FORSCHUNGSZENTRUM JÜLICH GmbH KFA

Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61–6402
Informationszentrum, Tel. (02461) 61–6658

Referenzkarte

FZJ-ZAM-RFK-0010

U. Schmidt

1. 8. 94

vi Referenzkarte

Erläuterungen zur Syntax

Zeichenketten oder Buchstaben in **Fettdruck** müssen in der Kommandozeile des vi so eingegeben werden, wie sie erscheinen. Begriffe in *Kursivschrift* sind durch die tatsächlichen Namen, Buchstaben oder Ziffern zu ersetzen. Befehle innerhalb des vi sind in Pfeile <...> eingeschlossen. Sie sind beim Eintippen nicht sichtbar. C-c heißt, daß die Tastenkombination <Ctrl>-Key zusammen mit Zeichen 'c' zu drücken ist.

Starten von vi

vi	Starte vi
vi *filename*	Editiere die Datei *'filename'*
vi *+n filename*	Editiere die Datei *'filename'* und positioniere den Cursor auf die n-te Zeile
vi -r *filename*	Restauriere die Datei *'filename'* nach einem Systemabsturz
vi *fn1 fn2 ...*	Editiere mehrere Dateien *'fn1', 'fn2', ...*
vi -R *filename*	Liste die Datei *'filename'* Entspricht dem Befehl: **view** *filename*

Sichern der Änderungen und/oder Beenden von vi

:q!	Wegwerfen der Änderungen und Verlassen von vi
:q	Verlassen von vi , wenn alle Änderungen zuvor gesichert wurden
:wq	Abspeichern der Änderungen und Verlassen von vi
ZZ	Abspeichern der Änderungen und Verlassen von vi
:w	Abspeichern der Änderungen auf Originaldatei
:w *newname*	Abspeichern der geänderten Datei auf eine neue Datei *'newname'*

Blättern im vi

C-F bzw. **C-f**	Vorwärtsblättern um eine ganze Seite (Forward)
C-D bzw. **C-d**	Vorwärtsblättern um eine halbe Seite (Down)
C-E bzw. **C-e**	Vorwärtsblättern um eine Zeile
C-B bzw. **C-b**	Zurückblättern um eine ganze Seite (Backward)
C-U bzw. **C-u**	Zurückblättern um eine halbe Seite (Up)
C-Y bzw. **C-y**	Zurückblättern um eine Zeile
< 1G >	Zurückblättern an den Anfang der Datei
< G >	Vorwärtsblättern an das Ende der Datei

Cursor Positionierung

Neben den Cursor-Tasten ← , → , ↑ , ↓ können folgende Befehle zur Positionierung des Cursors benutzt werden.

... *innerhalb der Datei*

:*n* bzw. ***n*G**	Setzt Cursor an den Anfang der n-ten Zeile
:1 bzw. **< 1G >**	Setzt Cursor an den Anfang der ersten Zeile
< G >	Setzt Cursor an den Anfang der letzten Zeile
< :*n*| **>**	Setzt Cursor auf die n-te Spalte der Zeile

... *innerhalb des Windows*

< H >	Setzt Cursor an den Anfang der obersten Zeile (High)
< *n*H >	Setzt Cursor an den Anfang der n-ten Zeile von oben
< M >	Setzt Cursor an den Anfang der mittleren Zeile (Middle)
< L >	Setzt Cursor an den Anfang der untersten Zeile (Low)
< *n*L >	Setzt Cursor an den Anfang der n-ten Zeile von unten

... *per Zeichen*

< h > bzw. **C-H**	
< BACKSPACE >	Setzt Cursor um ein Zeichen nach links (←)
< l >	
< SPACE >	Setzt Cursor um ein Zeichen nach rechts (→)
< F*c* **>**	Setzt Cursor nach links (←) auf das Zeichen *c* (Find)
< f*c* **>**	Setzt Cursor nach rechts (→) auf das Zeichen *c*
< T*c* **>**	Setzt Cursor nach links (←) auf das Zeichen hinter dem Zeichen *c* (Tight)
< t*c* **>**	Setzt Cursor nach rechts (→) auf das Zeichen vor dem Zeichen *c*

... *per Wort*

< e >	Setzt Cursor an das Wortende (End)
< b >	Setzt Cursor an den Wortanfang (Begin)
< w >	Setzt Cursor an den Anfang des nächsten Wortes

... *per Zeile*

< j > bzw. **C-N**	Setzt Cursor um eine Zeile nach unten (↓) (Next)
< k > bzw. C-P	Setzt Cursor um eine Zeile nach oben (↑) (Previous)
< 0 >	Setzt Cursor an den Zeilenanfang (|←)
< $ >	Setzt Cursor an das Zeilenende (→|)
< + >	Setzt Cursor an den Anfang der nächsten Zeile
< – >	Setzt Cursor an dan Anfang der vorhergehenden Zeile

... *per Satz oder Abschnitt*

< (>	Setzt Cursor an den Anfang des Satzes
<) >	Setzt Cursor an den Anfang des nächsten Satzes
< { >	Setzt Cursor an den Anfang des Abschnitts
< } >	Setzt Cursor an den Anfang des nächsten Abschnitts

Eingeben von Text (Input Mode)

Die folgenden Befehle versetzen in den Input-Mode, in dem jedes Zeichen so genommen wird, wie es eingegeben wird. Zurück in den Command-Mode kommt man durch Drücken der **<ESC>**-Taste.

< i >	Gibt Eingabe vor (←) dem Cursor frei (Input)
< I >	Gibt Eingabe am Zeilenanfang (|←) frei
< a >	Gibt Eingabe hinter (→) dem Cursor frei
< A >	Gibt Eingabe am Zeilenende (→|) frei (Append)
< o >	Erzeugt neue Zeile hinter (↓) der aktuellen Zeile
< O >	Erzeugt neue Zeile vor (↑) der aktuellen Zeile

Befehle innerhalb des Input Mode

< ESC >	Beendet den Input Mode
< BACKSPACE >	Löscht das aktuelle Eingabezeichen
< ENTER >	Erzeugt eine neue Zeile

Löschen von Text

< *n*x >	Löscht *'n'* Zeichen beginnend beim Cursor (→)
< *n*X >	Löscht *'n'* Zeichen vor (←) dem Cursor
< *n*dd >	Löscht *'n'* Zeilen ab der aktuellen Zeile (↓)
< D >	Löscht bis zum Zeilenende
< d*M* **>**	Löscht bis zur angegebenen Maßeinheit *'M'*

Beispiele:

d0	Löscht bis zum Zeilenanfang
dW	Löscht bis zum Ende des leer begrenzten Wortes
dG	Löscht bis zum Dateiende
d1G	Löscht bis zum Dateianfang

Ändern von Text

< r >	Ersetzt das Zeichen unterm Cursor durch genau ein Zeichen (Replace)
< R >	Ersetzt beliebig viele Zeichen in Folge ab dem Cursor. Beenden mit <ESC>.
< s >	Ersetzt genau ein Zeichen durch beliebig viele Zeichen (Substitute). Beenden mit <ESC>.
< S >	Ersetzt die aktuelle Zeile
< C >	Ersetzt den Rest der Zeile ab dem Cursor durch beliebig viele Zeichen (Change)
< ~ >	Wandelt das Zeichen von Groß- nach Kleinschreibung um und umgekehrt

Wiedereinfügen von Text

Nachdem Text gelöscht oder gemerkt wurde, kann er aus dem Puffer an gleicher oder anderer Stelle wieder in die Datei eingefügt werden.

< p >	Einfügen des Textes hinter dem Cursor bzw. hinter die aktuelle Zeile
< P >	Einfügen des Textes vor dem Cursor bzw. vor die aktuelle Zeile

Verschieben und Kopieren von Text

Zum Verschieben (move) bzw. Kopieren (copy) von Daten sind immer zwei Befehlsschritte notwendig: erstens Definition des Datenbereiches der verschoben bzw. kopiert werden soll und zweitens Definition des Ziels.

Verschieben

Beim Verschieben von Text werden die Originaldaten gelöscht und dabei im Puffer zwischengespeichert. Dazu können die unter Löschen aufgeführten Befehle benutzt werden. Nach dem Löschen muß der Puffer mit den Befehlen zum Wiedereinfügen von Text an die gewünschte Stelle geschrieben werden, bevor er durch andere Befehle zerstört wird.

Beispiele:

dd und P	Verschiebt eine Zeile vor die aktuelle Zeile
3dd und p	Verschiebt drei Zeilen hinter die aktuelle Zeile
5x und p	Verschiebt 5 Zeichen hinter die Cursorposition

Eine andere Möglichkeit bietet der **move**-Befehl, bei dem aber die Zeilennummern bekannt sein müssen:

:m*#*	Verschiebt die aktuelle Zeile hinter Zeile *'#'*
:*a,b***m***#*	Verschiebt die Zeilen *'a'* bis *'b'* hinter Zeile *'#'*

Beispiele:

:m3	Verschiebt die aktuelle Zeile hinter Zeile 3
:1,3m$	Verschiebt die erste bis dritte Zeile ans Ende der Datei

Kopieren

Beim Kopieren bleiben die Originaldaten erhalten, d.h. der zu kopierende Text muß in einen Puffer übertragen werden, aus dem er dann in die Datei wiedereingefügt werden kann. Die Übertragung der Daten in den Puffer geschieht mit dem Yank-Befehl:

< Y >	Merke die aktuelle Zeile im Puffer
< *n*yy >	Merke ab der aktuellen Zeile *'n'* Zeilen im Puffer
< Y*M* **>**	Merke die aktuelle Zeile bis zur angegebenen Maßeinheit *'M'* im Puffer

Beispiele:

Y und P	Kopiert eine Zeile vor die aktuelle Zeile
3yy und p	Kopiert drei Zeilen hinter die aktuelle Zeile
YG und p	Kopiert den Text bis zum Dateiende hinter die aktuelle Zeile

Eine andere Möglichkeit bietet der **copy**-Befehl, bei dem aber die Zeilennummern bekannt sein müssen:

:co*#*	Kopiert die aktuelle Zeile hinter Zeile *'#'*
:*a,b***co***#*	Kopiert die Zeilen *'a'* bis *'b'* hinter Zeile *'#'*

Beispiele:

:co3	Kopiert die aktuelle Zeile hinter Zeile 3
:1,3co$	Kopiert die erste bis dritte Zeile ans Ende der Datei

Suchen von Text

< /*string* **>**	Sucht nach dem nächsten Vorkommen von *'string'*
< ?*string* **>**	Sucht rückwärts nach dem vorherigen Vorkommen von *'string'*
< n >	Wiederholt den letzten Suchvorgang
< N >	Wiederholt den letzten Suchvorgang in umgekehrter Richtung
< / >	Wiederholt den letzten Suchvorgang in Richtung Dateiende (vorwärts)
< ? >	Wiederholt den letzten Suchvorgang in Richtung Dateianfang (rückwärts)

Suchen und Ersetzen (global change)

< :s/*str1***/***str2***/g >**	Ersetzt in der aktuellen Zeile alle Vorkommen von *'str1'* durch *'str2'*. Wird **'g'** für global weggelassen, wird nur das erste Vorkommen ersetzt.
< :*a,b* **s/***str1***/***str2***/g >**	Ersetzt von Zeile *'a'* bis Zeile *'b'* alle Vorkommen von *'str1'* durch *'str2'*. Wird **'g'** für global weggelassen, wird nur das erste Vorkommen in jeder Zeile ersetzt. Beispiel: '**:1,$ s/resource/Ressource/g**' ersetzt in der gesamten Datei alle Vorkommen von resource durch Ressource. Anstelle von '1,$' bezeichnet auch das %-Zeichen die gesamte Datei. Der Befehl '**:% s/resource/Ressource/g**' ist somit gleichbedeutend.
:g */str/ command*	Sucht in der gesamten Datei nach *'str'* und führt in der Zeile, in der *'str'* vorkommt den Befehl *'command'* aus. Beispiel: '**:g /resource/ s//Ressource/g**' ersetzt in der gesamten Datei alle Vorkommen von resource durch Ressource.
:v */str/ command*	Führt in jeder Zeile, in der *'str'* nicht vorkommt den Befehl *'command'* aus.
< & >	Wiederholt den letzten Ersetzungsbefehl

Arbeiten mit mehreren Dateien

:r *filename*	Fügt den Inhalt von Datei *'filename'* hinter der aktuellen Zeile der editierten Datei ein (Read)
:w! *filename*	Schreibt die gesamte editierte Datei auf die Datei *'filename'* (Overwrite)
:*a,b***w!** *filename*	Schreibt Zeile 'a' bis 'b' der editierten Datei auf die Datei *'filename'*. Beispiel: mit ':10,15w! TT' wird Zeile 11 bis 15 auf die Datei TT geschrieben.
:*a,b***w>>** *filename*	Hängt Zeile 'a' bis 'b' der editierten Datei an die Datei *'filename'*.
:n	Editiert die nächste Datei aus der Argumentliste
:e *filename*	Editiert zwischenzeitlich eine andere Datei. Zurück kommt man mit **:e#**.
C-G	Zeigt den Namen und die Position in der aktuellen Datei an.

Ausführen von Shell-Befehlen

:sh	Startet eine Subshell. Zurück in den Editor kommt man mit **C-d** oder **exit**.
:!*command*	Startet Shell und führt den Befehl *'command'* aus.
:r!*command*	Startet Shell, führt den Befehl aus und fügt die Ausgabe von *'command'* an der aktuellen Cursorposition ein.
!!*command*	Startet Shell, führt den Befehl aus und ersetzt die aktuelle Zeile der editierten Datei durch die Ausgabe von *'command'*.

Sonstige nützliche Befehle

< . >	Wiederholt den letzten Befehl
< J >	Hängt die folgende Zeile an die aktuelle Zeile (Join)
< J*n* **>**	Hängt die folgenden n-1 Zeilen an die aktuelle Zeile
< u >	Macht die letzte Aktion (Löschen oder Ändern) rückgängig (Undo)
< U >	Macht alle Änderungen in der Zeile rückgängig

Benutzung von vi-Optionen

:set all	Auflisten der gesetzten vi-Optionen
:set *option [value]*	Setzen bzw. Überschreiben des Defaults von Optionen

Einige nützliche Optionen sind beispielsweise:

autowrite	Schaltet die autowrite-Funktion ein.
ignorcase	Ignoriert Groß- Kleinschreibung beim Suchen.
number	Numeriert die Zeilen.
showmode	Zeigt unten rechts den Modus (Input, Replace) an.

Benutzung von vi-Macros

vi erlaubt das Erstellen von Macros, die ein Mapping einer Folge von Befehlen auf einen selbstgewählten Zeichenstring darstellt. Es wird weiterhin zwischen Macros im Command-Modus und im Input-Modus unterschieden.

Definition von Macros:

:map *macroname commands*	Macro für Command-Mode
:map! *macroname commands*	Macro für Input-Mode

Beispiel: '**:map Q :q!**' erlaubt es den Editor mit Q zu verlassen ohne die Änderungen zu sichern.

Löschen von Macros:

:unmap *macroname*	für Command-Mode-Macros
:unmap! *macroname*	für Input-Mode-Macros

Weitere Details zur Macroprogrammierung entnehme man weitergehender Literatur.

Diese Referenzkarte erhebt nicht den Anspruch auf Vollständigkeit. Sie enthält nur die wichtigsten der umfangreichen und mächtigen vi Befehle.