Visual Editor vi unter Unix

Die wichtigsten Kommandos

Aufruf und Modi des vi

vi *dateiname*	Aufruf zum Editieren einer vorhandenen oder neuen Datei
vi -r *dateiname*	Aufruf zum Wiederherstellen (recover) der Änderungen nach Abbruch einer Editor-Sitzung
vi *datei1 datei2 ...*	Aufruf zum Editieren mehrerer Dateien

vi arbeitet in einem temporären Puffer; Änderungen an der externen Datei werden erst nach Zurückschreiben des Puffers wirksam.

Kommandomodus des vi:	Alle Eingabezeichen im Textfenster werden als Kommandos (zum Bewegen, Ändern im Text etc.) interpretiert. *Eingegebene Kommandos sind nicht sichtbar!*
Texteingabemodus:	Eingegebene Zeichen werden im Textpuffer eingefügt bis Betätigen der ESC-Taste. Korrekturen bei der Eingabe sind mit BACKSPACE (letztes Zeichen), CTRL-w (letztes Wort), CTRL-u (letzte Zeile) möglich.
Kommandozeileneingabe:	Einige Kommandos werden in der Kommandozeile, der letzten Zeile auf dem Bildschirm *sichtbar hinter einem Doppelpunkt* eingegeben, abgeschlossen mit der RETURN-Taste.

Bewegen im Text, Positionieren des Fensters und des Cursors

j oder ↓		Cursor eine Zeile nach unten (gleiche Spalte)
k oder ↑		Cursor eine Zeile nach oben (gleiche Spalte)
l oder →		Cursor eine Position nach rechts (gleiche Zeile)
h oder ←		Cursor eine Position nach links (gleiche Zeile)
RETURN		zum Anfang der nächsten Zeile
CTRL-d	(down)	halbe (Bildschirm-) Seite nach unten
CTRL-u	(up)	halbe (Bildschirm-) Seite nach oben
CTRL-f	(forward)	eine (Bildschirm-) Seite nach unten
CTRL-b	(backward)	eine (Bildschirm-) Seite nach oben
w	(word)	zum Beginn des nächsten Wortes (1)
W		zum Beginn des nächsten Wortes (2)
b	(back)	zum Beginn des vorherigen Wortes (1)
B		zum Beginn des vorherigen Wortes (2)
e	(end)	zum Ende des nächsten/aktuellen Wortes

(1) Trennung durch Leerzeichen oder Satzzeichen wie . , -

(2) Trennung nur durch Leerzeichen

Den obigen Kommandos kann eine Zahl *n* vorangestellt werden zwecks *n*-maliger Ausführung der entsprechenden Bewegung.

^		zum ersten Nicht-Leerzeichen der Zeile
$		zum letzten Zeichen der Zeile
0		zur ersten Spalte der aktuellen Zeile
)		zum Begin des nächsten Satzes
(		zum Begin des vorherigen Satzes
}		zum Begin des nächsten Absatzes
{		zum Begin des vorherigen Absatzes
f*c*		zum nächsten Zeichen *c* in der Zeile
t*c*		zum nächsten Leerzeichen vor Zeichen *c*
F*c* , T*c*		dito, aber von rechts nach links
;		Wiederholen des letzten f, t, F oder T
H	(Home)	bewegt Cursor zur linken oberen Ecke des aktuellen Bildschirms
M	(Middle)	bewegt Cursor in die mittlere Zeile des aktuellen Bildschirms
L	(Last)	bewegt Cursor in die letzte Zeile des aktuellen Bildschirms
*n*G	(Goto)	Sprung zur Zeile *n*, 1G zum Dateianfang
G		Sprung zur letzten Zeile der Datei
*n*l		Positioniert auf Spalte *n* der Zeile
CTRL-g		Aktuelle Zeilennummer etc. anzeigen
m*c*		Zeile mit Markierung *c* versehen
'*c*		zu Zeile mit Markierung *c* springen

Sichern der Datei und Beenden des Editors

:w	(write)	Sichern der bearbeiteten Datei
:w dateiname		Sichern unter ggf. anderem Namen
:q	(quit)	Verlassen von vi (nur nach Sicherung möglich)
:q!		Verlassen von vi ohne Sichern der editierten Datei
ZZ oder :wq		Verlassen von vi mit Sichern der editierten Datei

Einfügen von Text

Die folgenden Kommandos wechseln in den Texteingabemodus bis Eingabe von ESC.

i (insert)	Einfügen vor der aktuellen Cursor-Position
I	Einfügen am Anfang der aktuellen Zeile
a (append)	Anfügen von Text hinter der aktuellen Cursor-Position
A	Anfügen von Text hinter dem Ende der aktuellen Zeile
o (open)	Einfügen von Text in einer neuen Zeile hinter der aktuellen
O	Einfügen von Text in einer neuen Zeile vor der aktuellen

Löschen von Text

x		Löschen des Zeichens, auf dem der Cursor steht
X		Löschen des Zeichens vor der Cursor-Position
D	(delete)	Löschen des Rests der Zeile ab dem Zeichen, auf dem der Cursor steht
dd		Löschen der aktuellen Zeile
*n*dd		Löschen von *n* Zeilen ab der aktuellen
d *bewegungskommando*		Löschen ab Cursor-Position bis zu der durch das *bewegungskommando* erreichten Stelle, zum Beispiel:
dG		Löschen ab Cursor-Position bis zum Dateiende
dW		Löschen ab Cursor-Position bis zum Wortende
d)		Löschen ab Cursor-Position bis zum Satzende
u	(undo)	letzte Löschung rückgängig machen

Ändern von Text

r*c*		Ersetzen des Zeichens, auf dem der Cursor steht, durch *c*
R		Überschreiben des Zeichens, auf dem der Cursor steht, und der folgenden durch Eingabe bis ESC
C	(change)	Ändern des Rests der Zeile ab dem Zeichen, auf dem der Cursor steht, durch Eingabe bis ESC
cc		Überschreiben der aktuellen Zeile durch Eingabe bis ESC
c *bewegungskommando*		Ändern ab Cursor-Position bis zu der durch das *bewegungskommando* erreichten Stelle durch Eingabe bis ESC
cG		Ändern ab Cursor-Position bis zum Dateiende
cW		Ändern ab Cursor-Position bis zum Wortende
c)		Ändern ab Cursor-Position bis zum Satzende
J	(join)	folgende Zeile an die aktuelle anhängen
u	(undo)	letzte Änderung rückgängig machen

Suchen und Ersetzen

/*string*		Suchen nach der Zeichenfolge *string* Richtung Dateiende und Positionieren auf nächstes Vorkommen
?*string*		dito; Richtung Dateianfang
n	(next)	Weitersuchen mit letztem *string*
N		Weitersuchen mit letztem *string* mit Wechsel der Suchrichtung

Allgemeiner kann statt der unmittelbar zu suchenden Zeichenfolge ein *regulärer Ausdruck* angegeben werden, in dem folgende Zeichen eine Sonderbedeutung haben:

^ (am Anfang)	das folgende wird genau am Zeilenanfang gesucht
$ (am Ende)	das vorhergehende wird genau am Zeilenende gesucht
\	hebt Sonderbedeutung des folgenden Zeichens auf außer bei runden Klammern
. (Punkt)	steht für irgendein (beliebiges) Zeichen

[*abc...*] , [*a-b*]		steht für eines der aufgezählten Zeichen *abc...* bzw. aus dem Bereich *a* bis *b*
[^*abc...*] , [^*a-b*]		steht für jedes Zeichen außer den aufgezählten *abc...* bzw. denen aus dem Bereich *a* bis *b*
*		das vorherige Zeichen oder eines der vier letzten Gebilde kann null-mal oder mehrfach auftreten
\(*reg. Ausdruck*\)		klammert Teilausdrücke für spätere Referenz bei Ersetzungen
: *von*,*bis* s/*alt*/*neu*/*opt*		Im Zeilenbereich *von* bis *bis* (z.B. $ für die letzte Zeile) die Zeichenfolge den regulären Ausdruck *alt* durch *neu* ersetzen; ohne Zusatz nur erstes Auftreten in Zeile, mit Option g: jedes Auftreten, mit Option c: manuelle Bestätigung (y RETURN) vor jeder Änderung In *neu* können mit \1, \2 usw. geklammerte Teile in *alt* referiert und übernommen werden.
u	(undo)	letzte Ersetzung rückgängig machen

Beispiel:

```
:1,$s/^.*\([A-Z].*\)$/\1/
```

löscht in allen Zeilen alles vor dem ersten Großbuchstaben, Zeilen ohne Großbuchstaben bleiben unverändert.

Text verschieben und kopieren, Register des vi

y *bewegungskommando*	Kopieren ab Cursor-Position bis zu der durch das *bewegungskommando* erreichten Stelle in ein vi-internes "Register" (englisch auch cut buffer). Beispiele
yG	Kopieren bis zum Dateiende
yW	Kopieren bis zum Wortende
y)	Kopieren bis zum Satzende
yy oder Y	Kopieren der gesamten aktuellen Zeile
*n*Y	Kopieren von *n* Zeilen ab der aktuellen Zeile
p	Inhalt des Registers (aus letztem y- oder d-Kommando) hinter die aktuelle Cursor-Position bzw. Zeile kopieren
P	Inhalt des Registers (aus letztem y- oder d-Kommando) vor die aktuelle Cursor-Position bzw. Zeile kopieren

Das genannte Register wird durch jedes y-(yank-) oder d-Kommando überschrieben. Es können bei den Kommandos y, Y, p und P aber auch sog. *benannte* Register durch Voranstellen von "*a* mit einem der 26 Kleinbuchstaben verwendet werden; durch Verwendung des entsprechenden Großbuchstaben bei y, Y wird an das Register angehängt, nicht überschrieben. Bei p und P kann außerdem durch Voranstellen von "1 bis "9 auf die letzten gelöschten Texte zugegriffen werden.

Bearbeiten mehrerer Dateien

:n	(next)	Wechsel in zur nächsten Datei, wenn vi mit mehreren Dateinamen aufgerufen wurde
:n!		dito, ohne Sichern der aktuellen Datei
:w	(write)	Sichern der bearbeiteten Datei
:r *dateiname*	(read)	Einlesen einer Datei hinter aktuelle Zeile
:e *dateiname*	(edit)	Editieren einer anderen Datei

:e! *dateiname*	dito, ohne Sichern der aktuellen Datei
:e#	Editieren der vorherigen Datei

Durch Verwendung der benannten Register aus dem vorherigen Abschnitt können Texte zwischen mehreren Dateien kopiert werden; der "implizite" wird bei Dateiwechsel gelöscht

Integration von Shell-Kommandos

:! *shell-kommando*	Ausführen des eingegebenen Shell-Kommandos; nach Beendigung muß einmal zusätzlich RETURN betätigt werden zur Rückkehr in den Kommandomodus von vi
! *bewegungskommando*	Von vi wird in der Kommandozeile ein Shell-Kommando eingelesen (bis RETURN); diesem wird der Text ab Cursor-Position bis zu der durch das Bewegungskommando erreichten Position als Standardeingabe übergeben und der Text durch die Ausgabe des Kommandos ersetzt

Persönliche Einstellungen mit dem set-Kommando

:set all	Zeigt Einstellungen aller Optionen
:set *option*	Option *option* einschalten
:set no*option*	Option *option* ausschalten
:set *option*=*wert*	Option *option* auf den Wert *wert* setzen (abhängig von der Option)

Auswahl einiger Optionen (mit Abkürzungen):

autoindent	ai	Automatisches Einrücken einer neuen Zeile wie die letzte im Texteingabemodus; CTRL-d rückt die Eingabeposition wieder vor.
showmode		Anzeige des Modus (z. B. INPUT MODE oder I im Texteingabemodus)
showmatch	sm	Zeigt im Texteingabemodus zugehörige öffnende Klammer bei Eingabe einer schließenden (oder klingelt)
mesg		no mesg unterdrückt Meldungen von z. B. write-Kommandos auf das eigene Terminal während der Editorsitzung
number	nu	Anzeige der Zeilennummern vor allen Textzeilen
ignorecase	ic	Bei Suchkommandos (Vergleich mit regulären Ausdrücken) werden Groß- und Kleinbuchstaben im Text nicht unterschieden
wrapscan	ws	Suche wird am Dateianfang fortgesetzt, wenn Dateiende erreicht wurde, und umgekehrt
report=*n*		Setzt die Zahl für gelöschte, kopierte Zeilen auf *n*, ab wann eine Meldung erfolgt

Mit set vorgenommene Einstellungen gelten zunächst nur für die jeweilige Sitzung. Sie können aber zu dauerhaften Voreinstellungen gemacht werden, wenn die entsprechenden set-Kommandos in einer Datei mit Namen `.exrc` im Home-Directory *ohne ':' vor den Kommandos* hinterlegt werden.
Beispiel: `set ic ms ws`

In der .exrc - Datei können weiter definiert werden:

Abkürzungen für die Texteingabe `abbr` *`wort ersetzung`*
Beispiel: `abbr hhu Heinrich-Heine-Universitaet`

Definitionen sog. Makros: `map` *`zeichen kommandofolge`*
Beispiel: `map X :set nu` <Ctrl-v> <Enter> (Ctrl-v ist nicht sichtbar!)

Die Datei .exrc ist *nur dann wirksam*, falls sie das Kommando `set exrc` enthält (Default `noexrc`).