$ **vi [option]** *fln* ... Die zu editierende Datei wird in einen Buffer kopiert. Originaldatei bleibt erhalten.

-r (recover) Journalfile auswerten (siehe Recover).

Soll der Inhalt der Datei nur betrachtet aber nicht verändert werden, so kann der Editor aufgerufen werden mit:

$ **view** *fln* Intern wird dabei eine **readonly**-Option gesetzt. (Verhindert ein unbeabsichtigtes Rückschreiben)

Befehle zum Verlassen des Editors müssen im Kommandomode abgesetzt werden. Dieser Mode wird durch **<ESC>** herbeigeführt.

- Schulungsanlage Esc-Taste drücken
- bei VT220 muß CTRL/3 statt <ESC> verwendet werden, da die Taste F11 <ESC> und x21 liefert.

ZZ Verlassen mit Rückschreiben (Achtung **CTRL/Z** stoppt den Job).

Anweisungen des ex zum Verlassen vi

:q	Editor verlassen
:q!	Verlassen ohne Rückschreiben
:wq [*file***]**	Verlassen mit Rückschreiben (in file)
:w [*file***]**	Rückschreiben ohne den Editor zu verlassen
:command	startet ex (expanded Editor) aus dem vi heraus, für ein Kommando

Kommandos des **vi** bestehen i.d.R. aus einem Operator (Buchstaben) dem ein Wiederholungsfaktor oder eine Bereichsangabe vorangestellt werden kann. Nach dem Operator kann ein Objekt angegeben werden.

Beispiel : **3dw** löscht 3 Worte

Die Wiederholung des Operatornamens bewirkt, daß das Kommando auf die ganze Zeile angewendet wird ohne <CR> einzugeben.

Beispiel: **dd** löscht die ganze Zeile

^G Zeigt den Status an (what is going on) z.B. ***"fln" [modified] line 9 of 27 --33%--***

Scrollen des Bildschirms (^ hier CTRL-Taste)

^D	Down 12 Zeilen (halben Bildschirm im Buffer nach unten)
^U	Up 12 Zeilen up (Halben Bildschirm)
^F	Forward vollen Bildschirm vorwärts
^B	Backward vollen Bildschirm rückwärts
^E	Eine Zeile nachziehen, Cursor bleibt in der Zeile wo er war
n^E	n Zeilen nachziehen. n > Bildschirm, dann Cursor in oberen Zeile, gleiche Spalte.
^Y	Eine Zeile am Bildschirmanfang nachziehen
n^Y	n Zeilen am Bildschirmanfang nachziehen
-	Anfang der vorherigen Zeile
+	Anfang der Folgezeile
H	Home Position am Bildschirm
M	Middle Mitte des Bildschirms
L	Last Letzte Bildschirmzeile
G	Zur letzen Zeile des Buffers gehen
$G	Zur letzen Zeile des Buffers gehen
nG	Zur angegeben Zeile des Buffers gehen

Cursor bewegen

B	ein Wort zurückgehen. Interpunktion dabei ignorieren.
b	ein Wort zurückgehen. Interpunktion beachten
W	ein Wort vorwärts. Interpunktion ignorieren
w	ein Wort vorwärts. Interpunktion beachten /.;,:->{}[]()
E	Ende des augenblicklichen Wortes. Interp. ignorieren
e	Ende des augenblicklichen Wortes gehen.
^	Anfang der Zeile gehen (bei manchen Tastaturen ^Blank)
0	dto Anfang der Zeile
$	An das Ende der Zeile gehen

Alternative Cursortasten des **vi** bewegen den Cursor in der gezeigten Richtung **h j k l**

Bewegung über mehrere Zeilen

+	Anfang nächste Zeile
-	Anfang vorherige Zeile
(	Zurück zum Anfang Satz (nach .<Blank> oder .<CR>)
)	Anfang Folgesatz (nach .<Blank> oder .<CR>)
{	Anfang Paragraph d.h zur vorherigen Leerzeile
}	Ans Ende vom Paragraph d.h zur nächsten Leerzeile
[[	Anfang Section. Eine Section wird durch { } jeweils in Spalte 1 gekennzeichnet (Kann gut bei der C- oder Script- Programmierung verwendet werden).
]]	Anfang der Folgesection.

Eingeben

Im vi sind die Tasten **aAiIoOcCsSR** der normalen Tastatur mit Kommandos belegt, die aus dem Kommand-Mode (Full-Screen Bewegung des Cursors möglich) in den Eingabemode umschalten.

Achtung: Der häufigste und frustrierendste Fehler ist, in der Eingabephase mit den Cursortasten zu arbeiten.

a	append	Einfügen hinter Cursorposition. Während des Einfügen wirkt die Taste als Löschtaste innerhalb des neuen Textes. Er kann dann überschrieben werden. **^h wirkt wie**
A	Append	am Zeilenende
i	insert	Einfügen vor der Cursorposition.
I	Insert	Einfügen am Zeilenanfang
o	open	Eine neue Eingabezeile wird **nach** der augenbl. Zeile aufgemacht.
O	Open	Eine neue Eingabezeile wird **vor** der augenbl. Zeile aufgemacht.

Ersetzen und ändern

cwTEXT<ESC>	Der neue TEXT ersetzt ab der Cursorposition bis zum Wortende den alten Begriff. Die Länge darf unterschiedlich sein.
Ctext<ESC>	Ersetzt ab Cursorposition bis Zeilenende.
*n***C**text<ESC>	Ersetzt ab Cursorposition bis Ende der *n*-ten Zeile
r*c* replace	Das Zeichen unter dem Cursor wird durch das Einzelzeichen ***c*** ersetzt
n**r***c* replace	Ersetzt **n** Zeichen ab dem Cursor durch ***c***
R*text*<ESC>	Ersetzt ab dem Cursor bis der Mode durch <ESC> aufgehoben wird (überschreiben).
s*kette*<ESC>	Das Einzelzeichen unter dem Cursor wird durch die angegebene Zeichenkette ersetzt (substitute).
n**s***kette*<ESC>	Die n Zeichen ab dem Cursor werden durch *kette* ersetzt (z.B. 5 Zeichen durch 20 Zeichen).
S*text*<ESC>	Ersetzt ganze Zeile durch *text*.
n**S***text*<ESC>	Ersetzt n Zeilen durch *text*
n~	Änderung Groß - Kleinbuchstaben
n**J**	Zusammenziehen von n Zeilen

Löschen (Wiederholungsfaktor möglich)

d löscht das angegebene Objekt **3dw**

dd löscht augenblickliche Zeile

D löscht Rest der Zeile

d<CR> löscht bis zum nächsten Zeilenende

d) löscht ab Cursorposition bis zum Ende des Satzes. Zeichenkette beendet durch . ! ? gefolgt von EOL (End-Of-Line) oder 2 Blanks.

d(löscht ab Cursorposition bis zum Anfang des Satzes. Wenn der Cursor am Satzanfang steht, wird der vorherige Satz gelöscht.

Suchen und markieren

f*c* find Sucht das nächste Vorkommen des angegebenen Zeichens c in der Zeile und positioniert den Cursor darauf.

F*c* Sucht rückwärts nach dem angegebenen Zeichen *c.*

t*c* Sucht Zeichen c positioniert den Cursor vor dem Zeichen.

T*c* Sucht rückwärts Zeichen c positioniert den Cursor davor

; wiederholt den letzten Suchbefehl.

, wiederholt den letzten Suchbefehl im umgekehrter Richtung.

n und **N** wiederholen auch letzten Suchbefehl

% Sucht passende Klammer () oder [] oder {}

m$_x$ Markiert die augenblickliche Position im Buffer $_x$ ist ein Buchstabe von a-z.

`x positioniert den Cursor an diese Marke

'x positioniert den Cursor an Zeilenanfang, in der sich die Marke x befindet.

d`x löscht von der Cursorposition bis zu der Marke

d'x löscht von der augenblicklichen Cursorposition bis zum Anfang der markierten Zeile

:`a,`ed löscht von Marke a bis Marke e

:'a,'ed löscht vom Anfang der Zeile in der sich die Marke **a** befindet bis zum Ende der Zeile in der sich die Marke **e** befindet.

Markierungen können bei: Lösch-, Kopier- und Bufferkommandos verwendet werden

Duplizieren

Der Editor hat einen unbenannten Buffer, in dem der letzte gelöschte oder **'changed away'** Text abgelegt ist (**Yank**-Buffer). Es gibt mehrere Möglichkeiten diesen Buffer anzusprechen. Zusätzlich kann der An-wender eigene, benannte Buffer a-z definieren und in der gleichen Weise ansprechen.

***n*yy** schreibt *n* Zeilen oder die augenblickliche Zeile in den Yank-Buffer (unbenanter Buffer)

***n*Y** vereinfachte Schreibweise für yy

P oder **p** holt Yank-Buffer zurück vor oder hinter dem Cursor

YP erzeugt eine Kopie der augenblicklichen Zeile im Yank-Buffer und fügt diese Kopie sofort unmittelbar **vor** der augenblicklichen Zeile ein. Cursor wird in Kopie positioniert.

Yp erzeugt eine Kopie der augenblicklichen Zeile im Yank-Buffer und fügt diese Kopie sofort unmittelbar **hinter** der augenblicklichen Zeile ein. Cursor wird in Kopie positioniert.

xp Bügelt Buchstabendreher aus, d.h aus ue wird eu

Versetzen und kopieren)

Um einen Text zu **versetzen**, muß er zunächst an seinem Ursprungs-ort gelöscht und in einen Buffer a-z gebracht werden. Es stehen die benannten Buffer **a** - **z** zur Verfügung.

"a5dd löscht 5 Zeilen und schreibt sie in den Buffer a.

"a5yy kopiert 5 Zeilen in den benannten Buffer **a**

"aP oder **"ap** _____ oder fügt den Inhalt des benannten Buffers **a** vor nach der augenblicklichenCursorposition ein. Es ist sogar möglich zwischen dem Löschen und dem Einfügen eine andere Datei zum Editieren zu holen. **:e fln<CR>** Falls Änderungen in der ursprünglichen Datei gemacht wurden, müssen sie entweder zurück geschrieben oder verworfen werden, bevor vi den SWITCH zur neuen Datei zuläßt.

Text verschieben

n*<<** ***n Zeilen um die Anzahl Zeichen nach links verschieben, die durch **set sw=x** festgelegt ist Default **set sw=8**

***n*>>** dto, jedoch nach rechts verschieben

>L von der Cursorposition bis zum zum Bildende um eine Tabposition nach rechts verschieben

<L von der Cursorposition bis zum zum Bildende um eine Tabposition nach links verschieben

Änderungen rückgängig machen, Wiederholung

U Undo Änderungen in der augenblicklichen Zeile werden rückgängig gemacht.

u Die letzte Änderung wird rückgängig gemacht.

. repeat Letzte Änderung wiederholen

"*n*p Löschpuffer *n* zurückholen (0 <= n <= 9)

"*c*p Löschpuffer *c* zurückholen (*c* ist ein Zeichen des Alphabets)

Der Editor sichert die letzten 9 gelöschten Blöcke in numerierten Textbuffern, die auch als Register 1 - 9 angesprochen werden können durch **"np**. Beispiel: **"1p** bringt ihnen die letzte Löschung zurück.

Wenn Sie nicht mehr wissen, wann Sie die Zeilen gelöscht haben, so können Sie mit Hilfe von u (undo) alle Register durchprobieren.

"1pu.u.u.u.u.

Recover

vom vi wird eine Journaldatei mitgeführt..

Der Benutzer bekommt beim Login eine Mail, wenn vorher eine Unterbrechung stattgefunden hat. Er muß dann in das Directory wechseln, in dem er sich beim Systemcrash befand und dort das nachfolgende Kommando absetzen:

$ vi -r fln

Eine Übersicht der vorhandenen Journaldateien erhalten Sie durch :

$ vi -r

Macros

Soll eine Taste definiert werden, die im Insert-Mode bereits eine Bedeutung hat, so muss vorher **CTRL V** eingegeben werden, damit der Editor vorgewarnt wird.
Ein ESC wird z.B. durch **CTRL V ESC** erzeugt.
Die Definition von Tasten während der Sitzung erfolgt durch
:map *taste folge*<CR> Belegung ist nur im Kommandmode aktiv
:map! *taste folge*<CR> Belegung auch im Insertmode aktiv
:**unmap** *taste* hebt Belegung auf
Beispiel : Die Taste q soll mit der Folge :wq<CR> belegt werden.

```
:map q :wq^V^V<CR><CR>
|      |  |  |
|      |  |_____ Wird mit in die Folge übernommen
|      |        und leitet die Ausführung ein
|      |_____   2 ^V damit eines in die Folge
|              übernommen wird
|_____Befehlsfolge wird durch : eingeleitet d,h. Umschaltung
      nach ex.
```

Abkürzungen

Die Definition erfolgt durch das Kommando **abbreviate**
:ab *abk ersatzkette*
:una *abk* Aufhebung der Abkürzung **unabbreviate**
Beispiel: :ab bsb BASIC-SOFTWARE-BERATUNGS GmbH
Ein einzeln stehendes **'bsb'** in Blanks eingeschlossen oder **<tab>bsb<tab>** wird automatisch durch die Abkürzung ersetzt.
:so *fln* führt die Kommandos aus der angegebenen Datei aus. IdR sind in derartigen Dateien Macros und Tastaturbelegungen abgelegt.
Es ist möglich, Schlüsselworte für Programmierspachen oder evtl. Sprachkonstrukte auf diese Weise zu definieren. Die Definitionen können in einer Kommandodatei abgelegt und durch **:so fln** aktiviert werden.
Achtung: Die Verwendung von vielen Abkürzungen beeinträchtigt das Antwortverhalten des Editors

```
:ab dw do while
:ab g goto
```

Die Startupdatei .exrc

In der Regel werden die Optionen, Macros und Abkürzungen in der Datei **.exrc** im Homedirectory abgelegt.
Diese Datei wird bei jedem Aufruf von **vi** automatisch abgearbeitet.

Achtung: In der Datei sind CTRL-Sequenzen enhalten. Diese werden sichtbar, wenn Sie die Datei mit dem **vi** bearbeiten. Sie sind unsichtbar wenn die Datei direkt zum Drucker geschickt wird und können den Drucker **in unerwünschter Weise steuern.**

Mit folgendem Kommando können die Sequenzen auch auf dem Drucker sichtbar werden !

cat -v .exrc | lp

Analysieren Sie die vorhandenen Datei **.exrc**

```
set  showmode a utoindent  autowrite  showmatch
set  wrapmargin=0  report=1
map  ^W  :set wrapmargin=8^M
map  ^X  {!}sort -b
map  ^Z  ZZ
map  ^[F  G
map  ^[H  1G
map  ^[J  DjdG$
map  ^[K  D
map  ^[L  O
map  ^[M  dd
map  ^[P  x
map  ^[Q  i
map  ^[S  ^Ej
map  ^[T  ^Yk
map  ^[U  ^F
map  ^[V  ^B
map! ^H  ^[xi
map! ^Z  ^[ZZ
map! ^[A  ^[ka
map! ^[B  ^[ja
map! ^[C  ^[la
map! ^[D  ^[i
map! ^[L  ^[^[DO
map! ^[M  ^[dda
map! ^[P  ^[lxi
map! ^[Q  ^[
map! ^[R  ^[
```

Die maps hier sind nur als Beispiel zu sehen.
Unbedingt empfohlen werden folgende Optionen in .exrc:

```
set  ruler showmatch  showmode  wrapmarging=0
set  tabstop=4 shitfwidth=4
```

nitialisierungmöglichkeiten
Environmentvariable EXINIT

Die Optionen können auch in der Environmentvariablen **EXINIT** in der Datei .**bashrc (.login in der ksh)** definiert werden. Sie stehen dann automatisch nach jedem Aufruf von vi zur Verfügung

EXINIT='set ai wm=8' #(Bourne-Shell)

Die Initialisierung mit der .exrc ist aber vorzuziehen.

Tips

showmatch zeigt Ihnen bei der Eingabe einer schließenden Klammer, die jeweils dazu passende.

% *Setzen Sie den Cursor auf eine beliebige Klammer % und der vi springt zum passenden Gegenstück*

*~ konvertiert Groß/Kleinbuchstaben **5~** die nächsten 5*

. (Punkt) Wiederholung des letzten Kommandos

xp *Bügelt Buchstabendreher aus, d.h aus ue wird eu*

10>>1 *0 Zeilen um shiftwidth einrücken*

"a5dd löscht 5 Zeilen und schreibt sie in den Buffer a.

"aP *oder* ***"ap*** *fügt den Inhalt des benannten Buffers **a** vor* _____ *oder nach der augenblicklichenCursorposition ein.*

Der ex Editor

Der **ex** ist ein eigenständiger, zeilenorientierter Editor.Er kann durch **ex** *dateiname* aufgerufen werden. Seine Kommandos können als Obermenge des **ed** betrachtet werden.. Aus dem **vi** können die **ex**-Kommandos durch einen vorangestellten **:** (Doppelpunkt) aktiviert werden. Innerhalb des **ex** werden sie ohne **:** verwendet.

:!cmd führt das Shell Kommando cmd aus
:!sh startet **/bin/sh**

:sh startet eine Subshell. Die verwendete Shell kann in **$SHELL**
oder mit **:set EXINIT** oder in **.exrc** festgelegt werden.
Rückkehr zum vi durch **CTRL/D** oder **exit**
:!*cmd* führt ein einzelnes UNIX-Kommando aus
:!! wiederhole das letzte UNIX-Kommando
:!!*cmd* die augenblickliche Zeile wird durch den Output des UNIX-
Kommandos *cmd* ersetzt.
:!% führe den augenblickliche Datei als Script durch. Die Datei
muß ausführbar sein. **:!chmod +x %** setzt das Flag.
% wird automatisch durch den Dateinamen ersetzt.
:cd [dir] ändert das Working-Directory
Q wechselt in den Line-Editor-ex
vi geht zurück zu vi

Ausgabe aus dem Buffer auf eine Datei

Alle nachfolgenden Anweisungen kann eine Bereichangabe vorangestellt werden (z.B. **:80,$w** *temp*), fehlt diese, so wird der gesamte Buffer geschrieben.

:w [*fln*] auf angegebene Datei oder in augenbl. Datei schreiben.
:w>> [*fln*] an Datei anhängen (append)
:w! [*fln*] auf Datei schreiben, normale Prüfung unterdrücken (bei noclobber)
:wq [*fln*] auf Datei schreiben und vi beenden
:x auf Datei schreiben und vi beenden

Einlesen einer Datei in den Buffer

:r [*fln*] fügt angegebene Datei an Cursorposition ein.
Default ist die augenblickliche Datei.
:nr [*fln*] fügt Datei hinter Zeile n in Buffer ein
:r !*cmd* fügt Output des UNIX-Kommandos in Buffer ein

Andere Dateien editieren

Beim Aufruf können dem vi mehrere Dateien mitgeteilt werden
vi *prog1.c abcd.c xyz*
Diese Aufzählung wird als Dateiliste bezeichnet.
Beim Editieren kann zwischen diesen Dateien gewechselt werden.

:args zeigt Dateiliste an. Die augenblickliche Datei ist in [ ] Klammern eingeschlossen
:e# vorherige Datei editieren
:n nächste (next) Datei editieren
:n! bisherige Änderungen verwerfen und nächste Datei editieren
:rew Dateiliste zurücksetzen, erste Datei editieren
:rew! dto, bisherige Änderungen ignorieren

:e *fln* angegebene Datei editieren
:e! *[fln]* bisherige Änderungen verwerfen und angegebene Datei editieren oder reedit der augenblicklichen Datei
:e +pos *fln* angegebene Datei editieren, dabei an der Position
(Zeile) ***pos*** beginnen
:n [+pos] *fln-list* spezifiziert eine neue Dateiliste, es soll mit der nächsten Datei an der Position **pos** begonnen werden.
:n +4 c b a es wird in Zeile 4 der Datei c positioniert

Mixed Anweisungen

:nu zeigt die augenblickliche Zeile mit ihrer Zeilennummer
:set ruler zeigt permanent die Position an
:s/*alt*/*neu*/opt Suche die Zeichenkette *alt* und ersetze sie durch die Kette *neu* .
Wirkt nur innerhalb der augenblicklichen Zeile.
opt bestimmt das weitere Verhalten
c Vor Austausch rückfragen (confirm)
g jede Kette *alt* durch *neu* ersetzen (global)
p geänderte Zeilen anzeigen

& letztes **:s** Kommando wiederholen

:g/*str*/*cmd* Kommando *cmd* auf alle Zeilen anwenden die die Zeichenkette *str* enthalten **:g/abc/s/alt/neu/c**
:g!/*str*/*cmd* Kommando *cmd* auf alle Zeilen anwenden, die **nicht** die Zeichenkette *str* enthalten
:v/*str*/*cmd* Kommando *cmd* auf alle Zeilen anwenden, die **nicht** die Zeichenkette *str* enthalten
:g/*str* gehe zur letzten Zeile im Buffer, die *str* enthält

Den Anweisungen **s, g** und **v** kann eine Bereichsangabe in der Form *a,e* vorangestellt werden.
:1,$s/alt/neu/gp

Options

Der Editor kann in seiner Arbeitsweise durch Options beeinflußt werden. Diese Options werden in einer Set-Anweisung gesetzt.
set muß nicht ausgeschrieben werden, es genügt **se**
Allgemeine Form :
:set *option* oder **:set** *option=value*

:set Alle Optionen zeigen deren Standardwert geändert wurde
:set all zeigt gesetzte Optionswerte
:se list Im Normalfall sind Tabulatorzeichen im Text nicht sichtbar.
Mit dieser Option werden TAB's durch ^I und das Zeilenende durch $ angezeigt.
:se nolist (Default) hebt diese Darstellung wieder auf.
:se ai Auto Indent. Der Cursor wird bei der fortlaufenden Texterfassung beim Beginn einer neuen Zeile automatisch unter das erste (non_Blank) Zeichen der vorherigen Zeile positioniert.
Achtung: Der Cursor kann nur durch ^D vor die erste Tabposition gesetzt werden. CTRL/H (Backspace) funktioniert nicht.
:se noai Hebt Auto Indent auf.
:se nu Number. Zeilen werden mit ihrer Zeilennummer angezeigt.
:se nonu (Default) **No Number**. Keine Zeilennummern anzeigen.
:se smd showmode zeigt in letzter Zeile den Eingabemode an
:se ruler (Linux) zeigt Spalte und Zeilenposition des Cursors
:se sm show match. Die zugehörige (und { wird gezeigt, wenn eine) oder } eingegeben wird.
:se wm=n Window Margin legt den Wrap-Bereich am rechten Bild-
schirmrand fest, in dem beginnende Worte an den Anfang der nächstenZeile übernommen werden.
:se wm=8 Wrapbereich sind 8 Zeichen
:se wm=0 Kein Wrapbereich, d.h es wird bis zur definierten Zeilenlänge übernommen.
:se sw=n shift width. für > und < set sw=4
:se ts=n tabstop set tabstop=n