

---

## 6 Bearbeitung von Texten

### 6.1 Lernziele

#### In diesem Kapitel lernen Sie

- ▶ eine Textdatei mit dem **vi**-Editor zu erstellen und bearbeiten.
- ▶ mögliche alternative Editoren einzuschätzen.

Unter UNIX gibt es mehrere Editoren mit verschiedenen Eigenschaften für unterschiedliche Einsatzgebiete:

**ed**: Zeilenorientierter Editor; Standardeditor von AT&T UNIX bis Version V

**ex**: Erweiterung des **ed**-Editors, ebenfalls ein zeilenorientierter Editor.

**vi**: Bildschirmorientierter (visual) Editor, der in den **ex**-Modus umschaltbar ist. Der **vi**-Texteditor ist nahezu auf jedem UNIX-System vorhanden. Deshalb ist er der Standard-Editor auf allen UNIX-Systemen. Für den UNIX-Systemadministrator ist es daher ein Muss, den **vi** zu beherrschen. Außerdem werden viele Eigenschaften des **vi** auch bei der Shell-Programmierung und Shell-Textverarbeitung gebraucht.

Neben **vi** ist **vim** (Vi IMproved) eine (stark) erweiterte Version des **vi**: Multilevel-Undo, mehrere Fenster und Dateien, Syntax-Hervorhebung, Kommandozeilen-Bearbeitung, Dateinamenvervollständigung, Online-Hilfe, visuelle Auswahl sind nennenswerte neue Features.

**gvim** erweitert **vim** mit einer (Gtk-basierten) grafischen Benutzeroberfläche: Menüs und Symbolleisten für häufig verwendete Befehle machen dem Einsteiger das Leben leichter. Dennoch funktionieren alle tastaturbasierten Kommandos wie unter der Konsole.

**pico bzw. nano** Ein kleiner, bildschirmorientierter Editor im Textmodus. Da alle Tastenkombinationen am Bildschirm angezeigt werden, lässt er sich schnell erlernen. **nano** ist die fast identische Nachfolgeversion des **pico**.

**emacs und xemacs**: Er zeichnet sich durch einen integrierten LISP-Interpreter aus. Deswegen gibt es zahlreiche Erweiterungen für diesen Editor, wie Taschenrechner, Entwicklungsumgebungen für fast alle Programmiersprachen, ein integriertes Mail-System und einen Web-Browser. Kurz: Es gibt fast nichts, was dieser Editor nicht kann.

Der Name leitet sich aus „Editor Macros“ ab, was recht treffend die Stärke dieses Editors beschreibt. Vor allem für Programmierer und Buchautoren, die eine Satzsprache wie  $\text{\LaTeX}$  oder HTML schreiben, leisten die Makrofähigkeit, das farbliche Hervorheben von Syntaxelementen sowie die beigefügten Makropakete wertvolle Dienste.

Es existieren zwei Versionen des **Emacs**:

**GNU Emacs:** Das Original aus dem GNU-Projekt.

**XEmacs:** Eine **Emacs**-Variante, die vor allem auf eine grafische Oberfläche und komfortable Bedienung abgestimmt ist.

**kwrite und kate:** Zwei Text-Editoren des KDE-Desktops. Unter KDE die Editoren der Wahl für ernsthaftere Anwendungen wie Programmierung oder zum Bearbeiten von Konfigurationsdateien, wobei **kate** die größeren Funktionalität bietet.

*Achtung:* Grafische Editoren wie **kwrite** hinterlassen gerne Zeilenumbruchzeichen oder ersetzen Tabulatorzeichen durch Leerzeichen, was für manche Konfigurationsdateien tödlich sein kann, und zu massiven Syntaxfehlern führt. Meist kann man die Editoren aber passend konfigurieren.



## 6.2 Der vi-Editor

### 6.2.1 Bearbeitungsmodi des vi-Editors

Der **vi** kann in drei verschiedenen Modi arbeiten:

- Befehlsmodus (Command Mode)
- Eingabemodus (Input Mode)
- Last-Line-Modus (auch Ex-Mode)

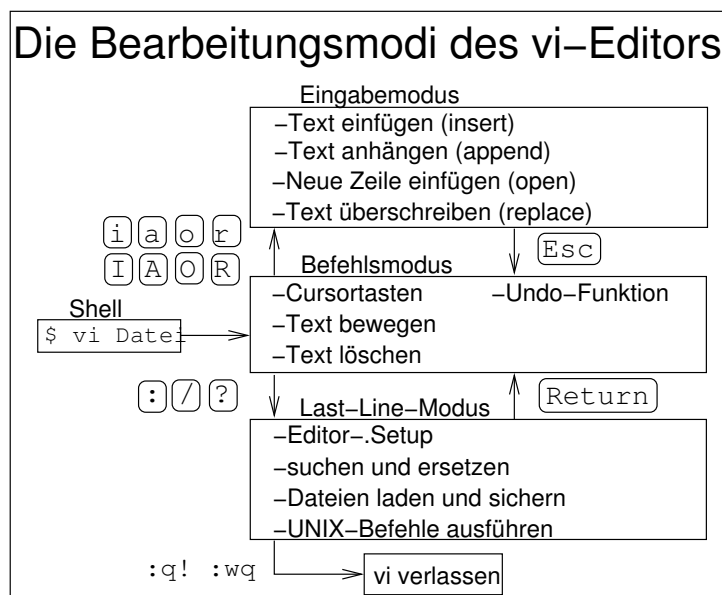
**Befehlsmodus** Eine **vi**-Sitzung beginnt im Befehlsmodus, von dem aus beliebige **vi**-Befehle abgesetzt werden können. Diese Befehle bestehen in der Regel aus wenigen Buchstaben, wie z.B. **dw** für „delete word“. Sie können folgende Funktionen erfüllen:

- Positionieren des Cursors
- Löschen, Kopieren und Verschieben von Text
- Umschalten in die anderen Modi

**Eingabemodus** In diesem Modus kann nur Text eingegeben werden. Suchen nach Text, Löschen, etc. muss in den anderen Modi durchgeführt werden. Das Positionieren mit den Cursortasten geht in vielen neueren Implementierungen.

**Last-Line-Modus** Bestimmte Operationen können nur im Last-Line-Modus durchgeführt werden:

- Text suchen
- Text laden und speichern
- Setzen von Editoroptionen
- Durchführen von Shell-Befehlen, ohne **vi** zu verlassen
- Verlassen des **vi** (ggf. ohne Abspeichern)



### 6.2.2 Aufrufen und Verlassen des vi-Editors

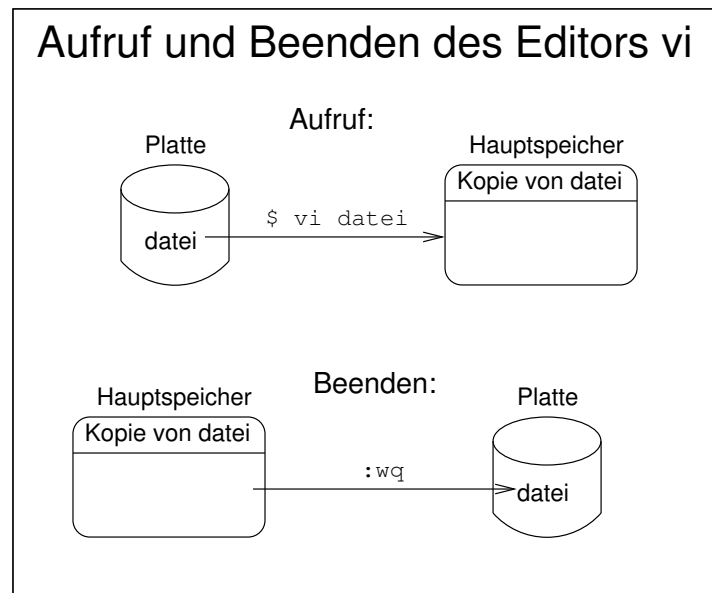
**vi-Editor aufrufen** Der **vi**-Editor wird aufgerufen mit dem Befehl

```
vi [option] [datei]
```

## Bearbeitung von Texten

---

Der **vi**-Editor kopiert beim Aufruf den Inhalt der zu editierenden Datei in einen Textpuffer im Hauptspeicher. Die Veränderungen beim Editieren erfolgen im Textpuffer. Die Originaldatei bleibt unverändert.



Der Inhalt einer bereits existierenden Datei wird am Bildschirm dargestellt. Der Cursor ist oben links positioniert, die Tilde-Zeichen (~) markieren Bildschirmzeilen, die noch nicht zum Textpuffer gehören.

**vi-Editor verlassen** Der Befehl `␣wq` schreibt den Textpuffer auf die editierte Datei zurück. Die alte Datei wird dadurch überschrieben.

Der Befehl `␣q!` beendet die Editorsitzung ohne Veränderung der editierten Datei.







*Beispiele:*

- 3dd Löscht die nächsten drei Zeilen
- 5w Springt fünf Worte weiter

Fehlt die Wiederholungsangabe, so wird standardmäßig der Faktor 1 angenommen. Fehlt die Bereichsangabe, wird die aktuelle Cursorposition, bzw. je nach Befehl die aktuelle Zeile angenommen.

**Positionieren des Cursors im Text** Im vi-Editor können Sie den Cursor *zeichenweise*, *zeilenweise* oder *abschnittsweise* im Text positionieren. In Abhängigkeit vom verwendeten Terminal können die *Pfeiltasten* zur zeichen- und zeilenweisen Positionierung verwendet werden. In Fällen, in denen die Pfeiltasten nicht verwendbar sind, gibt es alternative Tastenbefehle.



*Hinweis:* Das Positionieren des Cursors ist in älteren vi-Versionen nur im Befehlsmodus möglich.

### Cursorpositionierung

**zeichenweise:**

<code>h</code>	, <code>←</code>	Cursor nach links
<code>j</code>	, <code>↓</code>	Cursor nach unten
<code>k</code>	, <code>↑</code>	Cursor nach oben
<code>l</code>	, <code>→</code>	Cursor nach rechts (l wie rechts!)

**zeilenweise:**

<code>Return</code>	Beginn der nächsten Zeile
<code>^</code>	Zeilenbeginn
<code>\$</code>	Zeilenende

**wortweise:**


<code>w</code>	ein Wort weiter (zum nächsten Wortbeginn)
<code>b</code>	ein Wort zurück (back: zum letzten Wortbeginn)

<b>abschnittsweise:</b>	<b>H</b>	(High End) springt zur ersten Zeile der gerade angezeigten Bildschirmseite
	<b>L</b>	(Low End) springt zur letzten Zeile der gerade angezeigten Bildschirmseite
	<b>Ctrl-B</b>	(Backward) Vorhergehende Bildschirmseite
	<b>Ctrl-F</b>	(Forward) Nächste Bildschirmseite
	<b>G</b>	Sprung zum Dateiende
	<b>n G</b>	(Go) Sprung zu Zeile <i>n</i> , z.B. <b>1 G</b> positioniert an den Beginn der Datei
	<b>Ctrl-L</b>	baut den Bildschirm neu auf

### 6.2.4 Texte einfügen (Eingabemodus)

Um vom Befehlsmodus in den Eingabemodus zu wechseln, gibt es verschiedene Befehle, die in der nachfolgenden Tabelle aufgeführt sind. Sie bestimmen, an welcher Position der Text relativ zum Cursor in der Datei eingetragen wird.

<b>I</b>	(Insert) Am Zeilenanfang einfügen
<b>i</b>	(Insert) Vor dem Cursor einfügen
<b>a</b>	(Append) Nach dem Cursor einfügen (anhängen)
<b>A</b>	(Append) Text am Zeilenende anhängen
<b>O</b>	(Open) Öffnen einer neuen Zeile vor der aktuellen Zeile
<b>o</b>	(Open) Öffnen einer neuen Zeile nach der aktuellen Zeile

*Hinweis:* Durch Drücken der Taste **Esc** wird der Eingabemodus verlassen, und der Editor geht in den Befehlsmodus zurück. Unter Linux findet man meist den Editor **vim** vor, eine Weiterentwicklung des ursprünglichen **vi** mit einer Vielzahl neuer Kommandos. Einige der hier vorgestellten Kommandos, z.B. **I** und **A**, sind bei alten Versionen von **vi** nicht verfügbar. 

### 6.2.5 Text löschen

Im Befehlsmodus des vi-Editors stehen zwei Befehle zum Löschen von Text zur Verfügung:

- `x` (eXtinguish) Zeichen löschen
- `d` (Delete) Objekte löschen:

**Wichtige Objekte** (durch Cursorposition definiert):

- `w` Aktuelles Wort (`d w`=Delete Word)
- `d` Aktuelle Zeile (`d d`)
- `$` Cursorposition bis Ende der Zeile (`d $`)
- `^` Anfang der Zeile bis zum Cursor (`d ^`)
- `1 G` Anfang der Datei bis zur aktuellen Zeile (`d 1 G`)
- `G` Aktuelle Zeile bis Ende der Datei (`d G`)

#### Struktur der Löschbefehle

*[Wiederholung]*`x`

*[Wiederholung]*`d`Objekt

#### Löschbefehle des vi-Editors

- `x` Löscht das Zeichen unter dem Cursor
- `d d` Löscht die aktuelle Zeile
- `d w` Löscht das nächste Wort
- `3 d w` Löscht die nächsten drei Worte
- `d G` Löscht bis ans Ende der Datei

### 6.2.6 Kopieren und Verschieben von Text

Mit dem vi-Editor können Sie Text in Hilfspuffern (Zwischenspeichern) vorübergehend speichern, um ihn an beliebiger Stelle wieder einzufügen.

**Eintragen in den Standard-Hilfspuffer:** Jeder Löschbefehl (`d` oder `x`) überträgt das Gelöschte automatisch in den Hilfspuffer. Mit dem **yank**-Befehl (Befehl `Y`, `y y`) können Sie Text in den Hilfspuffer kopieren, ohne ihn im Text zu löschen.

**Zurückholen aus dem Standard-Hilfspuffer:** Mit dem Befehl **paste** (Befehle **P**, **p**) wird der zuletzt eingetragene Text aus dem Hilfspuffer nach bzw. vor dem Cursor eingefügt.

### Kopieren und Versetzen von Textteilen:

<b>d</b> <b>d</b>	Löscht Zeile und kopiert sie in den Standard-Hilfspuffer
<b>Y</b> , <b>y</b> <b>y</b>	(Yank) Kopiert die gegenwärtige Zeile in den Standard-Hilfspuffer
<b>3</b> <b>Y</b>	Kopiert drei Zeilen in den Standard-Hilfspuffer
<b>P</b>	(Paste) Fügt den Inhalt des Hilfspuffers vor dem Cursor ein
<b>p</b>	(Paste) Fügt den Inhalt des Hilfspuffers nach dem Cursor ein

### 6.2.7 Suchen und Ersetzen von Texten

Mit dem **vi**-Editor können Sie Suchmuster nur im Last-Line-Modus suchen. Abhängig vom Suchkommando wird das Suchmuster im Text hinter dem Cursor (Befehl **/**) bzw. vor dem Cursor (Befehl **?**) gesucht.

#### Nach Mustern suchen

<b>/</b> <i>Muster</i>	Sucht vorwärts ab der Cursorposition nach <i>Muster</i>
<b>?</b> <i>Muster</i>	Sucht rückwärts ab der Cursorposition nach <i>Muster</i>
<b>n</b>	(Next) Wiederholt letzten Suchvorgang (in derselben Richtung)
<b>N</b>	(Next) Wiederholt letzten Suchvorgang (in umgekehrter Richtung)

**Textobjekte ändern** Nachdem die Zeichenkette gefunden wurde, wird der Cursor an die entsprechende Stelle positioniert. Mit verschiedenen Befehlen **c**, **r**, **R** kann ein angegebenes Objekt ersetzt werden. Der Editor löscht dabei das Objekt und schaltet in den Eingabemodus um. Die nachfolgende Eingabe ersetzt das gelöschte Objekt. Die Eingabe wird durch Drücken von **Esc** abgeschlossen, und der Editor schaltet wieder in den Befehlsmodus zurück. Die Struktur der Änderungsbefehle orientiert sich an der Struktur der Löschbefehle.