SelfLinux-0.10.0

Praxisorientiertes vim-Tutorial

Autor: Johnny Graber (*selflinux@jgraber.ch*)
Formatierung: Frank Börner (*frank@frank-boerner.de*)

Dieser Text ist eine Zusammenstellung von Johnny Graber aus der Arbeit von Wolfgang Jährling (Praxisorientiertes VI-Tutorial von Pro-Linux) und der von Martin "Herbert" Dietze (Vi-Kommandos http://www.fh-wedel.de/~herbert/html/vi/).

Daher an dieser Stelle ein grosses Danke dafür, das wir auf die beiden Texte zurückgreifen konnten.

Inhaltsverzeichnis

1 Warum vi(m)?

Der Editor vi ist seit sehr langer Zeit ein Grundbestandteil von UNIX-Systemen. Damit ist er vermutlich eines der ältesten Programme, die noch verbreitet sind. Man mag sich fragen, warum man sich mit diesem Urgestein beschäftigen soll. Die Gründe sind wohl hauptsächlich die folgenden:

* vi ist auf wohl jedem UNIX-System anzutreffen (und auch auf vielen nicht-Unix-Systemen).
* vi ist ein extrem mächtiger Editor, aber dennoch klein und schnell.
* vi zu benutzen ist "cool" B-)

Es gibt viele vi-Varianten. Die zweifellos beliebteste ist der "Vi IMproved" (vim). Daher bezieht sich dieser Artikel zum Teil speziell auf diesen (selbstverständlich wurde dieser Artikel auch damit geschrieben).

2 Eine Datei erstellen

Als erstes werden wir uns anschauen, wie man in vim eine Datei erstellt und speichert. Starten Sie vim durch Eingabe von `vi` oder `vim`.

Im vi(m) gibt es 3 Modi. Nach dem Start befinden wir uns im Kommandomodus. Um einen Text einzugeben, müssen wir in den Eingabemodus wechseln. Dazu gibt es mehrere Möglichkeiten. Um den Eingabemodus am Ende der aktuellen Zeile zu beginnen (d.h. mit dem Cursor ans Ende springen und dort den Eingabemodus starten), können wir `A` (großes a; "Append" = "anhängen") drücken. Um ihn an der aktuellen Cursorposition zu beginnen, drücken Sie nun `i` ("insert" = "einfügen").

Jetzt können Sie einen beliebigen Text tippen. Zuletzt wollen wir die Änderungen speichern. Verlassen Sie dazu den Eingabemodus über die ESC Taste. Nun sind wir wieder im Kommandomodus.

Für viele Dinge, darunter auch das Öffnen und Speichern von Dateien, müssen wir in den "ex-Modus" wechseln. Das machen wir durch die Taste `:`. Nun kann ein ex-Kommando eingegeben werden.

Um den Text in die Datei "test01.txt" zu schreiben, geben Sie `:w test01.txt` ein, gefolgt von ENTER. Abbrechen können Sie ein im ex-Modus eingegebenes Kommando mit ESC. In beiden Fällen befinden Sie sich danach wieder im Kommandomodus.

3 Verlassen des vi

Um vi(m) zu beenden, brauchen wir wieder ein ex-Kommando. Geben Sie, wenn Sie im Kommandomodus sind, `:q` ein. Dadurch wird vi unter der Bedingung verlassen, dass die aktuelle Datei gespeichert wurde.

Sollten Sie einmal Ihre Änderungen verwerfen wollen, verlassen Sie vi mit `:q!`.

4 Eine Datei öffnen

Sie können vi beim Aufruf von der Shell einen Dateinamen übergeben. Dadurch wird diese Datei geöffnet. Geben Sie dazu beispielsweise `vi test01.txt` in der Shell ein.

Alternativ können Sie auch im vi mit dem ex-Kommando `:e test01.txt` eine Datei öffnen.

Wenn Sie eine Datei auf eine dieser Weisen geöffnet oder beim Speichern einen Dateinamen angegeben haben, brauchen Sie nun beim nächsten Speichern nicht mehr den Dateinamen eingeben. Es genügt das ex-Kommando `:w`. Um zu speichern und den Editor zu verlassen genügt `:wq`.

5 Mehrere Dateien öffnen

Sie können vim beim Starten auch mehrere Dateinamen übergeben: `vi *.cpp *.h`. Dabei wird die zuerst genannte Datei direkt geöffnet und kann bearbeitet werden. Wenn Sie mit der Bearbeitung der nächsten beginnen wollen, erreichen Sie dies über den ex-Befehl `:next`. Zur vorherigen Datei gelangen Sie entsprechend mit `:prev`.

6 Bestimmte Stellen wiederfinden

Beim Editieren einer langen Datei, insbesondere eines Quellcodes, ist es sehr praktisch, wenn man Markierungen setzen kann, um später schnell diese Stellen wiederzufinden.

Um eine Markierung zu setzen, drücken Sie (im Kommandomodus) die Zeichenfolge `mx`, wobei x ein beliebiger Kleinbuchstabe ist. Um später zu dieser Markierung zu springen, genügt ein (wieder im Kommandomodus) eingegebenes `'x`, wobei x wieder der gleiche Buchstabe ist. Sie können für Markierungen entweder Buchstaben (a-z und A-Z) oder ganze Wörter verwenden. Diese Markierungen können auch überall dort verwendet werden, wo Zeilenangaben auftreten.

7 Kopieren und Einfügen

Sie haben in vi nicht nur die Möglichkeit, 26 Markierungen zu setzen, Sie können auch 26 Puffer zum Kopieren und Einfügen verwenden (was ein großer Vorteil von vi gegenüber anderen Editoren ist).

Drücken Sie `v` um den Anfang des Bereiches zu markieren den Sie kopieren wollen. Bewegen Sie den Cursor zum Ende. Dabei wird der markierte Bereich hervorgehoben. Um den markierten Bereich z.B. in den Puffer h zu kopieren genügt nun ein `"hy` (Das Anführungszeichen steht für "Puffer", das "y" steht für "yank", also kopieren). Einfügen können Sie den Puffer bei Bedarf mit `"hp` ("p" für "paste, einfügen).

Anstatt einen Puffer über einen Buchstaben anzusprechen, können Sie auch einfach einen Standardpuffer verwenden. Dazu lassen Sie jeweils das Anführungszeichen und den Puffernamen weg.

8 Schnelles Navigieren durch den Text

Im Kommandomodus können Sie sich mit den Cursortasten durch den Text navigieren. Sie können dabei auch eine Zahl voranstellen. Wenn Sie z.B. `30` eingeben und danach die `NACH-LINKS` -Taste drücken, bewegt sich der Cursor um 30 Zeichen nach links.

Das Voranstellen von Zahlen als Wiederholungsfaktor funktioniert bei fast allen Befehlen.

Mit `b` können Sie ein Wort zurück und mit w eines nach vorne. Einen Satz zurück kommen Sie mit `(` und einen vor entsprechend mit `)`. Auch hier geben vorangestellte Zahlen den Wiederholungsfaktor an: `5(` bewegt den Cursor 5 Sätze zurück und mit `4w` gelangen Sie 4 Wörter nach vorn.

9 Löschen von Text

Im Eingabemodus ist es relativ umständlich, Text zu löschen. Wenn Sie eine komplette Zeile entfernen wollen, stehen Sie sogar vor einem sehr schwierigen Problem. Daran merkt man schon, dass der Eingabemodus wirklich nur zum Eingeben von Text gedacht ist und nicht zum Editieren.

Im Kommandomodus steht uns zum Löschen von Text das Kommando `d` zur Verfügung. Danach folgt ein Buchstabe, der kennzeichnet, auf was für eine Texteinheit wir uns beziehen (Wort, Satz, Zeile, ...). Der gelöschte Text wird zusätzlich in die Standard-Zwischenablage eingefügt.

Um ein Wort zu löschen, drücken Sie `dw`. Das Wort hinter dem Cursor löschen Sie entsprechend mit `db`. Auch hier ist es wieder möglich, eine Zahl voranzustellen. So löschen Sie mit `3dd` 3 Zeilen und mit `5dw` 5 Wörter.

Bis zum Anfang des aktuellen Satzes löschen Sie mit `d(`, bis zum Ende mit `d)`.

Bis zum Anfang der aktuellen Zeile kann man mit `d^` oder auch mit `d0` löschen und bis zum Ende mit `d$`.

Bei einigen Kommandos kann man auch einen Bereich voranstellen, auf den sie sich beziehen sollen. So entfernen Sie mit `1,5d` die ersten fünf Zeilen einer Datei. Statt Zeilennummern können Sie auch Markierungen verwenden: `'a,'bd`.

10 Suchen und Ersetzen allgemein

Suchen und Ersetzen können Sie in vi mit regulären Ausdrücken. Hier eine Einführung in reguläre Ausdrücke zu geben würde den Rahmen dieser Kurz-Einführung sprengen.

Wenn Sie nur nach einem Wort suchen, können Sie das mit `/suchbegriff/` machen. Wenn Groß-/Kleinschreibung ignoriert werden soll, verwenden Sie `/wort/i`.

Um Text zu ersetzen, müssen Sie in den ex-Modus wechseln. Folgendes Kommando ersetzt alle Vorkommen von "Unix" mit "Linux": `:1,$s/Unix/Linux/g` (Bereich: 1,$, also vom Anfang bis zum Ende der Datei "s" für "substitude", "ersetzen". Das "g" am Ende steht für "global". Wird das g weggelassen, wird bei jeder Zeile bloss das erste Vorkommen von Unix ersetzt. Sehr nett ist es, dass man eben auch innerhalb eines bestimmten Bereiches die Ersetzung durchführen lassen kann, d.h. man markiert z.B. per `ma` und `mb` zwei Zeilen und verwendet dann `:'a,'bs/foo/bar/g`.

Hängt man am Ende des Suchtextes noch ein /c an, fragt vi vor dem Ersetzen nach. Dies funktioniert auch in der Kombination mit /g. Somit lautet das vorherige Kommando, ergänzt um das Nachfragen, `:'a,'bs/foo/bar/gc`.

11 Praktische Tips

Wenn Sie in vi zwei Zeilen zusammenfügen wollen, können Sie das leider nicht so machen, wie Sie es von anderen Editoren gewohnt sind. In vi ist eine Zeile eine Einheit und das Newline-Zeichen am Ende ist kein Zeichen, das gelöscht werden kann (Moderne vi-Klone bieten diese Möglichkeit im Eingabemodus). Stattdessen müssen sie im Kommandomodus `J` drücken, wodurch die nächste Zeile an die aktuelle angefügt wird. Der Cursor braucht dabei nicht am Ende einer Zeile zu sein.

Recht praktisch ist auch die Möglichkeit, die Ausgabe eines Unix-Kommandos in den Text einzufügen. Wollen Sie z.B. die aktuelle Uhrzeit in den Text einfügen, verwenden Sie `:r !date`. Das Einfügen einer Datei geht hingegen per `:r datei`.

Buchstabendreher kann man per `xp` korrigieren, zwei Zeilen vertauschen können Sie per `ddp`. Probieren Sie es aus!

11.1 Rechtschreibprüfung mit aspell

Mit `GNU aspell` (http://aspell.net/) gibt es eine gute und freie Rechtschreibprüfung. Da auf ein grafisches Frontend verzichtet wurde, kann man `aspell` direkt im `vim` benutzen. Dafür müssen Sie ihre `.vimrc` lediglich um die untenstehende Zeile ergänzen:

.vimrc

```
:map <F12> :w!<CR>:!aspell --lang=german check %<CR>:e! %<CR>
```

Wenn Sie nun auf die Taste `F12` drücken, wird `aspell` gestartet und beginnt mit der Prüfung des geladenen Textes. An der Stelle von `F12` kann man auch andere Tasten verwenden, doch muss man darauf achten, nicht eine häufig gebrauchte Funktion zu überlagern.

11.2 Mit vim Verzeichnisse durchsehen

```
user@linux ~/ # vim .
```

Mit diesem Befehl starten Sie bei den neueren Klonen `vim` im **mc**-Modus. `vim` präsentiert nun keine neue Datei, sondern den Inhalt des aktuellen Ordners.

Mit `?` bekommen Sie eine Liste mit speziellen Befehlen, die in diesem Modus verwendet werden können. Sie können den Cursor auf das gewünschte Verzeichnis verschieben und dieses durch betätigen von `ENTER` betreten. Wenn Sie dies bei einer Datei machen, wird diese in `vim` geöffnet und kann wie gewöhnlich bearbeitet werden.

Zum Verlassen des **mc**-Modus dient das bekannte Kommando `:q`

12 vim-Konfiguration

Der `vim` besitzt eine Konfigurationsdatei mit dem Namen $HOME/.vimrc. Hier können ganz normale Befehle eingetragen werden, die beim Starten dann abgearbeitet werden. Während man Syntax-Highlighting für alle

möglichen Programmiersprachen mit `:syntax on` direkt beim Arbeiten einschaltet, kann man entsprechend in die .vimrc einfach `syntax on` eintragen. Vim beherrscht über 120 verschiedene Programmier- und Beschreibungssprachen. Natürlich kann man auch Hervorhebungsregeln selbst zusammenstellen.

Das Kommando `syntax on` funktioniert nur, wenn vim entsprechend compiliert wurde. Die meisten Distributionen enthalten mehrere vim-Pakete (z.B. vim-enhanced und vim-minimal), wobei `syntax on` nur in der ersten Version funktioniert.

Wer noch nie mit `vi` gearbeitet hat, wird nicht daran gewöhnt sein, dass die Backspace- und die Entf-Tasten Zeilenumbrüche nicht wie normale Zeichen behandelt (sie also nicht löschen können). Dieses Verhalten kann man durch Eintragen von `set bs=2` in die .vimrc ändern.

Wenn Sie beim Editieren automatische Einrückung bevorzugen, versuchen Sie mal `set ai`.

13 Wichtige vi-Kommandos

Konvention Im folgenden werden einige oft benutzte vim-Kommandos aufgelistet. Fast alle diese Kommandos sind für den Kommandomodus. ex-Kommandos werden durch das `:` am Anfang gekennzeichnet. Manche Kommandos haben noch ein [Count] vorangestellt. Das heißt, dass das Kommando normalerweise einmal, bei einer vorher gedrückten Zahl n aber n-mal ausgeführt wird.

13.1 Starten

Komando	Beschreibung
`vi`	Aufruf von vi mit leerem Text-Puffer.
`vi Dateiname`	Datei wird geladen und der Cursor bei der ersten Zeile platziert.
`vi + Dateiname`	Datei wird geladen und der Cursor bei der letzten Zeile platziert.
`vi +n`	Dateiname Datei wird geladen und der Cursor bei der n-ten Zeile platziert.
`vi +/Zeichenkette Dateiname`	Datei wird geladen, der Cursor bei der Zeile mit Zeichenkette plaziert.

13.2 Beenden

Komando	Beschreibung
`:wq`	Speichern und vi verlassen.
`:q`	vi verlassen, falls Datei unverändert.
`:q!`	vi verlassen, egal ob Datei verändert oder nicht.
`:w`	Datei speichern.

13.3 Dateien laden

Komando	Beschreibung
`e Datei`	Datei wird geladen, wenn sie existiert, ansonsten erzeugt.
`:next`	Die nächste Datei wird geladen, falls vi mit mehreren Dateien aufgerufen wurde.
`:prev`	Die vorherige Datei wird geladen, falls vi mit mehreren Dateien aufgerufen wurde.

13.4 Cursorbewegungen

Komando	Beschreibung
`[Count]j`	Den Cursor um eine (bzw. Count) Zeile runter u.s.w.
`[Count]k`	Den Cursor um eine (bzw. Count) Zeile rauf u.s.w.
`[Count]l`	Den Cursor um ein (bzw. Count) Zeichen rechts u.s.w.
`[Count]h`	Den Cursor um ein (bzw. Count) Zeichen links.
`[Count]w`	Den Cursor um ein (bzw. Count) Wort rechts.
`[Count]b`	Den Cursor um ein (bzw. Count) Wort links.
`[Count]h`	Den Cursor um ein (bzw. Count) Zeichen links.
`[Count]G`	Springe zum Ende der Datei oder, falls Count gegeben, zu Zeile

	Count.
`Ctrl-f`	Page-Down.
`Ctrl-b`	Page-Up.
`^`	Springe zum Anfang der aktuellen Zeile.
`$`	Springe zum Ende der aktuellen Zeile.

13.5 Text eingeben

Komando	Beschreibung
`i`	(insert), Eingabe vor dem aktuellen Zeichen.
`a`	(append), Eingabe nach dem aktuellen Zeichen.
`I`	(Insert), Eingabe am Anfang der aktuellen Zeile.
`A`	(Append), Eingabe am Ende der aktuellen Zeile.
`o`	neue Zeile und Eingabe nach der aktuellen Zeile.
`O`	neue Zeile und Eingabe vor der aktuellen Zeile.
`Ctrl-v`	Eingabe eines Steuerzeichens.

13.6 Text ändern

Komando	Beschreibung
`[Count]rZeichen`	(replace), Änderung des aktuellen Buchstaben in Zeichen.
`R`	(Replace), Überschreibmodus vom aktuellen Buchstaben aus.
`cwWort`	ersetzt das Wort vor dem Cursor durch Wort.
`ccZeichenkette`	ersetzt die aktuelle oder nächste Zeile durch Zeichenkette.
`J`	hängt die der aktuellen folgende Zeile an die aktuelle an.

13.7 Text löschen

Komando	Beschreibung
`[Count]x`	1 (bzw. Count) Zeichen unter dem Cursor (rechts) wird gelöscht.
`[Count]X`	1 (bzw. Count) Zeichen links vom dem Cursor wird gelöscht.
`D`	löscht von der Cursorposition bis zum Zeilenende.
`[Count]dd`	1 (bzw. Count) Zeilen werden gelöscht.
`[Count]d[Richtung]`	1 (bzw. Count) mal wird in Richtung [Richtung] gelöscht.

13.8 Die Zwischenablagen

Komando	Beschreibung
`"1..0, a..z`	Die Ablage 1..0 bzw. a..z für die nächste Aktion auswählen.
`[Count]y[Richtung]`	1 (bzw. Count) Bewegungen in [Richtung].
`[Count]yy`	1 (bzw. Count) Zeilen werden in die aktuelle Zwischenablage kopiert.
`Beliebige Löschaktion`	Gelöschter Text wird in die aktuelle Zwischenablage kopiert.
`p`	Inhalt der Zwischenablage wird hinter dem Cursor eingefügt.
`P`	Inhalt der Zwischenablage wird vor dem Cursor eingefügt.

13.9 Suchen und Ersetzen

Komando	Beschreibung
`/Regex`	Suche vorwärts nach dem regulären Ausdruck Regex.
`?Regex`	Suche rückwärts nach dem regulären Ausdruck Regex.
`n`	Wiederholt das letzte Suchkommando.
`N`	Wiederholt das letzte Suchkommando in die jeweils andere Richtung.
`fZeichen`	Sucht nach Zeichen in der aktuellen Zeile vorwärts.
`FZeichen`	Sucht nach Zeichen in der aktuellen Zeile rückwärts.
`:%s/Quelle/Ziel/`	Ersetzt Quelle textweit beim 1. Vorkommen in der Zeile durch Ziel.
`:%s/Quelle/Ziel/g`	Ersetzt Quelle im Text überall durch Ziel.
`:%s/Quelle/Ziel/gc`	Ersetzt Quelle im Text überall durch Ziel, fragt aber vorher nach.

13.10 Bookmarks

Komando	Beschreibung
`mKey`	Setzt eine Marke an der aktuellen Stelle unter dem Namen der Taste Key.
`'Key`	Springt zu der Zeile mit der Marke Key.
`` `Key ``	Springt zu der Stelle mit der Marke Key.

13.11 Sonstige Goodies

Komando	Beschreibung
`...`	Wiederholt die letzte Editieraktion,
`%`	(über einer Klammer) Springt auf die korrespondierende Klammer\end{center}.
`:u`	(undo) Rückgängig.
`:r`	(redo) Wieder herstellen.