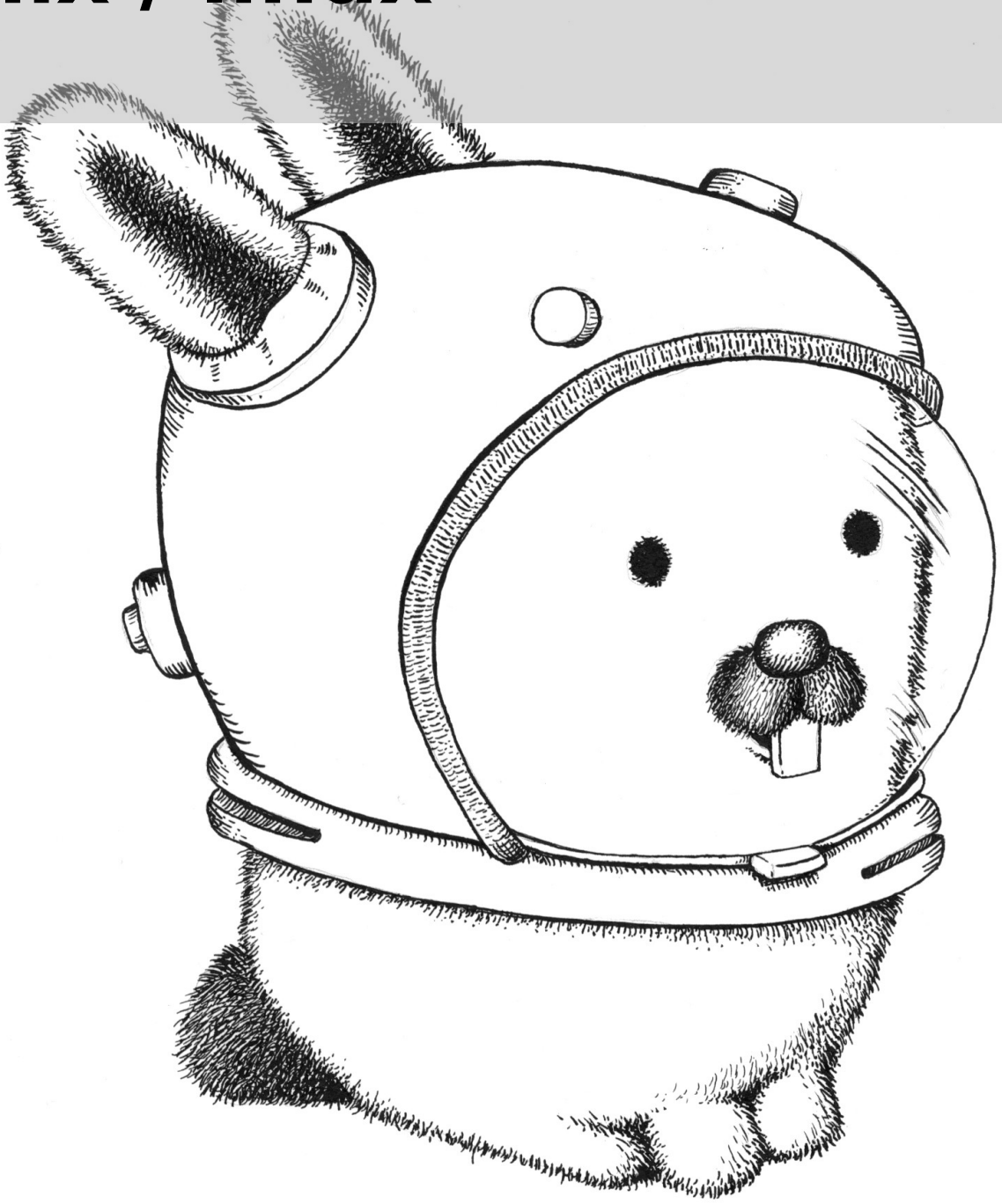
unix / linux



TextEditor vi und emacs

Skript Seite 61

Thema

Ziele:

- Die Editoren vi und emacs kennen lernen
- Textdateien anlegen und verändern können

Editoren

Text-Files bilden die Grundlage der Konfiguration unter Unix. (Beispielsweise sind die Benutzer in folgenden Text-Dateien abgelegt: **`/etc/passwd`**, **`/etc/shadow`**, **`/etc/group`**, **`/etc/gshadow`**).

Um diese Dateien zu bearbeiten – aber auch um Unix selber zu entwickeln – sind Text-Editoren notwendig.

Ähnlich wie bei den Shells gibt es verschiedene Editoren. Unter den Editoren, die sich in einem Terminal einsetzen lassen, haben sich vor allem **`vi`**, deren Klone und **`emacs`** durchgesetzt.

Editoren

Es ist sinnvoll, wenn sie einen 'Terminal'-Editor beherrschen, beispielsweise wenn

- sie an einem Server arbeiten
- sie nur in den Single-User-Mode booten können, werden sie keine X11-Oberfläche haben,
- wenn X11 falsch konfiguriert wurde

sind sie froh, wenn sie die verschiedenen KonfigurationsFiles trotzdem bearbeiten können.

Editoren: vi

Die ersten Unix Version hatten **ed** und **ex** als zeilen-orientierte Editoren im Lieferumfang. Als in den 70er Jahren immer mehr elektronische Terminals anstelle der alten Teletypewriter verwendet wurden, haben die Edito-ren die neuen Funktionen der neuen Terminals adaptiert – speziell dass eine ganze Seite Text in sekundenschnelle angezeigt werden kann. Das Ergebnis war der Editor vi.

Heute gibt es verschiedene Implementationen vom vi.

Editoren visual und emacs

Sowohl der vi als auch der Editor emacs arbeiten nicht direkt mit der Datei, die bearbeitet wird. Beide Editoren laden die Datei in den Speicher und bearbeiten den Text im Speicher.

Bevor der Editor beendet wird, muss der Benutzer – sofern er die Modifikationen behalten will – den Speicher zurück in die Datei speichern.

Der Speicher, der eine Kopie der Datei enthält wird Buffer genannt.

visual

Der **vi** Editor ist der UrEditor der bildschirm-orientierten Editoren. **vi** wurde ursprünglich 1976 von Bill Joy für BSD entwickelt.

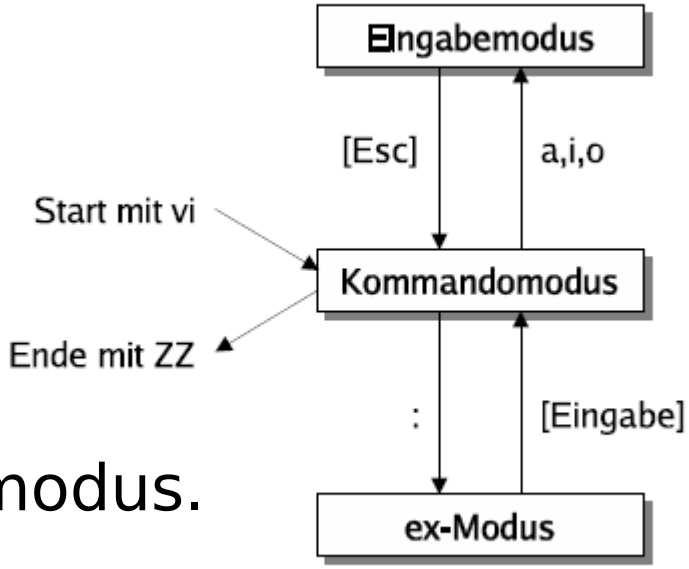
vi benötigt, dass er funktioniert ein Terminal und eine einfache ASCII-Tastatur, die auch noch eine ESCape Taste hat. Es sind keine Funktionstasten notwendig.

neben **vi** gibt es noch **ed** – ein Zeileneditor, vergleichbar mit dem DOS-Program edlin und **sed**. **ed** lässt sich einfach durch Skripte steuern.

visual

Der **vi** kennt 3 verschiedene Modi:

- **Kommando-**
- **Eingabe-**
- **ex-Modus**



Wenn der vi gestartet wird,
befindet er sich im Kommandomodus.

Im **Kommandomodus** stellen alle Tastatur-Eingaben Befehle dar, die nicht auf dem Bildschirm erscheinen und sofort ausgeführt werden (ohne dass die Befehle durch **[enter]** bestätigt werden müssen.

visual

Im **Eingabemodus** werden alle Tastatureingaben als Text in den aktuellen Buffer eingefügt. Mit **[ESC]** kann der Eingabemodus verlassen werden und sie befinden sich wieder im Kommandomodus.

Im **ex-Modus** können längere Befehlszeilen eingegeben werden. Den ex-Modus erreicht man durch die Eingabe von einem Doppelpunkt ":" und wird durch die Eingabe von **[enter]** abgeschlossen.

▶ Wenn man jeweils genau weiss in welchem Modi man sich befindet ist die Bedienung vom vi sehr viel einfacher!

visual Kommandomodus

Befehle im **Kommandomodus**:

Cursor bewegen:

Die Befehle können – wo es Sinn macht – mit einer Zahl *N* kombiniert werden. Der Befehl wird dann *N*-mal ausgeführt

N h|^h	*N* Zeichen nach links
N j |<lf>	*N* Linien nach unten
N k|^p	*N* Linien nach oben
N l |<sp>	*N* Zeichen nach rechts

$	Ans Ende der aktuellen Zeile
^	An den Anfang der aktuellen Zeile
0	Zum ersten Zeichen in der aktuellen Zeile
N +	An den Anfang der nächsten, *N*ten Zeile
N -	An den Anfang der letzten Zeile | *N*ten Zeile davor

N w	*N* Wörter vorwärts
N b	*N* Wörter rückwärts

visual Kommandomodus

Speichern und Beenden:

:q	Verlasse vi – sofern alle Änderungen gespeichert sind
:q!	Verlasse vi – auch wenn Änderungen **nicht** gespeichert sind

:w	Schreibe das File
:w <name>	Schreibe das File <name>
:w!	Schreibe das File, auch wenn es geschützt ist
:w! <name>	Schreibe das File <name> auch wenn es geschützt ist oder schon existiert
:x,y w <name>	Schreibe Zeilen x bis y ins File <name>
:x,y! w <name>	Schreibe Zeilen x bis y ins File <name> auch wenn es geschützt ist oder schon existiert.

:wq | ZZ	Schreibe das File und verlasse vi
:x [<file>]	Schreibe das File <name> und verlasse vi
:x! [<file>]	Schreibe das File <file> und :q

visual Kommandomodus

Anzeige, Scroll Befehle

^g	Zeit den aktuellen Status, Filename an
^l	Refresh von Display
N ^e	Scrollen des Bildschirmfenster zeilenweise nach oben
N ^y	Scrollen des Bildschirmfenster zeilenweise nach unten
N ^d	Scrollt den halben Bildschirm nach unten
N ^u	Scrollt den halben Bildschirm nach oben
N ^f	Scrollt den Bildschirm nach unten
N ^b	Scrollt den Bildschirm nach oben

visual Kommandomodus

Editier Befehle

N ~	ändert das Zeichen unter dem Cursor von Gross- auf Klein-Buchstabe oder umgekehrt.
N J	Fügt die aktuelle und nächste Zeile zusammen (Join)
N >>	rückt N Zeilen um eine Tabstop-Position ein
N <<	rückt N Zeilen um eine Tabstop-Position aus
N x	löscht das | *N* Zeichen unter dem Cursor
N X	löscht das | *N* Zeichen vor vom Cursor
d$ | D	löscht vom Cursor bis zum Ende der Zeile
d^	löscht vom Cursor bis zum Anfang der Zeile
N dw	löscht vom Cursor *N* Wörter
N dd	löscht *N* Zeilen
N r<chr>	ersetzt *N* Zeichen durch das Zeichen <chr>
N .	Wiederhole den letzten Befehl *N*-mal
u	undo des letzten Befehls

visual Kommandomodus

Editier Befehle, Wechsel in den Eingabemodus

i	(keines i) insert vor der aktuellen Cursor Position
I	(grosses i) insert am Anfang der aktuellen Zeile
a	append hinter der aktuellen Cursor Position
A	append am Ende der aktuellen Zeile
o	fügt eine Zeile nach (unterhalb) der aktuellen Zeile ein.
O	fügt eine Zeile vor (oberhalb) der aktuellen Zeile ein.
R	überschreibt die folgenden Zeichen
N cc	ändert die nächsten *N* Zeilen (beginnend bei der aktuellen Zeile)
N C	ändert die nächsten *N* Zeilen (beginnend bei der aktuellen Cursorposition)

visual Eingabemodus

Eingabemodus

[ESC] Verlassen des Eingabemodus

Je nach vi Implementation kann der Cursor mit dem Pfeiltasten und [Home], [End] [PgUp], [PgDn] bewegt werden.

visual ex-Modus

Befehle im **ex-Modus**

/text	Suchen vorwärts innerhalb vom Buffer nach dem Muster "test"
?text	Suchen rückwärts innerhalb vom Buffer nach dem Muster "test" Als Muster können auch Regex-Muster [2] verwendet werden.
N n	Sucht nach dem nächsten vorkommen vom letzten Suchmuster, in der gleichen Suchrichtung.
N N	Sucht nach dem nächsten vorkommen vom letzten Suchmuster, in der umgekehrten Suchrichtung.

:s/alt/neu/	ersetzt das Muster "alt" durch den Text "neu"
:x,y s/alt/neu/	ersetzt das Muster "alt" durch den Text "neu" in den Zeilen x biz y
:1,$<befehl>	Führt den Befehl in jeder Zeile des Puffers aus.
:%<befehl>	Führt den Befehl in jeder Zeile des Puffers aus.
:g/muster/<Befehl>	Führt den Befehl <Befehl> überall dort aus, wo das Muster in der Zeile vorkommt

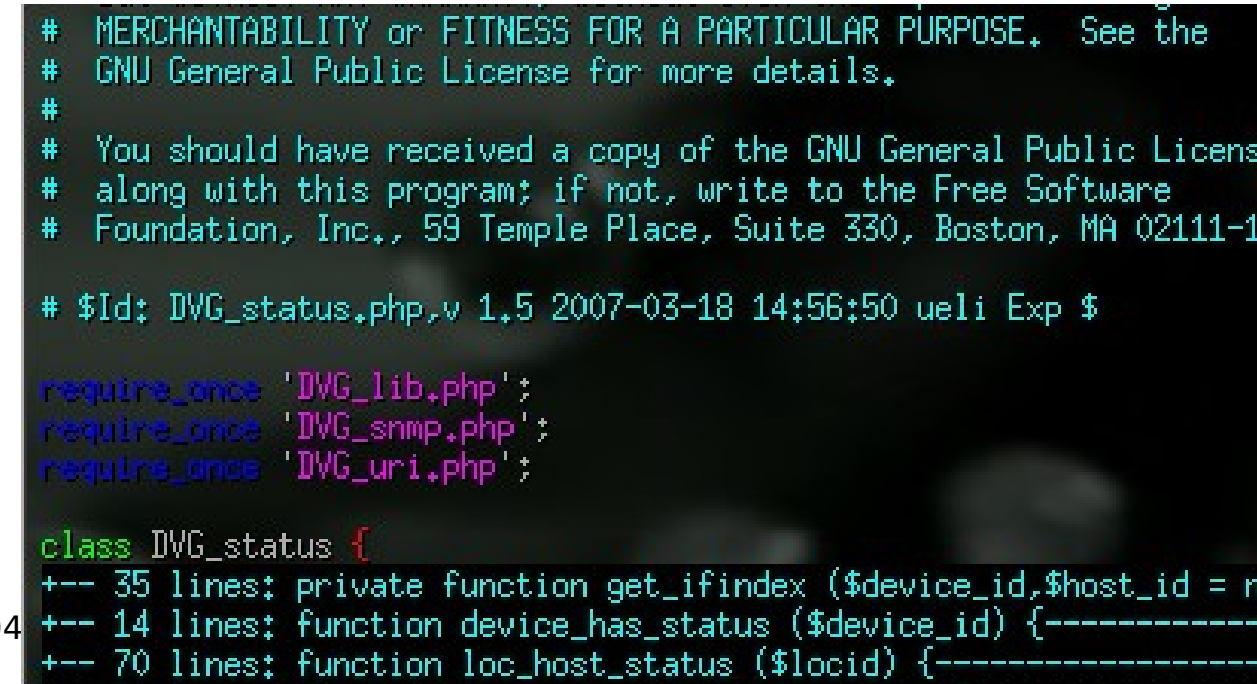
visual

Allgemein:

^V Verhindert, dass das nächste Zeichen als Sonder-/Befehls-Zeichen interpretiert wird. (Funktioniert sogar bei der CISCO-Command-Line)

Die vim Hompage [1] ist sehr viel detaillierter. Auch gibt es Portierungen des vim nach UNIX :), DOS, Windows, Amiga, OS/2, Mac-OS und viele anderen Betriebssysteme.

Andere nette Erweiterungen, die vim bietet, ist beispielsweiseSyntax-Highligthing und Folding des Textes:

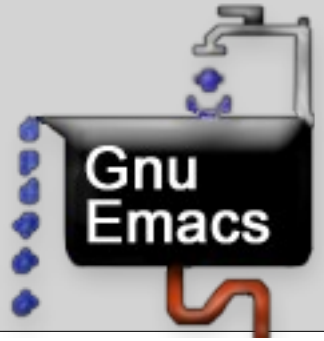


Fragen ?

Übungen

4.1: Welche Texteditoren sind auf Ihrem System installiert?
Wie können Sie das herausfinden

4.2: Finden Sie heraus, wie die interaktive vim-Einführung aufgerufen wird, und arbeiten Sie sie durch.



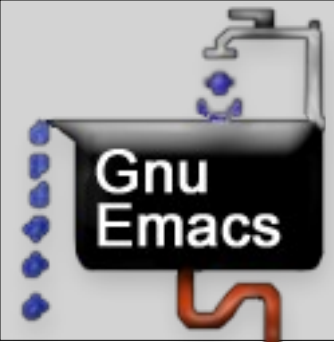
EMACS → Editor **MAC**ro**S**

oder **E**ight **M**egs **A**nd **C**onstantly **S**wapping
oder **E**macs **M**anuals **A**re **C**ryptic and **S**urreal

EMACS entstand 1976 am MIT als Sammlung von Makros für den Editor TECO.

1984 startete Richard Stallman ein Projekt um den EMACS neu zu schreiben. EMACS ist das UR-GNU Program der FSF.

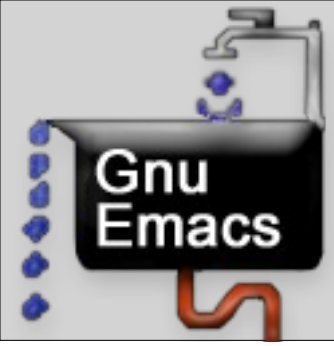
Portierungen von EMACS gibt es für jedes UNIX wie auch andere Betriebssysteme (MAC OS-X, Windows, DOS, ...)



EMACS hat mehr Funktionen als der vi Editor.

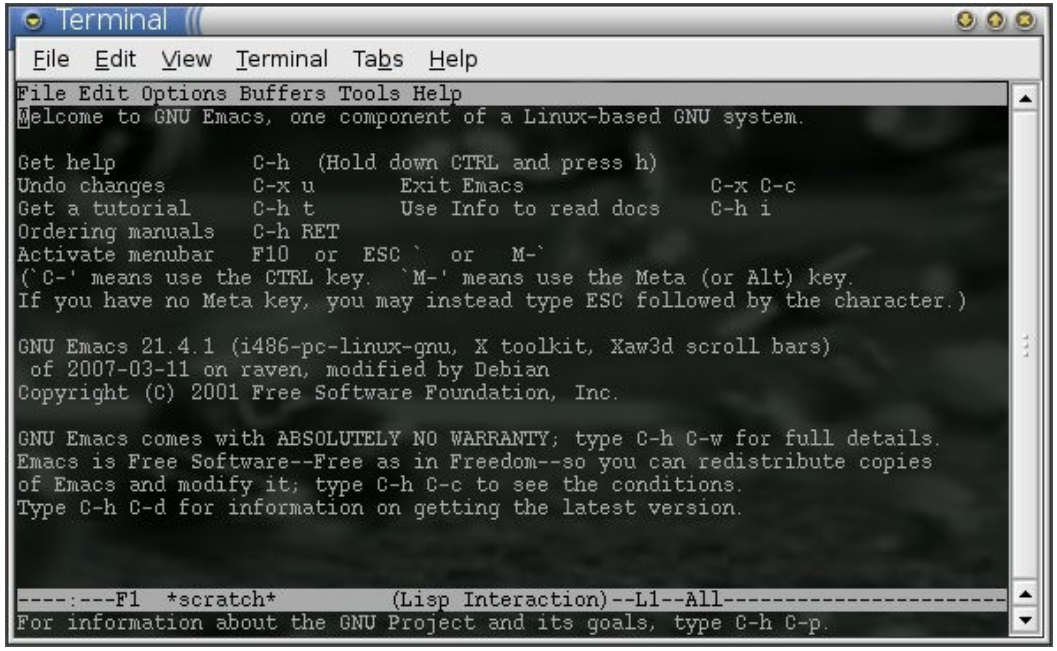
Das zeigt sich auch bei den Befehlen, die sich oft aus mehreren Control-Zeichen oder Meta-Sequenzen zusammen setzen.

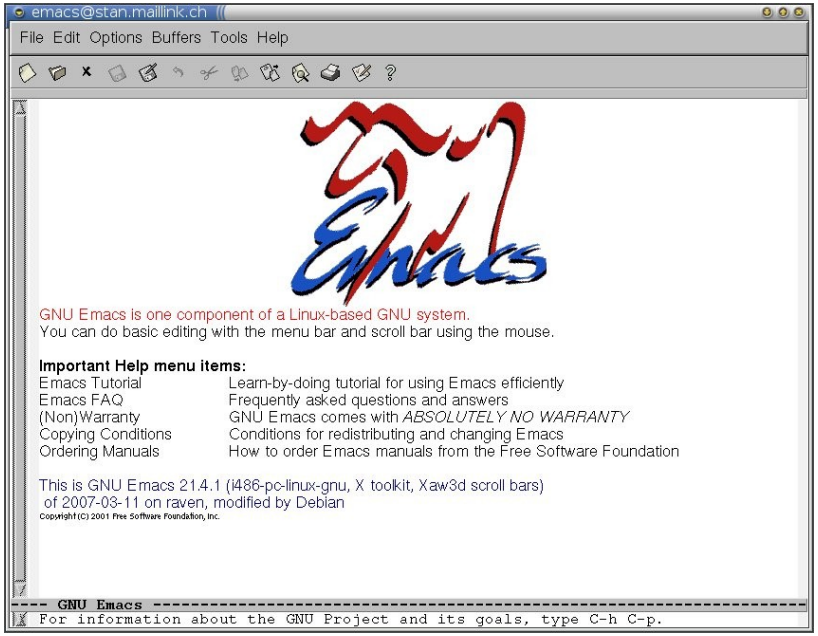
Im EMACS werden die Control-Zeichen und Meta-Sequenzen wie folgt geschrieben:
`[Ctrl]-c → C-c`
`[Meta]-c → [ESC]c` oder `[L-Alt] c`

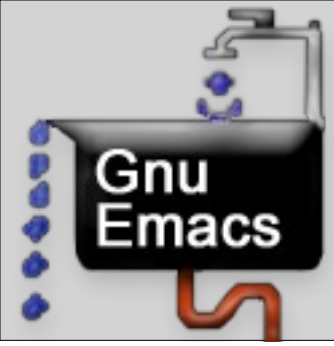


EMACS wird mir dem Befehl **emacs** gestartet. Entdeckt emacs eine graphische Oberfläche so startet er in einem neuen Fenster. An sonst startet **emacs** im aktuellen Terminal.

Optional können auch mehrere Dateien angegeben werden – jede Datei wird in einem eigenen Puffer geöffnet.







Befehle um den emacs zu steuern sind in der Reference-Karte [3] zu finden

Fragen ?

Emacs is a great operating system - it lacks a good editor, though.

Thomer M. Gil

Übungen

4.3: Arbeiten Sie die interaktive Emacs-Einführung durch.
(Die englischsprachige Version erhalten Sie über `[ctrl]-[h][t]` eine deutschsprachige ist über das Help-Menü des Emacs zu erreichen.)

Links

[1] Vim Homepage
http://www.vim.org/

[2] Regex Suchmuster
http://www.vim.org/htmldoc/pattern.html

[3] Reference Cards
vi: http://tnerual.eriogerg.free.fr/vimqrc.pdf
emacs: http://refcards.com/docs/wingb/xemacs/xemacs-refcard-a4.pdf