VI-Kurzreferenz

Stand: Februar 1996

1. Hinweise zur Notation

1.1 Typografie

- Variable Texte sind *kursiv* und ggf. $^{hoch-}$ oder $_{tief-gestellt}$.
- Alternativen für Kommandos sind durch Komma (,) getrennt. Ausnahme: Komma allein.
- Funktionstasten sind großgeschrieben und unterstrichen, z.B. ESC.
- ⇑*x* steht für cntrl-*x*.
- Beispiele: `Beispiel`

1.2 Abkürzungen

AC	aktuelles Zeichen (unter dem Cursor).
AZ	aktuelle Zeile.
AP	aktuelle Position (Zeile/Spalte).
AD	aktuelle Datei.
RA	Regulärer Ausdruck (Abschnitt 6).
SP	Standard-Puffer.
SW	Einrückschritt (Shiftwidth).

1.3 Begriffe

Bereich	durch ein Kommando zu bearbeitender Bereich. Wird durch ein Bewegungskommando (s. Abschnitt 9) beschrieben.
LW	Langwort: Wie Wort, Abschluß nur durch Blank.
Wort	Zeichenkette. Abschluß durch Blank oder Sonderzeichen.

2. SOS

Wenn der vi in einem undefinierten Zustand ist: mehrfach ESC drücken. Damit wird immer die Rückkehr in den Kommandomodus erzwungen.

Beenden ohne zu sichern: `ESC:q!`

3. Aufruf des vi

Beim Aufruf kann statt *datei* auch eine Dateiliste angegeben werden (z.B. `vi *.c`).

vi	Aufruf vi, beim Speichern einer Datei muß dann ein Name angegeben werden.
vi *file*	Normaler Aufruf.
vi -r	Liefert Liste aller geretteten Dateien.
vi -r *file*	Wiederaufsetzen nach Crash (recover).
vi + *file*	Position auf der letzten Zeile.
vi +*n file*	Position auf der *n*-ten Zeile.
vi +/*text file*	Position auf *text*.
vi -t *tag*	Startet Dateibearbeitung bei *tag*.
view *file*	Start im Readonly-Modus.

4. vi beenden

Aus dem vi-Kommandomodus:

:wq, :x, ZZ	Beenden und Speichern.
:q	Beenden ohne Speichern.
:q!	Abbrechen.

5. vi-Zustände

Normalzustand: vi-Kommandomodus (wird durch ESC erreicht).

Mit	kommt man in den
:	ex-Kommandomodus (für die Dauer eines Kommandos); zurück durch CR.
i,I,a,A,o,O,c,C,s,S,R	Input-Modus; Beenden mit ESC.
/,?	Eingabe eines Suchziels. Das Suchziel wird mit CR abgeschlossen.

6. Reguläre Ausdrücke (RA)

RA werden zur Zeilenadressierung und als Suchausdrücke verwendet.

c	Zeichen *c*.
c	Sonderbedeutung von *c* unterdrücken.
^	Zeilenanfang.
$	Zeilenende.
.	beliebiges einzelnes Zeichen.
[...]	jedes einzelne Zeichen aus ...
[^...]	jedes einzelne Zeichen nicht aus ...
*c**	beliebig viele Zeichen *c* (auch 0).
/<	Wortbeginn.
/>	Wortende.

Beispiele:

`^abc`	abc am Zeilenanfang
`*$`	alle Blanks am Zeilenende
`a[0-9]b`	a0b, a1b usw. bis a9b

7. Input

7.1 An der aktuellen Position

i	Einfügen ab der AP.
a	Einfügen nach der AP.

7.2 An Zeilen

I	Einfügen am Zeilenanfang.
A	Anfügen ans Zeilenende.

7.3 Neue Zeilen

Siehe dazu auch Option autoindent!

o	Neue Zeile nach der AZ öffnen.
O	Neue Zeile vor der AZ öffnen.

8. Kommandos im Input-Modus

8.1 Löschen

BS, ⇑h	löscht das letzte eingegebene Zeichen.
⇑w	löscht das letzte eingegebene Wort.
⇑x	löscht die letzte eingegebene Zeile.
CR	beginnt eine neue Zeile.

8.2 Einrückungen

⇑d	Cursor wird um SW nach links verschoben (nur bei autoindent).
0⇑d	beendet das automatische Einrücken.
^⇑d	unterbricht das automatische Einrücken für diese Zeile.
⇑t	um SW einrücken.

8.3 Sonderzeichen einfügen

⇑v*x*	*x* wird nicht als Steuerzeichen interpretiert sondern direkt in den Text übernommen.
x	*x* wird direkt in den Text übernommen.

8.4 Input-Modus beenden

ESC	beendet den Input-Modus.
DEL	bricht den Input-Modus ab.

9. Cursorbewegung

Diese Kommandos werden auch verwendet, um einen Bereich zu definieren!

9.1 Zeichenweise

BS, ⇑h, h	links.
j	unten.
k	oben.
SPACE, l	rechts.
⇑n	gleiche Spalte in der nächsten Zeile.
⇑p	gleiche Spalte in der vorhergehenden Zeile.

9.2 Wortweise

w	Anfang nächstes Wort.
W	Anfang nächstes LW.
b	Anfang vorhergehendes Wort.
B	Anfang vorhergehendes LW.
e	Wortende (nach rechts).
E	Ende LW (nach rechts).

9.3 Innerhalb der aktuellen Zeile

^, _	erstes Zeichen (kein Blank).
0	erste Spalte.
$	Letztes Zeichen.
|	erstes Zeichen.
n|	*n*-tes Zeichen.

9.4 Grössere Bereiche

CR, +	Anfang nächste Zeile.
-	Anfang vorhergehende Zeile.
G	letzte Zeile der Datei.
*n*G	Auf Zeile *n*.
H	Linke obere Ecke des Bildschirms.
M	Mittlere Bildschirmzeile.
L	Letzte Zeile des Bildschirms.
*n*L	*n*-te Zeile vom unteren Bildrand.
*n*H	*n*-te Zeile vom oberen Bildrand.
%	steht der Cursor auf einer der Klammern { } [ ] ( ), wird nach der jeweils korrespondierenden Klammer gesucht.
(	Anfang des vorhergehenden Absatzes.
)	Absatzende.
{	Anfang eines ROFF-Absatzes.
}	Ende eines ROFF-Absatzes.
[[	Anfang eines ROFF-Abschnitts bzw. die nächste {-Klammer.
]]	Ende eines ROFF-Abschnitts bzw. die vorherige {-Klammer.
'*a*	Setzen des Cursors an den Anfang der Zeile, die die Marke *a* enthält.
`*a*	Setzen des Cursors auf die Position, die mit *a* markiert wurde.
``	kehrt zur letzten Position zurück, bei der eine Änderung gemacht wurde.
"	wie ``, aber Zeilenanfang.

10. Fenster verschieben

⇑d	1/2 Fenstergröße nach unten.
⇑u	1/2 Fenstergröße nach oben.
⇑f	eine Seite vorwärts.
⇑b	eine Seite rückwärts.
⇑e	Text um eine Zeile nach unten.
⇑y	Text um eine Zeile nach oben.
z+	AZ an den oberen Rand des Fensters.
z-	AZ an den unteren Rand des Fensters.
z.	AZ in die Fenstermitte.

11. Suchen

11.1 Suchen nach Text in der Datei

/*RA*	sucht vorwärts nach *RA*.
?*RA*	sucht rückwärts nach *RA*.
n	wiederholt letzte Suche.
N	wiederholt letzte Suche, andere Richtung.

/*RA*/*cmd*, ?*RA*/*cmd*

RA suchen, dann *cmd* ausführen:

-*n*	*n* Zeilen vorher.
+*n*	*n* Zeilen dahinter.
z.	Zeile mit *RA* in Bildschirmmitte.
z-	*RA* in letzte Zeile.
z+	*RA* in erste Zeile.

/CR, ?CR wiederholt letzte Suche.

//*cmd*, ??*cmd*

Suche wiederholen, *cmd* ausführen.

11.2 Suchen nach einem Zeichen in der aktuellen Zeile

f*x*	suche x rechts.
F*x*	suche x links.
t*x*	suche x rechts; Cursor vor das Zeichen.
T*x*	suche x links; Cursor hinter das Zeichen.
;	wiederholt das letzte Suchkommando f,F,t,T.
,	wie ; jedoch andere Richtung.

12. Ersetzen

text wird immer mit ESC abgeschlossen.

12.1 Zeichen

r*x*	ersetzt das Zeichen unter dem Cursor durch *x*.
s*text*	ersetzt ein Zeichen durch *text*.
*n*s*text*	ersetzt *n* Zeichen durch *text*.
S*text*	ersetzt die ganze Zeile durch *text*.

12.2 Worte (ab der Cursorposition)

cw*text*	Wort durch *text*.
cW*text*	Langwort durch *text*.

12.3 Bereiche

R*text*	ersetzt ab AP zeichenweise durch *text*.
c^{ber}*text*	Ersetzt den Bereich.
C*text*	ersetzt den Rest der AZ durch *text*.
cc*text*	ersetzt die ganze AZ durch *text*.
*n*cc*text*	ersetzt *n* Zeilen durch *text*.

12.4 Suchen und Ersetzen

12.4.1 Syntax

:{*bereich*}s^{del}RA^{del}*text*^{del}{*mod*}

12.4.2 Elemente

bereich	Definiert den Suchbereich (s.u.).
del	Delimiter, z.b. /
text	neuer Text. Das Sonderzeichen & wird durch den die RA gefundenen Text ersetzt.
mod	Modifier:
g	Alle Vorkommen von RA in einer Zeile.
v	Überall, nicht im *bereich*.

12.4.3 Suchbereich *bereich*

%	Ganze Datei.
von,*bis*	Suchbereich:
Nummer	Zeilennummer.
.	AZ.
$	Letzte Zeile in der Datei.
'*x*	Markierung *x*.
/*text*/	Zeile mit *text*.

Dabei sind auch Ausdrücke möglich: `.+2` `$-3`

12.4.4 Beispiele

```
:s/aaa/bbb/
```

ersetzt in der AZ das erste aaa durch bbb.

```
:%s/aaa//g
```

löscht aaa in der ganzen Datei.

```
:.,/xyz/s/ptr/&_t/g
```

ersetzt von der AZ bis zur nächsten Zeile mit xyz alle ptr durch ptr_t.

```
:%s/ *$//g
```

Entfernt alle Leerzeichen am Zeilenende.

13. Löschen

Alle gelöschten Zeichen sind im SP.

13.1 Zeichen

x	löscht das Zeichen unter dem Cursor.
*n*x	löschen von *n* Zeichen

13.2 Worte

dw	Löschen eines Wortes.
dW	Löschen eines LW.
de	Löschen bis ans Wortende.
dE	Löschen bis an das Ende des aktuellen LW.

13.3 Bereiche

D	Löschen bis ans Zeilenende.
d%	Löschen bis zur korrespondierenden Klammer (Cursor muß auf einer Klammer stehen).
d*ber*	Löscht den gesamten Bereich.
*n*d*ber*	Löscht n Bereiche, z.B. `3dw`.

13.4 Zeilen

dd	Löschen der AZ.
*n*dd	Löschen von *n* Zeilen, beginnend bei der AZ.

14. Kopieren

14.1 Standard-Puffer

Alle Löschoperationen kopieren die gelöschten Zeichen in den SP.

Y,yy	kopiert die AZ in den SP.
*n*Y, *n*yy	kopiert *n* Zeilen ab der AZ in den SP.
P	fügt den Inhalt des SP vor die AZ ein.
p	fügt den Inhalt des SP nach der AZ ein.
y*ber*	kopiert *ber* in den SP.

14.2 Puffer mit Namen

"^{a}*n*yy	kopiert *n* Zeilen in den Puffer ^{a}.
"*a*y*ber*	kopiert *ber* in den Puffer *a*.
"*a*p	fügt den Inhalt des Puffers *a* nach der AZ ein.
"*a*P	fügt den Inhalt des Puffers *a* vor der AZ ein.

15. Zeilen links/rechts verschieben

>*ber*	Bereich um SW nach rechts schieben.
<*ber*	Bereich um SW nach links schieben.
<<	AZ um SW nach links schieben.
>>	AZ um SW nach rechts schieben.

16. tags

Siehe dazu auch das UNIX-Utility ctags!

Format des Tagfiles:

tag file vi-Suchkommando/Zeilennr.

:ta *tag*	vi sucht im tags-File nach der dem *tag* zugeordneten Datei, lädt sie und führt das Suchkommando aus bzw. positioniert auf die Zeile.
⇑]	wie :ta, tag ist das Wort rechts vom Cursor.

17. Makros

@*x*	führt Makro im Buffer *x* aus.
@@	wiederholt letzten Makroaufruf.
:map *m k*	weist dem Wort *m* das vi-Kommando *k* zu. Makronamen dürfen max. 10 Zeichen, Kommandos max. 100 Zeichen lang sein.
:unmap *m*	löscht das Makro *m*.
:ab *k t*	Definiert die Abkürzung *k* für den Text *t*. Im Input-Modus wird *k* zu *t* expandiert. *k* wird nicht in Wörtern expandiert!
:una *k*	löscht die Abkürzung *k*.
:map! *a b*	ersetzt bei der Eingabe (Inputmodus) jedes *a* durch *b*.

18. Dateioperationen

Die Ausführung eines Kommandos kann durch ein ! erzwungen werden, z.B. :q! bei ungesicherter Datei.

:r *file*	Datei hinter die AZ einlesen.
:e *file*, :vi *file*	Datei editieren.
:n	Nächste Datei in der Aufrufliste editieren.
:e!	Bisherige Änderungen verwerfen, AD nochmals editieren.
:e + *file*	Datei *file* laden, auf letzte Zeile positionieren.
:e +*n file*	Datei *file* laden, auf Zeile *n* positionieren.
:e +/*text file*	Datei laden, auf Zeile mit *text* positionieren.
:e #	vorhergehende Datei editieren.
:w	AD speichern.
:w *file*	AD unter neuem Namen speichern
:q	beenden.
:x, :n	nächste Datei aus der Aufrufzeile.
:n *file file*	wie vi *datei datei*.
:wq	AD schreiben, beenden.

Sonderzeichen:

#	Name der letzten verwendeten Datei.
%	Name der AD.

19. Undo

u	macht die letzte Änderung rückgängig.
U	macht alle Änderungen der aktuellen Zeile rückgängig.

20. Markierungen

m*c*	Markierung *c* auf AP setzen (*c*: a-z).
'*a*	Setzen des Cursors an den Anfang der Zeile, die die Marke *a* enthält.
`*a*	Setzen des Cursors auf die Position, die mit *a* markiert wurde.

21. Diverses

~	AC: Groß<->Klein.
J	Folgende Zeile an AZ anhängen (join).
:f, ⇑g	Statusinformation.
.	wiederholt die letzte Änderung.
z*n*CR	Windowgröße auf *n* Zeilen setzen.
⇑l	Neuaufbau des Bildschirms.
Q	Wechseln in den ex-Modus.
:vi	Wechsel aus ex- in vi-Modus.
:cd	wechsel ins Directory $HOME.
:cd *dir*	wechselt das Directory nach *dir*.
:pre	sichert den aktuellen Zustand so, daß nach weiteren Änderungen der gesicherte Zustand mit vi -r file wieder hergestellt werden kann.
:g /*text*/*cmd*	Die Zeilen, die *text* enthalten werden ausgewählt; auf sie wird *cmd* angewendet. *cmd* ist ein ex-Kommando.

22. Betriebssystem aufrufen

Die Shell wird mit der Option shell= definiert!

!^{ber}*cmd*	Bereich ^{ber} wird durch das Resultat des shell-Kommandos *cmd* ersetzt. ^{ber} ist Input für *cmd*.
!!*cmd*	ersetzt AZ.
!!!	wiederholt letztes Kommando.
:!*cmd*	*cmd* wird in einer shell ausgeführt.
:!!	letztes :!*cmd* wird wiederholt.
:r !*cmd*	*cmd* wird in einer shell ausgeführt, sein Output kommt hinter die AZ.
:sh	Start einer shell.

23.

vi-Optionen

Der vi kann mit mehreren Parametern gesteuert werden. Zur Voreinstellung können die Optionen auch im Profile .exrc gesetzt werden.

Setzen einer Option:	:set *option*CR
Setzen auf einen Wert:	:set *option=wert*
Rücksetzen einer Option:	:set no*option*CR
Anzeigen der gesetzten Optionen:	:set
Option abfragen:	:set *option*?CR

all	zeigt alle Parameter.
autoindent, ai	automatische Einrückung bei Insert.
autoprint, ap	Zeile wird bei Änderung sofort am Bildschirm neu aufgebaut.
autowrite, aw	Beim Verlassen des Editors oder Laden einer neuen Datei wird die AD zurückgeschrieben.
beautify, bf	entfernt alle Kontrollzeichen aus einem Text.
dir=*pfad*	Pfad für die temporären Files des vi.
errorbells, eb	bei Fehlermeldungen akustisches Zeichen (BEL).
hardtabs=*n*, hd=*n*	gibt die Abstände der Hardwaretabulatoren des Terminals an.
ignorecase, ic	Bei Textsuche (/?) wird Groß-/Kleinschreibung ignoriert.
lisp	Automatisches Einrücken für LISP; die Kommandos () {} [[]] werden entsprechend den LISP-Konventionen interpretiert.
list	alle Zeichen sichtbar darstellen.
magic	schaltet die Bedeutung der Sonderzeichen in ed und ex ein. Wenn nomagic gesetzt ist, werden nur noch ^ und $ als Sonderzeichen erkannt.
mesg	mesg: Alle Meldungen kommen bis zum Terminal durch; nomesg: alle Meldungen, die nicht vom vi selbst kommen, werden unterdrückt.
number, nu	Anzeige von Zeilennummern.
paragraphs, para	definiert die Absatzgrenzen für {}-Kommandos.
prompt	Toggle für Prompt-Zeichen : (nur im ex-Modus).
redraw	Beim Ändern wird die aktuelle Zeile bei jedem Zeichen neu aufgebaut.
remap	Makros werden rekursiv abgearbeitet.
report=*n*	Änderungen, die mehr als n Zeilen betreffen, werden mitgeteilt.
scroll=*n*	Anzahl Zeilen, um die bei ⇑d, ⇑u verschoben werden soll.
sections	definiert die Abschnittgrenzen für die [[ ]] Kommandos.
shell, sh	Pfad des Kommandointerpreters, der bei !-Kommandos gestartet wird.
shiftwidth, sw=*n*	Bei Shift-Kommandos (< > und ⇑d) wird um *n* Zeichen verschoben.
showmatch, sm	Im Input-Modus wird bei der Eingabe von) und } kurz auf die korrespondierende öffnende Klammer gesprungen (falls sie noch im Fenster ist).
tabstop=*n*	Abstand der Tabulatorpositionen.
taglength=*n*	Maximale Länge eines Tags.
tags=*file-liste*	durch Blanks getrennt: Liste der Tag-Files.
ts=*n*	Tabulatorpositionen alle *n* Spalten.
term	Terminaltyp aus Termcap.
terse	Fehlermeldungen werden in verkürzter Form ausgegeben.
warn	Vor jedem !-Kommando wird eine Warnung ausgegeben, wenn die Datei vorher nicht gesichert wurde.
window	gibt die Größe des Editorfensters an.
w300, w1200, w9600	setzt die Fenstergröße auf 8, 16 oder max. mögliche Zeilen.
wrapscan, ws	Ringsuche: EOF -> 1.Zeile -> AP.
wrapmargin, wm	wird hinter Zeilenlänge-wm eingegeben, kommt BEL.
writeany, wa	erlaubt das zurückschreiben auf alle Dateien (wie :w!).

24. vi-Profile

Falls eine Datei mit dem Namen .exrc im HOME-Directory vorhanden ist, wird ihr Inhalt ausgeführt, bevor Kommandos vom Terminal angenommen werden. Bewährtes Beispiel für ein Profile:

```
set autoindent
set ignorecase
set number
set report=1
set shiftwidth=3
set showmatch
set noterse
set autoprint
map q ⇑]
map Q :e #⇑M
```

Alternativ können Voreinstellungen über die Umgebungsvariable EXINIT eingestellt werden. Beispiel:

```
EXINIT="set ai ic nu" ; export
EXINIT
```

25. Notizen

Dipl.-Inform.(FH) Germar Heinicke
Hans-Fitz-Weg 4, 81476 München
Telefon (089) 74 57 65 78
Mobil (0172) 899 22 69

Email: germar.heinicke@t-online.de
http://home.t-online.de/home/germar.heinicke/