eBook Gratuit

APPRENEZ emacs

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#emacs

Table des matières

À propos

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

Chapitre 1: Démarrer avec emacs

Remarques

Emacs est un éditeur de texte dont la caractéristique principale est la capacité des utilisateurs à personnaliser par programmation presque tous les aspects de celui-ci. Ceci est facilité par un dialecte spécial du langage de programmation Lisp, appelé Emacs Lisp, créé spécifiquement pour être utilisé dans l'éditeur Emacs.

Il existe une multitude d'extensions écrites en Emacs Lisp qui ajoutent aux fonctionnalités d'Emacs. Ces extensions incluent des fonctions d'édition pour des langages de programmation spécifiques (similaires à ceux fournis par un IDE), des clients e-mail et IRC, des interfaces Git, des jeux tels que Tetris et 2048, etc.

De nombreux aspects de l'éditeur Emacs peuvent être utilisés sans connaissance de programmation. Les utilisateurs cherchant à personnaliser Emacs par programmation trouveront toutefois certaines fonctionnalités du langage Lisp d'Emacs telles que le système de documentation (autonome) incroyablement utile et accommodant.

Références externes:

1. Le site de Sacha Chua est un très bon endroit pour trouver plus de ressources d'apprentissage sur Emacs.

   une. Pour ceux qui ont besoin d'un appel plus visuel sur le parcours d'apprentissage d'Emacs

   b. Pour ceux qui voudraient obtenir facilement les raccourcis clavier

2. Wikemacs est basé sur mediawiki, et a donc un contenu structuré, des catégories navigables et autres. Commencez à explorer !

Versions

Version	Date de sortie
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

Version	Date de sortie
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

Installation ou configuration

Instructions détaillées sur la configuration ou l'installation d'emacs.

Les instructions officielles sont disponibles sur le site Web de GNU Emacs .

Systèmes Debian

Sur les systèmes avec le gestionnaire de paquets Debian (tels que Debian, Ubuntu et Mint), Emacs peut être installé via la simple commande:

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

Pour une version à la pointe de la technologie, on peut utiliser le ppa suivant:

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

Construire pour la source

Si votre distribution basée sur Debian n’a pas la version d’emac que vous voulez, vous pouvez la

construire à partir de zéro.

```
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*
```

Systèmes Redhat

Sur les systèmes avec le gestionnaire de paquets Redhat (tels que RHEL, CentOS et Fedora Core), Emacs peut être installé via la simple commande:

```
sudo yum install emacs
```

Arch Linux

Emacs peut être installé via la simple commande:

```
sudo pacman -Syu emacs
```

Gentoo et Funtoo

Sur les systèmes exécutant Portage, Emacs peut être installé via la simple commande:

```
sudo emerge emacs
```

GSRC (Collection de versions GNU)

Fonctionne sur tout système GNU / Linux pour obtenir la dernière version d'emacs sans utiliser le gestionnaire de paquets système (qui peut être obsolète) ou en téléchargeant l'archive ou le fichier binaire. Pour installer `gsrc` consultez sa documentation . Alors:

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

Systèmes 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
```

App Bundle

Des bundles d'applications précompilés pour les dernières versions stables et de développement peuvent être téléchargés sur https://emacsformacosx.com .

les fenêtres

Gestionnaire de paquets chocolatés

Emacs peut être installé avec

```
choco install emacs
```

Gestionnaire de package Scoop

Peut être installé à partir de seaux supplémentaires

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

Installateurs binaires officiels

- stable
- dernier

(Notez que les fichiers binaires officiels ne sont pas fournis avec certaines bibliothèques - par exemple, les bibliothèques pour les formats d'image)

Autres installateurs binaires

- Emacs avec AUCTeX et ESS pré-compilés
- GNU Emacs 64 bits pour MS Windows avec optimisation fournit un programme d'installation binaire 64 bits natif et optimisé avec le code source non modifié de git master et la version release, avec le support JPEG, GIF, PNG, TIFF, SVG, XML2 et GnuTLS boîte.

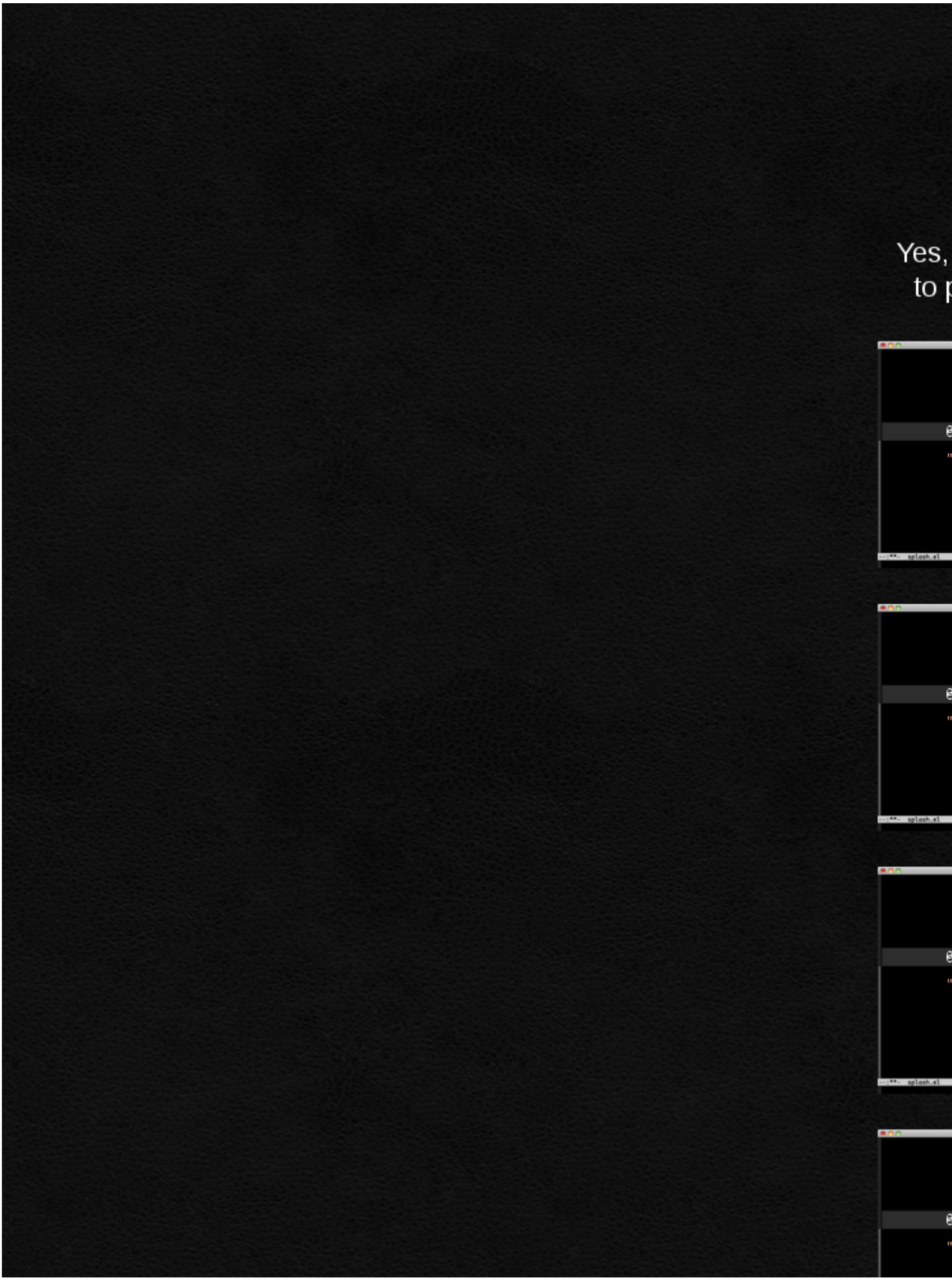
Tutoriel interactif Emacs

Dans Emacs, tapez `Ch t` (Control-h, t) pour obtenir un excellent didacticiel interactif dans Emacs. L'utilisateur apprend la navigation et l'édition de base en opérant sur le texte TUTORIAL lui-même, pendant qu'il lit le didacticiel. (Les modifications apportées au didacticiel sont supprimées lorsque le didacticiel est fermé. Par conséquent, chaque fois qu'un utilisateur demande le didacticiel, il s'agit d'une version par défaut du didacticiel.

La première chose à faire dans le tutoriel est de savoir comment comprendre les références `C-<chr>` et `M-<chr>` dans le texte. La deuxième chose est de savoir comment avancer et reculer dans le texte.

Tutoriels vidéo Emacs Rocks

Vous trouverez de bons didacticiels vidéo sur Emacs sur emacsrocks.com .


Yes,
to p

https://riptutorial.com/fr/emacs/topic/986/demarrer-avec-emacs

Chapitre 2: Aide dans Emacs

Remarques

Emacs est décrit comme un éditeur d'auto-documentation et fournit de nombreuses informations sur son utilisation dans l'éditeur même. Parmi les points d'entrée à cette documentation est un tutoriel, des informations sur les fonctions est disponible lié à un sujet donné, une information sur les liaisons entre les frappes au clavier et la documentation functions.The est accessible en utilisant le préfixe `Ch`, à *savoir* `Ctrl h`, ou `F1`, avec une liste d'autres choix disponibles en appuyant sur `?`

Examples

Tutoriel Emacs

`Ch t` exécute la fonction `help-with-tutorial` , qui ouvre un tampon contenant un didacticiel sur les fonctionnalités d'édition de base d'emacs, y compris le déplacement dans le texte et l'utilisation de fichiers, de tampons et de fenêtres.

Fonctions disponibles et liaisons de touches

En appuyant sur `Ch a` , vous exécuterez la fonction emacs `apropos-command` qui demandera à emacs de rechercher les mots (ou une expression rationnelle). Il affichera ensuite un tampon contenant une liste de noms et de descriptions liés à ce sujet, y compris des raccourcis clavier pour chacune des fonctions disponibles via les frappes.

En appuyant sur `Ch m` (`describe-mode`), vous obtenez un tampon décrivant les modes majeur et mineur, y compris les listes des fonctions disponibles et leurs raccourcis.

En appuyant sur `Ch b` (`describe-bindings`), vous obtenez un tampon répertoriant toutes les liaisons de touches actuelles. La liste comprend des liaisons globales ainsi que des liaisons pour les modes majeur et mineur actifs dans le tampon actuel.

Documentation de reliure à clé

`Ch k` exécute la fonction `describe-key` , qui recherche la fonction associée aux touches fournies et présente une description de la fonction qui sera exécutée lorsque ces touches sont enfoncées.

`Ch c` exécute `describe-key-briefly` la fonction `describe-key-briefly` , qui affiche uniquement le nom de la fonction mappée sur une séquence de touches donnée.

Documentation de fonction

`Ch f` lance la fonction `describe-function` , qui affiche des informations sur l'utilisation et le but d'une fonction donnée. Ceci est particulièrement utile pour les fonctions qui ne disposent pas d'une

liaison de clé mappée pouvant être utilisée pour la recherche de documentation via `Ch k` .

Lire Aide dans Emacs en ligne: https://riptutorial.com/fr/emacs/topic/4736/aide-dans-emacs

Chapitre 3: Barre

Examples

Installation de la barre via MELPA

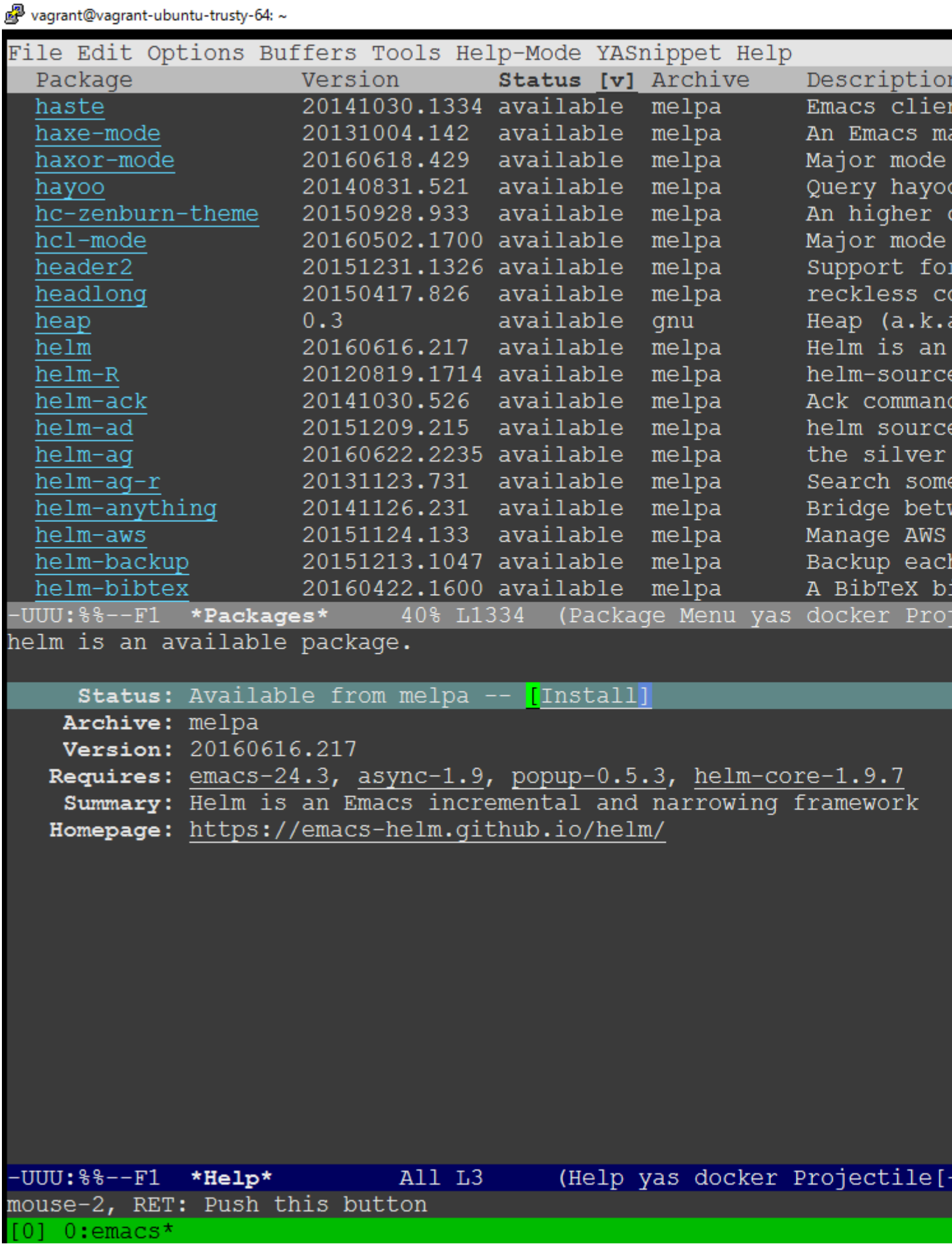
De emacs 24.4 package.el est disponible, et une façon d'installer la barre est de le faire via MELPA. Tout d'abord, ajoutez le référentiel MELPA en tant qu'archive de package en mettant le code suivant quelque part dans votre `~/.emacs` (ou `~/.emacs.d/init.el`).

```
(require 'package)

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

(package-initialize)
```

Ensuite, entrez `Mx list-packages` pour voir la `liste des packages` disponibles. Recherchez l'entrée de `helm` , placez votre curseur sur l'entrée de `helm` , appuyez sur `RET` . Vous verrez le tampon d'informations sur le paquet. Placez votre curseur sur `[Install]` et appuyez sur `RET` . Helm sera installé. La fenêtre de liste de packages et la fenêtre d'informations sur le package s'affichent dans l'image suivante.


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*

Lire Barre en ligne: https://riptutorial.com/fr/emacs/topic/5341/barre

Chapitre 4: emacs possède déjà une documentation de très haute qualité et bien organisée. pourquoi le dupliquer?

Introduction

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

Examples

Clés

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

3 clés

Certaines commandes Emacs sont appelées par un seul événement d'entrée; Par exemple, Cf avance d'un caractère dans le tampon. D'autres commandes prennent en charge deux ou plusieurs événements d'entrée, tels que Cx Cf et Cx 4 Cf.

Une séquence de touches, ou clé pour faire court, est une séquence d'un ou plusieurs événements d'entrée qui sont significatifs en tant qu'unité. Si une séquence de touches appelle une commande, nous l'appelons une clé complète; Par exemple, Cf, Cx Cf et Cx 4 Cf sont des clés complètes. Si une séquence de touches n'est pas assez longue pour invoquer une commande, nous l'appelons une clé de préfixe; à partir de l'exemple précédent, nous voyons que Cx et Cx 4 sont des clés de préfixe. Chaque séquence de touches est soit une clé complète, soit une clé de préfixe.

Lire emacs possède déjà une documentation de très haute qualité et bien organisée. pourquoi le dupliquer? en ligne: https://riptutorial.com/fr/emacs/topic/9077/emacs-possede-deja-une-documentation-de-tres-haute-qualite-et-bien-organisee--pourquoi-le-dupliquer-

Chapitre 5: Gérer les favoris dans Emacs

Examples

Comment marquer les fichiers fréquemment utilisés

Utilisez les commandes suivantes pour créer des signets et accéder aux signets depuis Emacs.

Disons que vous éditez un fichier appelé `foobar.org` et supposez que vous `foobar.org` fréquemment ce fichier pour éditer / visualiser le contenu.

Il serait pratique d'accéder à ce fichier avec quelques touches au lieu de naviguer dans la structure de fichier (Dired) et de visiter le fichier.

Pas:

1. Ouvrez `foobar.org` pour une fois en naviguant vers le fichier (*visitez le fichier* dans le jargon d'Emacs)
2. Lorsque le fichier est ouvert, tapez `Cx r m`, ce qui vous invitera à fournir le nom du signet du fichier. Disons `foobar` dans ce cas.
3. Fermez le fichier (`Cx k` - kill buffer) - sauvegardez si nécessaire
4. Maintenant, pour visiter le fichier, tapez simplement `Cx r l` - cela remplira une liste qui contiendra `foobar` .
5. Sélectionnez `foobar` et `appuyez sur la` touche `Entrée`
6. Utilisez `Mx bookmark-delete` pour supprimer tout signet inutile d'un fichier.

Remarque: la suppression d'un signet est analogue à la suppression d'un raccourci sur votre bureau Windows.
Le fichier principal sera en sécurité à son emplacement et seule la liste du fichier du menu des signets sera supprimée.

Lire Gérer les favoris dans Emacs en ligne: https://riptutorial.com/fr/emacs/topic/5837/gerer-les-favoris-dans-emacs

Chapitre 6: Gestion des packages

Examples

Installation automatique de paquets sur la start-up 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 '()))
```

Les références

- Comparaison des pensions alimentaires
- Article de blog sur la fonctionnalité de paquets sélectionnés par l'utilisateur dans emacs 25.1

Installation automatique du paquet avec utilisation-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)
```

Gestion automatique des paquets à l'aide de Cask

Cask est un outil de gestion de projet qui peut également être utilisé pour gérer facilement votre configuration emacs locale.

L'installation du fût est facile. Vous pouvez soit exécuter la commande suivante sur la ligne de commande:

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

Ou si vous êtes sur un Mac, vous pouvez l'installer en utilisant `homebrew` :

```
brew install cask
```

Une fois installé, vous créez un fichier `Cask` . Les fichiers cask répertorient toutes les dépendances de paquets à inclure dans votre configuration. Vous pouvez créer un nouveau fichier Cask à la racine de votre répertoire `~/.emacs` .

Vous devrez également initialiser Cask dans votre `~/.emacs.d/init.el` Si vous avez installé homebrew, ajoutez ces lignes:

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

Ou vous pouvez fournir le chemin d'accès à la casse, si vous avez utilisé le script d'installation:

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

Un simple fichier Cask ressemble à ceci:

```
(source gnu)
(source melpa)

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

Nous `flx` ici les dépôts sources à rechercher dans les paquets. Ensuite, nous `flx` que nous voulons les `flx` `projectile` , `flx` et `flx-ido` installés.

Une fois que vous avez un fichier Cask, vous pouvez installer toutes les dépendances avec la commande suivante sur la ligne de commande:

```
cask install
```

Gestion automatique des paquets avec el-get

el-get est un système de gestion de paquets open source pour GNU Emacs. el-get fonctionne avec `melpa` , ainsi qu'avec de nombreux systèmes de contrôle de version courants. Sa documentation comprend un auto-installateur simple pour vos `.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 maintient les installations de paquets dans une structure de répertoires sur `~/.emacs.d/el-get` . Il charge les définitions de `~/.emacs.d/el-get/.loaddefs.el` et suit l'état du paquet avec `~/.emacs.d/el-get/.status.el` . `(el-get 'sync)` installe ou supprime des paquets pour synchroniser l'état réel de la machine avec le package `.status.el` .

el-get est auto-hébergé - voici son propre statut 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))))
```

Lire Gestion des packages en ligne: https://riptutorial.com/fr/emacs/topic/2414/gestion-des-packages

Chapitre 7: Kits de démarrage

Remarques

Les kits de démarrage permettent aux *nouveaux utilisateurs* d'utiliser rapidement Emacs et d'éviter certains des obstacles posés par un système mature comme Emacs. *Les utilisateurs expérimentés* bénéficient également d'une configuration de kit des extensions organisées par d'autres.

Il faut déployer des efforts considérables pour maintenir un ensemble de packages et de paramètres qui continueront de fonctionner ensemble lorsque les packages s'améliorent (ou baissent) au fil du temps. De nombreux utilisateurs d'Emacs ne souhaitent pas faire cette maintenance, ils se tournent donc vers des kits de démarrage. L'assemblage et la maintenance d'un kit ont une ressemblance à petite échelle avec la gestion d'une distribution Linux.

Thèmes et personnalisation

Certains kits de démarrage sont à thème; Par exemple, pour des environnements de langage de programmation spécifiques, la création de musique ou l'émulation d'un autre éditeur. D'autres visent à fournir un évier de cuisine regroupant des modules confortables / productifs pour autant de situations ou de langues que possible.

La plupart des kits de démarrage comportent des dispositions relatives à l'extension et à la personnalisation. Un utilisateur remplacera des paramètres et des liaisons de clé spécifiques et pourra ajouter des packages qui ne sont pas encore fournis.

Kits populaires

Il existe de nombreux kits de démarrage disponibles. En théorie, quiconque publie son `~/.emacs.d` a créé un. Mais une poignée d'entre eux sont devenus populaires et bien entretenus par un ou plusieurs individus. Quelques exemples (par ordre de popularité subjective en fonction des étoiles Github) incluent Spacemacs , Prelude , Purcell , Emacs Starter Kit , Magnars et Emacs Live . Plus de détails sont listés dans la section **Exemples** ci-dessus et d'autres kits de démarrage sont listés sur ce wiki .

Sane Defaults est un "micro-kit" remarquable, qui fournit une poignée de paramètres permettant de supprimer certains comportements d'Emacs par défaut.

Un kit de démarrage est-il nécessaire?

Bien que l'utilisation de kits de démarrage suscite une certaine controverse , les avantages peuvent largement dépasser le coût de l'harmonisation d'une configuration dynamique d'Emacs. Les arguments contre les kits de démarrage concernent généralement: les utilisateurs ignorant certaines des nuances et le comportement natif d'Emacs, étant difficiles à déboguer, et même à

faire ressembler Emacs à un éditeur étranger (Spacemacs).

Examples

Spacemacs

Spacemacs est un kit de démarrage populaire pour emacs. Il propose une solution robuste de gestion des paquets et se concentre sur le mode maléfique populaire d'emacs, qui fournit de nombreux liens à partir de vim .

Il est appelé Spacemacs car il utilise la touche `Espace` comme clé principale (l'idée est similaire à la clé de leader de Vim).

L'installation est assez facile. Il suffit de télécharger et d'installer la distribution emacs standard, puis de cloner le dépôt git:

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

Ou, vous pouvez télécharger un `~/.emacs.d` zip localement à partir du site Web et le copier dans `~/.emacs.d`

Une fois que vous l'avez téléchargé (cloné), lancez-le, puis appuyez sur la barre d'espace pour explorer la liste interactive des raccourcis clavier soigneusement choisis. Vous pouvez également appuyer sur le bouton [?] Du tampon d'accueil pour essayer les premières associations de touches.

Prélude

Prelude est un autre kit de démarrage populaire. Il prend en charge de nombreux langages de programmation prêts à l'emploi, notamment: Sur les systèmes * nix, il peut être installé avec la commande suivante:

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

emacs-live

emacs-live est un autre kit de démarrage populaire emacs, avec un accent supplémentaire sur le codage de musique en direct utilisant des harmoniques .

Vous pouvez l'installer de deux manières:

1. Sur les systèmes * nix (par exemple, linux, OSX, etc.), exécutez la commande suivante sur la ligne de commande:

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

2. - Téléchargez le fichier zip depuis la page github.

- Sauvegardez votre `~/.emacs.d` dans votre répertoire personnel
- extraire le zip que vous avez téléchargé et le déplacer vers `~/.emacs.d` :

Scimax

Scimax est un kit de démarrage Emacs axé sur la recherche reproductible, destiné principalement aux scientifiques et aux ingénieurs. Scimax personnalise Org-Mode avec des fonctionnalités qui facilitent le référencement, l'exportation et le codage (en particulier Python).

Les instructions d'installation peuvent être trouvées sur la page de destination du projet .

Lire Kits de démarrage en ligne: https://riptutorial.com/fr/emacs/topic/1960/kits-de-demarrage

Chapitre 8: Les nombreuses variantes d'Emacs

Introduction

La plupart de cette documentation s'applique implicitement ou explicitement à GNU Emacs. Cela peut être la variante la plus connue d'Emacs, ainsi que la source de plusieurs forks, et la cible de certaines fusions.

Cette rubrique traite de certaines des variantes d'Emacs que l'on peut rencontrer et de leurs principales différences par rapport à GNU Emacs.

Examples

Spacemacs

Spacemacs (http://spacemacs.org/) est une variante d'Emacs qui tente de mettre fin au conflit à long terme entre les utilisateurs d'Emacs et de vim, en créant un Emacs qui se comporte comme vim.

Lire Les nombreuses variantes d'Emacs en ligne: https://riptutorial.com/fr/emacs/topic/9456/les-nombreuses-variantes-d-emacs

Chapitre 9: Liens de base

Examples

Quittez Emacs

Vous pouvez quitter Emacs avec le raccourci clavier suivant:

```
Cx Cc
```

Où `C` est la clé de `control` .

Suspendre Emacs

Vous pouvez suspendre Emacs en utilisant le raccourci clavier suivant:

```
Cz
```

Cela vous ramène à votre coquille. Si vous souhaitez reprendre votre session emacs, entrez `fg` dans votre terminal.

La gestion des fichiers

- Réenregistrez le fichier ouvert sous le même nom de fichier (Enregistrer):

  ```
  Cx Cs
  ```

- Ecrivez comme `filename` de `filename` (Enregistrer sous):

  ```
  Nom de filename Cx Cw
  ```

  Le nouveau nom de fichier sera demandé dans le mini-tampon.

- Créer un nouveau fichier **ou** charger un fichier existant (Nouveau / Charger):

  ```
  Cx Cf filename
  ```

  Avec le mnémonique ici pour f signifiant fichier. Vous serez invité à entrer un chemin de fichier dans le mini-tampon.

- Visitez un autre fichier

  ```
  Cx Cf
  ```

  Si le fichier n'existe pas encore, vous serez invité à créer le chemin du fichier à créer dans le mini-tampon.

Abandonner la commande en cours

Souvent, vous allez entrer dans un état où vous avez une séquence de commandes partiellement

typée en cours, mais vous voulez l'abandonner. Vous pouvez l’abandonner avec l’un des raccourcis suivants:

`Cg`

`Esc Esc Esc Esc`

Multiples fenêtres ou cadres

"Fenêtre" dans Emacs fait référence à ce que l'on pourrait sinon appeler un "volet" ou une "division d'écran". Certaines commandes de manipulation de fenêtres incluent:

- Séparer la fenêtre actuelle horizontalement: `Cx 2`
- Séparer la fenêtre actuelle verticalement: `Cx 3`
- Sélectionner la prochaine fenêtre: `Cx o`
- Fermer la fenêtre actuelle: `Cx 0`
- Fermez toutes les autres fenêtres, à l'exception de la version actuelle: `Cx 1`

Un "cadre" dans Emacs est ce que l'on pourrait sinon appeler une "fenêtre". Les cadres sont manipulés à l'aide de ces commandes:

- Créer un nouveau cadre: `Cx 5 2`
- Supprimer l'image actuelle: `Cx 5 0`
- Supprimer d'autres cadres: `Cx 5 1`

Il est possible de changer de fenêtre en utilisant

- `S-left` , `S-right` , `S-up` , `S-down` (`c.-` à-d. `Shift` en combinaison avec une touche fléchée) pour passer à la fenêtre voisine dans une direction, ou
- `Cx o` pour passer à la fenêtre suivante.

Tampons

- Exemple de liste de tampons

```
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
```

Le premier champ d'une ligne indique:

  - '.' le tampon est en cours.
  - '%' un tampon en lecture seule.

    - '*' le tampon est modifié.

- Sélectionnez un tampon. Vous pouvez sélectionner n'importe quel tampon ouvert avec le raccourci clavier suivant:

  `Cx b`

  Vous serez invité à indiquer le nom du tampon vers lequel vous souhaitez basculer.

- Liste des tampons:

  `Cx Cb`

- Save-some-buffer, donnant le choix du tampon à enregistrer ou non:

  `Cx s`

- Tuez un tampon:

  `Cx k`

- Opérations sur les tampons marqués:

  `S` Sauvegarder les tampons marqués

  `A` Affichez les tampons marqués dans ce cadre.

  `H` Affichez les tampons marqués dans un autre cadre.

  `V` Annuler les tampons marqués.

  `T` Basculez l’état en lecture seule des tampons marqués.

  `D` Tuez les tampons marqués.

  `Ms a Cs` Effectuez une recherche incrémentielle dans les tampons marqués.

  `Ms a CMs` Isearch pour l'expression régulière dans les tampons marqués.

  `U` Remplacez par regexp dans chacun des tampons marqués.

  `Q` Query remplace dans chacun des tampons marqués.

  `I` Comme ci-dessus, avec une expression régulière.

  `P` Imprimez les tampons marqués.

  `O` Liste des lignes dans tous les tampons marqués qui correspondent à une expression rationnelle donnée (comme la fonction `comporter ').

  `X` Placez le contenu des tampons marqués sur une commande shell.

  `N` Remplacez le contenu des tampons marqués par la sortie d'une commande shell.

`!` Exécutez une commande shell avec le fichier du tampon comme argument.

`E` Evaluez un formulaire dans chacun des tampons marqués. C'est une commande très flexible. Par exemple, si vous souhaitez que tous les tampons marqués soient en lecture seule, essayez d'utiliser (mode lecture seule 1) comme formulaire de saisie.

`W` - Comme ci-dessus, mais visualisez chaque tampon pendant que le formulaire est évalué.

`k` - Supprimez les lignes marquées du tampon *Ibuffer* , mais ne tuez pas le tampon associé.

`x` - Tue tous les tampons marqués pour suppression.

- Save-some-buffer, donnant le choix du tampon à enregistrer ou non:

  `Cx s`

- Passez au tampon suivant:

  `Cx à droite`

- Passer au tampon précédent:

  Cx À GAUCHE

Rechercher et remplacer

Dans Emacs, l'outil de recherche de base (`I-Search`) vous permet de rechercher après ou avant l'emplacement de votre curseur.

- Pour rechercher *quelque chose* après l'emplacement de votre curseur (`search-forward`), appuyez sur `Cs` *`sometext`* . Si vous voulez passer à la prochaine occurrence de *`sometext`* , appuyez à nouveau sur `Cs` (et ainsi de suite pour les prochaines occurrences). Lorsque le curseur se place au bon endroit, appuyez sur `Entrée` pour quitter l'invite de recherche.

- Pour effectuer une recherche avant l'emplacement de votre curseur (`search-backward`), utilisez `Cr de` la même manière que vous utilisiez auparavant.

- Pour passer de la `search-backward` à la `search-forward` , appuyez sur 2 fois `Cs` . Et appuyez 2 fois `Cr` pour une `search backward` en `search-forward` `search backward` quand vous êtes dans la `search-forward` rapide.

- Rechercher et remplacer:

  `M-%` (ou `Esc-%`) `oldtext Entrer newtext Entrer`

  - Confirmer: `y`
  - Skip: `n`
  - Quitter: `q`
  - Remplacez tout `:!`

Région - Couper, Copier, Coller

- Définir la marque à l'emplacement du curseur:

  `C-espace` ou `C- @`

- Kill region (Cut):

  `Cw`

- Copier la région pour tuer l'anneau:

  `Mw` ou `Esc-w`

- Yank (Paste) le plus récemment tué:

  `Cy`

- Yank (Paste) la dernière fois tué:

  `Mon` ou `esc-y`

Tuer

`kill` est la commande utilisée par Emacs pour la suppression de texte. La commande `kill` est analogue à la commande `cut` de Windows. Diverses commandes existent pour «tuer» un mot (`Md`), le reste de la ligne (`Ck`) ou des blocs de texte plus grands. Le texte supprimé est ajouté au `kill-ring` , à partir duquel il peut être `yanked` plus tard.

Sélectionner et couper (tuer)

Killing and yanking Semblable à la fonctionnalité `select-and-cut` de Windows, nous avons ici `C-spc` . Le raccourci clavier `C-spc` lance la sélection, l'utilisateur peut déplacer la marque à l'aide des touches fléchées ou d'une autre commande pour effectuer une sélection. Une fois la sélection terminée - appuyez sur `Cw` pour supprimer le texte sélectionné

Quelques commandes de base pouvant servir de référence rapide pour la commande `kill` (tirée du tutoriel d'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
```

Coup sec

`yank` décrit l'insertion d'un texte précédemment supprimé, *par exemple en* utilisant `Cy`, qui récupère le dernier texte tué. `Yank` commande `Yank` est analogue à la commande `paste` de Windows.

Yank texte tué précédemment

Nous savons que la commande `kill` ajoute le texte tué à un `kill-ring` . Pour récupérer le texte supprimé du `kill-ring` utilisez `ma` commande à plusieurs reprises jusqu'à ce que le texte souhaité soit tiré.

(Remarque: pour que la touche `My` fonctionne, la commande précédente devrait être un `YANK` sinon cela ne fonctionnerait pas)

Mouvement du curseur

En plus des mouvements du curseur à l'aide des touches fléchées, Accueil, Fin, Page précédente et Page suivante, emacs définit un nombre de touches pouvant déplacer le curseur sur des parties de texte plus petites ou plus grandes:

Par personnage:

- Caractère arrière: `Cb`
- Caractère avant: `Cf`

Par mot

- Mot en arrière: `Mb` (*c.-à-d.* `Alt b` ou `Meta b`)
- Mot en avant: `Mf`

Par ligne:

- Début de la ligne actuelle: `Ca`
- Début de la ligne courante premier caractère (non-espace): `Mm`
- Fin de la ligne en cours: `Ce`
- Ligne précédente: `Cp`
- Ligne suivante: `Cn`

Tampon entier:

- Début du tampon: `M- <`
- Fin du tampon: `M->`

Par 'block', selon le contexte (mode):

Fixations de touches typiques:

- Phrase / déclaration en arrière: `Ma`
- Phrase / déclaration en avant: `moi`
- Début de fonction: `MCa`
- Fin de fonction: `MCe`

Arguments de préfixe

Afin de déplacer plusieurs «étapes» à la fois, les commandes de mouvement peuvent recevoir un argument de préfixe en appuyant sur `ESC` ou `Cu` et un nombre avant les séquences de touches répertoriées. Pour `Cu` , le numéro est facultatif et la valeur par défaut est 4.
Par exemple, `ESC 3 Cn` déplace 3 lignes vers le bas, tandis que `Cu Mf` déplace 4 mots vers l'avant.

annuler

Annuler quelque chose que vous venez de faire:

`C-_` ou `Cx u` ou `C- /`

Cas

- Capitaliser le mot: `Mc`
- Convertissez le mot en majuscule: `Mu`
- Convertissez le mot en minuscule: `Ml`

Notation des liaisons clés

La documentation d'Emacs utilise une notation cohérente pour toutes les liaisons clés, ce qui est expliqué ici:

Accords clés

Un "accord clé" est obtenu en appuyant simultanément sur deux touches ou plus. Les accords clés sont notés en séparant toutes les clés par des tirets (`-`). Ils impliquent généralement des touches de modification, qui sont mises en avant:

- `C-` : contrôle;
- `S-` : décalage;
- `M-` : alt (le "M" signifie "Meta" pour des raisons historiques).

D'autres clés sont simplement désignées par leur nom, comme:

- `a: a` clé;
- `à gauche` : la touche fléchée gauche;
- `SPC` : la touche espace;
- `RET` : la clé de retour.

Voici des exemples d'accords clés:

- `Ca:` en appuyant sur `une commande` et en même temps;
- `S-droite` : appuyer simultanément sur la `touche shift` et `droite` ;
- `CMa` : appuyer sur `control` , `alt` et `a` simultanément.

Séquences clés

Les "séquences de touches" sont des séquences de clés (ou accords de touches), qui doivent être saisies les unes après les autres. Ils sont notés en séparant toutes les notations de clé (ou d'accord) par un espace.

Les exemples comprennent:

- `Cx b` : appuyer simultanément sur `control` et `x` , puis les relâcher et appuyer sur `b` ;
- `Cx Cf` : appuyer simultanément sur `control` et `x` , puis relâcher `x` et appuyer sur `f` (puisque les deux accords impliquent le modificateur de `contrôle` , il n'est pas nécessaire de le relâcher).

Utiliser ESC au lieu de Alt

Les accords de touches utilisant le modificateur Alt peuvent également être saisis en tant que séquence de touches commençant par `ESC` . Cela peut être utile lorsque vous utilisez Emacs sur une connexion distante qui ne transmet pas les accords de touches Alt, ou lorsque ces combinaisons de touches sont capturées, par *exemple,* par un gestionnaire de fenêtres.

Exemple:

`Mx` peut être entré comme `ESC x` .

Décrire les liaisons de clé dans les fichiers lasp d'Emacs

La même notation décrite ici peut être utilisée lors de la définition des liaisons de clé dans les fichiers lisp Emacs.

Exemple:

(global-set-key (kbd "Cx Cb") 'buffer-menu)
lie la séquence de touches `Cx Cb` à la commande `buffer-menu`

Lire Liens de base en ligne: https://riptutorial.com/fr/emacs/topic/3436/liens-de-base

Chapitre 10: Magit

Introduction

Magit est une interface vers le système de contrôle de version Git, implémenté en tant que package Emacs. Il vous permet d'interagir avec git dans Emacs.

Remarques

Magit est une interface vers le système de contrôle de version Git, implémenté en tant que package Emacs. Magit aspire à être une porcelaine complète de Git. Bien que nous ne puissions pas (encore) prétendre que Magit enveloppe et améliore chaque commande Git, il est suffisamment complet pour permettre aux utilisateurs expérimentés de Git d'effectuer presque toutes leurs tâches quotidiennes de contrôle de version directement depuis Emacs. Bien qu'il existe de nombreux clients Git, seuls Magit et Git eux-mêmes méritent d'être appelés porcelaines.

Notez que Magit peut s'interfacer avec Github (avec Magithub , voir aussi l' intégration de Github dans Emacs) et qu'Emacs a aussi des paquets pour travailler avec Gitlab , Bitbucket et d'autres.

Examples

Installation

Vous pouvez installer Magit de MELPA avec:

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

Utilisation de base: valide les éditions non mises en scène dans un référentiel existant

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

Lire Magit en ligne: https://riptutorial.com/fr/emacs/topic/3909/magit

Chapitre 11: Nomenclature d'Emacs

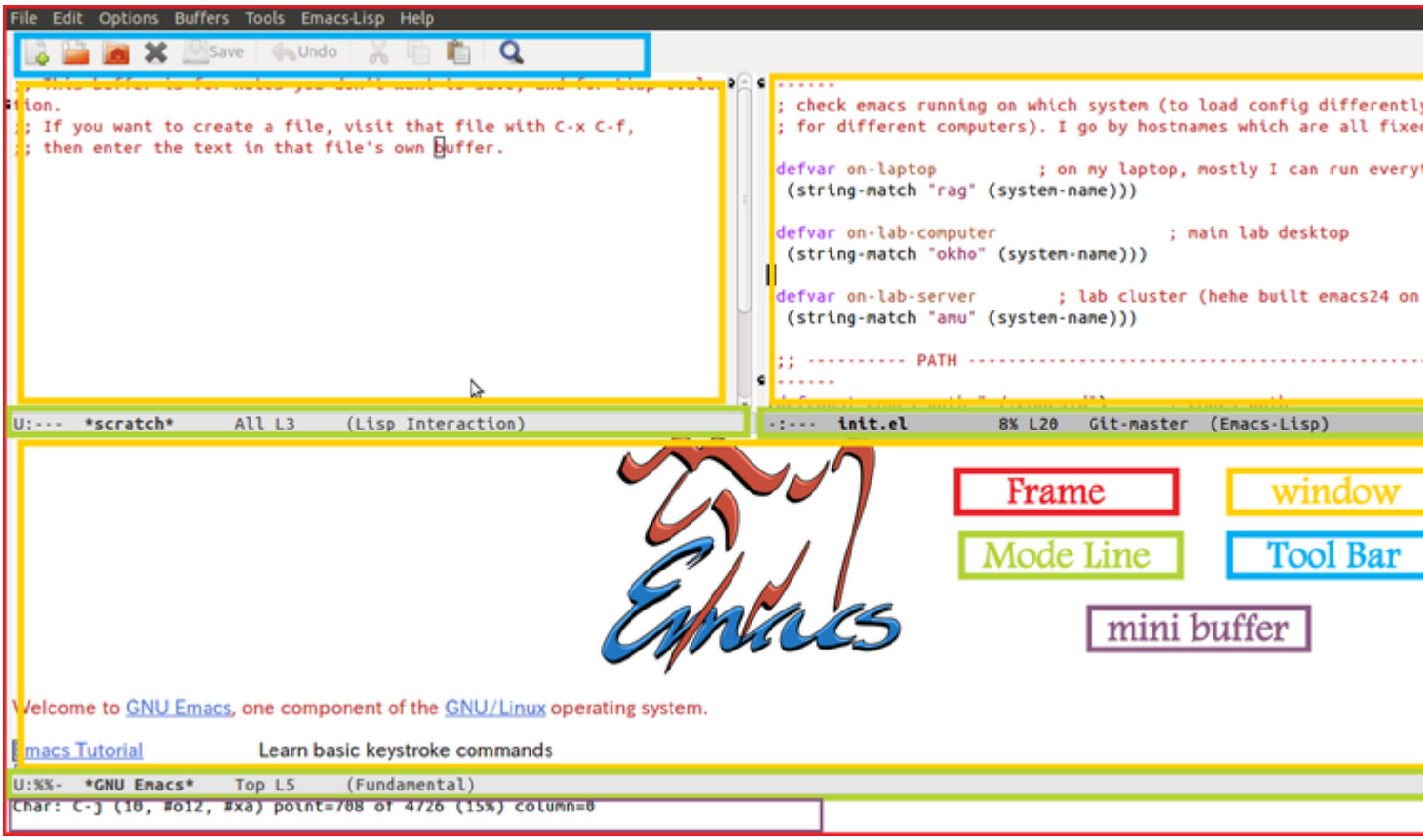
Examples

Fichiers et tampons

Dans Emacs, le *fichier* a la même signification que dans le système d'exploitation et est utilisé pour le stockage permanent des données. Un *tampon* est la représentation interne d'un fichier en cours d'édition. Les fichiers peuvent être lus dans des tampons à l'aide de `Cx Cf` , et des tampons peuvent être écrits dans des fichiers à l'aide de `Cx Cs` (enregistrer le fichier à son emplacement actuel) ou `Cx Cw` (écrire un fichier à un autre emplacement, équivalent à `Save as`).

Eléments de l'interface utilisateur

L'interface utilisateur d'Emacs utilise des termes qui ont été inventés tôt et qui peuvent être dérangeants pour les utilisateurs habitués à une terminologie plus moderne.



Cadre

Dans Emacs, ce qu'on appelle par ailleurs une fenêtre (la zone d'affichage utilisée par un programme) est appelée une *image* . Emacs commence à utiliser une *image* , mais des images supplémentaires peuvent être créées avec `Cx 5` .

Fenêtre

Un *cadre* contient une ou plusieurs *fenêtres* (généralement appelées volets), chacune affichant le contenu d'un *tampon* . Chaque *cadre* commence généralement par une seule *fenêtre* , mais des fenêtres supplémentaires peuvent être créées en divisant celles qui existent déjà. soit horizontalement en utilisant `Cx 2` ou verticalement avec `Cx 3` . Voir aussi Fenêtres ou cadres multiples .

Tampon

Le terme *tampon* fait référence au contenu affiché dans une *fenêtre* . Un tel contenu peut refléter le contenu d'un fichier dans le système de fichiers (ou peut-être une version mise à jour qui n'a pas encore été enregistrée sur le disque), mais plus généralement, il peut s'agir de n'importe quel type de texte.

Ligne de mode

Au bas de chaque *fenêtre se* trouve une *ligne de mode* qui décrit de manière synthétique le *tampon* affiché dans la fenêtre.

Barre d'outils

De manière similaire à de nombreux autres logiciels, une *barre d'outils* peut être affichée en haut de chaque *image* . Son contenu peut varier en fonction du type de *tampon en* cours de modification.

Minibuffer

Un *mini - tampon* , généralement affiché en bas de chaque *image* , permet d'interagir avec Emacs. Chaque fois qu'une commande demande une entrée utilisateur, elle est demandée dans le *mini - tampon* . Inversement, les messages affichés pour l'utilisateur à voir sont imprimés ici.

Point, marque et région

Emacs utilise les termes **point** , **marque** et **région** pour fournir plus de précision sur le texte sélectionné et la position du curseur. En comprenant ces termes, cela vous aidera à comprendre et à utiliser d'autres opérations et fonctions.

Le **point** est l'endroit dans un tampon où l'édition (*c'est-à-dire l'* insertion) est en cours et est généralement indiquée par un curseur.
La **marque** est un marqueur placé n'importe où dans le tampon à l'aide de commandes telles que

`set-mark-command` (`C-SPC`) ou `exchange-point-and-mark` (`Cx Cx`).
La **région** est la zone située entre le point et la marque et de nombreuses commandes opèrent sur la région, par *exemple* pour la supprimer, la vérifier, la mettre en retrait ou la compiler.

Lorsque vous cliquez sur votre souris sur un emplacement dans un tampon, vous voyez le point. Lorsque vous sélectionnez du texte, vous définissez la région (le texte sélectionné) et la marque (au début de votre sélection).

Tuer et tuer

Tuer et *tirer* plus ou moins correspond à ce qu'on appelle habituellement «couper» et «coller».

Meurtre

tuer signifie supprimer du texte et le copier sur le *kill-ring* (ce qui peut être considéré comme une sorte de "presse-papier" dans la terminologie "couper et coller"). Le *kill ring* est ainsi nommé car il stocke plusieurs morceaux de texte *tués* , auxquels on peut accéder ultérieurement dans un ordre cyclique.

Diverses commandes existent pour *tuer* un mot (`Md`), le reste de la ligne (`Ck`) ou des blocs de texte plus volumineux (tels que la région actuellement sélectionnée: `Cw`).

D'autres commandes existent, qui sauvegardent le texte sur le *kill ring* , sans le *tuer* (de la même manière que la "copie" dans les terminologies modernes). Par exemple, `Mw` , qui agit sur la région actuellement sélectionnée.

Taper

Les entrées dans le *kill ring* peuvent ensuite être *récupérées* dans un tampon. On peut généralement *tirer* le dernier texte *tué en* utilisant, *par exemple,* `Cy` (ce qui est similaire à l'opération "coller" dans une terminologie plus moderne). Mais d’autres commandes peuvent accéder aux anciennes entrées du *kill ring* et les *récupérer* .

Modes

Mode majeur

Emacs peut adapter son comportement au type spécifique de texte édité dans un *tampon* . L'ensemble des personnalisations spécifiques d'Emacs pour un type de texte particulier est appelé "mode majeur". Chaque tampon a exactement un *mode majeur en* fonction de son type de contenu.

Les modes principaux peuvent modifier la signification de certaines clés, définir des règles de mise en évidence de la syntaxe ou d'indentation, et installer de nouvelles liaisons de clés

(commençant généralement par `Cc`) pour les commandes spécifiques au mode. Emacs est livré avec un large éventail de modes majeurs, répartis en trois catégories principales:

- prise en charge du texte (par exemple, langages de balisage),
- prise en charge des langages de programmation,
- applications dans emacs (par exemple dired, gnus, ...). Les tampons utilisant ce dernier groupe de modes majeurs ne sont généralement pas associés à des fichiers, mais servent plutôt d'interface utilisateur.

Mode mineur

Les modes mineurs sont des fonctionnalités facultatives pouvant être activées ou désactivées. *Les modes mineurs* peuvent être activés pour des tampons spécifiques (modes tampon-local) ou tous les tampons (modes globaux). Contrairement aux *principaux modes,* un certain nombre de *modes mineurs* peuvent être activés pour un tampon donné.

Emacs fournit beaucoup de modes mineurs. Quelques exemples incluent:

- *Mode de remplissage* automatique pour envelopper automatiquement les lignes de texte à mesure que vous tapez.
- *Mode Flyspell* pour mettre en évidence les fautes d'orthographe lors de la *frappe* .
- *Mode ligne visuelle* pour envelopper les longues lignes à l'écran.
- *Mode de marque transitoire* pour mettre en évidence la région actuelle.

Lire Nomenclature d'Emacs en ligne: https://riptutorial.com/fr/emacs/topic/3683/nomenclature-d-emacs

Chapitre 12: Org-mode

Remarques

Org est un mode permettant de conserver des notes, de gérer des listes TODO et de planifier des projets avec un système de texte simple rapide et efficace. C'est aussi un système auteur avec un support unique pour la programmation alphabétisée et la recherche reproductible.

Site officiel du mode org

Examples

Syntaxe de balisage

Org fournit un langage de balisage complet qui aide à structurer le document et se reflète aussi précisément que possible lors de l'exportation vers d'autres formats (comme HTML ou LaTeX).

Structure

Titre du document

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

Sectionnement

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

Des listes

```
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
```

Cases à cocher

Chaque élément d'une liste simple peut être transformé en une case à cocher en commençant par la chaîne '[]'.

```
* 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-rubrique
- `Cc Cx o` propriété de org-toggle
- `Cc #` org-update-statistics-cookies

Accentuation et monospace

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

Le texte du code et la chaîne verbatim ne sont pas traités pour la syntaxe du mode organisation, il est exporté tel quel.

Liens et références

Liens

Org-mode reconnaîtra les formats d'URL et les activera en tant que liens cliquables. Cependant, les liens peuvent être explicitement déclarés comme ceci:

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

Ou bien :

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

Notes de bas de page

Les notes de bas de page peuvent être nommées:

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

ou anonyme et en ligne:

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

Fixations de base

Pour parcourir le niveau de contour affiché:

- `Tab` Cycle level level pour une rubrique
- `Maj-Tab` Niveau du contour du cycle pour l'ensemble du document

Pour parcourir les états `TODO` :

- `Shift-Right Arrow`
- `Shift-Left Arrow`

Pour augmenter ou diminuer le niveau hiérarchique d'un titre

- `Meta-Right Arrow` Niveau inférieur ("augmenter l'indentation")
- `Meta-Left Arrow` Niveau supérieur ("diminution de l'indentation")

Pour faire pivoter la priorité pour un titre donné:

- `Flèche vers le haut`
- `Flèche vers le bas`

Pour déplacer un titre vers le haut ou le bas:

- `Flèche Meta-Up`
- `Flèche Meta-Down`

(`Meta` fait référence à différentes clés sur différents claviers. Le plus souvent, il s'agit d' `Alt` ou de).

Blocs de code

Pour ajouter un bloc de code, entourez-le avec le `#+BEGIN_SRC language` `#+END_SRC` et `#+END_SRC` . *la langue* devrait correspondre au mode majeur pour la langue en question, par exemple le mode majeur pour Emacs Lisp est `emacs-lisp-mode` , alors écrivez `#+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
```

Vous pouvez ouvrir le bloc de code dans un tampon séparé en tapant `Cc '` (pour `org-edit-special`). Si vous ne possédez pas le mode principal pour la langue spécifiée, cela vous donnera un message d'erreur tel que `No such language mode: foo-mode` .

Si le contenu que vous souhaitez placer dans le bloc ne se trouve dans aucun langage de programmation, vous pouvez utiliser `#+BEGIN_EXAMPLE` et `#+END_EXAMPLE` place.

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

Il existe des modèles simples pour les deux. Au début de la ligne, tapez `<s` ou `<e` , puis appuyez sur `TAB` . Il deviendra un bloc avec des marqueurs de début et de fin pour `SRC` ou `EXAMPLE` , respectivement.

Ces marqueurs ne sont pas sensibles à la casse, vous pouvez donc écrire `#+begin_src` etc à votre place.

les tables

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

Pour ajouter une table en mode org, entourez simplement vos colonnes d'une barre (`|`)

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

Lorsque vous appuyez sur `Entrée` depuis une colonne, le mode org crée automatiquement une nouvelle ligne avec les barres.

- `Tab` et `Return` seront respectivement déplacés vers la cellule ou la ligne suivante (ou en créer une nouvelle s'il n'y en a pas)
- Vous pouvez échanger les lignes et les colonnes avec `M-ArrowKey`
- `MS-Down` et `MS-Right` créeront respectivement une ligne (au-dessus du courant) et une colonne (à gauche du courant)
- `MS-Up` et `MS-Left` supprimeront respectivement la ligne en cours et la colonne en cours
- `Cc je` crée un séparateur

Lire Org-mode en ligne: https://riptutorial.com/fr/emacs/topic/6259/org-mode

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec emacs	Adobe, Community, ebpa, Ehvince, eyqs, IntFloat, Jack Henahan, joon, legoscia, mellowmaroon, Michel de Ruiter, Nemanja Trifunovic, omul, pcurry, Prasanna, salotz, squiter
2	Aide dans Emacs	mellowmaroon, Terje D., ygram
3	Barre	Yuki Inoue
4	emacs possède déjà une documentation de très haute qualité et bien organisée. pourquoi le dupliquer?	erjoalgo
5	Gérer les favoris dans Emacs	Francesco, Prasanna
6	Gestion des packages	Adobe, Kaushal Modi, leeor, pcurry, salotz, squiter
7	Kits de démarrage	dangom, Ehvince, Kaushal Modi, leeor, Micah Elliott, Xinyang Li
8	Les nombreuses variantes d'Emacs	pcurry
9	Liens de 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.
10	Magit	boehm_s, Ehvince, glallen, mellowmaroon, squiter
11	Nomenclature d'Emacs	Doug Harris, Francesco, Nikana Reklawyks, Stephen Leppik, Terje D.
12	Org-mode	Christopher Bottoms, Francesco, legoscia, SuperBear, Wazam