Mac OS X Consoliero

Weiterführende Dokumentationen für Administratoren.

Der vi-Editor

Christoph Müller, PTS

Version	Ersteller	Datum	Prüfung	Druckdatum	Freigabe
1.1	Christoph Müller	5.10.2004 21:47	15.10.2004	21.10.2004 22:03	Christoph Müller

Mac OS X Consoliero:
Der vi-Editor

Inhaltsverzeichnis

Mac OS X Consoliero:
Der vi-Editor

Einleitung

Wer kennt es nicht: Da editiert man fleißig Konfigurationsdateien mit pico. Keine Probleme, alles ist klar. Nun loggt man sich auf einen anderen PC ein und prompt gibt es da kein pico mehr. Deswegen wäre es doch schön wenn man einen Editor hätte der auf allen UNIX-Derivaten vorhanden wäre, so müsste man die Befehle nur einmal lernen. vi ist ein solcher Editor welcher bei jeder Distribution dabei ist.
Viele Benutzer schrecken jedoch von vi zurück. Die vielen Features und Funktionen können verwirren. Diejenigen welche vi schon einmal angeschaut haben sind verwirrt, dass der Rechner immer Warntöne von sich gibt, und wenn man was schreibt, nie das erscheint was man eigentlich geschrieben hat.
Dabei ist es wie mit allem! Eigentlich ist ganz einfach und logisch. Zudem ist man als Administrator mit einer Hand voll Befehlen völlig ausreichend ausgestattet um seine Konfigurationsdateien elegant zu bearbeiten.

Christoph Müller, www.pts.ch

Konventionen

Wenn im Text ein **^X** angezeigt wird, bedeutet das einen so genannten „controll character". Eingegeben wird dieser mit „ctrl" + „X" Taste. Befehle sind in Courier und **Fett** gehalten. Also in etwa:

```
vi testfile.txt
```

Ausgaben des Terminal werden in Courier gehalten, werden aber nicht fett gedruckt.

```
tcsh: was: Command not found
```

Starten und Verlassen von vi

Was die meisten Leute verwirrt ist, dass vi in „command-mode" startet. In diesem Modus können Befehle an vi übergeben werden. Möchte man Text editieren oder einfügen, muss man dazu im „input-mode" sein. Aber alles der Reihe nach.

vi wird wie pico mit dem Syntax: Befehl Dateinamen gestartet. Mit

```
vi testfile.txt
```

öffnet man also eine vorhandene Datei. Wenn diese Datei leer ist, wird auf dem Bildschirm eine ganze Menge Tilden (~) angezeigt. Jede Zeile am Ende der Datei wird so angezeigt. Am unteren Ende des Terminals wird der Dateiname, die Anzahl der Zeilen und die der Zeichen welche in der Datei vorhanden sind, angegeben.

```
~
~
"testfile.txt" 1L, 22C
```

Um vi wieder zu verlassen muss man im „command-mode" sein. Normalerweise startet vi in diesem Modus. Um in diesen Modus zu gelangen, drückt man die „Escape"- oder „Esc"-Taste. Wenn man schon im „command-mode" ist, zeigt uns ein Warnton an, dass wir uns schon im richtigen Modus befinden.
Die „Esc"-Taste bringt einem nur in den „command-mode", nicht aber zurück. Der Befehl vi zu verlassen, ist **:q**. Tippen Sie also einen Doppelpunkt „:" gefolgt von einem „q".

Normalerweise verlässt man einen Editor in dem man die gemachten Änderungen speichert. Der Befehl dazu ist **:w**. Dieser Befehl kann mit dem Verlassen-Befehl kombiniert werden: **:wq**. Möchte man vi verlassen, ohne dass die Änderungen gespeichert werden, macht man das mit **:q!**.

Möchte man die Änderungen unter einem neuen Namen abspeichern, kombiniert man ebenfalls den Schreib-Befehl mit dem Dateinamen. Also **:w config.plist_old**. Möchte man vi verlassen und speichern, kann man das natürlich kombinieren. Die Reihenfolge der Befehle muss in ihrer logischen Abfolge eingegeben werden. Man kann also nicht vi verlassen und dann speichern. So macht **:qw datei2** keinen Sinn. Sondern **:wq datei2**.

Der command-mode und der insert-mode

Wie wir gesehen haben, startet vi im command-mode. Das heißt wir sehen den Text, können aber trotzdem Zeichen über die Tastatur eingeben welcher aber nicht in der zu editierenden Datei erscheint. Wollen wir Text in eine Datei schreiben, müssen wir dazu in den insert-mode wechseln.

Spielen wir das an einem Beispiel durch. Wir möchten gerne in hostconfig-Datei unseren Appletalk-Namen ändern. O.K. ich weiß, wir könnten das auch in der Systemsteuerung tun, aber das wäre ja langweilig.

Öffnen wir also die Datei als Superuser (sudo):

```
[Ti-Surfer:/Users/pts] la% sudo vi /etc/hostconfig
```

Nun befinden wir uns im Editor. Wir sehen die Datei im command-mode. Mit den Pfeiltasten können wir uns in der Datei bewegen. Mit **^F** und **^B** können wir eine Seite vor, oder eine Seite zurück blättern.

Wir suchen also mit diesen Möglichkeiten unseren Eintrag in der hostconfig-Datei, bis wir den entsprechenden Eintrag gefunden haben. Dieser ist in unserem Beispiel fast am Ende der Datei. Das Problem ist nun aber, wir können mit dem Cursor zwar auf die entsprechende Stelle, können aber nichts einfügen.
Dazu müssen wir in den insert-mode wechseln. Dies geschieht durch drücken der Taste **i**. vi zeigt uns an, dass wir im insert-mode sind (Abbildung 1). vi zeigt es, farblich getrennt, am Ende des Terminals an.

Löschen Sie nun den alten Namen und schreiben Sie einen Neuen darüber.

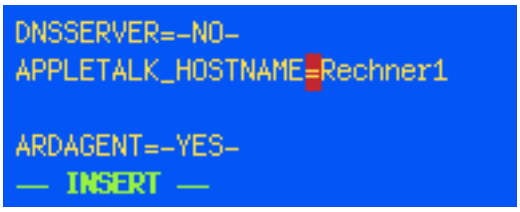

Abbildung 1

Haben Sie Ihre Änderungen vorgenommen, wechseln Sie mit der Esc-Taste wieder in den command-mode und speichern Ihre Änderungen mit dem Befehl **:wq**

Natürlich können Sie die Änderungen auch verwerfen, indem Sie ein **:q!** eingeben.

Mehr zum Öffnen von Dateien

Wir haben jetzt eine einzelne Datei geöffnet. vi kann aber auch mehrere Dateien miteinander öffnen. Wenn Sie wissen wie die Dateien heißen, geben Sie diese im Terminal einfach durch einen Abstand getrennt ein.

```
[Ti-Surfer:~] pts% vi test.txt test2.txt
```

Im vi-Editor können Sie dann im command-mode mit **:n** (für next) zur nächsten Datei wechseln. Natürlich kann man auch wildcards zum öffnen von Dateien einsetzen. Um zum Beispiel alle html-Dateien zu öffnen, kann man das in etwa so tun:

```
[Ti-Surfer:~] pts% vi *.html
```

Importieren von Dateien in vi

In vi kann man auch mehrere Dateien zu einer zusammenfügen. Dazu öffnet man zuerst die erste Datei mit vi. Im command-mode gibt man dann den **:r** gefolgt vom Namen der zweiten Datei an. Die zweite Datei wird ab der Zeile eingefügt, auf welcher der Cursor steht. Um eine Datei am Ende anzuhängen, geht man mit dem Cursor auf die letzte Zeile.

```
[Ti-Surfer:~] pts% vi test.txt

Das ist ein Test
Das ist eine weitere Zeile
~
~
~
~
~
~
~
~
~
~
~
:r test2.txt
```

```
Das ist ein Test
Das ist eine weitere Zeile
Das ist der Text der zweiten Datei.
~
~
~
~
~
~
~
~
~
~
~
"test2.txt" 1L, 36C
```

Löschen und erfassen von Text in vi

Das löschen und erfassen von Text in vi ist etwas komplizierter als zum Beispiel in pico. Hat man sich aber einmal daran gewöhnt, möchte man diese Features nicht mehr missen. In pico platziert man den Cursor genau dort, wo man eine Veränderung haben möchte. Diese Änderung gibt man ein und sie passiert unmittelbar.
In vi kann man das auch. Jedoch muss man jeweils noch spezifizieren, wo die Veränderung stattfinden soll. Was am Anfang unglaublich schwerfällig ist, entpuppt sich bald schon als power-feature. Probieren Sie es aus, und gewöhnen Sie sich daran.

Die Befehle in den folgenden Tabellen werden nur im command-mode ausgeführt und führen dazu, dass vi in den insert-mode wechselt.

Tabelle 1: vi-Befehle um Text hinzuzufügen

Befehl:	Funktion:
a	Fügt Text nach dem Cursor ein
A	Fügt Text am Ende der aktuellen Zeile ein
i	Fügt Text vor dem Cursor ein
I	Fügt Text am Anfang der Zeile ein
o	Fügt eine leere Zeile nach dem Cursor ein
O	Fügt eine leere Zeile vor dem Cursor ein

Tabelle 2: vi-Befehle um Text zu löschen

Befehl:	Funktion:
x	Löscht ein Zeichen unter dem Cursor
X	Löscht ein Zeichen hinter dem Cursor
dd	Löscht die aktuelle Zeile
5dd	Löscht fünf Zeilen beginnend mit der aktuellen Zeile. Dies funktioniert mit jeder Anzahl Zeilen
dw	Löscht das aktuelle Wort
cw	Wechselt das aktuelle Wort. Löscht es und ersetzt es mit dem Wort welches man danach eintippt
r	Ersetzt das Zeichen auf dem der Cursor liegt mit einem anderen

R	Ersetzt den vorhandenen Text mit dem neu eingegebenen

Tabelle 3: Andere nützliche vi-Befehle

Befehl:	Funktion:
Y	Kopiert die aktuelle Zeile
ye	Kopiert das aktuelle Wort
p	Fügt die kopierte Zeile ein
J	Fügt die aktuelle und die nächste Zeile zusammen
u	Widerruft die letzte Änderung
U	Widerruft alle Änderungen auf der aktuellen Zeile
.	Wiederholt den letzten Befehl

Copy Paste in vi

Um ein einzelnes Wort zu kopieren, brauchen wir also, wie aus der Liste von oben ersichtlich ist, den Befehl **ye**. Mit dem Befehl **p** kopieren wir das Wort an der aktuellen Cursor-Position wieder in das Dokument. vi hat aber nicht nur eine Zwischenablage, sondern mehrere solcher. Diese werden „general purpose buffers" genannt und können selber definiert werden. Es ist in vi also möglich eine Konfigurationsdatei durchzugehen und während dem Durchsehen mehrere Zeilen oder Wörter, nacheinander zu kopieren.

Das füllen und definieren dieser Buffer geschieht, indem ich den normalen Kopierbefehl **y** benutze. Vor dem Kopierbefehl sage ich vi noch, dass ich es gerne in einem Buffer gespeichert hätte. Dies geschieht mit dem Befehl **"**. Genau einem Anführungszeichen! Danach muss ich noch den Namen des Buffers definieren. Also in etwa so:

```
"1y
```

Dieser Befehl kopiert die aktuelle Zeile in den Buffer „1".

Um diese, im Buffer 1, gespeicherte Zeile wieder einzusetzen benutze ich den Einfüg-Befehl **p**. Wie beim Kopieren muss ich vi mitteilen, welchen Buffer ich verwenden möchte. Also so:

```
"1p
```

Suchen und Ersetzen in vi

Einer der großen Vorteile von vi gegenüber pico, ist sicher die Möglichkeit, dass man Textmuster (einen regulären Ausdruck, um es in UNIX-Lingo zu sagen) in einer Datei suchen und ersetzen kann. In vi können Sie im command-mode mit **/suchtext** nach dem Text suchen. Drücken Sie die Taste **N** um zum nächsten Fundort zu kommen.

Um den Text auch gleichzeitig zu ersetzen, muss man den Befehl **:%s** mit folgenden Syntax verwenden: **.%s/suchen nach/ersetzen mit/**. In Beispiel unten wird der Text Windows mit WindowsXP ersetzt.

```
[Ti-Surfer:~] pts% vi search.txt

1 Apple
2 Windows
3 Linux
4 BSD
~
~
~
~
:%s/Windows/WindowsXP/

1 Apple
2 WindowsXP
3 Linux
4 BSD
~
~
~
~
"search.txt" 4L, 34C
```

Enthält der Text nicht nur ein gesuchtes Textmuster welches geändert werden soll, kann dem Befehl einfach noch ein **g** anhängt werden. Somit sind ab der aktuellen Cursorposition alle zutreffenden Textstellen ersetzt.

```
:%s/Windows/WindowsXP/g
```

Christoph Müller - www.pts.ch

Publishing Tools Support
Rüschlikon, 21.10.2003

Bei Fragen oder Anmerkungen, kontaktieren Sie mich bitte unter chm@pts.ch

Weitere detaillierte Informationen erhalten Sie aus meinem Buch: **„Mac OS X „Consoliero-Client" Praxis Handbuch"**: ISBN-Nr. 3-905647-17-6.