**EBook Gratuito**

# APPRENDIMENTO emacs

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#emacs

# Sommario

# Di

# Capitolo 1: Iniziare con emacs

## Osservazioni

Emacs è un editor di testo la cui caratteristica principale è la capacità degli utenti di personalizzare a livello di programmazione quasi tutti gli aspetti di esso. Questo è facilitato attraverso uno speciale dialetto del linguaggio di programmazione Lisp, chiamato Emacs Lisp, creato appositamente per essere utilizzato nell'editor di Emacs.

Ci sono una moltitudine di estensioni scritte in Emacs Lisp che si aggiungono alle funzionalità di Emacs. Queste estensioni includono funzionalità di editing per specifici linguaggi di programmazione (simile a quello che potrebbe fornire un IDE), e-mail e client IRC, frontend Git, giochi come Tetris e 2048 e molto altro.

Molti aspetti dell'editor di Emacs possono essere utilizzati senza conoscenze di programmazione. Gli utenti che cercano di personalizzare a livello di codice Emacs, tuttavia, troveranno alcune funzionalità del linguaggio Lisp di Emacs come il sistema di (auto) documentazione incredibilmente utile e accomodante.

**Riferimenti esterni:**

1. Il sito di Sacha Chua è un ottimo sito per trovare più risorse di apprendimento su Emacs.

   un. Per coloro che hanno bisogno di un maggiore appeal visivo sul percorso di apprendimento di Emacs

   b. Per coloro che vorrebbero ottenere facilmente le associazioni chiave

2. Wikemacs è basato su mediawiki e quindi ha contenuti strutturati, categorie sfogliabili e simili. Inizia ad esplorare !

## Versioni

| Versione | Data di rilascio |
|---|---|
| 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 |

| Versione | Data di rilascio |
|---|---|
| 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

### Installazione o configurazione

Istruzioni dettagliate su come installare o installare emacs.

Le istruzioni ufficiali sono disponibili sul sito Web GNU Emacs .

# Sistemi Debian

Sui sistemi con il gestore di pacchetti Debian (come Debian, Ubuntu e Mint) Emacs può essere installato tramite il semplice comando:

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

Per una versione all'avanguardia, è possibile utilizzare il seguente ppa:

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

## Costruisci per la fonte

Se la tua distro basata su Debian non ha la versione di emacs che desideri, puoi crearla da zero.

```
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*
```

# Sistemi Redhat

Sui sistemi con il gestore di pacchetti Redhat (come RHEL, CentOS e Fedora Core) Emacs può essere installato tramite il semplice comando:

```
sudo yum install emacs
```

# Arch Linux

Emacs può essere installato tramite il semplice comando:

```
sudo pacman -Syu emacs
```

# Gentoo e Funtoo

Sui sistemi che eseguono Portage, Emacs può essere installato tramite il semplice comando:

```
sudo emerge emacs
```

# GSRC (Collezione di rilascio di GNU Source)

Funziona su qualsiasi sistema GNU / Linux per ottenere l'ultima versione di emacs senza utilizzare il gestore di pacchetti di sistema (che potrebbe non essere aggiornato) o scaricare l'archivio o il file binario. Per installare `gsrc` vedi la sua documentazione . Poi:

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

# Sistemi 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
```

## Pacchetto di app

I pacchetti di app precompilati per le ultime versioni stabili e di sviluppo possono essere scaricati su https://emacsformacosx.com .

# finestre

## Responsabile pacchetto Chocolatey

Emacs può essere installato con

```
choco install emacs
```

## Scoop gestore pacchetti

Può essere installato dalla benna extra

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

## Installatori binari ufficiali

- stabile
- più recente

(Si noti che i binari ufficiali non vengono forniti con alcune librerie, ad es. Librerie per i formati di immagine)

## Altri programmi di installazione binari

- Emacs con AUCTeX e ESS precompilati
- GNU Emacs a 64 bit per MS Windows con ottimizzazione fornisce un programma di installazione binario nativo e ottimizzato a 64 bit con codice sorgente non modificato da Git Master e versione di rilascio, con supporto JPEG, GIF, PNG, TIFF, SVG, XML2 e GnuTLS out-of- scatola.

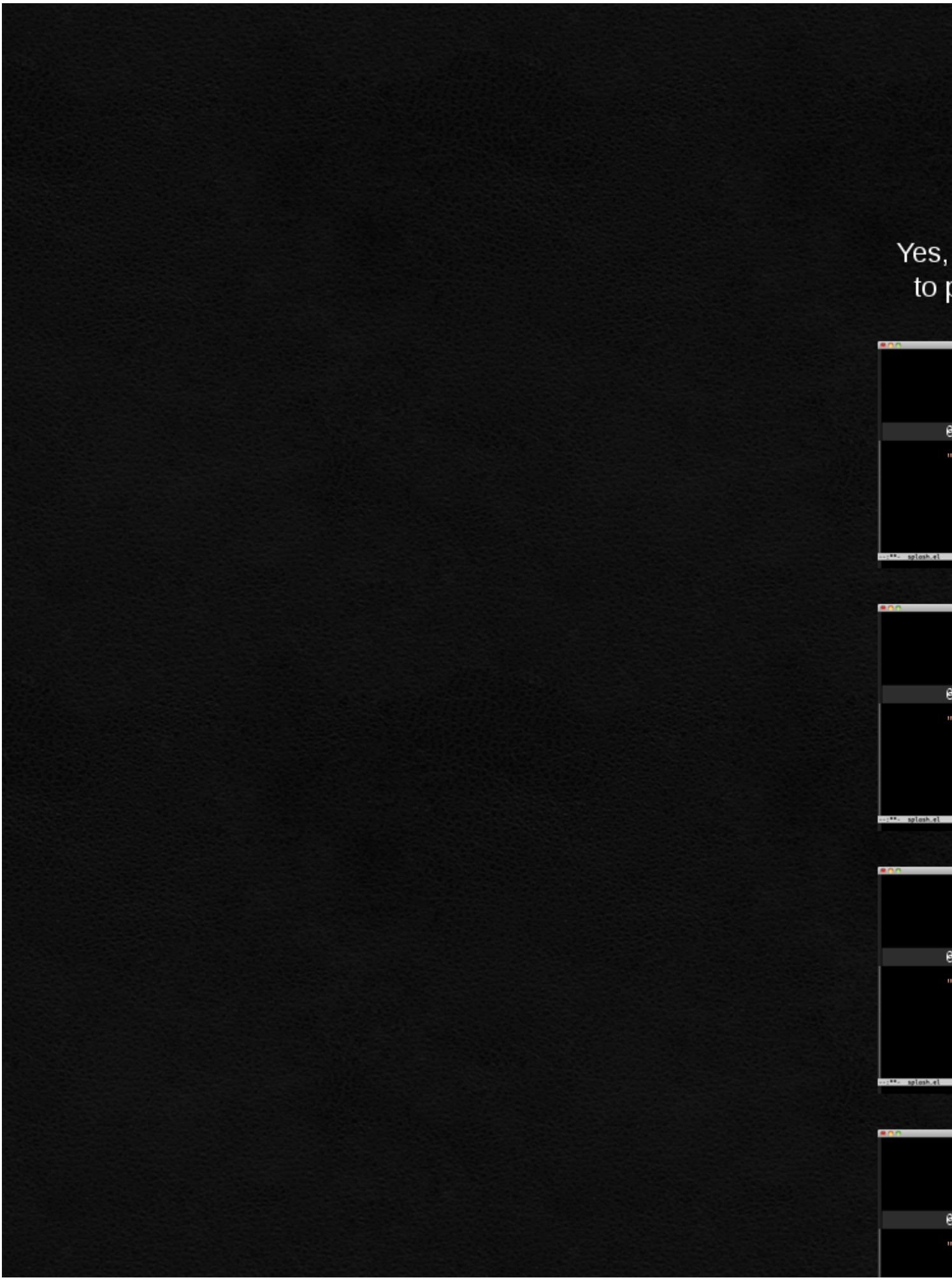
### Tutorial interattivo di Emacs

All'interno di Emacs, digita `Ch t` (Control-h, t) per ottenere un eccellente tutorial interattivo all'interno di Emacs. L'utente apprende la navigazione e il montaggio di base operando sul testo TUTORIAL stesso, mentre leggono il tutorial. (Le modifiche al tutorial vengono scartate quando il tutorial è chiuso, quindi ogni volta che un utente richiede il tutorial, è una versione predefinita pulita del tutorial.

Sicuramente, la prima cosa nel tutorial è come capire i riferimenti `C-<chr>` e `M-<chr>` nel testo. La seconda cosa è come andare avanti e indietro nel testo.

### Emacs Rocks Tutorial video

Buoni tutorial video su Emacs possono essere trovati su emacsrocks.com .


Yes,
to

# Capitolo 2: Aiuto all'interno di Emacs

## Osservazioni

Emacs è descritto come un editor di auto-documentazione e fornisce molte informazioni su come utilizzarlo all'interno dell'editor stesso. Tra i punti di accesso a questa documentazione vi sono un tutorial, informazioni su quali funzioni sono disponibili relative a un determinato argomento, informazioni sui collegamenti tra tasti e funzioni. Si accede alla documentazione usando il prefisso `Ch` , *cioè* `Ctrl h` o `F1` , con una lista di ulteriori scelte disponibili premendo `?`

## Examples

### Esercitazione Emacs

`Ch t` corre la funzione di `help-with-tutorial` , che apre un buffer contenente un tutorial sulla funzionalità di editing di base emacs, tra cui muoversi nel testo, e lavorare con i file, tamponi, e le finestre.

### Funzioni disponibili e associazioni di tasti

Premendo `Ch a` si eseguirà la funzione `apropos-command` che fa in modo che emacs richieda le parole (o un'espressione regolare) da cercare. Mostrerà quindi un buffer contenente un elenco di nomi e descrizioni relativi a quell'argomento, compresi i binding di chiavi per ciascuna delle funzioni disponibili tramite i tasti.

Premendo `Ch m` ( `describe-mode` ) si fornisce un buffer che descrive le modalità principali e secondarie in vigore, inclusi gli elenchi delle funzioni disponibili e le loro associazioni di tasti.

Premendo `Ch b` ( `describe-bindings` ) si fornisce un buffer che elenca tutti i collegamenti attuali dei tasti. L'elenco include binding globali e binding per le modalità major e secondarie attive nel buffer corrente.

### Documentazione chiave vincolante

`Ch k` esegue la funzione `describe-key` , che ricerca la funzione associata ai tratti di tasto forniti e presenta una descrizione della funzione che verrà eseguita quando questi tasti vengono premuti.

`Ch c` esegue `describe-key-briefly` la funzione `describe-key-briefly` , che visualizza solo il nome della funzione mappato su una determinata sequenza di tasti.

### Documentazione delle funzioni

`Ch f` esegue la funzione `describe-function` , che mostra informazioni sull'uso e lo scopo di una determinata funzione. Questo è particolarmente utile per le funzioni che non hanno un legame chiave mappato che può essere usato per la ricerca della documentazione tramite `Ch k` .

Leggi Aiuto all'interno di Emacs online: https://riptutorial.com/it/emacs/topic/4736/aiuto-all-interno-di-emacs

# Capitolo 3: emacs ha già una documentazione di alta qualità e ben organizzata. perché duplicarlo?

## introduzione

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

## Examples

### chiavi

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

3 chiavi

Alcuni comandi di Emacs sono richiamati da un solo evento di input; per esempio, Cf sposta un carattere nel buffer. Altri comandi richiedono due o più eventi di input da invocare, come Cx Cf e Cx 4 Cf.

Una sequenza di tasti, o tasto in breve, è una sequenza di uno o più eventi di input che è significativa come un'unità. Se una sequenza di tasti richiama un comando, lo chiamiamo una chiave completa; ad esempio, Cf, Cx Cf e Cx 4 Cf sono tutte chiavi complete. Se una sequenza di tasti non è abbastanza lunga per invocare un comando, la chiamiamo chiave di prefisso; dall'esempio precedente, vediamo che Cx e Cx 4 sono chiavi di prefisso. Ogni sequenza di tasti è una chiave completa o una chiave di prefisso.

Leggi emacs ha già una documentazione di alta qualità e ben organizzata. perché duplicarlo? online: https://riptutorial.com/it/emacs/topic/9077/emacs-ha-gia-una-documentazione-di-alta-qualita-e-ben-organizzata--perche-duplicarlo-

# Capitolo 4: Gestione dei pacchetti

## Examples

### Installazione automatica dei pacchetti su start-up di 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 '()))
```

---

# Riferimenti

- Confronto dei repository dei pacchetti
- Post del blog sui pacchetti selezionati dall'utente in emacs 25.1

## Installazione automatica dei pacchetti con pacchetto di utilizzo

```
;; 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)
```

## Gestione automatica dei pacchetti tramite Cask

Cask è uno strumento di gestione del progetto che può essere utilizzato anche per gestire facilmente la configurazione di emacs locale.

Installare la botte è facile. È possibile eseguire il seguente comando sulla riga di comando:

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

Oppure se sei su un Mac, puoi installarlo usando `homebrew` :

```
brew install cask
```

Una volta installato, si crea un file `Cask` . I file Cask elencano tutte le dipendenze del pacchetto che dovrebbero essere incluse nella configurazione. Puoi creare un nuovo file Cask nella root della tua directory `~/.emacs` .

Avrai anche bisogno di inizializzare Cask nel tuo `~/.emacs.d/init.el` Se hai installato usando homebrew, aggiungi queste righe:

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

Oppure puoi fornire il percorso alla botte, se hai usato lo script di installazione:

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

Un semplice file Cask ha questo aspetto:

```
(source gnu)
(source melpa)

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

Qui stiamo specificando i repository sorgente in cui cercare i pacchetti. Quindi stiamo specificando che vogliamo installare i pacchetti `projectile` , `flx` e `flx-ido` .

Una volta che hai un file Cask, puoi installare tutte le dipendenze con il comando follwoing sulla riga di comando:

```
cask install
```

## Gestione automatica dei pacchetti con el-get

el-get è un sistema di gestione dei pacchetti open source per GNU Emacs. el-get funziona con `melpa` , così come con molti sistemi di controllo delle versioni più comuni. La sua documentazione include un semplice auto-installazione per i tuoi `.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 installazioni di pacchetti in una struttura di directory in `~/.emacs.d/el-get` . Carica le definizioni da `~/.emacs.d/el-get/.loaddefs.el` e tiene traccia dello stato del pacchetto con `~/.emacs.d/el-get/.status.el` . `(el-get 'sync)` installa o rimuove i pacchetti per portare lo stato macchina reale in sincrono con il pacchetto `.status.el` .

el-get è auto-ospitato - qui è il suo stato da `.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))))
```

Leggi Gestione dei pacchetti online: https://riptutorial.com/it/emacs/topic/2414/gestione-dei-pacchetti

# Capitolo 5: Gestisci i segnalibri all'interno di Emacs

## Examples

### Come contrassegnare i file utilizzati di frequente

Utilizzare i seguenti comandi per creare segnalibri e accedere ai segnalibri da Emacs.

Diciamo che stai modificando un file chiamato `foobar.org` e supponiamo che visiti questo file frequentemente per modificare / visualizzare i contenuti.

Sarebbe conveniente accedere a questo file con un paio di tratti chiave piuttosto che navigare attraverso la struttura dei file (Dired) e visitare il file.

**passi:**

1. Apri `foobar.org` per una volta navigando nel file ( *visita il file* nel gergo di Emacs)
2. Mentre il file è aperto, digitare `Cx r m` questo richiederà di fornire il nome del segnalibro per il file. Diciamo `foobar` in questo caso.
3. Chiudere il file ( `Cx k` - kill buffer) - salvare se necessario
4. Ora a visitare il file, basta digitare `Cx r l` - questo compilerà un elenco che conterrà `foobar` .
5. Seleziona il `foobar` e premi il tasto `Invio`
6. Utilizzare `Mx bookmark-delete` per eliminare qualsiasi segnalibro non necessario di un file.

*Nota: l'* eliminazione di un segnalibro è analoga all'eliminazione di un collegamento sul desktop di Windows.
Il file principale sarà al sicuro nella sua posizione e verrà rimosso solo l'elenco del file dal menu dei segnalibri.

Leggi Gestisci i segnalibri all'interno di Emacs online:
https://riptutorial.com/it/emacs/topic/5837/gestisci-i-segnalibri-all-interno-di-emacs

# Capitolo 6: Keybindings di base

## Examples

### Esci da Emacs

Puoi uscire da Emacs con la seguente combinazione di tasti:

```
Cx Cc
```

Dove `C` è la chiave di `control` .

# Sospendi Emacs

Puoi sospendere Emacs usando la seguente combinazione di tasti:

```
Cz
```

Ti riporta alla tua shell. Se vuoi riprendere la sessione di emacs, inserisci `fg` nel tuo terminale.

### Gestione dei file

- Re-Salva il file aperto con lo stesso nome file (Salva):

  ```
  Cx Cs
  ```

- Scrivi come `filename` (Salva come):

  ```
  Nome filename Cx Cw
  ```

  Il nuovo nome del file verrà richiesto nel minibuffer.

- Crea un nuovo file **o** carica un file esistente (Nuovo / Carica):

  ```
  Nome filename Cx Cf
  ```

  Con il mnemonico qui per il file f significato. Ti verrà richiesto un percorso file nel minibuffer.

- Visita il file alternativo

  ```
  Cx Cf
  ```

  Se il file non esiste ancora, ti verrà richiesto il percorso del file da creare nel minibuffer.

### Interrompi il comando corrente

Spesso si entra in uno stato in cui si ha una sequenza di comandi parzialmente digitata in corso, ma si desidera annullarla. Puoi interromperlo con una delle seguenti combinazioni di tasti:

`cg`

`Esc Esc Esc`

## Finestre o cornici multiple

"Finestra" in Emacs si riferisce a ciò che altrimenti si potrebbe chiamare "pannello" o "divisione schermo". Alcuni comandi di manipolazione delle finestre includono:

- Dividi la finestra corrente in orizzontale: `Cx 2`
- Dividi la finestra corrente verticalmente: `Cx 3`
- Seleziona la finestra successiva: `Cx o`
- Chiudi finestra corrente: `Cx 0`
- Chiudi tutte le altre finestre, tranne quella attuale: `Cx 1`

Un "frame" in Emacs è ciò che altrimenti potrebbe essere chiamato una "finestra". I frame sono manipolati usando questi comandi:

- Crea nuovo frame: `Cx 5 2`
- Elimina il frame corrente: `Cx 5 0`
- Elimina altri frame: `Cx 5 1`

Il passaggio a finestre può essere effettuato utilizzando

- `S-sinistra` , `S-destra` , `S-up` , `S-down` (ovvero, `Shift` in combinazione con un tasto freccia) per passare alla finestra adiacente in una direzione, o
- `Cx o` per passare alla finestra successiva.

## buffer

- Esempio di un elenco di buffer

```
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
```

  Il primo campo di una riga indica:

  - '' il buffer è corrente.
  - '%' un buffer di sola lettura.
  - '*' il buffer è modificato.

- Seleziona buffer. È possibile selezionare qualsiasi buffer aperto con la seguente

combinazione di tasti:

```
Cx b
```

Ti verrà richiesto il nome del buffer che desideri passare.

- Elenco dei buffer:

```
Cx Cb
```

- Save-some-buffer, dando la scelta di quale buffer salvare o meno:

```
Cx s
```

- Uccidi un buffer:

```
Cx k
```

- Operazioni sui buffer contrassegnati:

`S` Salva i buffer marcati

`A` Visualizza i buffer marcati in questo frame.

`H` Visualizza i buffer marcati in un altro frame.

`V` Ripristina i buffer contrassegnati.

`T` Attiva lo stato di sola lettura dei buffer contrassegnati.

`D` Uccidi i buffer marcati.

`Ms a Cs` Effettua ricerca incrementale nei buffer marcati.

`Ms a CMs` Isearch per regexp nei buffer marcati.

`U` Sostituisci con regexp in ciascuno dei buffer marcati.

`Q` Query sostituisce in ciascuno dei buffer contrassegnati.

`I` Come sopra, con un'espressione regolare.

`P` Stampa i buffer marcati.

`O` Elenca le righe in tutti i buffer marcati che corrispondono ad una espressione regolare (come la funzione `occurr ').

`X` Passare il contenuto dei buffer marcati a un comando shell.

`N` Sostituisce il contenuto dei buffer contrassegnati con l'output di un comando shell.

`!` Esegui un comando shell con il file del buffer come argomento.

`E` Valutare un modulo in ciascuno dei buffer contrassegnati. Questo è un comando molto

flessibile. Ad esempio, se si desidera che tutti i buffer contrassegnati siano di sola lettura, provare a utilizzare (modalità sola lettura 1) come modulo di input.

`W` - Come sopra, ma visualizza ogni buffer mentre il modulo viene valutato.

`k` - Rimuovi le righe contrassegnate dal buffer di *Ibuffer* , ma non uccidere il buffer associato.

`x` - Elimina tutti i buffer contrassegnati per l'eliminazione.

- Save-some-buffer, dando la scelta di quale buffer salvare o meno:

  `Cx s`

- Passa al buffer successivo:

  `Cx a DESTRA`

- Passa al buffer precedente:

  Cx SINISTRA

## Cerca e sostituisci

In Emacs, lo strumento di ricerca di base ( `I-Search` ) consente di cercare prima o dopo la posizione del cursore.

- Per cercare il *testo* dopo la posizione del cursore ( `search-forward` ), premi `Cs al` *`sometext`* . Se vuoi passare alla prossima occorrenza del *`sometext`* , premi nuovamente `Cs` (e così via per le prossime occorrenze). Quando il cursore si posiziona nella posizione corretta, premere `Invio` per uscire dal prompt di ricerca.

- Per cercare prima la posizione del cursore ( `search-backward` ), utilizzare `Cr` nello stesso modo in cui è stato utilizzato in precedenza.

- Per passare dalla `search-backward` alla `search-forward` , premere 2 volte `Cs` . E premi 2 volte `Cr` per `search backward` quando sei in `search-forward` prompt di `search-forward` .

- Cerca e sostituisci:

  `M-%` (o `Esc-%` ) `oldtext Immettere nuovo newtext Invio`

  - Conferma: `y`
  - Skip: `n`
  - Esci: `q`
  - Sostituisci tutto `:!`

## Regione - Taglia, Copia, Incolla

- Imposta il segno nella posizione del cursore:

  `C-spazio` o `C- @`

- Kill region (Cut):

  `Cw`

- Copia regione per uccidere squilli:

  `Mw` O `Esc-w`

- Yank (Paste) più recentemente ucciso:

  `Cy`

- Yank (Paste) l'ultima volta ucciso:

  `My` O `Esc-y`

# Uccidere

`kill` è il comando utilizzato da Emacs per la cancellazione del testo. Il comando `kill` è analogo al comando di `cut` in Windows. Esistono vari comandi che "uccidono" una parola ( `Md` ), il resto della linea ( `Ck` ) o blocchi di testo più grandi. Il testo cancellato viene aggiunto al `kill-ring` , da cui può successivamente essere `yanked` .

## Seleziona e taglia (uccidi)

Killing e yanking Simile alla funzione `select-and-cut` di Windows, qui abbiamo `C-spc` .
L'associazione chiave `C-spc` avvierà la selezione, l'utente può spostare il segno con l'aiuto di tasti freccia o altro comando per effettuare una selezione. Una volta completata la selezione, `premi Cw` per eliminare il testo selezionato

Alcuni comandi di base che possono essere utilizzati come riferimento rapido per il comando `kill` (tratto dal tutorial di 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
```

# strattone

`yank` descrive l'inserimento di testo cancellato in precedenza, *ad es.* usando `Cy` che tira il testo ucciso più di recente. `Yank` comando `Yank` è analogo al comando `paste` in Windows.

## Testo strattonato precedentemente ucciso

Sappiamo che il comando `kill` aggiunge il testo ucciso a un `kill-ring` . Per recuperare il testo

cancellato dal `kill-ring` usare ripetutamente il comando `My` fino a quando il testo desiderato viene strattonato.

*(Nota: se il tasto `My` funziona, il comando precedente dovrebbe essere uno `YANK` altrimenti non funzionerebbe)*

## Cursore (punto) movimento

Oltre ai movimenti del cursore usando i tasti freccia, Home, Fine, Pagina su e Pagina giù, emacs definisce un numero di sequenze di tasti che possono spostare il cursore su parti di testo più piccole o più grandi:

**Per carattere:**

- Personaggio all'indietro: `Cb`
- Carattere in avanti: `Cfr`

**A parole**

- Parola indietro: `Mb` ( *cioè* `Alt b` , o `Meta b` )
- Forward word: `Mf`

**Per linea:**

- Inizio della linea corrente: `Ca`
- Inizio della riga corrente (carattere non spaziale): `Mm`
- Fine della linea attuale: `Ce`
- `Riga` precedente: `Cp`
- Prossima riga: `Cn`

**Buffer completo:**

- Inizio del buffer: `M- <`
- Fine del buffer: `M->`

**Per "blocco", a seconda del contesto (modalità):**

Tipici attacchi chiave:

- Frase / affermazione all'indietro: `Ma`
- Inoltro frase / affermazione: `Me`
- Inizio della funzione: `MCa`
- Fine della funzione: `MCe`

**Argomenti prefisso**

Per spostare più "passi" contemporaneamente, ai comandi di movimento può essere assegnato un argomento di prefisso premendo `ESC` o `Cu` e un numero prima delle sequenze di tasti elencate. Per `Cu` , il numero è facoltativo e il valore predefinito è 4.
*Ad esempio,* `ESC 3 Cn` sposta 3 linee in basso, mentre `Cu Mf` sposta in avanti di 4 parole.

### Disfare

Per annullare qualcosa che hai appena fatto:

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

### Astuccio

- Parola in maiuscolo: `Mc`
- Convertire la parola in maiuscolo: `Mu`
- Convertire la parola in minuscolo: `Ml`

### Notazione dei tasti chiave

La documentazione di Emacs utilizza una notazione coerente per tutte le associazioni di tasti, che è spiegata qui:

## Accordi chiave

Un "accordo chiave" si ottiene premendo due o più tasti contemporaneamente. Gli accordi chiave sono contraddistinti dalla separazione di tutti i tasti da trattini ( `-` ). Di solito coinvolgono i tasti modificatori, che sono messi in primo piano:

- `C-` : controllo;
- `S-` : shift;
- `M-` : alt (la "M" sta per "Meta" per ragioni storiche).

Altre chiavi sono semplicemente indicate dal loro nome, come:

- `R`: Il `a` chiave;
- `a sinistra` : il tasto freccia sinistra;
- `SPC` : la chiave dello spazio;
- `RET` : la chiave di ritorno.

Esempi di accordi chiave includono quindi:

- `Ca` : premendo `control` e `a` contemporaneamente;
- `S-destra` : premendo `shift` e `destra` contemporaneamente;
- `CMa` : premendo `control` , `alt` e `a` simultaneamente.

## Sequenze chiave

Le "sequenze di tasti" sono sequenze di tasti (o accordi chiave), che devono essere digitati uno dopo l'altro. Sono indicati separando tutte le notazioni chiave (o accordi) da uno spazio.

Esempi inclusi:

- `Cx b` : premendo simultaneamente `control` e `x` , quindi rilasciandoli e premendo `b` ;
- `Cx Cf` : premendo simultaneamente `control` e `x` , rilasciando `x` e premendo `f` (poiché entrambi gli accordi coinvolgono il modificatore di `controllo` , non è necessario rilasciarlo).

## Usare ESC invece di Alt

Gli accordi chiave che usano il modificatore Alt possono anche essere inseriti come sequenza di tasti a partire da `ESC` . Questo può essere utile quando usi Emacs su una connessione remota che non trasmette gli accordi chiave Alt, o quando queste combinazioni di tasti sono catturate, *ad esempio* da un gestore di finestre.

Esempio:

`Mx` può essere inserito come `ESC x` .

## Descrivere i binding dei tasti nei file lisp di Emacs

La stessa notazione qui descritta può essere utilizzata quando si definiscono i collegamenti dei tasti nei file lisp di Emacs.

Esempio:

(menu-buffer globale-set-chiave (kbd "Cx Cb") ')
lega la sequenza di tasti `Cx Cb` al `buffer-menu` comando

Leggi Keybindings di base online: https://riptutorial.com/it/emacs/topic/3436/keybindings-di-base

# Capitolo 7: Le molte varianti di Emacs

## introduzione

La maggior parte di questa documentazione si applica implicitamente o esplicitamente a GNU Emacs. Questa potrebbe essere la variante più nota di Emacs, nonché la fonte di diverse forcelle e l'obiettivo di alcune fusioni.

Questo argomento discute alcune delle varianti di Emacs che si possono incontrare e le loro principali differenze rispetto a GNU Emacs.

## Examples

### Spacemacs

Spacemacs ( http://spacemacs.org/) è una variante di Emacs che tenta di porre fine al conflitto a lungo termine tra Emacs e gli utenti vim, creando un Emacs che si comporta come Vim.

Leggi Le molte varianti di Emacs online: https://riptutorial.com/it/emacs/topic/9456/le-molte-varianti-di-emacs

# Capitolo 8: magit

## introduzione

Magit è un'interfaccia per il sistema di controllo della versione Git, implementato come pacchetto Emacs. Ti permette di interagire con git in Emacs.

## Osservazioni

Magit è un'interfaccia per il sistema di controllo della versione Git, implementato come pacchetto Emacs. Magit aspira ad essere una porcellana Git completa. Anche se non possiamo (ancora) affermare, che Magit comprenda e migliori ogni singolo comando Git, è abbastanza completo da permettere anche agli utenti Git esperti di eseguire quasi tutte le loro attività quotidiane di controllo della versione direttamente da Emacs. Mentre esistono molti buoni client Git, solo Magit e Git meritano di essere chiamati porcellane.

Nota che Magit può interfacciarsi con Github (con Magithub , vedi anche l' integrazione con Github in Emacs ) e che Emacs ha anche pacchetti per lavorare con Gitlab , Bitbucket e altri.

## Examples

### Installazione

È possibile installare Magit da MELPA con:

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

### Utilizzo di base: commit modifiche non applicate all'interno di un repository esistente

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

Leggi magit online: https://riptutorial.com/it/emacs/topic/3909/magit

# Capitolo 9: Nomenclatura Emacs

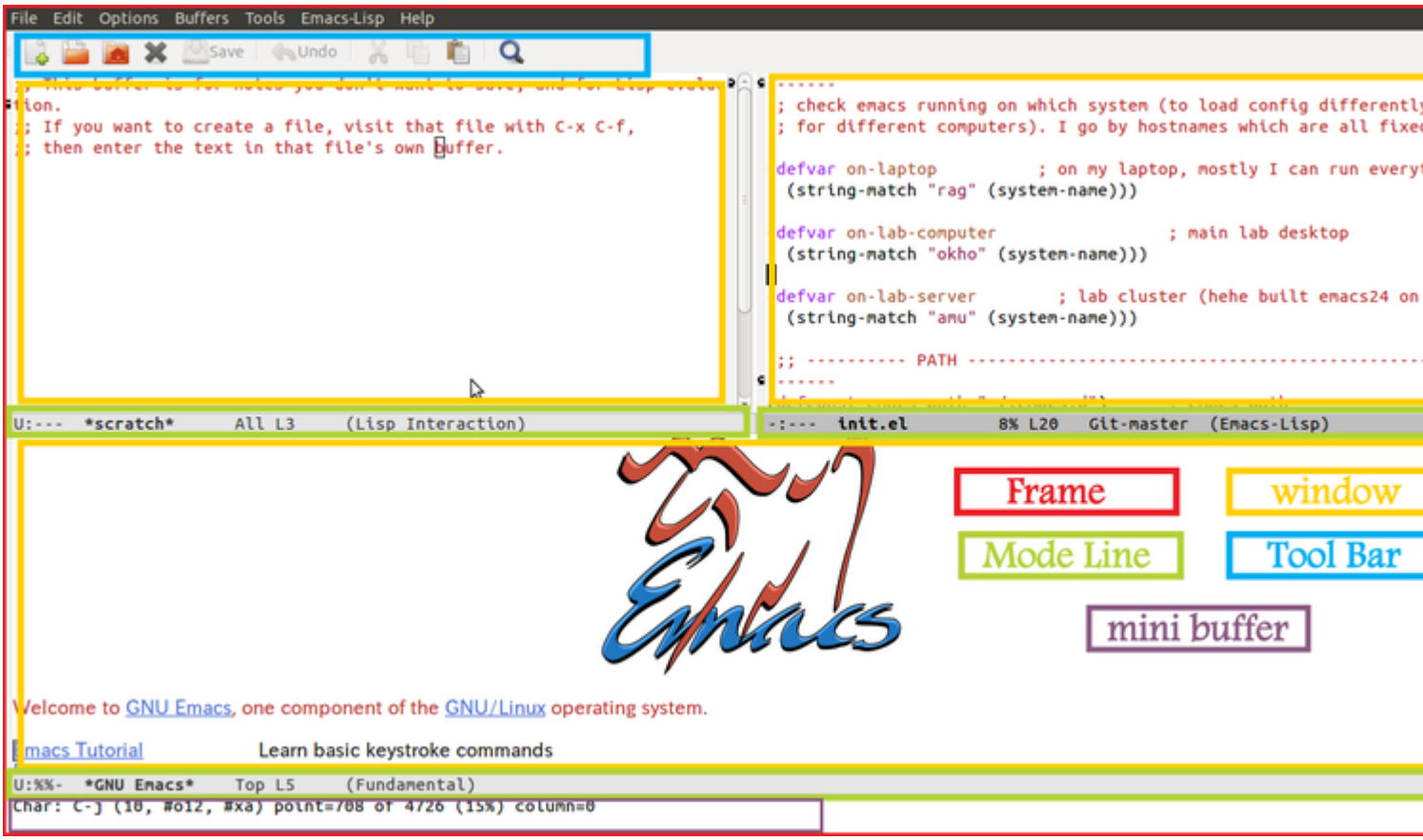
## Examples

### File e buffer

In Emacs, il *file* ha lo stesso significato del sistema operativo e viene utilizzato per l'archiviazione permanente dei dati. Un *buffer* è la rappresentazione interna di un file in fase di modifica. I file possono essere letti nei buffer utilizzando `Cx Cf` e i buffer possono essere scritti su file utilizzando `Cx C` (salva il file nella posizione corrente) o `Cx Cw` (scrive il file in un'altra posizione, richiedendolo - l'equivalente di `Salva con nome` ).

### Elementi dell'interfaccia utente

L'interfaccia utente di Emacs utilizza termini che sono stati coniati in anticipo e possono destabilizzare gli utenti abituati a una terminologia più moderna.




# Telaio

In Emacs, quella che viene altrimenti chiamata una finestra (l'area del display usata da un programma) è chiamata *frame* . Emacs inizia a usare un *frame* , anche se è possibile creare frame aggiuntivi usando `Cx 5` .

# Finestra

Una *cornice* contiene una o più *finestre* (altrimenti di solito chiamate riquadri), ciascuna delle quali mostra il contenuto di un *buffer* . Ogni *frame* solitamente inizia con una sola *finestra* , ma è possibile creare finestre aggiuntive suddividendo quelle esistenti; o orizzontalmente usando `Cx 2` o verticalmente con `Cx 3` . Vedi anche Multipli finestre o cornici .

# Buffer

Il termine *buffer si* riferisce al contenuto visualizzato in una *finestra* . Tale contenuto può riflettere il contenuto di un file nel file system (o forse una versione aggiornata che non è ancora stata salvata sul disco), ma più in generale può essere qualsiasi tipo di testo.

# Linea di modo

Nella parte inferiore di ogni *finestra* c'è una *riga della modalità* , che sinteticamente descrive il *buffer* visualizzato nella finestra.

# Barra degli strumenti

In modo simile a molti altri software, una *barra degli strumenti* può essere visualizzata nella parte superiore di ciascun *fotogramma* . Il suo contenuto può variare a seconda del tipo di *buffer* attualmente modificato.

# minibuffer

Un *minibuffer* , solitamente visualizzato nella parte inferiore di ciascun *fotogramma* , consente di interagire con Emacs. Ogni volta che un comando richiede l'input dell'utente, viene richiesto nel *minibuffer* . Al contrario, i messaggi visualizzati per l'utente da vedere vengono stampati lì.

### Punto, segno e regione

Emacs utilizza i termini **point** , **mark** e **region** per fornire maggiore precisione sul testo selezionato e sulla posizione del cursore. Comprendendo questi termini, ti aiuterà a capire e usare altre operazioni e funzioni.

Il **punto** è il posto in un buffer in cui è in corso la modifica ( *cioè l'* inserimento) e viene solitamente indicato da un cursore.
Il **segno** è un indicatore posizionato in qualsiasi punto del buffer utilizzando comandi come `set-mark-command` ( `C-SPC` ) o `exchange-point-and-mark` ( `Cx Cx` ).
La **regione** è l'area tra il punto e il segno e molti comandi operano nella regione, ad *esempio*

eliminazione, controllo ortografico, rientro o compilazione.

Quando fai clic con il mouse su una posizione in un buffer, stai vedendo il punto. Quando selezioni il testo, stai impostando la regione (il testo selezionato) e il segno (all'inizio della selezione).

### Uccidere e strappare

*Uccidere* e *tirare* più o meno corrisponde a ciò che viene solitamente chiamato "tagliare" e "incollare".

## uccisione

*uccidere* significa eliminare il testo e copiarlo *sull'anello uccidere* (che potrebbe essere visto come una sorta di "blocco appunti" nella terminologia "taglia e incolla"). Il *kill ring* è così chiamato perché memorizza diversi pezzi di testo *ucciso* , che possono essere successivamente consultati in ordine ciclico.

Esistono vari comandi che *uccidono* una parola ( `Md` ), il resto della riga ( `Ck` ) o blocchi di testo più grandi (come la regione attualmente selezionata: `Cw` ).

Esistono altri comandi, che salvano il testo *sull'anello di uccisione* , senza effettivamente *ucciderlo* (in modo simile a "copiare" in terminologie moderne). Ad esempio, `Mw` , che agisce sulla regione attualmente selezionata.

## strattoni

Le voci nel *kill ring* possono essere successivamente *ridotte* in un buffer. Di solito si può *tirare* il testo *ucciso* più recentemente usando *ad esempio* `Cy` (che è simile all'operazione "incolla" in una terminologia più moderna). Ma altri comandi possono accedere e *tirare* le voci più vecchie dal *kill ring* .

### Modalità

## Modalità principale

Emacs può adattare il suo comportamento al tipo specifico di testo modificato in un *buffer* . L'insieme di personalizzazioni specifiche di Emac per un particolare tipo di testo è chiamato "modalità principale". Ogni buffer ha esattamente una *modalità principale a* seconda del suo tipo di contenuto.

*Le modalità principali* possono modificare il significato di alcune chiavi, definire le regole di sintassi o di indentazione della sintassi e installare nuove associazioni di tasti (che di solito iniziano con `Cc` ) per i comandi specifici della modalità. Emacs offre una vasta gamma di modalità principali, che si dividono in tre categorie principali:

- supporto per il testo (ad esempio linguaggi di markup),
- supporto per i linguaggi di programmazione,
- applicazioni all'interno di emacs (ad es. dired, gnus, ...). I buffer che utilizzano questo ultimo gruppo di modalità principali di solito non sono associati ai file, ma servono piuttosto come interfaccia utente.

# Modalità secondaria

*Le modalità secondarie* sono funzioni opzionali che possono essere attivate e disattivate. *Le modalità minori* possono essere abilitate per specifici buffer (modalità locale buffer) o tutti i buffer (modalità globali). A differenza *delle modalità principali , è* possibile attivare un numero qualsiasi di *modalità secondaria* per un determinato buffer.

Emacs offre molte modalità secondarie. Alcuni esempi includono:

- *Modalità di riempimento* automatico per avvolgere automaticamente le righe di testo durante la digitazione.
- *Modalità Flyspell* per evidenziare gli errori di ortografia durante la digitazione.
- *Modalità Linea visiva* per avvolgere lunghe linee per adattarsi allo schermo.
- *Modalità segno transitorio* per evidenziare la regione corrente.

Leggi Nomenclatura Emacs online: https://riptutorial.com/it/emacs/topic/3683/nomenclatura-emacs

# Capitolo 10: Org-mode

## Osservazioni

Org è una modalità per conservare le note, mantenere gli elenchi TODO e pianificare il progetto con un sistema di testo semplice veloce ed efficace. È anche un sistema di authoring con supporto unico per la programmazione alfabetica e la ricerca riproducibile.

sito ufficiale di modalità org

## Examples

### Sintassi di markup

Org offre un linguaggio di marcatura completo che aiuta a strutturare il documento e si riflette nel modo più accurato possibile quando si esporta in altri formati (come HTML o LaTeX).

# Struttura

## Titolo del documento

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

## Sezionando

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

## elenchi

```
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
```

## caselle di controllo

Ogni elemento in un elenco semplice può essere trasformato in una casella di controllo avviandolo con la stringa '[]'.

```
* 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
```

- `Casella` di controllo `Cc Cc` org-toggle
- `Cc Cx Cb` casella di controllo dell'attivatore
- `MS` -org-insert-todo-heading
- `Cc Cx o` org-toggle-ordered-property
- `Cc #` org-update-statistics-cookies

# Enfasi e monospazio

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

Il testo nel codice e la stringa verbatim non viene elaborato per la sintassi specifica della modalità Org, viene esportato letteralmente.

# Collegamenti e riferimenti

## link

La modalità Org riconoscerà i formati di URL e li attiverà come link cliccabili. Tuttavia, i collegamenti possono essere esplicitamente dichiarati in questo modo:

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

o in alternativa:

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

## Le note

Le note a piè di pagina possono essere denominate:

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

o anonimo e in linea:

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

## Collegamenti chiave di base

Per ciclare il livello del contorno mostrato:

- `Tab` Ciclo di livello del contorno per una sola intestazione
- `Maiusc-Tab Esegue il` ciclo di livello struttura per l'intero documento

Per scorrere gli stati di `TODO` :

- `Maiuscole-Freccia destra`
- `Maiuscole-Freccia sinistra`

Per aumentare o diminuire il livello gerarchico di un'intestazione

- `Freccia Meta-destra` Aumenta il livello ("aumenta il rientro")
- `Freccia Meta-Sinistra Aumenta il` livello ("diminuisci il rientro")

Per ciclare la priorità per una data intestazione:

- `Shift-Up Arrow`
- `Shift-Down Arrow`

Per spostare un titolo su o giù:

- `Meta-Up Arrow`
- `Meta-Down Arrow`

( `Meta si` riferisce a diversi tasti su tastiere diverse. Molto spesso è `Alt` o  ).

## Blocchi di codice

Per aggiungere un blocco di codice, racchiudilo con `#+BEGIN_SRC language` e `#+END_SRC` . *la lingua* dovrebbe corrispondere alla modalità principale per la lingua in questione, ad esempio la modalità principale per Emacs Lisp è `emacs-lisp-mode` , quindi scrivi `#+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
```

È possibile aprire il blocco di codice in un buffer separato digitando `Cc '` (per `org-edit-special` ). Se non si dispone della modalità principale per la lingua specificata, verrà visualizzato un messaggio di errore come `No such language mode: foo-mode` .

Se il contenuto che desideri inserire nel blocco non è in alcun linguaggio di programmazione, puoi utilizzare `#+BEGIN_EXAMPLE` e `#+END_EXAMPLE` .

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

Ci sono modelli semplici per entrambi. All'inizio della riga, digitare `<s` o `<e` , quindi premere `TAB` . Si espanderà in un blocco con marker di inizio e fine rispettivamente per `SRC` o `EXAMPLE` .

Questi marcatori sono tutti insensibili alle maiuscole e minuscole, quindi se preferisci puoi scrivere `#+begin_src` ecc.

### tabelle

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

Per aggiungere una tabella in org-mode, semplicemente circondare le colonne con una barra ( `|` )

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

Quando premi `Return` dall'interno di una colonna, org-mode creerà automaticamente una nuova riga con le barre.

- `Tab` e `Return` passeranno rispettivamente alla cella o riga successiva (o ne creeranno una nuova se non ce ne sono)
- Puoi scambiare le righe e le colonne con `M-ArrowKey`
- `M-Down` e `M-Right` creeranno rispettivamente una riga (sopra la corrente) e una colonna (a sinistra della corrente)
- `M-Up` e `M-Left` rimuoveranno rispettivamente la riga corrente e la colonna corrente
- `Cc` creo un separatore

Leggi Org-mode online: https://riptutorial.com/it/emacs/topic/6259/org-mode

# Capitolo 11: Starter Kit

## Osservazioni

I kit di avvio consentono ai *nuovi utenti* di iniziare a utilizzare Emacs rapidamente ed evitare alcuni degli ostacoli di installazione che provengono da un sistema maturo come Emacs, uno che è cresciuto attraverso decenni di evoluzione e naturalmente ha alcune peculiarità storiche. *Anche gli utenti esperti* traggono vantaggio dall'avere una configurazione di kit di estensioni curate da altri.

Richiede uno sforzo considerevole per mantenere un insieme di pacchetti e impostazioni che continueranno a funzionare bene insieme con il miglioramento dei pacchetti (o bit-rot) nel tempo. Molti utenti di Emacs non desiderano effettuare questa manutenzione, quindi si rivolgono ai kit di avvio. L'assemblaggio e la manutenzione di un kit hanno una somiglianza su piccola scala con la gestione di una distribuzione Linux.

## Temi e personalizzazione

Alcuni kit di avvio sono a tema; ad esempio, per specifici ambienti di linguaggio di programmazione, creazione di musica o emulazione di un altro editor. Altri mirano a fornire un lavello in cucina per raggruppare moduli confortevoli / produttivi per quante più situazioni o lingue possibili.

La maggior parte dei kit di avviamento ha disposizioni per l'estensione e la personalizzazione. Un utente sovrascriverà particolari associazioni e impostazioni di tasti e sarà in grado di aggiungere pacchetti non ancora forniti.

## Kit popolari

Sono disponibili molti kit di avvio. In teoria, chiunque pubblica il proprio `~/.emacs.d` ha creato uno. Ma una manciata è diventata popolare e ben mantenuta da uno o più individui. Alcuni esempi (in ordine di popolarità soggettiva basata su stelle Github) includono Spacemacs , Prelude , Purcell , Emacs Starter Kit , Magnars ed Emacs Live . Maggiori dettagli sono elencati nella sezione **Esempi** sopra e più starter kit sono elencati su questo wiki .

Un "micro-kit" notevolissimo è Sane Defaults , che fornisce una serie di impostazioni per rimuovere alcuni dei comportamenti di sorpresa per i nuovi arrivati di Emacs.

## È necessario un kit di base?

Sebbene vi siano alcune controversie sull'utilizzo dei kit di avvio, per molti i vantaggi possono superare di gran lunga i costi per capire come armonizzare una configurazione dinamica di Emacs. Gli argomenti contro i kit di avvio di solito riguardano: gli utenti non sono consapevoli di alcune delle sfumature e del comportamento nativo di Emacs, essendo difficili da eseguire il debug e persino rendendo Emacs più simile a un editor esterno (Spacemacs).

# Examples

## Spacemacs

Spacemacs è un popolare kit di avvio per emacs. È dotato di una solida soluzione di gestione dei pacchetti e si concentra sulla popolare modalità malvagia di emacs, che fornisce molte delle combinazioni di tasti da vim .

Si chiama Spacemacs perché usa il tasto `Space` come chiave del leader (l'idea è simile alla chiave del leader di Vim).

L'installazione è abbastanza facile. Basta scaricare e installare la distribuzione emacs standard e quindi clonare il repository git:

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

Oppure, puoi scaricare un zip localmente dal sito web e copiarlo in `~/.emacs.d`

Una volta scaricato (clonato), avvialo, quindi premi la barra spaziatrice per esplorare l'elenco interattivo dei binding di tasti scelti con cura. Puoi anche premere il pulsante [?] Del buffer di casa per provare alcuni primi collegamenti dei tasti.

## Preludio

Il preludio è un altro famoso antipasto. Offre un buon supporto per vari linguaggi di programmazione, incluso, in particolare, il clojure. Sui sistemi * nix può essere installato con il seguente comando:

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

## emacs-live

emacs-live è un altro famoso kit di avvio di emacs, con un'attenzione particolare alla codifica di musica dal vivo usando l' overtone .

Puoi installarlo in 2 modi:

1. Sui sistemi * nix (ad esempio linux, OSX, ecc.), Eseguire il seguente comando sulla riga di comando:

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

2. - Scarica il file zip dalla pagina github.
   - Esegui il backup del tuo attuale `~/.emacs.d` nella tua home directory
   - estrai la zip che hai scaricato e `~/.emacs.d` in `~/.emacs.d` :

### Scimax

Scimax è uno starter kit Emacs incentrato sulla ricerca riproducibile, rivolto principalmente a scienziati e ingegneri. Scimax personalizza la modalità Org con funzionalità che semplificano il riferimento incrociato, l'esportazione e la codifica (in particolare Python).

Le istruzioni di installazione sono disponibili sulla pagina di destinazione del progetto .

Leggi Starter Kit online: https://riptutorial.com/it/emacs/topic/1960/starter-kit

# Capitolo 12: Timone

## Examples

### Installazione del timone tramite MELPA

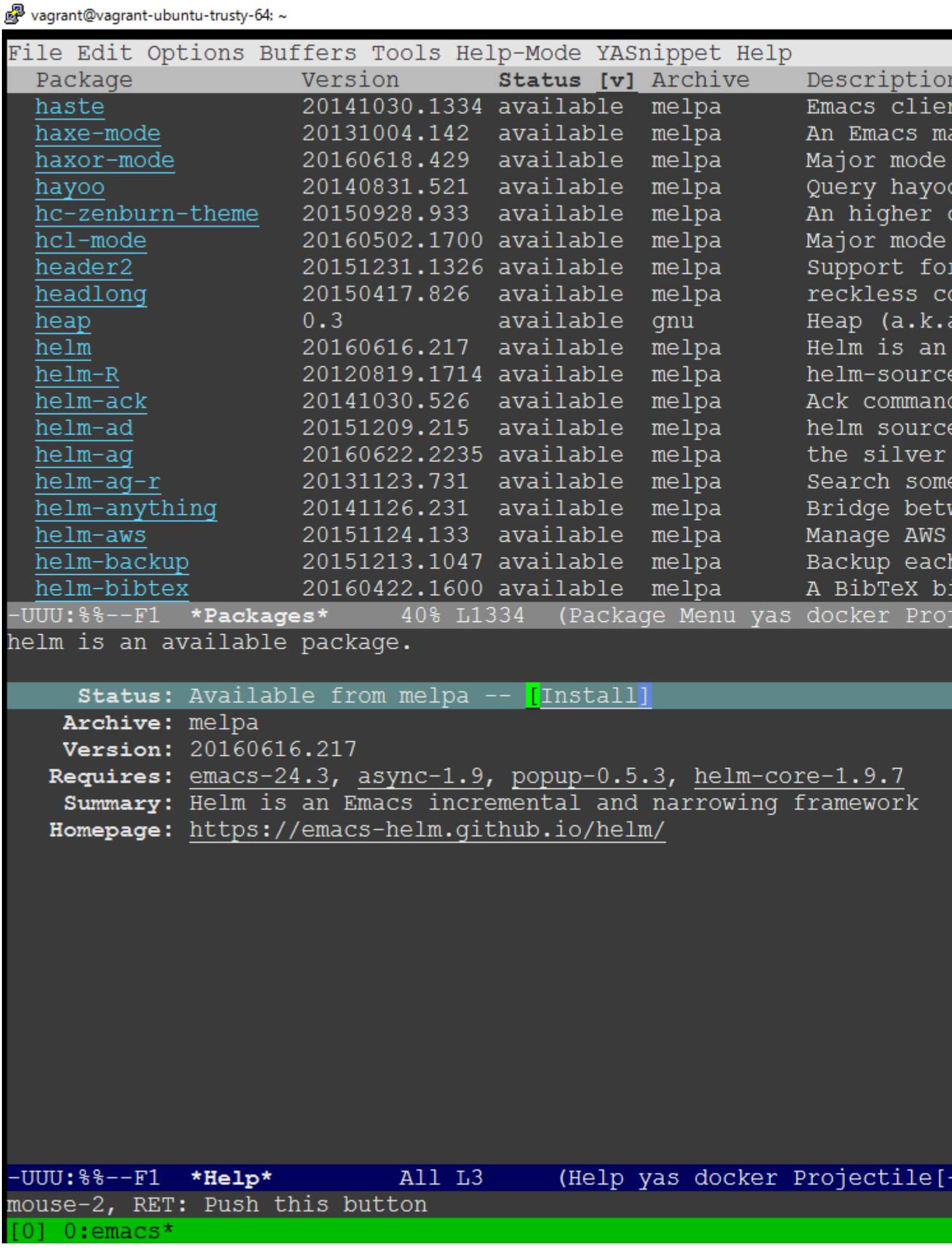
Da emacs 24.4 package.el è disponibile, e un modo per installare helm è farlo tramite MELPA. Innanzitutto, aggiungi il repository MELPA come archivio di pacchetti inserendo il seguente codice da qualche parte nel tuo `~/.emacs` (o, `~/.emacs.d/init.el` ).

```
(require 'package)

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

(package-initialize)
```

Quindi, immettere `Mx list-packages` per visualizzare l'elenco di pacchetti disponibili. Cerca l'ingresso `helm` , metti il cursore sulla voce `helm` , premi `RET` . Vedrai il buffer delle informazioni sul pacchetto. Metti il cursore su `[Install]` e premi `RET` . Helm sarà installato. La finestra dell'elenco dei pacchetti e la finestra delle informazioni sul pacchetto sono mostrate nell'immagine seguente.


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*

Leggi Timone online: https://riptutorial.com/it/emacs/topic/5341/timone

# Titoli di coda

| S. No | Capitoli | Contributors |
|---|---|---|
| 1 | Iniziare con emacs | Adobe, Community, ebpa, Ehvince, eyqs, IntFloat, Jack Henahan, joon, legoscia, mellowmaroon, Michel de Ruiter, Nemanja Trifunovic, omul, pcurry, Prasanna, salotz, squiter |
| 2 | Aiuto all'interno di Emacs | mellowmaroon, Terje D., ygram |
| 3 | emacs ha già una documentazione di alta qualità e ben organizzata. perché duplicarlo? | erjoalgo |
| 4 | Gestione dei pacchetti | Adobe, Kaushal Modi, leeor, pcurry, salotz, squiter |
| 5 | Gestisci i segnalibri all'interno di Emacs | Francesco, Prasanna |
| 6 | Keybindings di base | 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 | Le molte varianti di Emacs | pcurry |
| 8 | magit | boehm_s, Ehvince, glallen, mellowmaroon, squiter |
| 9 | Nomenclatura Emacs | Doug Harris, Francesco, Nikana Reklawyks, Stephen Leppik, Terje D. |
| 10 | Org-mode | Christopher Bottoms, Francesco, legoscia, SuperBear, Wazam |
| 11 | Starter Kit | dangom, Ehvince, Kaushal Modi, leeor, Micah Elliott, Xinyang Li |
| 12 | Timone | Yuki Inoue |