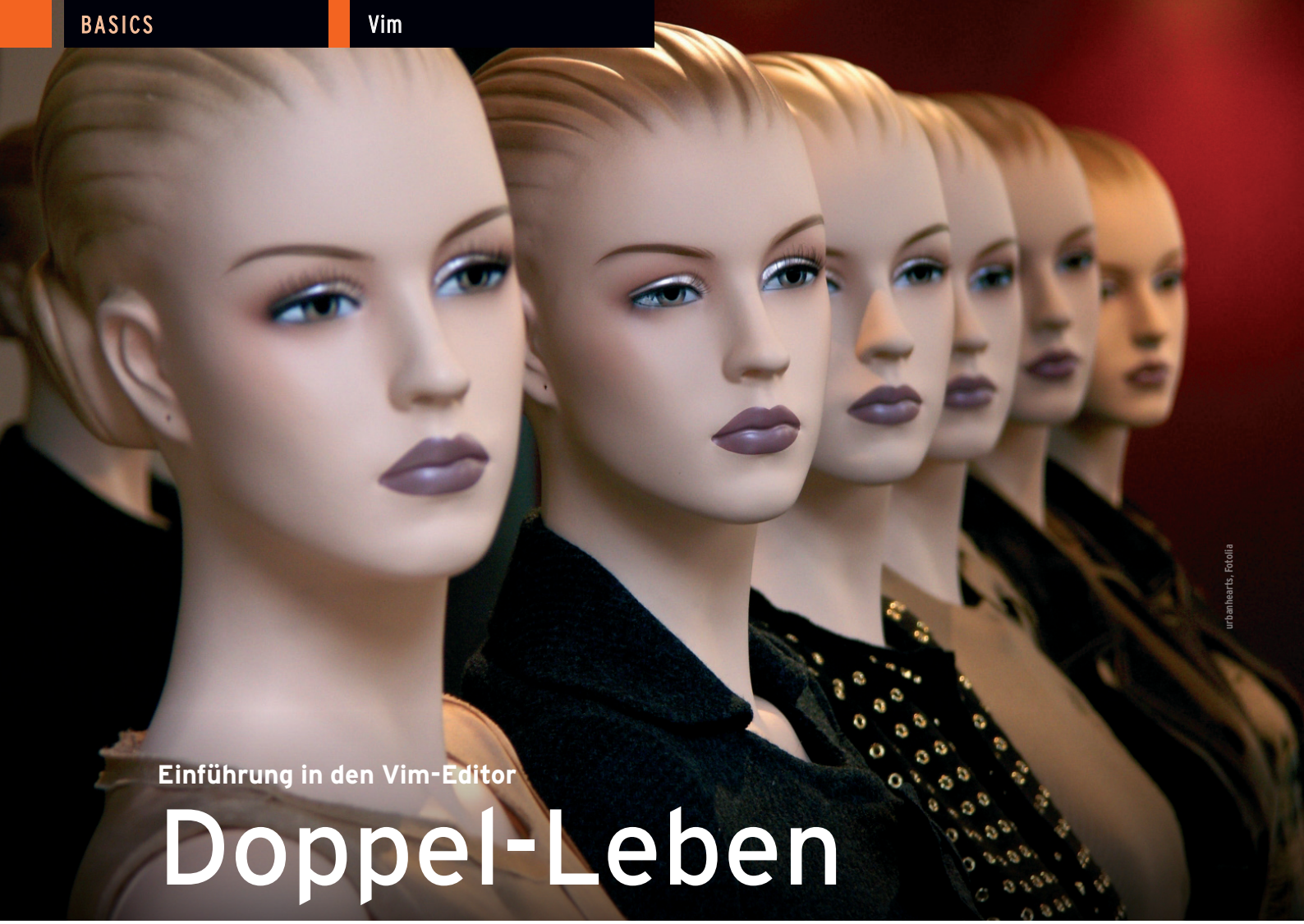

Einführung in den Vim-Editor

Doppel-Leben

Der Vim-Editor fehlt in kaum einer noch so kleinen Linux-Distribution. Grund genug für jeden Administrator, sich mit ihm zu beschäftigen. Wer ihn besser kennt, profitiert davon durch effizientes Arbeiten. Bruce Byfield

Texteditoren sind die grundlegenden Werkzeuge für jeden Administrator und jede Administratorin, mit denen sie Konfigurationsdateien bearbeiten und Shellskripts schreiben. Ein Editor, der im Gegensatz zu dem kürzlich an dieser Stelle behandelten Emacs [1], fast immer installiert ist, ist ein Vi-ähnlicher Editor wie Vim [2].

Ungefähr schon seit der Entdeckung des Feuers dauert bekanntlich der Flamewar zwischen Emacs- und Vi-Anhängern an. Ganz ohne Wertung ist es aber wahrscheinlicher, dass irgendein Vi-Klon Teil einer minimalistischen Linux-Distribution ist. Neben Vim, um den es in diesem Artikel vorwiegend geht, gibt es beispielsweise auf BSD-Systemen auch Nvi [3], bei Minix findet sich Elvis [4], selbst die Busybox-Tools bringen ihren eigenen Vi-Ableger mit [5].

Auf Standard-Linux-Distributionen ist Vim aber zweifellos der Favorit, nicht zuletzt, weil er sich mit Makros und Skripts gut an die eigenen Bedürfnisse anpassen lässt. Vim ist heute so populär, dass die meisten Anwender sich mit dem Namen „Vi“ eigentlich auf ihn beziehen und dabei vergessen, dass er sich gegenüber dem Ahnen wesentlich weiterentwickelt hat und deshalb den Namen „Vi IMproved“ (verbesserter Vi) trägt. Mittlerweile hat selbst Vim schon einige Jahre auf dem Buckel und kann dieses Jahr sein 20-jähriges Jubiläum feiern.

Viele Linux-/Unix-Neueinsteiger finden Vim bei der ersten Benutzung sehr eigenwillig. Die Trennung zwischen Kommando- und Editiermodus ist ungewohnt (dazu später mehr), und anekdotisch sind die Geschichten von Anwendern, die es nicht schaffen, den Editor zu verlassen. Deshalb sind bei Linux-Anfängern Editor-Alternativen wie Joe [6] oder Nano [7] beliebt.

Spätestens, wenn man im Single-User-Modus landet oder in einer anderen minimalen Recover-Umgebung, fehlen diese Tools aber, und man ist doch wieder auf zumindest rudimentäre Beherrschung der Vim-Bedienung angewiesen. Natürlich ist der Funktionsumfang riesig und die richtige Beherrschung des Editors beinahe ein Lebenswerk, aber man kann sich dennoch leicht an einem Nachmittag ausreichende Vim-Kenntnisse aneignen.

Interfaces und Modes

Sie starten den Vim-Editor, indem Sie »vim« eingeben. Wollen Sie eine existierende Datei öffnen, folgt deren Name dahinter. Existiert die Datei noch nicht, legt Vim sie an. Auf vielen Distributionen gibt es einen Alias für Vim, dann genügt es, »vi« einzugeben.

Wenn Sie Vim mit einer neuen Datei starten, erscheint das Fenster beunruhigend leer, gibt es keine Menüs oder Befehlskürzel zu sehen, die der Orientierung dienen könnten. Beim Start ohne weitere Parameter gibt Vim immerhin einige Hinweise zum Einstieg (Abbildung 1). Wer arglos nun seinen Text eingeben oder den Cursor bewegen möchte, erleidet gleich sein erstes Vim-Frusterlebnis.

Das Rechteck in der linken oberen Ecke markiert die aktuelle Position in der Datei, die Tilde-Zeichen am linken Rand stehen für leere Zeilen. Die Zahlen geben die Position des Cursors numerisch an. „All“

oder im deutsch lokalisierten Vim „Alles“ steht dafür, dass derzeit das komplette Dokument im Fenster Platz findet.

Warum ist es nun nicht möglich, einfach loszutippen? Der Grund dafür liegt darin, dass Vim zwei Betriebsmodi unterscheidet: Er startet im Befehlsmodus, in dem der Anwender Dateien speichert, durch eine Datei navigiert und Zeichen löscht. Um in den Eingabe-Modus zu gelangen, geben Sie »i« ein. Der Editor quittiert das, indem er unten links »INSERT« anzeigt (Abbildung 2).

Im Eingabemodus können Sie Text ganz normal eintippen. Wenn die Terminal-Einstellungen stimmen und Sie eine moderne Vi-Variante wie Vim verwenden, ist es möglich, den Cursor mit den Pfeiltasten der Tastatur zu bewegen. Mit [Entf] oder der Backspace-Taste löschen Sie einzelne Zeichen, mit der Eingabetaste beginnen Sie eine neue Zeile. Um zurück in den Kommandomodus zu gelangen, drücken Sie [Esc]. Die Abwesenheit jeglicher Statusmeldungen in der linken unteren Ecke zeigt den Befehlsmodus an.

Der Vim-Editor wird normalerweise nur mit der Tastatur bedient. Auch wenn es einige Ansätze dazu gibt, Vim mit der Maus zu bedienen [8], zum Beispiel zum Scrollen, bleibt die Tastatureingabe die erste Wahl. Erfahrene Vim-Anwender bewegen sich ohne Maus mindestens genauso schnell durch ihre Dateien, und sie müssen auch nicht mit der Hand zwischen Maus und Tastatur hin- und herwechseln.

Vim ist sogar für Tastaturen ausgelegt, die keine besonderen Tasten besitzen. So können Anwender auch ohne Pfeiltasten durch die Dateien navigieren, daber dazu später mehr. Erst sei noch darauf hingewiesen, dass Vim auch bei Tastaturbefehlen zwischen Klein- und Großschreibung unterscheidet. Ein »J« bedeutet also normalerweise etwas anderes als ein »j«. Häufig hängen die damit verknüpften Befehle inhaltlich zusammen. So ersetzt beispielsweise »r« das aktuelle Zeichen, während »R« mehrere Zeichen ersetzt. Die Unterschiede sind dennoch groß genug, dass es sich lohnt, sich die Bedeutung der Tastaturbefehle einzeln zu merken.

Manche Befehle lassen sich auch nicht einfach so eingeben. Ihnen müssen Sie einen Doppelpunkt oder ein anderes Präfix oder eine andere Kommandotaste voranstellen. Wenn Sie zum Beispiel die Hilfe lesen möchten, geben Sie »:help« ein, gefolgt von der Eingabetaste. Mit »:q« verlassen Sie die Hilfe wieder. Andere Befehle besitzen eigene Präfixe, beispielsweise die Navigationstools, die mit der [Strg]-Taste eingeleitet werden.

Um den Editor zu beenden und die Datei(en) zu speichern, geben Sie »:qa« oder »:qw« ein. Wollen Sie den Editor verlassen ohne zu speichern, verwenden Sie stattdessen »:q!«. Mit »:w new *Dateiname*« speichern Sie den Inhalt des Editors in einer neuen Datei. Auch diese Befehle erscheinen beim Eintippen unten links, genauso wie Ihre Fehlermeldungen, wenn etwas schiefgeht (Abbildung 3).

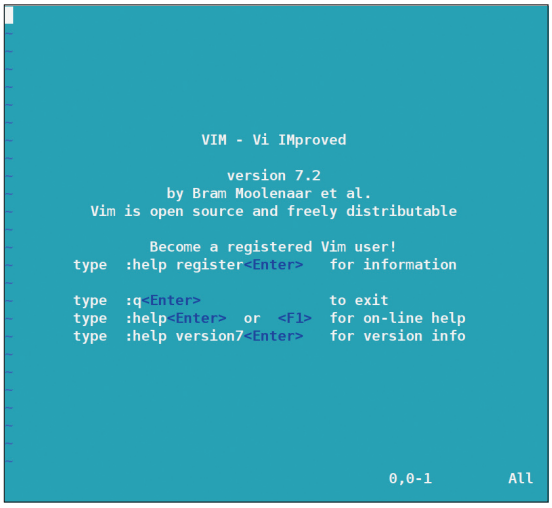

Abbildung 1: Ohne weitere Parameter gestartet, gibt Vim immerhin einige Wegweiser zu weiterer Hilfestellung.

Navigation

Die bis hierher gewonnenen Kenntnisse reichen aus, um Vim einigermaßen zu bedienen, aber weder besonders elegant noch sehr effizient. Die Profi-Stufe beginnt damit, sich nicht mit den Cursor-Tasten durch eine Datei zu bewegen, sondern mit den dafür vorgesehenen Buchstaben-Tasten. Die Taste »j« bewegt den Cursor eine Zeile nach unten, »k« nach oben. Die »l«-Taste verschiebt den Cursor nach rechts, die Taste »h« nach links. Alternativ dazu können Sie im Befehlsmodus den Cursor mit der Backspace-Taste ein Zeichen zurück bewegen, mit der Eingabe-Taste eine Zeile nach vorne.

Mit »0« springen Sie an den Anfang einer Zeile, mit »$« ans Ende. Diese Kürzel sind sehr praktisch, denn Vim beendet eine Zeile nur dann, wenn Sie sie explizit mit der Eingabetaste umbrechen. Ein automatischer Zeilenumbruch findet nicht statt. So ist zum Beispiel in HTML-Dateien oft ein ganzer Absatz eine Zeile, bis nach dem schließenden Tag ein Umbruch folgt. Mit »w« springt der Cursor zum Anfang des nächsten Worts, mit »b« zum Anfang des vorigen.

Vim bietet auch Tastaturkommandos, um sich in größeren Dateien schneller zu bewegen. So bewegt sich [Strg] + [f] bildschirmweise nach vorne, [Strg] + [b] ent-

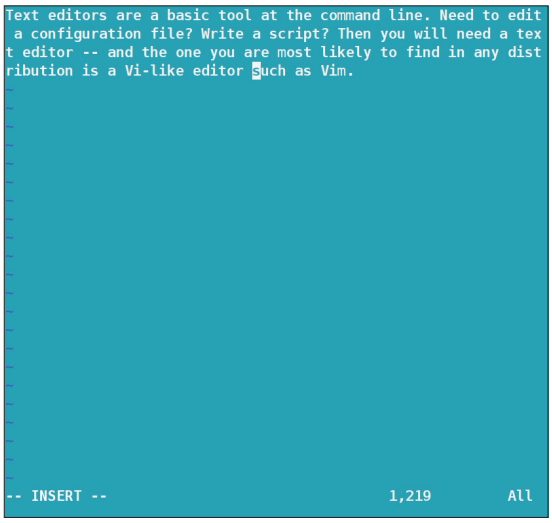

Abbildung 2: Vim besitzt zwei Modi: den Befehls- und den Eingabemodus. Zwischen den beiden Modi umzuschalten, ist der erste Schlüssel zur Bedienung des Editors.

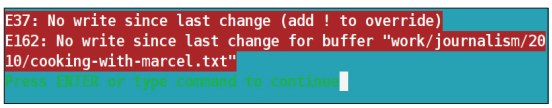

Abbildung 3: Ist die aktuelle Datei nicht gesichert, weist Vim beim Versuch zu beenden darauf hin.

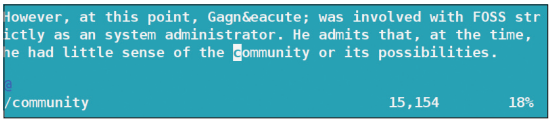

Abbildung 4: Die Vim-Suche springt direkt zum gefundenen String.

sprechend in die andere Richtung. Geht Ihnen das zu schnell, bieten [Strg] + [d] respektive [Strg] + [v] die halbe Schrittweite. Sie können auch direkt zu einer Zeile springen, wenn Sie »*Zeilennummer*G« eingeben, also die Zeilennummer gefolgt von einem »G«. Geben Sie nur »G« ein, springt Vim gleich zur letzten Zeile der Datei. Wenn Sie diese Befehle beherrschen, können Sie sich ungleich schneller durch Ihre Dateien bewegen als mit den Cursor-Tasten.

Textarbeit

Die meiste Zeit werden Sie vermutlich im Eingabemodus verbringen. Neben dem schon erwähnten Tastaturkürzel »i« gibt es noch einige andere Arten, in den Eingabemodus zu gelangen, die je nach Anwendungsfall effizienter sind. Zum Beispiel aktiviert »a« (append) den Eingabemodus so, dass Sie Ihren Text hinter der aktuellen Cursor-Position eingeben. Mit »I« fügen Sie neue Inhalte zu Beginn einer Zeile hinzu, »A« analog dazu am Ende. Verwandt dazu sind die Tastaturkürzel »o« und »O«, die oberhalb respektive unterhalb der aktuellen Zeile neuen Text einfügen lassen.

Fortgeschrittene Eingabemethoden sind zum Beispiel »cw«, das das aktuelle Wort (bis zum folgenden Leerzeichen) löscht und durch die Zeichen ersetzt, die Sie bis zum Beenden des Eingabemodus eintippen. Mit »R« ersetzen Sie alles ab der aktuellen Cursor-Position. Ähnlich dazu ersetzen Sie mit »C« die Zeichen der aktuellen Zeile, während »cc« erst die komplette Zeile löscht. Wie bereits erwähnt, ersetzen Sie mit »r« nur ein Zeichen und gelangen sofort zurück in den Befehlsmodus.

Ein einzelnes Zeichen löscht im Befehlsmodus das Tastenkürzel »x«. Eine bestimmte Anzahl von Zeichen löschen Sie mit »*Anzahl*x«. Der Tastenbefehl »D« löscht die Rest der Zeile rechts vom aktuellen Ort des Cursors, »dd« dagegen die ganze Zeile. Weil diese Befehle mit nur wenigen Tastendrucken einigen Schaden anrichten können, ist es praktisch, dass Vim über eine Undo-Funktion verfügt, die Sie mit »u« aktivieren. Der Editor führt eine lange Undo-History, mit der sie schrittweise zurückgehen können. Hilft alles nichts, können Sie immer noch mit »:q!« den Editor verlassen, ohne zu speichern. Das Undo für das Undo erreichen Sie mit dem Befehl »:redo«.

Cut-and-Paste ist üblicherweise eine Funktion, die man heute mit der Bedienung einer Maus assoziiert. Vim-Benutzer schaffen das aber aber mit dem Tastaturkommando »yy« (yank), das die aktuelle Zeile ausschneidet. Die Taste »p« fügt die Zeile wieder ein. Mehrere Zeilen schneidet der Befehl »*Zeilenzahl*yy« aus.

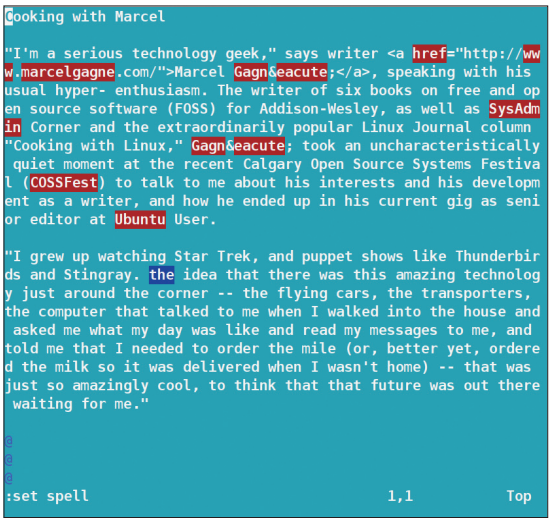

Abbildung 5: Vim kann in Texten die Rechtschreibung und Grammatik überprüfen.

Auch Suchen ist mit Vim nicht schwer. Dazu tippen Sie einfach im Befehlsmodus »/*Suchbegriff*« ein, und es geht los. Mit »n« springen Sie zum nächsten Fundort, mit »p« zum vorigen (Abbildung 4).

Die Rechtschreibprüfung aktiviert der Befehl »:set spell«. Rechtschreibfehler zeigt der Editor dabei in rot an, mögliche Grammatikfehler in blau (Abbildung 5). Mit »]s« bewegen Sie sich nach vorne durch die Fundstellen. mit »[s« zurück. Bei jedem Fehler zeigt »z=« eine Anzahl von Alternativen an, aus denen Sie über die Nummer auswählen oder den Fehler mit der Eingabe-Taste ignorieren. Über den Befehl »zg« fügen Sie ein als Fehler gekennzeichnetes Wort dem Wörterbuch hinzu.

Dieses Feature hängt wie viele weiter von den installierten Skripts und Plugins ab. Angesichts des Funktionsumfangs, den Vim schon so bietet, verwundert es nicht, dass viele Anwender den Editor auch zur Textverarbeitung verwenden. Wer mit seinen Befehlen vertraut ist, kann damit tatsächlich wesentlich schneller arbeiten als beispielsweise mit Open Office.

Bequem

Wie bei anderen Kommandozeilen-Tools gilt auch bei Vim eine spezielle Kosten-Nutzen-Rechnung: Der Lernaufwand mag höher sein als bei Desktop-Programmen wie Open Office oder Koffice, aber in den Händen des geübten Benutzers sind sie unschlagbar.

Jeder Linux- und Unix-Anwender sollte sich mindestens so viele Vim-Kenntnisse aneignen, dass er eine Datei öffnen, editieren und speichern kann, denn schnell findet man sich in einer Umgebung wieder, in der Vi(m) der einzige verfügbare Editor ist. Dafür genügen schon ein halbes Dutzend Befehle. Die kann man sich auf eine Karteikarte schreiben und neben den Computer legen, bis man genügend mit ihnen vertraut ist. Das kann man dann mit einer weiteren Handvoll von Befehlen wiederholen. Innerhalb weniger Tage meistert man so den Weg vom totalen Einsteiger zum geübten Vim-Anwender. *(ofr)* ■

Infos

[1] Vim: [http://www.vim.org/]

[2] Bruce Byfield, Emacs Basics, ADMIN 06/2010, S. 66

[3] Nvi: [http://en.wikipedia.org/wiki/Nvi]

[4] Elvis: [http://elvis.the-little-red-haired-girl.org/]

[5] Busybox: [http://www.busybox.net/]

[6] Joe's Own Editor: [http://sourceforge.net/projects/joe-editor/]

[7] GNU Nano: [http://www.nano-editor.org/]

[8] Mausbenutzung in Vim: [http://vim.wikia.com/wiki/Using_the_mouse_for_Vim_in_an_xterm]