Kurzanleitung zum vi

Der vi (sprich: "wie ei") ist ein ASCII Editor und auf jedem UNIX-Rechner zu finden. Auch wenn man ihn nicht liebt, sollte man doch ein paar grundlegende Tastengriffe kennen. Die angeführten Befehle sind bei weitem nicht alles was er auf Lager hat.
Diese Seite liegt *nur für mich* auf diesem Server. Die Informationen sind aus verschiedenen Internetseiten zusammengetragen, ohne Rücksicht auf Copyrights.

Inhalt

[weiter]

Einführung

[Seitenanfang] [weiter]

vi Betriebsarten

Der vi kennt drei Betriebsarten:

1. Der *visual mode*, in dem jeder Tastendruck als Kommando interpretiert wird. Dies ist die Standardbetriebsart des vi!
2. Der *ex mode*, in dem über die Kommandozeile komplexe Befehle, wie z.B. Suchen und Ersetzen eingegeben werden können.
3. Der *input mode*, in dem Text eingegeben werden kann.

Der Visual Mode

Der Visual Mode ist wie gesagt die Standardbetriebsart des vi, d.h., man befindet sich direkt nach dem Start des vi darin. Aus allen anderen Betriebsarten kommt man jederzeit durch Drücken der *Escape* Taste zurück.

Die Idee dahinter ist, dass man, solange kein Text eingegeben wird, ohne Hilfe von Maus oder erweiterter Tastatur (Pfeiltasten usw.) in der editierten Datei durch Bewegen des Cursors, Springen und mit Hilfe von Bookmarks navigieren kann. Das ermöglicht Arbeit schnelles Arbeiten auch auf Terminals ohne erweiterte Tastatur.

Der Ex Mode

Der Ex Mode dient dazu, auch komplexere Kommandos oder Makros eingeben zu können, die durch jeweils einfache Tastendrücke im Visual Mode so nicht möglich wären.

Man erreicht den Ex Mode aus dem Visual Mode heraus und zwar durch Drücken von ":". Ein Kommando im Ex Mode wird abgebrochen durch *Escape* oder beendet durch *Enter*.

Der Input Mode

Der Input Mode dient zum Eingeben von Text. Hier werden die normalen Tasten als einzugebende Buchstaben interpretiert. Andere Befehle aus dem Visual Mode, die auf Tasten liegen, die so nicht druckbar sind, wie z.B. *Ctrl-F* und *Ctrl-B* (*PageDown*und *PageUp*), stehen weiterhin zur Verfügung.

Der Input Mode kann durch verschiedene Visual Kommandos eingeleitet werden. Mehr dazu unter Text eingeben.

Der Input Mode wird verlassen durch *Escape*, man landet somit wieder im Visual Mode.

Wichtige vi Kommandos

[Seitenanfang] [weiter]

Konvention

Im Folgenden werden einige oft benutzte vi-Kommandos aufgelistet. Fast alle dieser Kommandos sind *Visual* Kommandos. *Ex* Kommandos werden durch das *":"* am Anfang gekennzeichnet.

Manche (meist *Visual*) Kommandos haben noch ein *[Count]* vorangestellt. Das heißt, dass das Kommando normalerweise einmal, bei einer vorher gedrückten Zahl *n* aber *n*-mal ausgeführt wird.

Starten des vi

Der vi kann mit oder ohne Angabe eines Dateinamens gestartet werden. Wird ein Dateiname angegeben, wird die Datei geladen oder, falls sie noch nicht existiert, neu erzeugt. Aufrufe können sein:

`vi`	Aufruf von vi mit leerem Text-Puffer.
`vi Dateiname`	Datei wird geladen und der Cursor bei der ersten Zeile plaziert.
`vi + Dateiname`	Datei wird geladen und der Cursor bei der letzten Zeile plaziert.
`vi +n Dateiname`	Datei wird geladen und der Cursor bei der *n*-ten Zeile plaziert.
`vi +/Zeichenkette Dateiname`	Datei wird geladen und der Cursor bei der Zeile mit *Zeichenkette* plaziert.

Hinweis: Die meisten vi-Versionen beherrschen auch das Bearbeiten mehrerer Dateien, allerdings unterscheiden sich die Implementierungen meistens.

Moderne Implementierungen wie z.B. *Elvis* können auch den Bildschirm in Fenster unterteilen. Hierzu verweise ich aber auf die Dokumentation des jeweiligen vi-Clones!

Beenden des vi

`:wq`	Speichern und vi verlassen.
`:q`	vi verlassen, falls Datei unverändert
`:q!`	vi verlassen, egal ob Datei verändert oder nicht.
`:w`	Datei speichern

Dateien laden

`e Datei`	*Datei* wird geladen, wenn sie existiert, ansonsten erzeugt.
`:next`	Die nächste Datei wird geladen, falls vi mit mehreren Dateien aufgerufen wurde.
`:prev`	Die vorherige Datei wird geladen, falls vi mit mehreren Dateien aufgerufen wurde.

Cursorbewegungen

`[Count]j`	Den Cursor um eine (bzw. *Count*) Zeile runter. usw.).
`[Count]k`	Den Cursor um eine (bzw. *Count*) Zeile rauf. usw.).
`[Count]l`	Den Cursor um ein (bzw. *Count*) Zeichen rechts. usw.).
`[Count]h`	Den Cursor um ein (bzw. *Count*) Zeichen links.
`[Count]w`	Den Cursor um ein (bzw. *Count*) Wort rechts.
`[Count]b`	Den Cursor um ein (bzw. *Count*) Wort links.
`[Count]h`	Den Cursor um ein (bzw. *Count*) Zeichen links.
`[Count]h`	Den Cursor um ein (bzw. *Count*) Zeichen links.
`[Count]G`	Springe zum Ende der Datei oder, falls *Count* gegeben, zu Zeile *Count*.
Ctrl-f	Page-Down.
Ctrl-b	Page-Up.
`^`	Springe zum Anfang der aktuellen Zeile.
`$`	Springe zum Ende der aktuellen Zeile.

Text eingeben

`i`	(insert), Eingabe vor dem aktuellen Zeichen.
`a`	(append), Eingabe nach dem aktuellen Zeichen.
`I`	(Insert), Eingabe am Anfang der aktuellen Zeile.
`A`	(Append), Eingabe am Ende der aktuellen Zeile.
`o`	neue Zeile und Eingabe nach der aktuellen Zeile.
`O`	neue Zeile und Eingabe vor der aktuellen Zeile.
Ctrl-v	Eingabe eines Steuerzeichens.

Text ändern

[Count]rZeichen	(replace), Änderung des aktuellen Buchstaben in *Zeichen*.
R	(Replace), Überschreibemodus vom aktuellen Buchstaben aus.
cw*Wort*	ersetzt das Wort vor dem Cursor durch *Wort*.
cc*Zeichenkette*	ersetzt die aktuelle oder nächste Zeile durch *Zeichenkette*
J	hängt die der aktuellen folgende Zeile an die aktuelle an und positioniert den Cursor "dazwischen".

Text löschen

[Count]x	1 (bzw. *Count*) Zeichen unter dem Cursor (nach rechts) wird gelöscht.
[Count]X	1 (bzw. *Count*) Zeichen links vom dem Cursor wird gelöscht.
D	löscht von der Cursorposition bis zum Zeilenende.
*[Count]*dd	1 (bzw. *Count*) Zeilen werden gelöscht.
*[Count]*d*[Richtung]*	1 (bzw. *Count*) mal wird in *Richtung* (rechts, links, oben, unten, wortweise, was es eben so gibt!) gelöscht.

Die Zwischenablagen im vi

Der vi hat ziemlich viele Zwischenablagen. Zum Einen sind das die, die beim Löschen automatisch gefüllt werden (man kann auch ohne zu löschen Text in diese Ablagen schieben), zum Anderen gibt es noch 26 weitere, die man selbst belegen kann.

Die sogenannten *Delete Buffer*, also die Zwischenablagen, die durch das Löschen (z.B. durch dd) gefüllt werden, sind wie ein Stack organisiert, d.h., nach jedem Löschen verschiebt sich der Inhalt der Ablagen um Einen nach hinten. Diese Ablagen werden adressiert über die Zifferntasten 1 bis 0, es gibt also 10 Stück.

Außerdem kann noch beliebig Text in die anderen 26 Ablagen schmeißen, die durch die Tasten a-z adressiert sind. Die Vorgehensweise bei beiden Arten ist identisch: Zwischenablage auswählen und dann kopieren, einfügen oder was auch immer.

"*1..0, a..z*	Die Ablage *1..0* bzw. *a..z* für die nächste Aktion auswälen.
*[Count]*y*[Richtung]*	1 (bzw. *Count*) mal wird in *Richtung* (rechts, links, oben, unten, wortweise, was es eben so gibt!) in die aktuelle Zwischenablage kopiert.
*[Count]*yy	1 (bzw. *Count*) Zeilen werden in die aktuelle Zwischenablage kopiert.
Beliebige Löschaktion	Gelöschter Text wird in die aktuelle Zwischenablage kopiert.
p	Der Inhalt der aktuellen Zwischenablage wird hinter dem Cursor eingefügt.
P	Der Inhalt der aktuellen Zwischenablage wird vor dem Cursor eingefügt.

Suchen und Ersetzen

`/Regex`	Suche vorwärts nach dem regulären Ausdruck *Regex*.
`?Regex`	Suche rückwärts nach dem regulären Ausdruck *Regex*.
`n`	Wiederholt das letzte Suchkommando.
`N`	Wiederholt das letzte Suchkommando in die jeweils andere Richtung.
`fZeichen`	Sucht nach *Zeichen* in der aktuellen Zeile vorwärts.
`FZeichen`	Sucht nach *Zeichen* in der aktuellen Zeile rückwärts.
`:%s/Quelle/Ziel/`	Ersetzt *Quelle* im Text *einmal* durch *Ziel*.
`:%s/Quelle/Ziel/g`	Ersetzt *Quelle* im Text *überall* durch *Ziel*.

Bookmarks im vi

`mKey`	Setzt eine Marke an der aktuellen Stelle unter dem Namen der Taste *Key*.
`'Key`	Springt zu der *Zeile* mit der Marke *Key*.
`` `Key ``	Springt zu der *Stelle* mit der Marke *Key*.

Sonstige Goodies

`.`	Wiederholt die letzte Editieraktion, z.B. die Texteingabe seit das letzte Mal der *Visual Mode* verlassen wurde, Suchen und Ersetzen oder was auch immer.
`%` über einer Klammer	Springt mit dem Cursor auf die korrespondierende öffnende bzw. schließende Klammer.
`:tag` *C-Identifier* `Ctrl-]` (auf *C-Identifier*)	Sucht in der aktuellen C-Tags-Datei nach *C-Identifier* und öffnet bei Erfolg einen Buffer mit der entsprechenden Quelltextdatei an der entsprechenden Stelle. Siehe hierzu das Utility *ctags*, das mit jedem vi installiert wird.

Last updated on *Tue, 19 Dec 2000 15:03:26 GMT+0100* © 2000 by Olaf Klein