Kostenloses eBook

LERNEN

emacs

Free unaffiliated eBook created from
Stack Overflow contributors.

#emacs

Inhaltsverzeichnis

Über

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

Kapitel 1: Erste Schritte mit Emacs

Bemerkungen

Emacs ist ein Texteditor, dessen wichtigste Funktion die Möglichkeit ist, dass Benutzer nahezu alle Aspekte des Programms programmgesteuert anpassen können. Dies wird durch einen speziellen Dialekt der Programmiersprache Lisp (Emacs Lisp) erleichtert, der speziell für die Verwendung im Emacs-Editor erstellt wurde.

In Emacs Lisp gibt es eine Vielzahl von Erweiterungen, die die Funktionalität von Emacs erweitern. Diese Erweiterungen umfassen Bearbeitungsfunktionen für bestimmte Programmiersprachen (ähnlich wie die IDE möglicherweise bietet), E-Mail- und IRC-Clients, Git-Frontends, Spiele wie Tetris und 2048 und vieles mehr.

Viele Aspekte des Emacs-Editors können ohne Programmierkenntnisse verwendet werden. Benutzer, die Emacs programmgesteuert anpassen möchten, werden jedoch bestimmte Funktionen der Sprache Emacs Lisp, wie das (Selbst-) Dokumentationssystem, als unglaublich hilfreich und entgegenkommend empfinden.

Externe Referenzen:

1. Die Website von Sacha chua ist ein sehr guter Ort, um auf Emacs weitere Lernressourcen zu finden.

   ein. Für diejenigen, die einen visuelleren Eindruck auf dem Emacs-Lernpfad benötigen

   b. Für diejenigen, die die Tastenkombinationen einfach erhalten möchten

2. Wikemacs basiert auf mediawiki und hat daher strukturierte Inhalte, durchsuchbare Kategorien und dergleichen. Beginnen Sie mit der Erkundung !

Versionen

Ausführung	Veröffentlichungsdatum
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

Ausführung	Veröffentlichungsdatum
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 oder Setup

Detaillierte Anweisungen zum Einrichten oder Installieren von emacs.

Offizielle Anweisungen sind auf der GNU Emacs-Website verfügbar .

Debian-Systeme

Auf Systemen mit dem Debian-Paketmanager (wie Debian, Ubuntu und Mint) können Emacs mit dem einfachen Befehl installiert werden:

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

Für eine blutende Version kann man folgendes ppa verwenden:

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

Bau für Quelle

Wenn Ihre Debian-basierte Distribution nicht die von Ihnen gewünschte Version von Emacs hat,

können Sie sie von Grund auf neu erstellen.

```
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*
```

Redhat-Systeme

Auf Systemen mit dem Redhat-Paketmanager (wie RHEL, CentOS und Fedora Core) können Emacs mit dem einfachen Befehl installiert werden:

```
sudo yum install emacs
```

Arch Linux

Emacs kann mit dem einfachen Befehl installiert werden:

```
sudo pacman -Syu emacs
```

Gentoo und Funtoo

Auf Systemen, auf denen Portage ausgeführt wird, können Emacs mit dem einfachen Befehl installiert werden:

```
sudo emerge emacs
```

GSRC (GNU Source Release Collection)

Funktioniert auf jedem GNU / Linux-System, um die neueste Version von emacs zu erhalten, ohne den Systempaket-Manager (der veraltet sein kann) zu verwenden oder das Archiv oder die Binärdatei herunterzuladen. Um `gsrc` zu installieren, `gsrc` Sie die Dokumentation . Dann:

```
cd gsrc
make -C gnu/emacs install
```

```
# add the binaries to your PATH
source ./setup.sh
```

Darwin-Systeme

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

Vorkompilierte App-Pakete für die neuesten Stable- und Entwicklungsversionen können unter https://emacsformacosx.com heruntergeladen werden.

Windows

Schokoladen- Paketmanager

Emacs können mit installiert werden

```
choco install emacs
```

Scoop- Paketmanager

Kann vom Extras-Eimer installiert werden

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

Offizielle Binäre Installateure

- stabil
- neueste

(Beachten Sie, dass offizielle Binärdateien nicht mit einigen Bibliotheken geliefert werden, z. B. Bibliotheken für Bildformate.)

Andere binäre Installer

- Emacs mit vorgefertigten AUCTeX und ESS

- 64-Bit-GNU-Emacs für MS Windows mit Optimierung bietet ein natives und optimiertes 64-Bit-Binärinstallationsprogramm mit unverändertem Quellcode von git master und Release-Version. JPEG, GIF, PNG, TIFF, SVG, XML2 und GnuTLS unterstützen Out-of- Box.

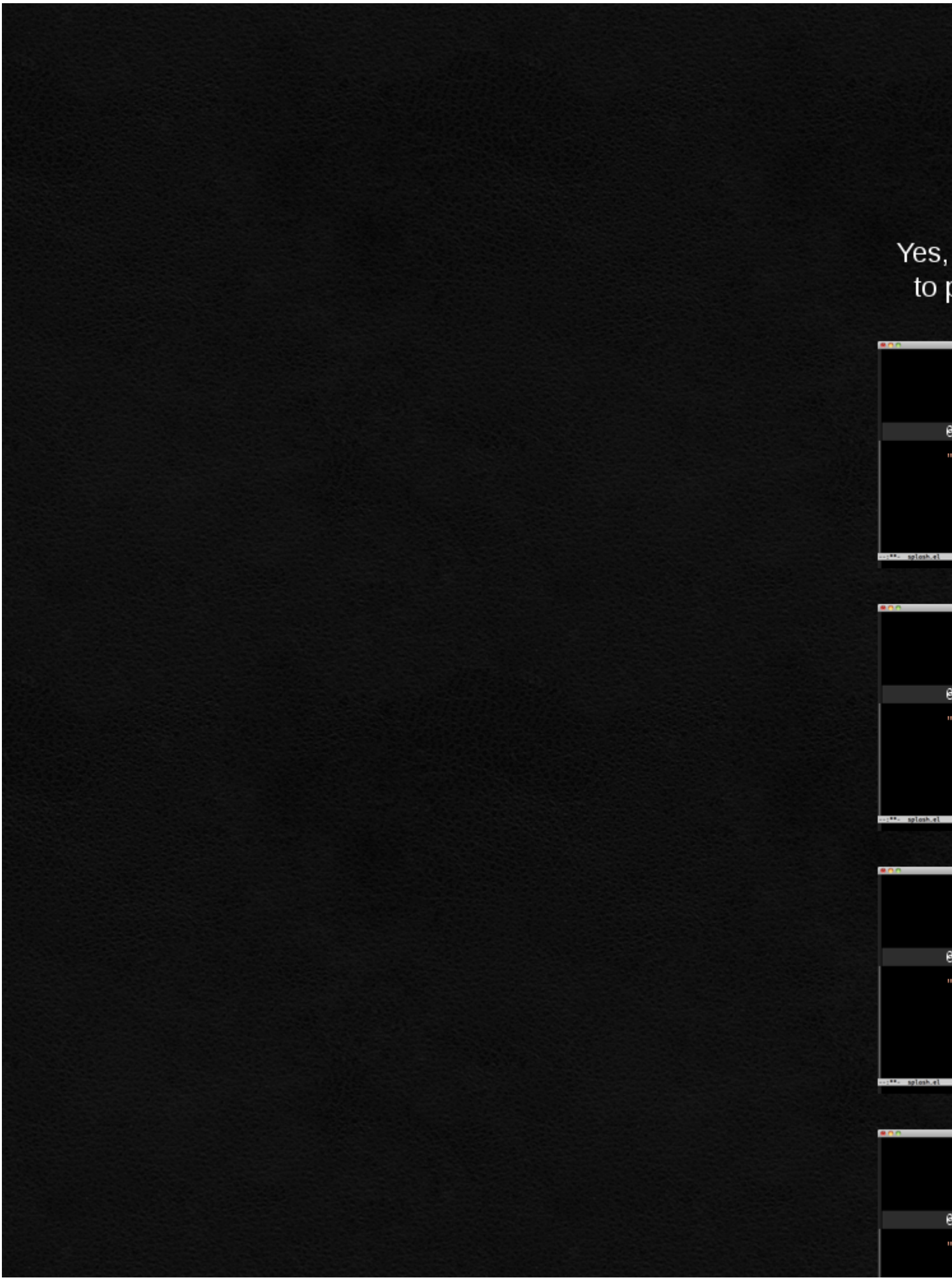
Interaktives Emacs-Tutorial

Geben Sie in Emacs `Ch t` (Strg-h, t) ein, um ein hervorragendes interaktives Lernprogramm in Emacs zu erhalten. Der Benutzer lernt die grundlegende Navigation und Bearbeitung, indem er den TUTORIAL-Text selbst bearbeitet, während er das Tutorial liest. (Änderungen am Lernprogramm werden verworfen, wenn das Lernprogramm geschlossen wird. Wenn also ein Benutzer das Lernprogramm anfordert, handelt es sich um eine saubere Standardversion des Lernprogramms.

Das Erste, was in diesem Tutorial hilfreich ist, ist das Verstehen der `C-<chr>` und `M-<chr>` im Text. Die zweite Sache ist, wie man im Text vorwärts und rückwärts blättert.

Emacs Rocks Video-Tutorials

Gute Video-Tutorials zu Emacs finden Sie unter emacsrocks.com


Yes,
to p

https://riptutorial.com/de/emacs/topic/986/erste-schritte-mit-emacs

Kapitel 2: Die vielen Varianten von Emacs

Einführung

Die meisten dieser Dokumentation gelten implizit oder explizit für GNU Emacs. Dies kann die bekannteste Variante von Emacs sein, die Quelle mehrerer Gabeln und das Ziel einiger Zusammenführungen.

In diesem Thema werden einige der möglichen Varianten von Emacs sowie deren hauptsächliche Unterschiede zu GNU Emacs beschrieben.

Examples

Spacemacs

Spacemacs (http://spacemacs.org/) ist eine Variante von Emacs, die versucht, den langfristigen Konflikt zwischen Emacs und Vim-Benutzern zu beenden, indem ein Emacs erstellt wird, der sich wie Vim verhält.

Die vielen Varianten von Emacs online lesen: https://riptutorial.com/de/emacs/topic/9456/die-vielen-varianten-von-emacs

Kapitel 3: emacs verfügt bereits über eine qualitativ hochwertige, gut organisierte Dokumentation. warum es duplizieren?

Einführung

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

Examples

Schlüssel

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

3 Schlüssel

Einige Emacs-Befehle werden von nur einem Eingabeereignis aufgerufen. Zum Beispiel bewegt sich Cf um ein Zeichen im Puffer vorwärts. Für andere Befehle sind zwei oder mehr Eingabeereignisse erforderlich, z. B. Cx Cf und Cx 4 Cf.

Eine Tastenfolge oder kurz für eine Taste ist eine Folge von einem oder mehreren Eingabeereignissen, die als Einheit sinnvoll ist. Wenn eine Tastenfolge einen Befehl aufruft, nennen wir ihn einen vollständigen Schlüssel. Zum Beispiel sind Cf, Cx Cf und Cx 4 Cf vollständige Schlüssel. Wenn eine Tastenfolge nicht lang genug ist, um einen Befehl aufzurufen, nennen wir sie eine Präfix-Taste. Im vorherigen Beispiel sehen wir, dass Cx und Cx 4 Präfixschlüssel sind. Jede Tastenfolge ist entweder ein vollständiger Schlüssel oder ein Präfixschlüssel.

emacs verfügt bereits über eine qualitativ hochwertige, gut organisierte Dokumentation. warum es duplizieren? online lesen: https://riptutorial.com/de/emacs/topic/9077/emacs-verfugt-bereits-uber-eine-qualitativ-hochwertige--gut-organisierte-dokumentation--warum-es-duplizieren-

Kapitel 4: Emacs-Nomenklatur

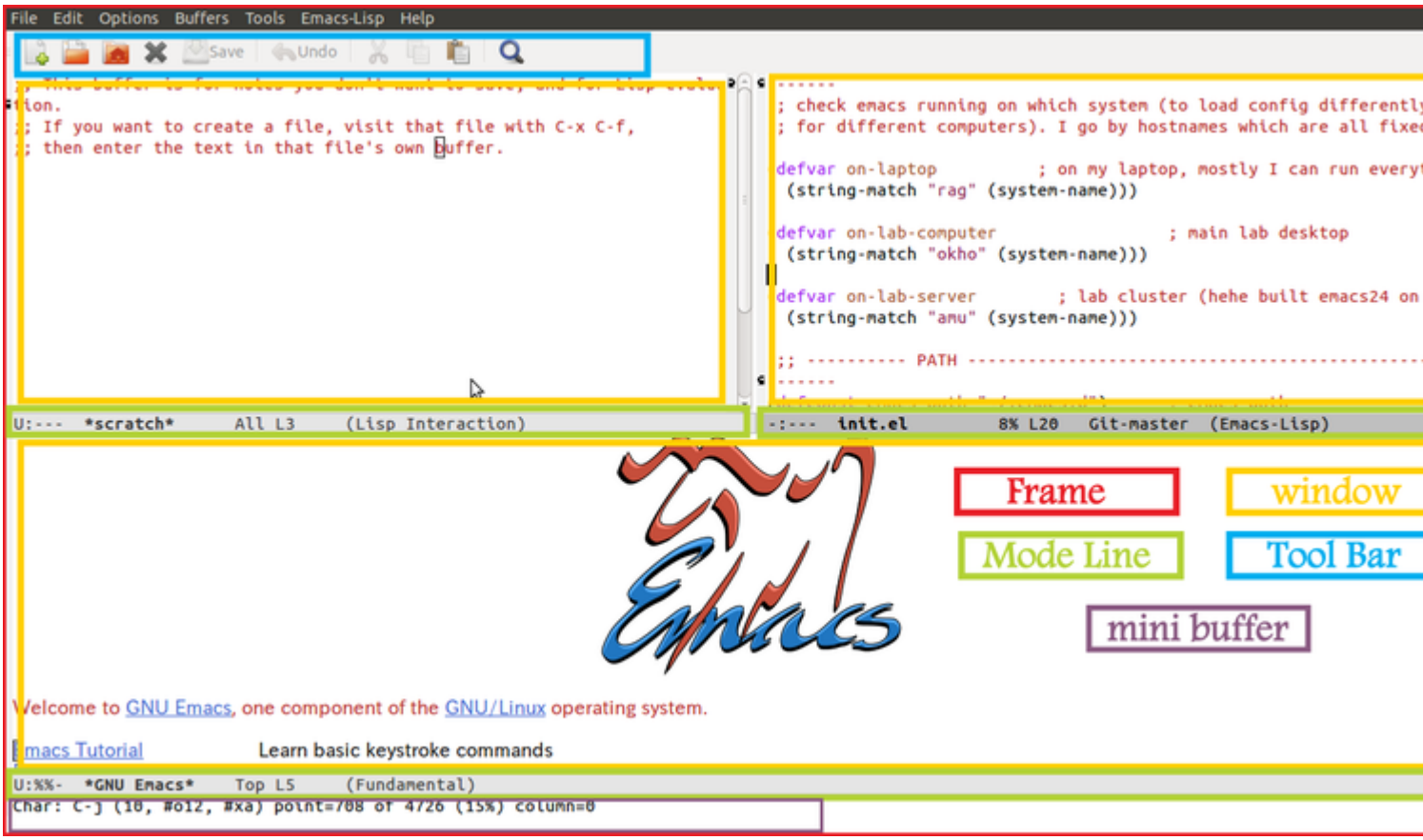
Examples

Dateien und Puffer

In Emacs hat *datei* dieselbe Bedeutung wie im Betriebssystem und wird zum dauerhaften Speichern von Daten verwendet. Ein *Puffer* ist die interne Darstellung einer zu bearbeitenden Datei. Dateien können mit `Cx Cf` in Puffer eingelesen werden, und Puffer können mit `Cx Cs` (Datei am aktuellen Speicherort speichern) oder `Cx Cw` (Datei an einen anderen Speicherort schreiben, dazu aufgefordert werden - dies entspricht `Speichern als`).

Elemente der Benutzeroberfläche

Die Benutzeroberfläche von Emacs verwendet Begriffe, die zu einem frühen Zeitpunkt geprägt wurden und für Benutzer mit einer moderneren Terminologie beunruhigend sind.



Rahmen

Was in Emacs sonst als Fenster (der Anzeigebereich eines Programms) bezeichnet wird, wird als *Frame bezeichnet* . Emacs verwendet nun ein *Bild* , obwohl mit `Cx 5` weitere Bilder erstellt werden können.

Fenster

Ein *Frame* enthält ein oder mehrere *Fenster* (normalerweise als *Fenster* bezeichnet), von denen jedes den Inhalt eines *Puffers zeigt* . Jeder *Frame* beginnt normalerweise mit nur einem *Fenster* , es können jedoch weitere Fenster erstellt werden, indem vorhandene Fenster aufgeteilt werden. entweder horizontal mit `Cx 2` oder vertikal mit `Cx 3` . Siehe auch Fenster oder Rahmen multiplizieren .

Puffer

Der Begriff *Puffer* bezieht sich auf den in einem *Fenster* angezeigten Inhalt. Dieser Inhalt kann den Inhalt einer Datei im Dateisystem (oder möglicherweise eine aktualisierte Version, die noch nicht auf der Festplatte gespeichert wurde) widerspiegeln, allgemeiner kann es sich jedoch um beliebige Textarten handeln.

Moduszeile

Am unteren Rand jedes *Fensters* befindet sich eine *Moduszeile* , die den im Fenster angezeigten *Puffer* synthetisch beschreibt.

Werkzeugleiste

In ähnlicher Weise wie viele andere Software kann eine *Werkzeugleiste* am oberen Rand jedes *Rahmens* angezeigt. Sein Inhalt kann je nach der Art des *Puffers* in Abhängigkeit zur Zeit bearbeitet wird.

Minibuffer

Ein *Minipuffer* , der normalerweise am unteren Rand jedes *Frames* angezeigt *wird* , ermöglicht die Interaktion mit Emacs. Jedes Mal, wenn ein Befehl nach einer Benutzereingabe fragt, wird dies im *Minibuffer angefordert* . Umgekehrt werden hier für den Benutzer angezeigte Meldungen gedruckt.

Punkt, Marke und Region

Emacs verwendet die Begriffe **Punkt** , **Marke** und **Region** , um den ausgewählten Text und die Position des Cursors genauer zu bestimmen. Durch das Verständnis dieser Begriffe können Sie andere Vorgänge und Funktionen besser verstehen und verwenden.

Der **Punkt** ist die Stelle in einem Puffer, an der gerade eine Bearbeitung (*dh* Einfügung) stattfindet und normalerweise durch einen Cursor angezeigt wird.
Die **Marke** ist eine **Marke** , die an beliebiger Stelle im Puffer mit Befehlen wie `set-mark-command` (`C-`

`SPC`) oder `exchange-point-and-mark` (`Cx Cx`) platziert wird.
Der **Bereich** ist der Bereich zwischen Punkt und Marke, und viele Befehle wirken sich auf den Bereich aus, *z. B. zum* Löschen, zur Rechtschreibprüfung, zum Einrücken oder zum Kompilieren.

Wenn Sie mit der Maus auf eine Position in einem Puffer klicken, wird der Punkt angezeigt. Wenn Sie Text auswählen, legen Sie die Region (den ausgewählten Text) und die Markierung (am Anfang Ihrer Auswahl) fest.

Töten und reißen

Töten und *Ruckeln entspricht* mehr oder weniger dem, was üblicherweise als "Schneiden" und "Einfügen" bezeichnet wird.

Tötung

töten bedeutet, Text zu löschen und in den *Kill-Ring zu* kopieren (was in der Terminologie "Ausschneiden & Einfügen" als eine Art "Zwischenablage" angesehen werden kann). Der *Kill-Ring hat* seinen Namen, weil er mehrere *getötete* Texte speichert, auf die später in zyklischer Reihenfolge zugegriffen werden kann.

Es gibt verschiedene Befehle, *die* ein Wort (`Md`), den Rest der Zeile (`Ck`) oder größere Textblöcke (beispielsweise die aktuell ausgewählte Region: `Cw`) *töten* .

Es gibt andere Befehle, die Text im *Killring* speichern, ohne ihn tatsächlich zu *töten* (ähnlich wie das Kopieren in modernen Terminologien). Zum Beispiel `Mw` , das auf die aktuell ausgewählte Region wirkt.

Ruckeln

Einträge im *Kill - Ring* kann später wieder in einen Puffer *gezerrt* werden. Typischerweise *kann man* den zuletzt *getöteten* Text *z. B. mit* `Cy` *mitnehmen* (was der "Einfügen" *-Operation* in einer moderneren Terminologie ähnelt). Andere Befehle können jedoch auf ältere Einträge vom *Kill-Ring* zugreifen und diese *mitnehmen* .

Modi

Hauptmodus

Emacs kann sein Verhalten an den spezifischen Text anpassen, der in einem *Puffer* bearbeitet wird. Der Satz spezifischer Emacs-Anpassungen für einen bestimmten Texttyp wird als "Hauptmodus" bezeichnet. Jeder Puffer hat je nach Inhaltstyp genau einen *Hauptmodus* .

Hauptmodi können die Bedeutung einiger Schlüssel ändern, Syntax-Markierungs- oder Einrückungsregeln definieren und neue Tastenzuordnungen (normalerweise beginnend mit `Cc`) für

modusspezifische Befehle installieren. Emacs-Schiffe mit einer großen Auswahl an Hauptmodi lassen sich in drei Hauptkategorien unterteilen:

- Unterstützung für Text (zB Auszeichnungssprachen),
- Unterstützung für Programmiersprachen
- Anwendungen innerhalb von Emacs (zB dired, gnus, ...). Puffer, die diese letzte Gruppe von Hauptmodi verwenden, werden normalerweise nicht mit Dateien verknüpft, sondern dienen als Benutzeroberfläche.

Minor-Modus

Minor-Modi sind optionale Funktionen, die ein- und ausgeschaltet werden können. *Kleinere Modi* können für bestimmte Puffer (lokale Puffermodi) oder für alle Puffer (globale Modi) aktiviert werden. Im Gegensatz zu den *Hauptmodi* kann für einen bestimmten Puffer eine beliebige Anzahl von *Nebenmodi* aktiviert werden.

Emacs bietet viele kleinere Modi. Einige Beispiele sind:

- *Auto-fill-Modus* , um Textzeilen während der Eingabe automatisch umzuwandeln.
- *Flyspell-Modus* , um Rechtschreibfehler während der *Eingabe* hervorzuheben.
- *Visual Line-Modus* , um lange Zeilen an den Bildschirm anzupassen.
- *Transient Mark-Modus* , um die aktuelle Region hervorzuheben.

Emacs-Nomenklatur online lesen: https://riptutorial.com/de/emacs/topic/3683/emacs-nomenklatur

Kapitel 5: Grundlegende Tastenkombinationen

Examples

Beenden Sie Emacs

Sie können Emacs mit der folgenden Tastenkombination beenden:

```
Cx Cc
```

Wo `C` der `control` .

Emacs suspendieren

Sie können Emacs mit der folgenden Tastenkombination aussetzen:

```
Cz
```

Es bringt dich zurück zu deiner Schale. Wenn Sie Ihre emacs-Sitzung fortsetzen möchten, geben Sie in Ihrem Terminal `fg` ein.

Dateibehandlung

- Geöffnete Datei unter demselben Dateinamen erneut speichern (Speichern):

  ```
  Cx Cs
  ```

- Schreiben Sie als `filename` (Speichern unter):

  ```
  Cx Cw filename
  ```

  Der neue Dateiname wird im Minibuffer abgefragt.

- Neue Datei erstellen **oder** vorhandene Datei laden (Neu / Laden):

  ```
  Cx Cf filename
  ```

  Mit der Mnemonik hier für die f-Datei. Sie werden aufgefordert, einen Dateipfad im Minibuffer anzugeben.

- Alternative Datei besuchen

  ```
  Cx Cf
  ```

  Wenn die Datei noch nicht vorhanden ist, werden Sie aufgefordert, den Pfad der Datei im Minibuffer zu erstellen.

Aktuellen Befehl abbrechen

Oft geraten Sie in einen Zustand, in dem Sie eine teilweise eingegebene Befehlsfolge haben, die Sie jedoch abbrechen möchten. Sie können es mit einer der folgenden Tastenkombinationen abbrechen:

`Cg`

`Esc Esc Esc`

Multiples Fenster oder Frames

"Fenster" in Emacs bezieht sich auf etwas, das sonst als "Teilfenster" oder "Bildschirmaufteilung" bezeichnet werden kann. Einige Fenstermanipulationsbefehle umfassen:

- Aktuelles Fenster horizontal `teilen` : `Cx 2`
- Aktuelles Fenster vertikal `teilen` : `Cx 3`
- Nächstes Fenster `auswählen` : `Cx o`
- Aktuelles Fenster schließen: `Cx 0`
- Schließen Sie alle anderen Fenster mit Ausnahme des aktuellen Fensters: `Cx 1`

Ein "Frame" in Emacs ist das, was man sonst als "Fenster" bezeichnen könnte. Frames werden mit diesen Befehlen bearbeitet:

- Neuen Frame erstellen: `Cx 5 2`
- Aktuelles Bild `löschen` : `Cx 5 0`
- Löschen Sie andere Bilder: `Cx 5 1`

Wechselfenster können mit erreicht werden

- `S-links` , `S-rechts` , `S-auf` , `S-Ab` (d. `H`. In Verbindung mit einer Pfeiltaste umschalten), um in eine Richtung zum Nachbarfenster zu wechseln, oder
- `Cx o` um zum nächsten Fenster zu wechseln.

Puffer

- Beispiel für eine Pufferliste

```
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
```

  Das erste Feld einer Zeile zeigt an:

  - '.' Der Puffer ist aktuell.

    - '%' ist ein Nur-Lese-Puffer.
    - '*' Der Puffer wird geändert.

- Puffer auswählen. Sie können einen beliebigen offenen Puffer mit der folgenden Tastenkombination auswählen:

  `Cx b`

  Sie werden aufgefordert, den Namen des Puffers anzugeben, zu dem Sie wechseln möchten.

- Puffer auflisten:

  `Cx Cb`

- Save-Some-Buffer, der die Wahl gibt, welcher Puffer gespeichert werden soll oder nicht:

  `Cx s`

- Töte einen Puffer:

  `Cx k`

- Operationen an markierten Puffern:

  `S` Speichern Sie die markierten Puffer

  `A` Zeigen Sie die markierten Puffer in diesem Rahmen an.

  `H` Zeigen Sie die markierten Puffer in einem anderen Frame an.

  `V` Setzen Sie die markierten Puffer zurück.

  `T` Nur-Lese-Status markierter Puffer umschalten.

  `D` Töten Sie die markierten Puffer.

  `Ms a Cs` Inkrementelle Suche in den markierten Puffern.

  `Ms a CMs` Suche nach Regex in den markierten Puffern.

  `U` Ersetzen Sie sie durch Regex in jedem der markierten Puffer.

  `Q` Abfrage in jedem der markierten Puffer ersetzen.

  `Ich` wie oben, mit einem regulären Ausdruck.

  `P` Drucken Sie die markierten Puffer aus.

  `O` Listet die Zeilen in allen markierten Puffern auf, die mit einem bestimmten Regex übereinstimmen (wie bei der Funktion `come ').

  `X` Pipe den Inhalt der markierten Puffer an einen Shell-Befehl.

  `N` Ersetzen Sie den Inhalt der markierten Puffer durch die Ausgabe eines Shell-Befehls.

`!` Führen Sie einen Shell-Befehl mit der Pufferdatei als Argument aus.

`E` Auswerten eines Formulars in jedem der markierten Puffer. Dies ist ein sehr flexibler Befehl. Wenn Sie beispielsweise alle markierten Puffer schreibgeschützt machen möchten, versuchen Sie (schreibgeschützter Modus 1) als Eingabeformular.

`W` - Wie oben, aber sehen Sie sich jeden Puffer an, während das Formular ausgewertet wird.

`k` - Entfernen Sie die markierten Zeilen aus dem *Ibuffer*- Puffer, aber *beenden Sie* den zugehörigen Puffer nicht.

`x` - Tötet alle zum Löschen markierten Puffer.

- Save-Some-Buffer, der die Wahl gibt, welcher Puffer gespeichert werden soll oder nicht:

  `Cx s`

- Wechseln Sie zum nächsten Puffer:

  `Cx RECHTS`

- Zum vorherigen Puffer wechseln:

  Cx LEFT

Suchen und Ersetzen

In Emacs können Sie mit dem einfachen Suchwerkzeug (`I-Search`) nach oder vor dem Cursor suchen.

- Um nach einem *Text* nach der Position des Cursors zu `search-forward` (`search-forward`), *`sometext`* `Cs` . Wenn Sie zum nächsten Vorkommen eines bestimmten *`sometext`* , drücken *`sometext`* einfach erneut `Cs` (usw.). Wenn der Cursor am rechten Ort landet, drücken Sie die `Eingabetaste` , um die Suchabfrage zu verlassen.

- Um vor dem Cursor zu `search-backward`), verwenden Sie `Cr auf` die gleiche Weise wie zuvor.

- Drücken Sie 2-mal `Cs,` um vom `search-backward` zum `search-forward` zu wechseln. Drücken Sie 2-mal `Cr` , `search backward` zu `search backward` wenn Sie sich in `search-forward` Suchvorwärtsaufforderung befinden.

- Suchen und ersetzen:

  `M-%` (oder `Esc-%) oldtext Geben Sie newtext eingeben`

  - Bestätigen Sie: `y`
  - Überspringen: `n`
  - Beenden: `q`
  - Ersetzen Sie alle `:!`

Region - Ausschneiden, Kopieren, Einfügen

- Markierung an der Cursorposition setzen:

  `C-Raum` oder `C- @`

- Kill region (Cut):

  `Cw`

- Kopieren Sie die Region, um den Ring zu töten:

  `Mw` oder `Esc-w`

- Yank (Paste) kürzlich getötet:

  `Cy`

- Yank (Paste) als nächster getötet:

  `Mein` oder `Esc-y`

Töten

`kill` ist der Befehl, mit dem Emacs Text löscht. Der Befehl " `kill` Befehl " `cut` in Windows. Es gibt verschiedene Befehle, die ein Wort (`Md`), den Rest der Zeile (`Ck`) oder größere Textblöcke "töten". Der gelöschte Text wird dem `kill-ring` hinzugefügt, aus dem er später gezogen werden `yanked` .

Auswählen und schneiden (töten)

Töten und Rucken Ähnlich wie beim `select-and-cut` in Windows haben wir hier `C-spc` . Die Tastenkombination `C-spc` startet die Auswahl, der Benutzer kann die Markierung mit Hilfe der Pfeiltasten oder eines anderen Befehls verschieben, um eine Auswahl zu treffen. Sobald die Auswahl abgeschlossen ist, drücken Sie `Cw` , um den ausgewählten Text zu beenden

Einige grundlegende Befehle, die als Schnellreferenz für den `kill` Befehl verwendet werden können (aus dem Emacs-Tutorial)

```
    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
```

Ruck

`yank` beschreibt das Einfügen eines zuvor gelöschten Textes, *z. B.* mit `Cy` , wodurch der zuletzt getötete Text gerissen wird. `Yank` Befehl entspricht dem `paste` Befehl in Windows.

Yank-Text zuvor getötet

Wir wissen, dass der `kill` Befehl den getöteten Text einem `kill-ring` hinzufügt. Um den gelöschten Text aus dem `kill-ring` abzurufen, verwenden Sie den Befehl `Mein` Kommando so oft, bis der gewünschte Text gezogen ist.

(Hinweis: Damit die `My-` Taste funktioniert, sollte der vorherige Befehl ein `YANK` sonst würde es nicht funktionieren.)

Cursor (Punkt) Bewegung

Neben den Cursorbewegungen mit den Pfeiltasten, Start, Ende, Bild auf und Bild ab, definiert emacs eine Anzahl von Tastenanschlägen, mit denen der Cursor über kleinere oder größere Textteile bewegt werden kann:

Nach Charakter:

- Rückwärtszeichen: `Cb`
- Weiterleitungszeichen: `Cf`

Durch wort

- Rückwärtswort: `Mb` (*dh* `Alt b` oder `Meta b`)
- Wort `weiterleiten` : `Mf`

Durch Zeile:

- Beginn der aktuellen Zeile: `Ca`
- Beginn der aktuellen Zeile erstes Zeichen (kein Leerzeichen): `Mm`
- Ende der aktuellen Zeile: `Ce`
- Vorherige Zeile: `Cp`
- Nächste Zeile: `Cn`

Gesamter Puffer:

- Beginn des Puffers: `M- <`
- Ende des Puffers: `M->`

Durch 'Block' abhängig vom Kontext (Modus):

Typische Tastenkombinationen:

- Rückwärtssatz / Aussage: `Ma`
- Satz / Aussage weiterleiten: `Ich`
- Beginn der Funktion: `MCa`
- Ende der Funktion: `MCe`

Präfix-Argumente

Um mehrere "Schritte" gleichzeitig zu verschieben, können die Bewegungsbefehle durch Drücken von `ESC` oder `Cu` und einer Zahl vor den aufgelisteten Tastenanschlägen ein Präfixargument gegeben werden. Für `Cu` ist die Anzahl optional und der Standardwert 4.
Eg `ESC 3 Cn` verschiebt 3 Zeilen nach unten, während `Cu Mf` vorwärts 4 Worte bewegt.

Rückgängig machen

Um etwas rückgängig zu machen, haben Sie gerade getan:

`C-_` oder `Cxu` oder `C- /`

Fall

- Großbuchstabe: `Mc`
- Konvertieren Sie das Wort in Großbuchstaben: `Mu`
- Konvertiere das Wort in Kleinbuchstaben: `Ml`

Notationen für Tastenkombinationen

Die Dokumentation von Emacs verwendet eine konsistente Notation für alle Tastenkombinationen, die hier erläutert wird:

Tastenakkorde

Ein "Tastenakkord" wird durch gleichzeitiges Drücken von zwei oder mehr Tasten erhalten. Tastenakkorde werden durch Trennen aller Tasten durch Striche (`-`) gekennzeichnet. In der Regel handelt es sich dabei um Modifikatortasten, die nach vorne gestellt werden:

- `C-` : Kontrolle;
- `S-` : Schicht;
- `M-` : alt (aus historischen Gründen steht das "M" für "Meta").

Andere Schlüssel werden einfach durch ihren Namen gekennzeichnet, wie:

- `a` : der `a` Schlüssel;
- `links` : die linke Pfeiltaste;
- `SPC` : die Leertaste;
- `RET` : die Rücktaste.

Beispiele für Key Chords sind daher:

- `Ca:` Pressen `Steuerung` und `eine` gleichzeitig;
- `S-rechts` : gleichzeitiges Drücken von `Shift` und `rechts` ;
- `CMa:` Drücken `Steuerung, Alt` und `ein` gleichzeitig.

Schlüsselsequenzen

"Tastenfolgen" sind Tastenfolgen (oder Tastenakkorde), die nacheinander eingegeben werden müssen. Sie werden durch Trennen aller Tasten- (oder Akkord-) Notationen durch ein Leerzeichen gekennzeichnet.

Beispiele beinhalten:

- `Cx b` : gleichzeitiges Drücken von `Steuerung` und `x` , Loslassen und Drücken von `b` ;
- `Cx Cf` : gleichzeitiges Drücken von `Control` und `x` , dann loslassen von `x` und `f` (da beide Akkorde den `Control-` Modifier enthalten, ist es nicht erforderlich, ihn freizugeben).

Verwenden Sie ESC anstelle von Alt

Tastenakkorde, die den Alt-Modifikator verwenden, können auch als Tastenfolge eingegeben werden, beginnend mit `ESC` . Dies kann nützlich sein, wenn Sie Emacs über eine Remote-Verbindung verwenden, die keine Alt-Tastenakkorde überträgt, oder wenn diese Tastenkombinationen *beispielsweise* von einem Fenstermanager erfasst werden.

Beispiel:

`Mx` kann als `ESC x` eingegeben werden.

Beschreiben der Schlüsselbindungen in Emacs-Lisp-Dateien

Die gleiche Notation, die hier beschrieben wird, kann verwendet werden, wenn Tastenzuordnungen in Emacs-Lisp-Dateien definiert werden.

Beispiel:

(global-set-key (kbd "Cx Cb") 'Puffermenü)
bindet die Tastenfolge `Cx Cb` in den `buffer-menu`

Grundlegende Tastenkombinationen online lesen:
https://riptutorial.com/de/emacs/topic/3436/grundlegende-tastenkombinationen

Kapitel 6: Helm

Examples

Helm über MELPA installieren

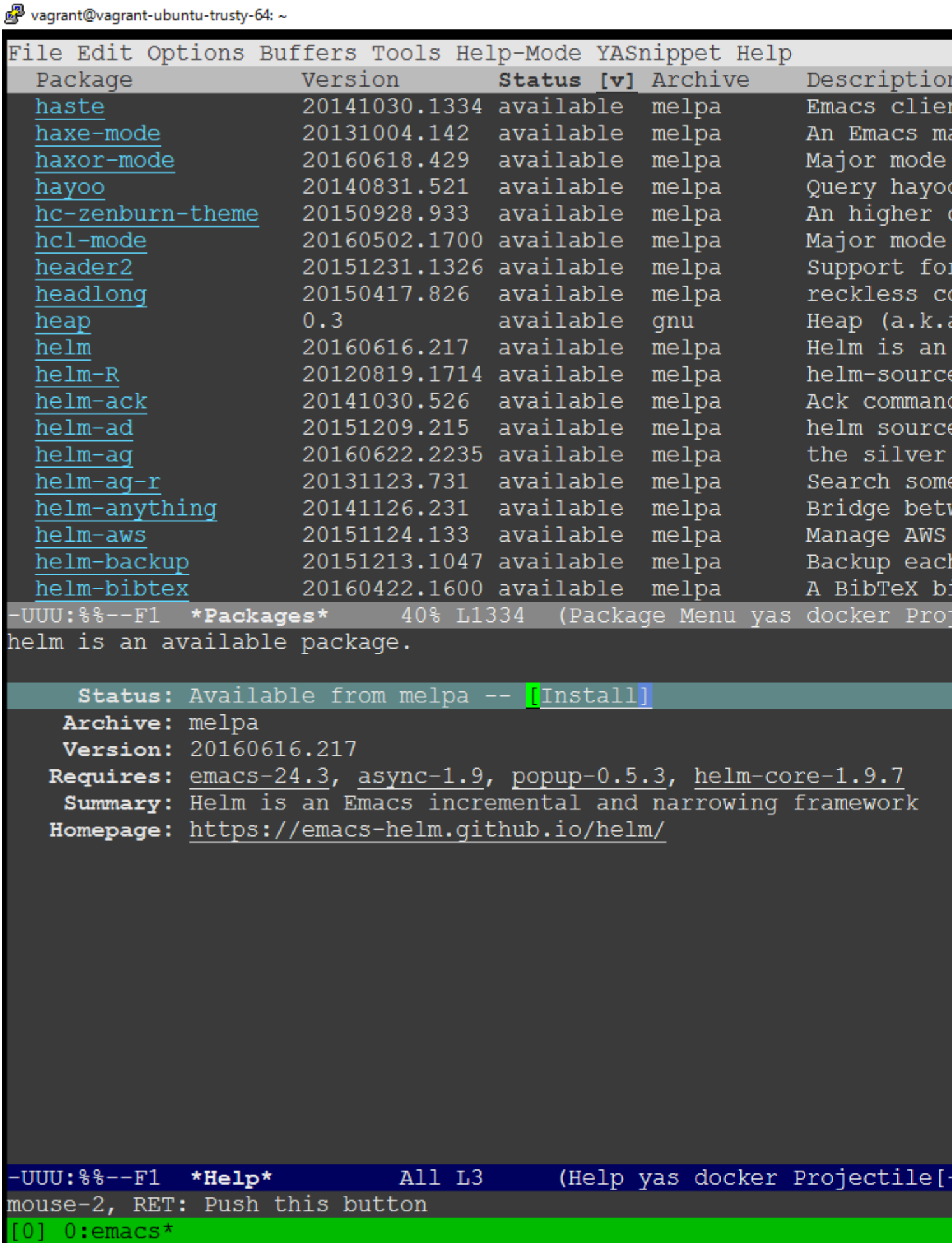
Von emacs 24.4 ist package.el verfügbar. Eine Möglichkeit, helm zu installieren, ist die Installation über MELPA. Fügen Sie zunächst das MELPA-Repository als Paketarchiv hinzu, indem Sie den folgenden Code irgendwo in Ihre `~/.emacs` (oder `~/.emacs.d/init.el`) `~/.emacs.d/init.el` .

```
(require 'package)

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

(package-initialize)
```

Geben Sie anschließend `Mx-` Listenpakete ein, um die `Liste der verfügbaren Pakete` anzuzeigen. Suchen Sie nach dem `helm` , setzen Sie den Cursor auf den `helm` und drücken Sie `RET` . Sie sehen den Paketinformationspuffer. Stellen Sie den Cursor auf `[Install]` und drücken Sie `RET` . Helm wird installiert. Das Paketlistenfenster und das Paketinformationsfenster sind in der folgenden Abbildung dargestellt.


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*

Helm online lesen: https://riptutorial.com/de/emacs/topic/5341/helm

Kapitel 7: Hilfe in Emacs

Bemerkungen

Emacs wird als selbstdokumentierender Editor beschrieben und enthält viele Informationen zur Verwendung im Editor. Zu den Einstiegspunkten dieser Dokumentation gehören ein Tutorial, Informationen darüber, welche Funktionen zu einem bestimmten Thema verfügbar sind, Informationen zu den Verknüpfungen zwischen Tastatureingaben und Funktionen. Der Zugriff auf die Dokumentation erfolgt mit dem Präfix `Ch` , *dh* `Ctrl h` oder `F1` eine Liste mit weiteren Auswahlmöglichkeiten durch Drücken von `?`

Examples

Emacs Tutorial

`Ch t` führt die Funktion `help-with-tutorial` , die einen Puffer öffnet, der ein Tutorial zu den grundlegenden Bearbeitungsfunktionen von emacs enthält, z. B. das Verschieben von Text sowie das Arbeiten mit Dateien, Puffern und Fenstern.

Verfügbare Funktionen und Tastenbindungen

Durch Drücken von `Ch a` wird der Emacs `apropos-command` der Emacs dazu auffordert, nach Wörtern (oder einem regulären Ausdruck) zu suchen. Daraufhin wird ein Puffer mit einer Liste von Namen und Beschreibungen angezeigt, die sich auf dieses Thema beziehen, einschließlich der Tastenzuordnungen für die einzelnen Funktionen, die über Tastenanschläge verfügbar sind.

Durch Drücken von `Ch m` (`describe-mode`) wird ein Puffer angezeigt, der die Haupt- und Nebenmodi beschreibt, einschließlich der Auflistung verfügbarer Funktionen und ihrer Tastenzuordnungen.

Durch Drücken von `Ch b` (`describe-bindings`) wird ein Puffer mit einer Liste aller aktuellen Tastenzuordnungen angezeigt. Die Auflistung enthält globale Bindungen sowie Bindungen für die aktiven Haupt- und Nebenmodi im aktuellen Puffer.

Dokumentation der Schlüsselbindung

`Ch k` führt die Funktion description `describe-key` , die die mit den angegebenen Tastenkombinationen verknüpfte Funktion nachschlägt und eine Beschreibung der Funktion enthält, die ausgeführt wird, wenn diese Tasten gedrückt werden.

`Ch c` führt die Funktion " `describe-key-briefly` , die nur den Funktionsnamen anzeigt, der der angegebenen Tastenfolge zugeordnet ist.

Funktionsdokumentation

`Ch f` führt die Funktion `describe-function` , die Informationen über die Verwendung und den Zweck

einer bestimmten Funktion anzeigt. Dies ist besonders nützlich für Funktionen, die keine zugeordnete Tastenbindung haben, die für die Suche nach Dokumentationen über `Ch k verwendet werden kann` .

Hilfe in Emacs online lesen: https://riptutorial.com/de/emacs/topic/4736/hilfe-in-emacs

Kapitel 8: Lesezeichen in Emacs verwalten

Examples

Wie werden häufig verwendete Dateien mit einem Lesezeichen versehen?

Mit den folgenden Befehlen können Sie in Emacs Lesezeichen erstellen und auf Lesezeichen zugreifen.

Nehmen wir an, Sie bearbeiten eine Datei namens `foobar.org` und nehmen an, dass Sie diese Datei häufig besuchen, um den Inhalt zu bearbeiten / anzuzeigen.

Es ist praktisch, auf diese Datei mit ein paar Tastenkombinationen zuzugreifen, anstatt durch die Dateistruktur (Dired) zu navigieren und die Datei aufzurufen.

Schritte:

1. Öffnen Sie `foobar.org` einmal, indem Sie zu der Datei navigieren (*besuchen Sie die Datei* in Emacs lingo).
2. Während die Datei geöffnet ist, geben Sie `Cx rm ein` . Daraufhin werden Sie aufgefordert, den Lesezeichennamen für die Datei anzugeben. Sagen wir in diesem Fall `foobar` .
3. Schließen Sie die Datei (`Cx k` - kill buffer) - speichern Sie sie bei Bedarf
4. Um die Datei `aufzurufen` , geben Sie einfach `Cx r l` ein. `foobar` wird eine Liste mit `foobar` .
5. Wählen Sie `foobar` und drücken Sie die `Eingabetaste`
6. Verwenden Sie `Mx bookmark-delete` , um alle nicht benötigten Lesezeichen einer Datei zu löschen.

Hinweis: Das Löschen eines Lesezeichens entspricht dem Löschen einer Verknüpfung auf Ihrem Windows-Desktop.
Die Hauptdatei ist an ihrem Speicherort sicher und nur die Liste der Datei aus dem Lesezeichenmenü wird entfernt.

Lesezeichen in Emacs verwalten online lesen:
https://riptutorial.com/de/emacs/topic/5837/lesezeichen-in-emacs-verwalten

Kapitel 9: Magit

Einführung

Magit ist eine Schnittstelle zum Versionskontrollsystem Git, implementiert als Emacs-Paket. Sie können mit Git in Emacs interagieren.

Bemerkungen

Magit ist eine Schnittstelle zum Versionskontrollsystem Git, implementiert als Emacs-Paket. Magit strebt danach, ein komplettes Git-Porzellan zu sein. Obwohl wir (noch) nicht behaupten können, dass Magit jeden einzelnen Git-Befehl umschließt und verbessert, ist er vollständig genug, um selbst erfahrenen Git-Benutzern zu ermöglichen, fast alle ihre täglichen Versionskontrollaufgaben direkt in Emacs auszuführen. Obwohl es viele gute Git-Kunden gibt, verdienen nur Magit und Git den Namen Porzellan.

Beachten Sie, dass Magit sich mit Github verbinden kann (mit Magithub , siehe auch Integration von Github in Emacs), und dass Emacs auch Pakete für Gitlab , Bitbucket und andere enthält.

Examples

Installation

Sie können Magit von MELPA installieren mit:

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

Grundlegende Verwendung: Festschreiben nicht bereitgestellter Änderungen in einem vorhandenen Repo

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

Magit online lesen: https://riptutorial.com/de/emacs/topic/3909/magit

Kapitel 10: Org-Modus

Bemerkungen

Org ist ein Modus zum Verwalten von Notizen, Verwalten von TODO-Listen und zur Projektplanung mit einem schnellen und effektiven Klartext-System. Es ist auch ein Autorensystem mit einzigartiger Unterstützung für die Programmierung und die reproduzierbare Recherche.

Offizielle Website von org Mode

Examples

Markup-Syntax

Org bietet eine vollständige Auszeichnungssprache, die das Strukturieren des Dokuments erleichtert und beim Export in andere Formate (wie HTML oder LaTeX) so genau wie möglich wiedergegeben wird.

Struktur

Dokumenttitel

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

Schneiden

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

Listen

```
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
```

Ankreuzfelder

Jedes Element in einer einfachen Liste kann in ein Kontrollkästchen umgewandelt werden, indem Sie es mit der Zeichenfolge '[]' starten.

```
* 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-Ankreuzfeld
- `Cc Cx Cb` Org-Toggle-Ankreuzfeld
- `MS-` org-insert-todo-Überschrift
- `Cc Cx oder` org-toggle-order-property
- `Cc #` org-update-statistics-cookies

Betonung und Monospace

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

Text im Code und die verbatim-Zeichenfolge wird nicht für die Org-Modus-spezifische Syntax verarbeitet, sondern wird wörtlich exportiert.

Links und Referenzen

Links

Der Org-Modus erkennt URL-Formate und aktiviert sie als anklickbare Links. Links können jedoch explizit folgendermaßen deklariert werden:

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

oder alternativ :

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

Fußnoten

Fußnoten können entweder benannt werden:

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

oder anonym und inline:

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

Grundlegende Schlüsselbindungen

So wechseln Sie die angezeigte Gliederungsebene:

- `Registerkarte` Zykluskonturebene für eine Überschrift
- `Shift-Tab` Umrissebene für das gesamte Dokument

Um durch die `TODO` Zustände zu `TODO` :

- `Umschalt-Rechtspfeil`
- `Pfeil nach links verschieben`

Hierarchieebene für eine Überschrift erhöhen oder verringern

- `Meta-` Rechtspfeil Niedriger machen ("Einzug vergrößern")
- `Meta-` Linkspfeil Höhere Stufe einstellen ("Einzug verringern")

So wechseln Sie die Priorität für eine bestimmte Überschrift:

- `Pfeil nach oben verschieben`
- `Pfeil nach unten verschieben`

So verschieben Sie eine Überschrift nach oben oder unten:

- `Meta-Up-Pfeil`
- `Meta-Abwärtspfeil`

(`Meta` bezieht sich auf verschiedene Tasten auf verschiedenen Tastaturen. Meistens ist es entweder `Alt` oder).

Code-Blöcke

Um einen Code-Block hinzuzufügen, umgeben Sie ihn mit `#+BEGIN_SRC language` und `#+END_SRC` . *Die Sprache* sollte dem Hauptmodus für die betreffende *Sprache* entsprechen. Der `#+BEGIN_SRC emacs-lisp` für Emacs Lisp ist beispielsweise `emacs-lisp-mode` . Schreiben Sie also `#+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
```

Sie können den Codeblock in einem separaten Puffer öffnen, indem Sie `Cc '` (für `org-edit-special`) eingeben. Wenn Sie nicht über den Hauptmodus für die angegebene Sprache verfügen, wird eine Fehlermeldung angezeigt, z. B. `No such language mode: foo-mode` .

Wenn sich der Inhalt, den Sie in den Block `#+BEGIN_EXAMPLE` möchten, nicht in einer Programmiersprache befindet, können `#+END_EXAMPLE` stattdessen `#+BEGIN_EXAMPLE` und `#+END_EXAMPLE` verwenden.

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

Für beide gibt es einfache Vorlagen . Geben Sie am Anfang der Zeile entweder `<s` oder `<e` und drücken Sie `TAB` . Es wird zu einem Block mit Anfangs- und Endmarkierungen für `SRC` bzw. `EXAMPLE` .

Bei diesen Markierungen wird die Groß- und Kleinschreibung nicht `#+begin_src` Sie können stattdessen auch `#+begin_src` usw. schreiben.

Tabellen

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

Um eine Tabelle im Organisationsmodus hinzuzufügen, umgeben Sie Ihre Spalten einfach mit einer Leiste (`|`).

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

Wenn Sie die `Eingabetaste` drücken aus dem Inneren einer Säule, org-Modus wird automatisch eine neue Zeile mit den Stäben erstellen.

- `Tab` und `Return` bewegen sich jeweils in die nächste Zelle oder Zeile (oder erstellen eine neue, falls keine vorhanden ist).
- Sie können die Zeilen und Spalten mit `M-ArrowKey vertauschen`
- `MS-Down` und `MS-Right` erzeugen jeweils eine Zeile (über dem aktuellen) und eine Spalte (links vom aktuellen).
- `MS-Up` und `MS-Left` entfernen jeweils die aktuelle Zeile und die aktuelle Spalte
- `Cc Ich` erstelle ein Trennzeichen

Org-Modus online lesen: https://riptutorial.com/de/emacs/topic/6259/org-modus

Kapitel 11: Paketverwaltung

Examples

Automatische Paketinstallation beim Start von 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 '()))
```

Verweise

- Vergleich von Paket-Repos
- Blogbeitrag über vom Benutzer ausgewählte Pakete in Emacs 25.1

Automatische Paketinstallation mit use-package

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

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

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

Automatische Paketverwaltung mit Cask

Cask ist ein Projektmanagement-Tool, mit dem Sie Ihre lokalen Emacs-Konfigurationen problemlos verwalten können.

Ein Fass ist einfach zu installieren. Sie können entweder den folgenden Befehl in der Befehlszeile ausführen:

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

Wenn Sie sich auf einem Mac befinden, können Sie ihn mit `homebrew` installieren:

```
brew install cask
```

Nach der Installation erstellen Sie eine `Cask` Datei. Cask-Dateien listen alle Paketabhängigkeiten auf, die in Ihrer Konfiguration enthalten sein sollten. Sie können eine neue Cask-Datei im Stammverzeichnis Ihres Verzeichnisses `~/.emacs` .

Sie müssen Cask auch in Ihrem `~/.emacs.d/init.el` . Wenn Sie Homebrew installiert haben, fügen Sie folgende Zeilen hinzu:

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

Oder Sie können den Pfad zum Fass angeben, wenn Sie das Installationsskript verwendet haben:

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

Eine einfache Cask-Datei sieht folgendermaßen aus:

```
(source gnu)
(source melpa)

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

Hier geben wir Quell-Repositorys an, in denen nach Paketen `flx` soll. Dann `flx` wir fest, dass die Pakete `projectile` , `flx` und `flx-ido` installiert werden sollen.

Sobald Sie eine Cask-Datei haben, können Sie alle Abhängigkeiten mit dem folgenden Befehl in der Befehlszeile installieren:

```
cask install
```

Automatische Paketverwaltung mit el-get

el-get ist ein Open Source-Paketverwaltungssystem für GNU Emacs. el-get arbeitet sowohl mit `melpa` als auch mit vielen gängigen Versionskontrollsystemen. Die Dokumentation enthält einen einfachen Selbstinstaller für Ihre `.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 verwaltet `~/.emacs.d/el-get` in einer Verzeichnisstruktur unter `~/.emacs.d/el-get` . Es lädt Definitionen aus `~/.emacs.d/el-get/.loaddefs.el` und verfolgt den Paketstatus mit `~/.emacs.d/el-get/.status.el` . `(el-get 'sync)` installiert oder entfernt Pakete, um den tatsächlichen Maschinenstatus mit dem Paket `.status.el` .

el-get ist selbst gehostet - hier ist der Status von `.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))))
```

Paketverwaltung online lesen: https://riptutorial.com/de/emacs/topic/2414/paketverwaltung

Kapitel 12: Starter-Kits

Bemerkungen

Mit Starter-Kits können *neue Benutzer* Emacs schnell einsetzen und einige der Setup-Hürden umgehen, die aus einem ausgereiften System wie Emacs stammen - eines, das durch jahrzehntelange Evolution gewachsen ist und natürlich historische Probleme hat. *Erfahrene Benutzer* profitieren auch von einer Kit-Konfiguration von Erweiterungen, die von anderen kuratiert werden.

Es ist ein erheblicher Aufwand erforderlich, um eine Reihe von Paketen und Einstellungen beizubehalten, die weiterhin gut zusammenarbeiten, da sich die Pakete im Laufe der Zeit verbessern (oder etwas verrotten). Viele Emacs-Benutzer möchten diese Wartung nicht durchführen, daher wenden sie sich an Starter-Kits. Die Montage und Wartung eines Kits hat eine kleine Ähnlichkeit mit der Verwaltung einer Linux-Distribution.

Themen und Anpassung

Einige Starter-Kits sind themenorientiert. B. für bestimmte Programmiersprachenumgebungen oder für die Musikerstellung oder Emulation eines anderen Editors. Andere zielen darauf ab, eine Küchenspüle mit Bündeln von komfortablen / produktiven Modulen für so viele Situationen oder Sprachen wie möglich bereitzustellen.

Die meisten Starter-Kits enthalten Erweiterungen und Anpassungen. Ein Benutzer überschreibt bestimmte Schlüsselbindungen und -einstellungen und kann Pakete hinzufügen, die noch nicht bereitgestellt wurden.

Beliebte Kits

Es gibt viele Starter-Kits. Theoretisch hat jeder, der seine `~/.emacs.d` eine erstellt. Aber eine Handvoll ist beliebt und wird von einer oder mehreren Personen gut gepflegt. Einige Beispiele (in der Reihenfolge der subjektiven Beliebtheit basierend auf Github-Sternen) umfassen Spacemacs , Prelude , Purcell , Emacs Starter Kit , Magnars und Emacs Live . Weitere Details finden Sie oben im Abschnitt **Beispiele.** Weitere Starter-Kits sind in diesem Wiki aufgeführt .

Ein bemerkenswertes "Micro-Kit" ist Sane Defaults , das eine Handvoll Einstellungen enthält, um einige der von Emacs standardmäßig vorgegebenen Verhaltensweisen der Überraschung für Neuankömmlinge zu entfernen.

Ist ein Starter-Kit erforderlich?

Obwohl es einige Kontroversen um die Verwendung von Starter-Kits gibt, können die Vorteile für viele die Kosten bei der Abstimmung eines dynamischen Emacs-Setups bei weitem überwiegen. Argumente gegen Starter-Kits beziehen sich in der Regel auf: Benutzer, die einige der Nuancen

und das Verhalten von Emacs nicht kennen, sind schwer zu debuggen und lassen Emacs sogar wie einen fremden Editor (Spacemacs) aussehen.

Examples

Spacemacs

Spacemacs ist ein beliebtes Starter-Kit für Emacs. Es verfügt über eine robuste Paketverwaltungslösung und konzentriert sich auf den beliebten bösen Modus von Emacs, der viele der Schlüsselbindungen von vim bereitstellt.

Es wird Spacemacs genannt , weil sie die `Leertaste` als Führer Schlüssel verwendet (die Idee ist ähnlich wie Vim Führer - Taste).

Die Installation ist ziemlich einfach. Laden Sie einfach die Standard-Emacs-Distribution herunter und installieren Sie sie. Anschließend klonen Sie das Git-Repo:

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

Sie können auch eine ZIP- `~/.emacs.d` lokal von der Website herunterladen und einfach nach `~/.emacs.d`

Starten Sie es, nachdem Sie es heruntergeladen (geklont) haben, und drücken Sie die Leertaste, um die interaktive Liste sorgfältig ausgewählter Tastenkombinationen zu durchsuchen. Sie können auch die [?] - Taste des Home-Puffers drücken, um die ersten Tastenkombinationen auszuprobieren.

Auftakt

Prelude ist ein weiteres beliebtes Starter-Kit. Es bietet eine gute Unterstützung für verschiedene vorprogrammierte Programmiersprachen, insbesondere Clojure. Auf * nix-Systemen kann es mit dem folgenden Befehl installiert werden:

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

Emacs-Live

emacs-live ist ein weiteres beliebtes Emacs-Starter-Kit mit zusätzlichem Fokus auf Live-Musik-Codierung mit Oberton .

Sie können es auf zwei Arten installieren:

1. Führen Sie auf * nix-Systemen (z. B. Linux, OSX usw.) den folgenden Befehl in der Befehlszeile aus:

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

2.
   - Laden Sie die Zip von der Github-Seite herunter.
   - Sichern Sie Ihre aktuelle `~/.emacs.d` in Ihrem Home-Verzeichnis
   - `~/.emacs.d` heruntergeladene Zip- `~/.emacs.d` und verschiebe sie nach `~/.emacs.d` :

Scimax

Scimax ist ein Emacs-Starter-Kit für reproduzierbare Forschung, das sich hauptsächlich an Wissenschaftler und Ingenieure richtet. Scimax passt den Org-Modus mit Funktionen an, die das Querverweisen, Exportieren und Codieren (insbesondere Python) vereinfachen.

Installationsanweisungen finden Sie auf der Landingpage des Projekts .

Starter-Kits online lesen: https://riptutorial.com/de/emacs/topic/1960/starter-kits

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Emacs	Adobe, Community, ebpa, Ehvince, eyqs, IntFloat, Jack Henahan, joon, legoscia, mellowmaroon, Michel de Ruiter, Nemanja Trifunovic, omul, pcurry, Prasanna, salotz, squiter
2	Die vielen Varianten von Emacs	pcurry
3	emacs verfügt bereits über eine qualitativ hochwertige, gut organisierte Dokumentation. warum es duplizieren?	erjoalgo
4	Emacs-Nomenklatur	Doug Harris, Francesco, Nikana Reklawyks, Stephen Leppik, Terje D.
5	Grundlegende Tastenkombinationen	Adeel Ansari, Arjun J Rao, boehm_s, Francesco, Idan, Jeff Bencteux, julienc, leeor, Meaningful Username, mellowmaroon, Nikana Reklawyks, Prasanna, SuperBear, Tej Chajed, Terje D.
6	Helm	Yuki Inoue
7	Hilfe in Emacs	mellowmaroon, Terje D., ygram
8	Lesezeichen in Emacs verwalten	Francesco, Prasanna
9	Magit	boehm_s, Ehvince, glallen, mellowmaroon, squiter
10	Org-Modus	Christopher Bottoms, Francesco, legoscia, SuperBear, Wazam
11	Paketverwaltung	Adobe, Kaushal Modi, leeor, pcurry, salotz, squiter
12	Starter-Kits	dangom, Ehvince, Kaushal Modi, leeor, Micah Elliott, Xinyang Li