presenta una descripción de la función que se ejecutará cuando se presionen estas teclas.

`Ch c` ejecuta la función `describe-key-briefly` , que solo muestra el nombre de la función asignada a la secuencia de teclas dada.

Documentación de funciones

`Ch f` ejecuta la función `describe-function` , que muestra información sobre el uso y el propósito de

una función determinada. Esto es especialmente útil para las funciones que no tienen un enlace de teclas asignado que se puede usar para la búsqueda de documentación a través de `Ch k` .

Lea Ayuda dentro de Emacs en línea: https://riptutorial.com/es/emacs/topic/4736/ayuda-dentro-de-emacs

Capítulo 4: Emacs ya tiene una documentación de alta calidad y bien organizada. ¿Por qué duplicarlo?

Introducción

https://www.gnu.org/software/emacs/manual/html_node/emacs/index.html#Top

Examples

Llaves

https://www.gnu.org/software/emacs/manual/html_node/emacs/Keys.html#Keys

3 llaves

Algunos comandos de Emacs son invocados por un solo evento de entrada; por ejemplo, Cf avanza un carácter en el búfer. Otros comandos toman dos o más eventos de entrada para invocar, como Cx Cf y Cx 4 Cf.

Una secuencia de teclas, o teclas para abreviar, es una secuencia de uno o más eventos de entrada que es significativa como una unidad. Si una secuencia de teclas invoca un comando, la llamamos una tecla completa; por ejemplo, Cf, Cx Cf y Cx 4 Cf son teclas completas. Si una secuencia de teclas no es lo suficientemente larga para invocar un comando, lo llamamos una tecla de prefijo; Del ejemplo anterior, vemos que Cx y Cx 4 son claves de prefijo. Cada secuencia de teclas es una tecla completa o una tecla de prefijo.

Lea Emacs ya tiene una documentación de alta calidad y bien organizada. ¿Por qué duplicarlo? en línea: https://riptutorial.com/es/emacs/topic/9077/emacs-ya-tiene-una-documentacion-de-alta-calidad-y-bien-organizada---por-que-duplicarlo-

Capítulo 5: Gestión de paquetes

Examples

Instalación automática de paquetes en el arranque de emacs.

```
;; package.el is available since emacs 24
(require 'package)

;; Add melpa package source when using package list
(add-to-list 'package-archives '("melpa" . "http://melpa.org/packages/") t)

;; Load emacs packages and activate them
;; This must come before configurations of installed packages.
;; Don't delete this line.
(package-initialize)
;; `package-initialize' call is required before any of the below
;; can happen

;; If you do not put the "(package-initialize)" in your ~/.emacs.d/init.el (or
;; ~/.emacs), package.el will do it for you starting emacs 25.1.

;; Below manual maintenance of packages should not be required starting emacs
;; 25.1 with the introduction of `package-selected-packages' variable. This
;; variable is automatically updated by emacs each time you install or delete a
;; package. After this variable is synced across multiple machines, you can
;; install the missing packages using the new
;; `package-install-selected-packages' command in emacs 25.1.

;; To clarify, below technique is useful on emacs 24.5 and older versions.
;; Request some packages:
(defconst my-package-list '()
  "List of my favorite packages")

(defvar my-missing-packages '()
  "List populated at each startup that contains the list of packages that need
to be installed.")

(dolist (p my-package-list)
  (when (not (package-installed-p p))
    (add-to-list 'my-missing-packages p)))

(when my-missing-packages
  (message "Emacs is now refreshing its package database...")
  (package-refresh-contents)
  ;; Install the missing packages
  (dolist (p my-missing-packages)
    (message "Installing `%s' .." p)
    (package-install p))
  (setq my-missing-packages '()))
```

Referencias

- Comparación de repos de paquetes
- Publicación en el blog sobre paquetes seleccionados por el usuario en emacs 25.1

Instalación automática de paquetes con use-package

```
;; disable automatic loading of packages after the init file
(setq package-enable-at-startup nil)
;; instead load them explicitly
(package-initialize)
;; refresh package descriptions
(unless package-archive-contents
   (package-refresh-contents))

;;; use-package initialization
;;; install use-package if not already done
(if (not (package-installed-p 'use-package))
    (progn
      (package-refresh-contents)
      (package-install 'use-package)))
;;; use-package for all others
(require 'use-package)

;; install your packages
(use-package helm
  :ensure t)
(use-package magit
  :ensure t)
```

Gestión automática de paquetes utilizando Cask.

Cask es una herramienta de administración de proyectos que también se puede usar para administrar fácilmente la configuración local de emacs.

Instalar el barril es fácil. Puede ejecutar el siguiente comando en la línea de comandos:

```
curl -fsSL https://raw.githubusercontent.com/cask/cask/master/go | python
```

O si estás en un mac, puedes instalarlo usando `homebrew` :

```
brew install cask
```

Una vez instalado, creas un archivo `Cask` . Los archivos de Cask enumeran todas las dependencias de paquetes que deben incluirse en su configuración. Puede crear un nuevo archivo Cask en la raíz de su directorio `~/.emacs` .

También deberá inicializar Cask en su `~/.emacs.d/init.el` Si instaló utilizando homebrew, agregue estas líneas:

```
(require 'cask "/usr/local/share/emacs/site-lisp/cask/cask.el")
(cask-initialize)
```

O puede proporcionar la ruta al barril, si usó el script de instalación:

```
(require 'cask "~/.cask/cask.el")
(cask-initialize)
```

Un simple archivo de Cask se ve así:

```
(source gnu)
(source melpa)

(depends-on "projectile")
(depends-on "flx")
(depends-on "flx-ido")
```

Aquí estamos especificando los repositorios de origen para buscar paquetes. Luego, estamos especificando que queremos que se instalen los paquetes `projectile` , `flx` y `flx-ido` .

Una vez que tenga un archivo Cask, puede instalar todas las dependencias con el siguiente comando en la línea de comandos:

```
cask install
```

Gestión automática de paquetes con el-get

el-get es un sistema de gestión de paquetes de código abierto para Emacs de GNU. el-get funciona con `melpa` , así como con muchos sistemas de control de versiones comunes. Su documentación incluye un simple auto-instalador para tus `.emacs` :

```
(unless (require 'el-get nil t)
  (url-retrieve
    "https://raw.github.com/dimitri/el-get/master/el-get-install.el"
    (lambda (s)
      (let (el-get-master-branch)
        (goto-char (point-max))
        (eval-print-last-sexp)))))

(el-get 'sync)
```

el-get mantiene las instalaciones de paquetes en una estructura de directorio en `~/.emacs.d/el-get` . Carga definiciones de `~/.emacs.d/el-get/.loaddefs.el` y rastrea el estado del paquete con `~/.emacs.d/el-get/.status.el` . `(el-get 'sync)` instala o elimina paquetes para sincronizar el estado real de la máquina con el paquete `.status.el` .

el-get es auto-alojado - aquí está su propio estado de `.status.el` :

```
(el-get status "installed" recipe
  (:name el-get :website "https://github.com/dimitri/el-get#readme" :description "Manage the
external elisp bits and pieces you depend upon." :type github :branch "master" :pkgname
"dimitri/el-get" :info "." :compile
         ("el-get.*\\.el$" "methods/")
         :features el-get :post-init
         (when
             (memq 'el-get
                   (bound-and-true-p package-activated-list))
```

```
(message "Deleting melpa bootstrap el-get")
(unless package--initialized
  (package-initialize t))
(when
    (package-installed-p 'el-get)
  (let
      ((feats
        (delete-dups
         (el-get-package-features
          (el-get-elpa-package-directory 'el-get)))))
    (el-get-elpa-delete-package 'el-get)
    (dolist
        (feat feats)
      (unload-feature feat t))))
(require 'el-get))))
```

Lea Gestión de paquetes en línea: https://riptutorial.com/es/emacs/topic/2414/gestion-de-paquetes

Capítulo 6: Keybindings básicos

Examples

Salir de Emacs

Puedes salir de Emacs con el siguiente enlace de teclas:

```
Cx Cc
```

Donde `C` es la tecla de `control` .

Suspender emacs

Puedes suspender Emacs usando el siguiente enlace de teclas:

```
Cz
```

Te lleva de vuelta a tu caparazón. Si desea reanudar su sesión de emacs, ingrese `fg` en su terminal.

Manejo de archivos

- Vuelva a guardar el archivo abierto con el mismo nombre de archivo (Guardar):

  ```
  Cx Cs
  ```

- Escribir como `filename` (Guardar como):

  ```
  Cx Cw filename
  ```

  El nuevo nombre del archivo aparecerá en el minibúfer.

- Crear un nuevo archivo **o** cargar un archivo existente (Nuevo / Cargar):

  ```
  Cx Cf filename
  ```

  Con el mnemotécnico aquí para el archivo de significado f. Se le pedirá una ruta de archivo en el minibúfer.

- Visita archivo alternativo

  ```
  Cx Cf
  ```

  Si el archivo aún no existe, se le solicitará la ruta del archivo que desea crear en el minibúfer.

Abortar el comando actual

A menudo entrará en un estado en el que tiene una secuencia de comandos parcialmente escrita en curso, pero desea abortarla. Puedes abortarlo con cualquiera de las siguientes combinaciones

de teclas:

`CG`

`Esc Esc Esc`

Múltiples ventanas o marcos

"Ventana" en Emacs se refiere a lo que de otra manera podría llamarse "panel" o "división de pantalla". Algunos comandos de manipulación de ventanas incluyen:

- Dividir la ventana actual horizontalmente: `Cx 2`
- Dividir la ventana actual verticalmente: `Cx 3`
- Seleccione la siguiente ventana: `Cx o`
- Cerrar ventana actual: `Cx 0`
- Cerrar todas las demás ventanas, excepto la actual: `Cx 1`

Un "marco" en Emacs es lo que de otra manera podría llamarse una "ventana". Los marcos son manipulados usando estos comandos:

- Crear nuevo marco: `Cx 5 2`
- Borrar cuadro actual: `Cx 5 0`
- Borrar otros marcos: `Cx 5 1`

Cambiando ventanas se puede lograr usando

- `S-izquierda` , `S-derecha` , `S-arriba` , `S-abajo` (es decir, `Shift` junto con una tecla de flecha) para cambiar a la ventana adyacente en una dirección, o
- `Cxo` para pasar a la siguiente ventana.

Tampones

- Ejemplo de una lista de búfer

```
CRM Buffer                Size  Mode              Filename[/Process]
. * .emacs                 3294  Emacs-Lisp        ~/.emacs
 %  *Help*                  101  Help
    search.c              86055  C                 ~/cvs/emacs/src/search.c
 %  src                   20959  Dired by name     ~/cvs/emacs/src/
  * *mail*                    42  Mail
 %  HELLO                  1607  Fundamental       ~/cvs/emacs/etc/HELLO
 %  NEWS                 481184  Outline           ~/cvs/emacs/etc/NEWS
    *scratch*                191  Lisp Interaction
  * *
Messages*             1554  Messages
```

  El primer campo de una línea indica:

  - '.' el buffer es actual
  - '%' un búfer de sólo lectura.
  - '*' se modifica el búfer.

- Seleccione búfer. Puede seleccionar de cualquier búfer abierto con la siguiente combinación de teclas:

  ```
  Cx b
  ```

  Se le pedirá el nombre del búfer al que desea cambiar.

- Lista de buffers:

  ```
  Cx Cb
  ```

- Save-some-buffer, dando la opción de qué buffer guardar o no:

  ```
  Cx s
  ```

- Mata un búfer:

  ```
  Cx k
  ```

- Operaciones en buffers marcados:

  `S` Guarda los buffers marcados

  `A` Ver los buffers marcados en este cuadro.

  `H` Ver los buffers marcados en otro marco.

  `V` Revertir los buffers marcados.

  `T` Alternar el estado de solo lectura de los buffers marcados.

  `D` Mata a los buffers marcados.

  `Ms a Cs` Hacer búsqueda incremental en los buffers marcados.

  `Ms a CMs` Isearch para regexp en los buffers marcados.

  `U` Reemplazar por regexp en cada uno de los buffers marcados.

  `Q` Query reemplazar en cada uno de los buffers marcados.

  `I` Como arriba, con una expresión regular.

  `P` Imprimir los buffers marcados.

  `O` Enumere las líneas en todos los búferes marcados que coincidan con una expresión regular dada (como la función "ocurrir").

  `X` Canalice el contenido de los buffers marcados a un comando de shell.

  `N` Reemplace el contenido de los buffers marcados con la salida de un comando de shell.

  `!` Ejecute un comando de shell con el archivo del búfer como argumento.

`E` Evaluar un formulario en cada uno de los buffers marcados. Este es un comando muy flexible. Por ejemplo, si desea que todos los búferes marcados sean de solo lectura, intente usar (modo de solo lectura 1) como formulario de entrada.

`W` - Como arriba, pero vea cada búfer mientras se evalúa el formulario.

`k` : *elimine* las líneas marcadas del búfer de *Ibuffer* , pero no *elimine el* búfer asociado.

`x` - Mata todos los buffers marcados para borrarlos.

- Save-some-buffer, dando la opción de qué buffer guardar o no:

  `Cx s`

- Cambiar al siguiente búfer:

  `Cx DERECHA`

- Cambiar al búfer anterior:

  Cx IZQUIERDA

Buscar y reemplazar

En Emacs, la herramienta de búsqueda básica (`I-Search`) le permite buscar después o antes de la ubicación de su cursor.

- Para buscar en *algún momento* después de la ubicación de su cursor (`search-forward`) *`sometext`* `Cs en` *`sometext`* . Si desea ir a la siguiente *`sometext`* de *`sometext`* , simplemente presione `Cs` nuevamente (y así sucesivamente para las próximas apariciones). Cuando el cursor cae en la ubicación correcta, presione `Entrar` para salir del indicador de búsqueda.
- Para buscar antes de la ubicación de su cursor (`search-backward`), use `Cr de` la misma manera que usó antes.
- Para cambiar de `search-backward` a `search-forward` , presione 2 veces `Cs` . Y presione 2 veces `Cr` para `search backward` cuando esté en el indicador de `search-forward` .
- Buscar y reemplazar:

  `M-%` (o `Esc-%`) texto `oldtext Entrar newtext Entrar`

  - Confirmar: `y`
  - Saltar: `n`
  - Salir: `q`
  - Reemplazar todos `:!`

Región - Cortar, Copiar, Pegar

- Establecer marca en la ubicación del cursor:

`C-espacio` o `C- @`

- Matar región (Corte):

  `Cw`

- Copiar región para matar el anillo:

  `Mw` o `Esc-w`

- Yank (Pegar) matado más recientemente:

  `Cy`

- Yank (Pegar) siguiente último muerto:

  `Mi` o `Esc-y`

Matar

`kill` es el comando utilizado por Emacs para la eliminación de texto. El comando `kill` es análogo al comando `cut` en Windows. Existen varios comandos que 'matan' una palabra (`Md`), el resto de la línea (`Ck`) o bloques de texto más grandes. El texto suprimido se añade al `kill-ring` , de la que más tarde puede ser `yanked` .

Seleccionar y cortar (matar)

Matar y tirar Similar a la función de `select-and-cut` en Windows, aquí tenemos `C-spc` . La `combinación de` teclas `C-spc` iniciará la selección, el usuario puede mover la marca con la ayuda de las teclas de flecha u otro comando para realizar una selección. Una vez que se complete la selección, presione la `tecla Cw` para eliminar el texto seleccionado

Algunos comandos básicos que se pueden usar como referencia rápida para el comando `kill` (tomado del tutorial de Emacs)

```
    M-DEL       Kill the word immediately before the cursor
    M-d         Kill the next word after the cursor

    C-k         Kill from the cursor position to end of line
    M-k         Kill to the end of the current sentence
```

Tirón

`yank` describe la inserción de un texto previamente eliminado, *por ejemplo* , utilizando `Cy` que elimina el texto eliminado más recientemente. `Yank` comando `Yank` es análogo al comando `paste` en Windows.

Yank texto matado previamente

Sabemos que el comando `kill` agrega el texto matado a un `kill-ring` . Para recuperar el texto eliminado del `kill-ring` use `Mi` comando repetidamente hasta que se elimine el texto deseado.

(Nota: para que la tecla `Mi` funcione el comando anterior debe ser un `YANK` contrario no funcionará)

Movimiento del cursor (punto)

Además de los movimientos del cursor con las teclas de flecha, Inicio, Fin, Página arriba y Página abajo, emacs define una serie de pulsaciones que pueden mover el cursor sobre fragmentos de texto más pequeños o más grandes:

Por carácter:

- Carácter al revés: `Cb`
- Carácter adelante: `Cf`

Por palabra

- Palabra hacia atrás: `Mb` (*es decir,* `Alt b` , o `Meta b`)
- Palabra hacia adelante: `Mf`

Nombre del autor:

- Inicio de la línea actual: `Ca`
- A partir del primer carácter de la línea actual (sin espacio): `Mm`
- Fin de la línea actual: `Ce`
- Línea anterior: `Cp`
- Siguiente línea: `Cn`

Búfer completo:

- Comienzo del buffer: `M- <`
- Fin de la memoria intermedia: `M->`

Por 'bloque', dependiendo del contexto (modo):

Uniones de teclas típicas:

- Frase / declaración atrasada: `Ma`
- Frase / declaración directa: `yo`
- Inicio de la función: `MCa`
- Fin de la función: `MCe`

Argumentos prefijo

Para mover varios 'pasos' a la vez, a los comandos de movimiento se les puede dar un argumento de prefijo presionando `ESC` o `Cu` y un número antes de las teclas enumeradas. Para `Cu` ,

el número es opcional y el valor predeterminado es 4.
Por ejemplo, `ESC 3 Cn` mueve 3 líneas hacia abajo, mientras que `Cu Mf` mueve 4 palabras hacia adelante.

Deshacer

Para deshacer algo que acabas de hacer:

`C-_` o `Cx u` o `C- /`

Caso

- Capitalizar palabra: `Mc`
- Convertir la palabra a mayúsculas: `Mu`
- Convertir palabra a minúsculas: `Ml`

Notación de enlaces clave

La documentación de Emacs utiliza una notación coherente para todos los enlaces de claves, que se explica aquí:

Acordes clave

Se obtiene un "acorde de tecla" presionando dos o más teclas simultáneamente. Los acordes clave se indican separando todas las claves mediante guiones (`-`). Por lo general, involucran teclas modificadoras, que se colocan al frente

- `C-` : control;
- `S-` : cambio;
- `M-` : alt (la "M" significa "Meta" por razones históricas).

Otras claves son simplemente denotadas por su nombre, como:

- `R`: La `a` llave;
- `izquierda` : la tecla de flecha izquierda;
- `SPC` : la tecla de espacio;
- `RET` : la tecla de retorno.

Ejemplos de acordes clave incluyen:

- `Ca`: presión `de control` y `un` simultáneamente;
- `S-derecha` : presionando `shift` y `derecha` simultáneamente;
- `CMa` : presionando `control` , `alt` y `a` simultáneamente.

Secuencias clave

Las "secuencias de teclas" son secuencias de teclas (o acordes de teclas), que deben escribirse una después de la otra. Se denotan separando todas las notaciones clave (o acorde) por un espacio.

Ejemplos incluyen:

- `Cx b` : presionando `control` y `x` simultáneamente, luego soltándolos y presionando `b` ;
- `Cx Cf` : presionando `control` y `x` simultáneamente, luego soltando `x` y presionando `f` (dado que ambos acordes involucran el modificador de `control` , no es necesario soltarlo).

Usando ESC en lugar de Alt

Los acordes clave que usan el modificador Alt también se pueden ingresar como una secuencia de teclas que comienza con `ESC` . Esto puede ser útil cuando se usa Emacs a través de una conexión remota que no transmite acordes de tecla Alt, o cuando estas combinaciones de teclas son capturadas, por *ejemplo,* por un administrador de ventanas.

Ejemplo:

`Mx` se puede introducir como `ESC x` .

Describiendo enlaces de teclas en archivos de Emacs lisp

La misma notación que se describe aquí se puede usar al definir enlaces de teclas en los archivos de Emacs lisp.

Ejemplo:

(global-set-key (kbd "Cx Cb") 'buffer-menu)
une la secuencia de teclas `Cx Cb` al comando `buffer-menu`

Lea Keybindings básicos en línea: https://riptutorial.com/es/emacs/topic/3436/keybindings-basicos

Capítulo 7: Kits de inicio

Observaciones

Los kits de inicio permiten a los *nuevos usuarios* comenzar a usar Emacs rápidamente y evitar algunos de los obstáculos de configuración que provienen de un sistema maduro como Emacs, uno que ha crecido a lo largo de décadas de evolución y, naturalmente, tiene algunas peculiaridades históricas. *Los usuarios experimentados* también se benefician de tener una configuración de extensiones de kit que son curadas por otros.

Requiere un esfuerzo considerable mantener un conjunto de paquetes y configuraciones que continuarán funcionando bien juntos a medida que los paquetes mejoran (o se descomponen) con el tiempo. Muchos usuarios de Emacs no desean realizar este mantenimiento, por lo que recurren a los kits de inicio. El montaje y mantenimiento de un kit se parece en pequeña escala a la administración de una distribución de Linux.

Temas y personalización

Algunos kits de inicio son temáticos; por ejemplo, para entornos específicos de lenguaje de programación, creación de música o emulación de otro editor. Otros apuntan a proporcionar un fregadero de cocina de paquetes cómodos / productivos para tantas situaciones o idiomas como sea posible.

La mayoría de los kits de inicio tienen disposiciones para la extensión y personalización. Un usuario anulará los enlaces y configuraciones particulares, y podrá agregar paquetes que aún no se han proporcionado.

Kits populares

Hay muchos kits de inicio disponibles. En teoría, cualquiera que publique su `~/.emacs.d` ha creado uno. Pero unos pocos se han vuelto populares y bien mantenidos por uno o más individuos. Algunos ejemplos (en orden de popularidad subjetiva según las estrellas Github) incluyen Spacemacs , Prelude , Purcell , Emacs Starter Kit , Magnars y Emacs Live . Más detalles se enumeran en la sección de **ejemplos** anterior y más kits de inicio se enumeran en este wiki .

Un notable "micro-kit" es Sane Defaults , que proporciona una serie de configuraciones para eliminar algunos de los comportamientos predeterminados de sorpresa para los recién llegados de Emacs.

¿Se necesita un kit de inicio?

Si bien existe cierta controversia sobre el uso de los kits de inicio, para muchos los beneficios pueden superar el costo de cómo armonizar una configuración dinámica de Emacs. Los argumentos en contra de los kits de inicio generalmente se relacionan con: los usuarios

desconocen algunos de los matices y el comportamiento nativo de Emacs, son difíciles de depurar e incluso hacen que Emacs se parezca más a un editor extranjero (Spacemacs).

Examples

Spacemacs

Spacemacs es un kit de inicio popular para emacs. Cuenta con una solución de administración de paquetes robusta y se centra en el popular modo malvado de emacs, que proporciona muchas de las combinaciones de teclas de vim .

Se llama Spacemacs porque usa la tecla `Espacio` como la tecla líder (la idea es similar a la tecla líder de Vim).

La instalación es bastante fácil. Solo descargue e instale la distribución estándar de emacs y luego clone el repositorio git:

```
git clone https://github.com/syl20bnr/spacemacs ~/.emacs.d
```

O bien, puede descargar un `~/.emacs.d` zip localmente desde el sitio web y simplemente copiarlo a `~/.emacs.d`

Una vez que lo hayas descargado (clonado), inícialo, luego presiona la barra espaciadora para explorar la lista interactiva de enlaces de teclas elegidos cuidadosamente. También puede presionar el botón [?] Del búfer de inicio para probar algunas de las primeras combinaciones de teclas.

Preludio

Preludio es otro kit de inicio popular. Cuenta con un buen soporte para varios lenguajes de programación listos para usar que incluyen, especialmente, clojure. En los sistemas * nix se puede instalar con el siguiente comando:

```
curl -L https://git.io/epre | sh
```

emacs-live

emacs-live es otro popular kit de inicio de emacs, con un enfoque adicional en la codificación de música en vivo usando sobretonos .

Puedes instalarlo de 2 maneras:

1. En los sistemas * nix (por ejemplo, linux, OSX, etc.), ejecute el siguiente comando en la línea de comandos:

   bash <(curl -fksSL https://raw.github.com/overtone/emacs-live/master/installer/install-emacs-live.sh)

2.
   - Descarga el zip desde la página github.
   - Copia de seguridad de su actual `~/.emacs.d` en su directorio de inicio
   - extraiga el zip que descargó y muévalo a `~/.emacs.d` :

Scimax

Scimax es un kit de inicio de Emacs centrado en la investigación reproducible, dirigido principalmente a científicos e ingenieros. Scimax personaliza Org-Mode con características que simplifican las referencias cruzadas, la exportación y la codificación (en particular Python).

Las instrucciones de instalación se pueden encontrar en la página de destino del proyecto .

Lea Kits de inicio en línea: https://riptutorial.com/es/emacs/topic/1960/kits-de-inicio

Capítulo 8: Las muchas variantes de Emacs

Introducción

La mayor parte de esta documentación se aplica implícita o explícitamente a GNU Emacs. Esta puede ser la variante más conocida de Emacs, así como el origen de varias bifurcaciones y el objetivo de algunas fusiones.

Este tema discute algunas de las variantes de Emacs que uno puede encontrar, y sus diferencias principales con Emacs de GNU.

Examples

Spacemacs

Spacemacs (http://spacemacs.org/) es una variante de Emacs que intenta terminar el conflicto a largo plazo entre Emacs y los usuarios de vim, al crear un Emacs que se comporta como vim.

Lea Las muchas variantes de Emacs en línea: https://riptutorial.com/es/emacs/topic/9456/las-muchas-variantes-de-emacs

Capítulo 9: Magit

Introducción

Magit es una interfaz para el sistema de control de versiones Git, implementado como un paquete de Emacs. Te permite interactuar con git en Emacs.

Observaciones

Magit es una interfaz para el sistema de control de versiones Git, implementado como un paquete de Emacs. Magit aspira a ser una completa porcelana Git. Si bien no podemos (aún) afirmar que Magit envuelve y mejora todos y cada uno de los comandos de Git, es lo suficientemente completo como para permitir que incluso los usuarios experimentados de Git realicen casi todas sus tareas diarias de control de versiones directamente desde Emacs. Si bien existen muchos clientes finos de Git, solo Magit y Git merecen ser llamados porcelanas.

Tenga en cuenta que Magit puede interactuar con Github (con Magithub , vea también la integración de Github en Emacs) y que Emacs también tiene paquetes para trabajar con Gitlab , Bitbucket y otros.

Examples

Instalación

Puedes instalar Magit desde MELPA con:

```
M-x package-install RET magit RET
```

Uso básico: confirma ediciones sin etapas dentro de un repositorio existente

```
M-x magit-status
s RET <file-to-stage> RET
c c <commit message>
C-c C-c
q
```

Lea Magit en línea: https://riptutorial.com/es/emacs/topic/3909/magit

Capítulo 10: Modo org

Observaciones

Org es un modo para mantener notas, mantener listas de tareas pendientes y planificar proyectos con un sistema de texto simple rápido y efectivo. También es un sistema de creación con soporte único para la programación alfabetizada y la investigación reproducible.

Sitio oficial de org mode

Examples

Sintaxis de marcado

Org proporciona un lenguaje de marcado completo que ayuda a estructurar el documento y se refleja con la mayor precisión posible al exportar a otros formatos (como HTML o LaTeX).

Estructura

Titulo del documento

```
#+TITLE: This is the title of the document
```

Seccionamiento

```
* First level
** Second level
```

Liza

```
Ordered list (items can also be numbered like '1)', with a perenthesis):
1. foo
2. bar
3. baz
```

```
Unordered list (items can also start with '+' or '*'):
- foo
- bar
- baz
```

```
Description
- lorem ipsum :: this is example text
- foo bar :: these are placeholder words
```

Casillas de verificación

Cada elemento de una lista simple se puede convertir en una casilla de verificación al iniciarlo con la cadena '[]'.

```
* TODO [2/4] (or [50%])
  - [-] call people [1/3]
    - [ ] Peter
      - [X] Sarah
      - [ ] Sam
  - [X] order food
  - [ ] think about what music to play
  - [X] talk to the neighbors
```

- `Cc Cc` org-toggle-checkbox
- `Cc Cx Cb` org-toggle-checkbox
- `MS-` org-insert-todo-header
- `Cc Cx o` org-toggle-pedidos-propiedad
- `Cc #` org-update-statistics-cookies

Énfasis y monoespacio

```
You can make words *bold*, /italic/, _underlined_, =verbatim=
and ~code~, and, if you must, `+strike-through+'.
```

El texto en el código y la cadena textual no se procesa para la sintaxis específica del modo Org, se exporta textualmente.

Enlaces y referencias

Campo de golf

Org-mode reconocerá los formatos de URL y los activará como enlaces en los que se puede hacer clic. Sin embargo, los enlaces pueden ser explícitamente declarados así:

```
You will find more information in the [[http://orgmode.org/org.html][Org Manual]].
```

o alternativamente :

```
The org manual is located here: [[http://orgmode.org/org.html]]
```

Notas al pie

Las notas al pie pueden ser nombradas:

```
See the org manual[fn:manual] to get more details.
...
[fn:manual] You will find it here: http://orgmode.org/org.html
```

o anónimo y en línea:

```
See the org manual[fn:: You will find it here: http://orgmode.org/org.html]
to get more details.
```

Encuadernaciones básicas

Para ciclar el nivel de contorno mostrado:

- Nivel de esquema de ciclo de `tabulación` para un encabezado
- `Shift-Tab` Ciclo nivel de esquema para todo el documento

Para recorrer los estados de `TODO` :

- `Mayúsculas-Flecha derecha`
- `Mayúsculas-Flecha izquierda`

Para aumentar o disminuir el nivel jerárquico de un encabezado.

- `Meta-Derecha Flecha` Hacer nivel inferior ("aumentar sangría")
- `Meta-flecha izquierda` Hacer nivel más alto ("disminuir sangría")

Para ciclar la prioridad de un encabezado dado:

- `Flecha de cambio`
- `Flecha de desplazamiento hacia abajo`

Para mover un encabezado hacia arriba o hacia abajo:

- `Meta-Up Arrow`
- `Flecha hacia abajo`

(`Meta se` refiere a diferentes teclas en diferentes teclados. La mayoría de las veces es `Alt` o).

Bloques de código

Para agregar un bloque de código, `#+BEGIN_SRC language #+END_SRC` y `#+END_SRC` . *el idioma* debe corresponder al modo mayor para el idioma en cuestión, por ejemplo, el modo principal para Emacs Lisp es el modo `emacs-lisp-mode` , por lo tanto, escriba `#+BEGIN_SRC emacs-lisp` .

```
#+BEGIN_SRC emacs-lisp
(defun hello-world ()
  (interactive)
  (message "hello world"))
#+END_SRC

#+BEGIN_SRC python
print "hello world"
```

```
#+END_SRC
```

Puede abrir el bloque de código en un búfer separado escribiendo `Cc '` (para `org-edit-special`). Si no tiene el modo principal para el idioma especificado, aparecerá un mensaje de error como `No such language mode: foo-mode` .

Si el contenido que desea poner en el bloque no está en ningún lenguaje de programación, puede usar `#+BEGIN_EXAMPLE` y `#+END_EXAMPLE` en `#+END_EXAMPLE` lugar.

```
#+BEGIN_EXAMPLE
output from a command I just ran
#+END_EXAMPLE
```

Hay plantillas fáciles para ambos. Al principio de la línea, escriba `<s` o `<e` , y luego presione `TAB` . Se expandirá en un bloque con marcadores de inicio y fin para `SRC` o `EXAMPLE` , respectivamente.

Todos estos marcadores no distinguen entre mayúsculas y minúsculas, por lo que puede escribir `#+begin_src` etc. si lo prefiere.

Mesas

```
| Name  | Phone | Age |
|-------+-------+-----|
| Peter | 1234  | 17  |
| Anna  | 4321  | 25  |
```

Para agregar una tabla en modo org, simplemente rodee sus columnas con una barra (`|`)

```
| column1 | column2 | this column is wider |
```

Cuando presionas `Retorno` desde dentro de una columna, org-mode creará automáticamente una nueva fila con las barras.

- `La pestaña` y el `retorno` se moverán respectivamente a la siguiente celda o fila (o crearán una nueva si no hay)
- Puede intercambiar las filas y columnas con `M-ArrowKey`
- `MS-Down` y `MS-Right` crearán respectivamente una fila (sobre la corriente) y una columna (a la izquierda de la actual)
- `MS-Up` y `MS-Left` eliminarán respectivamente la fila actual y la columna actual
- `Cc` creo un separador

Lea Modo org en línea: https://riptutorial.com/es/emacs/topic/6259/modo-org

Capítulo 11: Nomenclatura de emacs

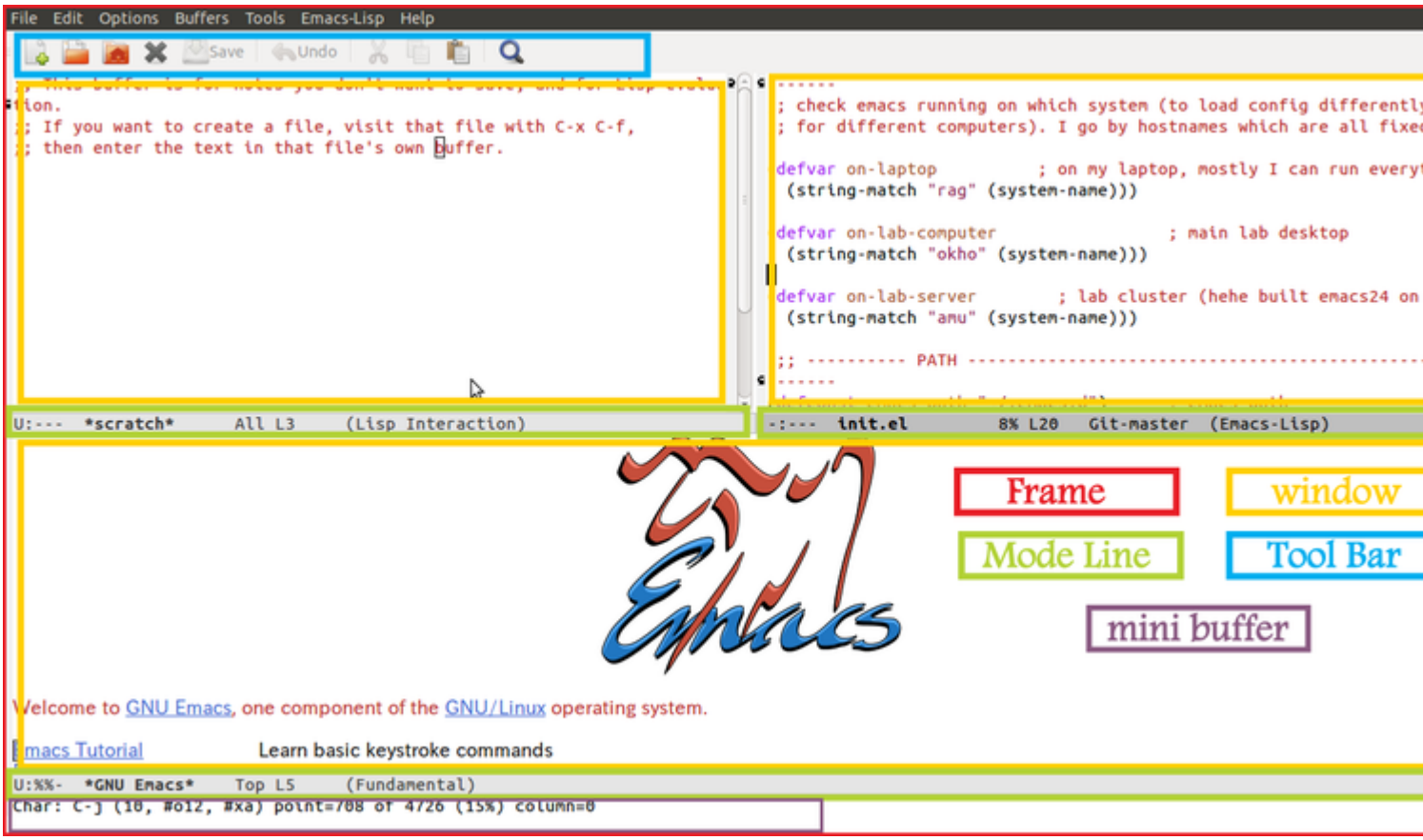
Examples

Archivos y buffers

En Emacs, el *archivo* tiene el mismo significado que en el sistema operativo y se usa para el almacenamiento permanente de datos. Un *búfer* es la representación interna de un archivo que se está editando. Los archivos se pueden leer en buffers usando `Cx Cf` , y los buffers se pueden escribir en archivos usando `Cx Cs` (guardar archivo en su ubicación actual) o `Cx Cw` (escribir el archivo en una ubicación diferente, solicitándolo, el equivalente de `Guardar como`).

Elementos de la interfaz de usuario

La interfaz de usuario de Emacs usa términos que fueron acuñados temprano y puede ser desconcertante para los usuarios acostumbrados a una terminología más moderna.



Cuadro

En Emacs, lo que de otro modo se llama una ventana (el área de la pantalla utilizada por un programa) se llama un *marco* . Emacs comienza a usar un *marco* , aunque se pueden crear marcos adicionales con `Cx 5` .

Ventana

Un *marco* contiene una o más *ventanas* (de lo contrario, generalmente llamados paneles), cada uno mostrando el contenido de un *búfer* . Por lo general, cada *cuadro* comienza con una sola *ventana* , pero se pueden crear ventanas adicionales dividiendo las existentes; ya sea horizontalmente utilizando `Cx 2` o verticalmente con `Cx 3` . Véase también Múltiples ventanas o marcos .

Buffer

El término *búfer se* refiere al contenido que se muestra en una *ventana* . Dicho contenido puede reflejar el contenido de un archivo en el sistema de archivos (o tal vez una versión actualizada que aún no se haya guardado en el disco), pero en general puede ser cualquier tipo de texto.

Línea de modo

En la parte inferior de cada *ventana* hay una *línea de modo* , que describe de forma sintética el *búfer que se* muestra en la ventana.

Barra de herramientas

De manera similar a muchos otros softwares, se puede mostrar una *barra de herramientas* en la parte superior de cada *marco* . Su contenido puede variar según el tipo de *búfer que* se está editando actualmente.

Minibuffer

Un *minibúfer* , que generalmente se muestra en la parte inferior de cada *cuadro* , permite interactuar con Emacs. Cada vez que un comando solicita la entrada del usuario, se solicita en el *minibúfer* . A la inversa, los mensajes mostrados para que el usuario los vea se imprimen allí.

Punto, marca y región.

Emacs usa los términos **punto** , **marca** y **región** para proporcionar más precisión sobre el texto seleccionado y la posición del cursor. Al comprender estos términos, le ayudará a comprender y utilizar otras operaciones y funciones.

El **punto** es el lugar en un búfer donde se está realizando la edición (*es decir, la* inserción), y generalmente se indica mediante un cursor.
La **marca** es un marcador colocado en cualquier lugar del búfer usando comandos como `set-mark-command` (`C-SPC`) o `exchange-point-and-mark` (`Cx Cx`).

La **región** es el área entre el punto y la marca, y muchos comandos operan en la región para, por *ejemplo* , eliminarla, verificar la ortografía, sangrar o compilarla.

Cuando hace clic con el mouse en una ubicación en un búfer, está viendo el punto. Cuando selecciona texto, está configurando la región (el texto seleccionado) y la marca (al comienzo de su selección).

Matar y tirar

Matar y *tirar* más o menos corresponde a lo que generalmente se llama "cortar" y "pegar".

Asesinato

matar significa borrar texto, y copiarlo en el *anillo de matanza* (que podría verse como una especie de "portapapeles" en la terminología de "cortar y pegar"). El *anillo de eliminación* se llama así porque almacena varios fragmentos de texto *eliminado* , a los que luego se puede acceder en orden cíclico.

Existen varios comandos que *eliminan* una palabra (`Md`), el resto de la línea (`Ck`) o bloques de texto más grandes (como la región seleccionada actualmente: `Cw`).

Existen otros comandos, que guardan el texto en el *anillo de interrupción* , sin *matarlo* realmente (de forma similar a "copiar" en las terminologías modernas). Por ejemplo, `Mw` , que actúa sobre la región seleccionada actualmente.

Tirón

Las entradas en el *histórico de recortes* más adelante pueden ser *arrancados* de nuevo en una memoria intermedia. Por lo general, se puede *tirar* el texto *eliminado* más recientemente usando, *por ejemplo,* `Cy` (que es similar a la operación de "pegar" en una terminología más moderna). Pero otros comandos pueden acceder y *arrancar* entradas antiguas del *anillo de interrupción* .

Modos

Modo mayor

Emacs puede adaptar su comportamiento al tipo específico de texto editado en un *búfer* . El conjunto de personalizaciones específicas de Emacs para un tipo particular de texto se denomina "modo principal". Cada búfer tiene exactamente un *modo principal* dependiendo de su tipo de contenido.

Los modos principales pueden cambiar el significado de algunas teclas, definir el resaltado de sintaxis o las reglas de sangrado, e instalar nuevos enlaces de teclas (generalmente comenzando con `Cc`) para los comandos específicos del modo. Emacs se envía con una amplia gama de

modos principales, que se dividen en tres categorías principales:

- soporte para texto (por ejemplo, lenguajes de marcado),
- soporte para lenguajes de programación,
- aplicaciones dentro de emacs (ej. dired, gnus, ...). Los buffers que usan este último grupo de modos principales generalmente no están asociados a los archivos, sino que sirven como una interfaz de usuario.

Modo menor

Los modos menores son características opcionales que se pueden activar y desactivar. *Los modos menores* pueden habilitarse para búferes específicos (modos de búfer locales) o todos los búferes (modos globales). En contraste con *los modos principales ,* se puede activar cualquier número de *modo menor* para un búfer dado.

Emacs proporciona muchos modos menores. Algunos ejemplos incluyen:

- *Modo de relleno* automático para ajustar automáticamente las líneas de texto a medida que escribe.
- *Modo Flyspell* para resaltar los errores de ortografía mientras escribe.
- *Modo de línea visual* para envolver líneas largas para adaptarse a la pantalla.
- *Modo de marca transitoria* para resaltar la región actual.

Lea Nomenclatura de emacs en línea: https://riptutorial.com/es/emacs/topic/3683/nomenclatura-de-emacs

Capítulo 12: Timón

Examples

Instalación de timón vía MELPA

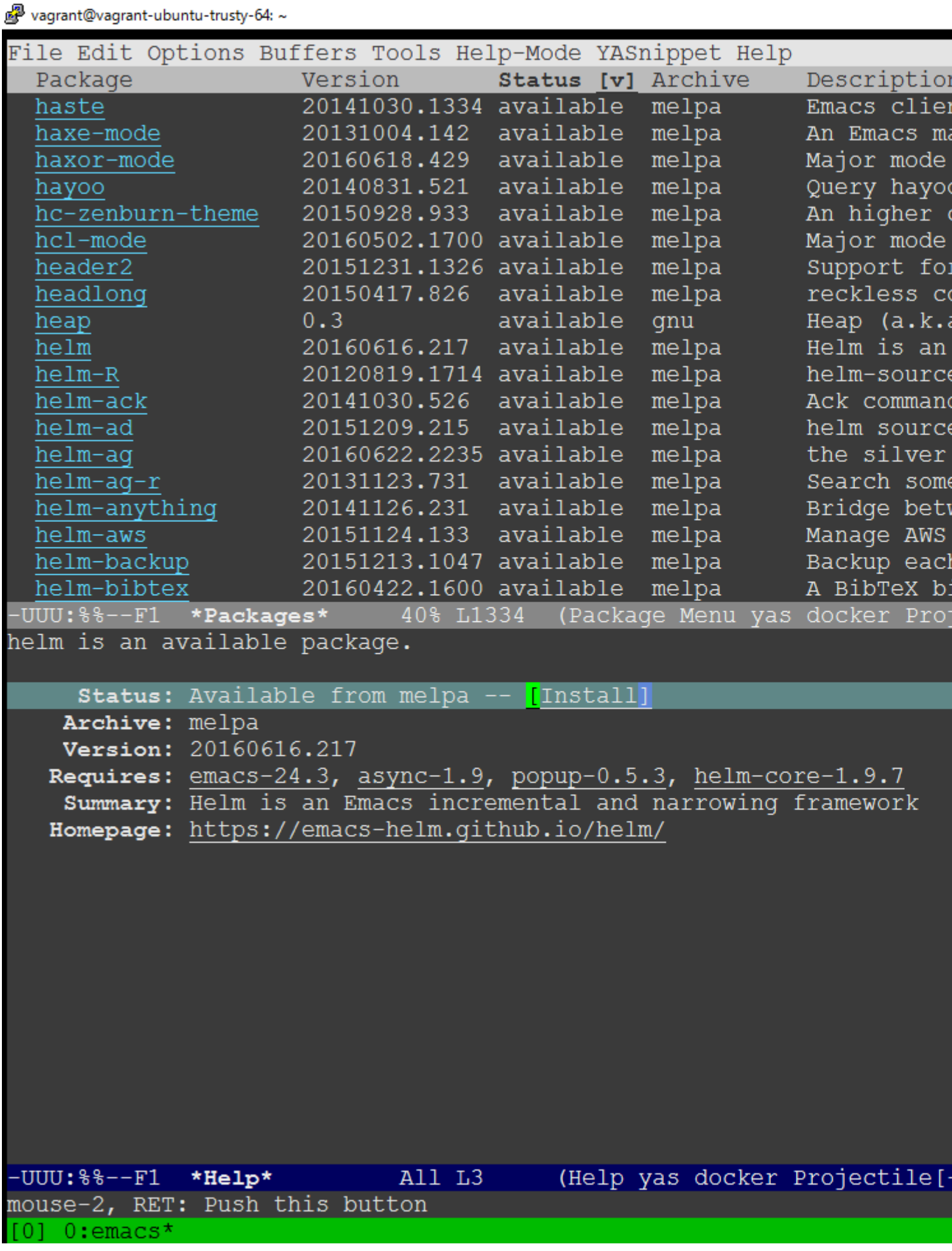
Desde emacs 24.4 package.el está disponible, y una forma de instalar helm es hacerlo a través de MELPA. Primero, agregue el repositorio de MELPA como archivo de paquete colocando el siguiente código en algún lugar de su `~/.emacs` (o, `~/.emacs.d/init.el`).

```
(require 'package)

;; add the repository before the package-initialize.
(add-to-list 'package-archives '("melpa" . "http://melpa.milkbox.net/packages/"))

(package-initialize)
```

A continuación, ingrese `Mx list-packages` para ver la `lista de paquetes` disponibles. Busque la entrada del `helm` , coloque el cursor en la entrada del `helm` y presione `RET` . Verás el búfer de información del paquete. Ponga el cursor en `[Install]` y presione `RET` . Helm será instalado. La ventana de la lista de paquetes y la ventana de información del paquete se muestran en la siguiente imagen.


vagrant@vagrant-ubuntu-trusty-64: ~
File Edit Options Buffers Tools Help-Mode YASnippet Help
Package Version Status [v] Archive Description
haste 20141030.1334 available melpa Emacs clien
haxe-mode 20131004.142 available melpa An Emacs ma
haxor-mode 20160618.429 available melpa Major mode
hayoo 20140831.521 available melpa Query hayoo
hc-zenburn-theme 20150928.933 available melpa An higher
hcl-mode 20160502.1700 available melpa Major mode
header2 20151231.1326 available melpa Support for
headlong 20150417.826 available melpa reckless co
heap 0.3 available gnu Heap (a.k.a
helm 20160616.217 available melpa Helm is an
helm-R 20120819.1714 available melpa helm-source
helm-ack 20141030.526 available melpa Ack command
helm-ad 20151209.215 available melpa helm source
helm-ag 20160622.2235 available melpa the silver
helm-ag-r 20131123.731 available melpa Search some
helm-anything 20141126.231 available melpa Bridge betw
helm-aws 20151124.133 available melpa Manage AWS
helm-backup 20151213.1047 available melpa Backup each
helm-bibtex 20160422.1600 available melpa A BibTeX bi
-UUU:%%--F1 *Packages* 40% L1334 (Package Menu yas docker Proj
helm is an available package.
Status: Available from melpa -- [Install]
Archive: melpa
Version: 20160616.217
Requires: emacs-24.3, async-1.9, popup-0.5.3, helm-core-1.9.7
Summary: Helm is an Emacs incremental and narrowing framework
Homepage: https://emacs-helm.github.io/helm/
-UUU:%%--F1 *Help* All L3 (Help yas docker Projectile[
mouse-2, RET: Push this button
[0] 0:emacs*

Lea Timón en línea: https://riptutorial.com/es/emacs/topic/5341/timon

Creditos

S. No	Capítulos	Contributors
1	Empezando con emacs	Adobe, Community, ebpa, Ehvince, eyqs, IntFloat, Jack Henahan, joon, legoscia, mellowmaroon, Michel de Ruiter, Nemanja Trifunovic, omul, pcurry, Prasanna, salotz, squiter
2	Administrar marcadores dentro de Emacs	Francesco, Prasanna
3	Ayuda dentro de Emacs	mellowmaroon, Terje D., ygram
4	Emacs ya tiene una documentación de alta calidad y bien organizada. ¿Por qué duplicarlo?	erjoalgo
5	Gestión de paquetes	Adobe, Kaushal Modi, leeor, pcurry, salotz, squiter
6	Keybindings básicos	Adeel Ansari, Arjun J Rao, boehm_s, Francesco, Idan, Jeff Bencteux, julienc, leeor, Meaningful Username, mellowmaroon, Nikana Reklawyks, Prasanna, SuperBear, Tej Chajed, Terje D.
7	Kits de inicio	dangom, Ehvince, Kaushal Modi, leeor, Micah Elliott, Xinyang Li
8	Las muchas variantes de Emacs	pcurry
9	Magit	boehm_s, Ehvince, glallen, mellowmaroon, squiter
10	Modo org	Christopher Bottoms, Francesco, legoscia, SuperBear, Wazam
11	Nomenclatura de emacs	Doug Harris, Francesco, Nikana Reklawyks, Stephen Leppik, Terje D.
12	Timón	Yuki Inoue