

# vim Version 7: Warum wechseln?

UNIX-Stammtisch Dresden, 5.7.2006 u.Z.

Reinhard Wobst, r.wobst@gmx.de

## Gründe für Wechsel:

- ◆ leckere neue Features, die das Leben leichter machen
- ◆ problemloses Weiterarbeiten (praktisch 100% aufwärtskompatibel), also sehr geringe Kosten
- ◆ sehr einfache Übersetzung unter Linux, individuelle Konfigurationsmöglichkeiten (spricht gegen fertige rpm's)

→ geringer Aufwand beim Wechsel

Mehr zur Installation weiter unten.

## Neues Features

### Quantitative Fortschritte

- ◆ 481 Dateitypen automatisch erkannt (Syntaxfärbung) - vim6.3: 73 weniger
- ◆ syntaxabhängiges Einrücken für 74 Sprachen und Dateiarten (vim6.3: 20 weniger).

- ◆ Weitere Syntaxfiles können jederzeit in `$VIMRUNTIME/indent/` bzw. `.../syntax/` untergebracht werden (über [1] zu finden).

## Klammerfärbung:

- ◆ Steht der Cursor auf einer Klammer - (, [, { -, wird die zugehörige andere Klammer (falls vorhanden) zusammen mit dieser eingefärbt. **Beispiel**
- ◆ Ist nur ein "kleines Feature", aber wird schnell hilfreich. (Erster Unterschied zu vim6, den man bemerkt.)
- ◆ Unterschied zu "showmatch" (vi-Option, `":set sm"` oder `":set showmatch"`): Dort nur beim Eintippen; geht immer: %-Kommando. **Beispiel**
- ◆ Klammernfärbung mit `":NoMatchParen"` ab- und mit `":DoMatchParen"` wieder anschalten (oder `set NoMatchParen` gleich in `.vimrc` unterbringen).

## Undo-Zweige

- ◆ **vi:** Undo-Kommando `"u"` ist reflexiv, d.h., pendelt zwischen letztem und aktuellem Stand
- ◆ **vim:** Undo-Stack bis zur Tiefe 1000 - rückwärts mit `"u"`, vorwärts mit `"CTRL/R"`. **Beispiel**
- ◆ **Problem:** 4 Änderungen durchgeführt, Änderung 2 war falsch; zurück gehen, korrigieren: Wie war doch gleich die letzte Änderung 4? (Besonders beim Programmieren knifflig.) Stört öfter, als man denkt.

- ◆ **vim7** - die Lösung: "u" und "CTRL/R" wirken wie gewohnt, gleichzeitig kann man aber mit "g+" und "g-" auf einer Zeitachse vor- und rückwärts gehen. **Beispiel**
- ◆ Liste der Änderungen mit ":undolist"; es gibt auch Befehle wie ":earlier 15m" (15 Minuten zurückgehen). Mehr unter ":help undo-branches".
- ◆ Keine Navigation in Baum - vim will und kann keine Versionsverwaltung sein! (Außerdem: Wie sollen Änderungen bezeichnet werden?)

## Tabbed Windows

- ◆ vim war schon immer mehrfensterfähig, Navigation allerdings umständlich (CTRL/W + Cursorstaste z.B.). Wichtig allerdings für vimdiff.
- ◆ vim7 hat endlich "tabbed windows":  
`vim -p file1 file2 ...`
- ◆ Neuen Tab öffnen: ":tabe file"
- ◆ Navigation zwischen Tabs: gt und gT (vorwärts/rückwärts), oder 3gt (springe zum 3. Tab).

### Beispiel

- ◆ ":tabdo cmd" führt Kommando cmd in allen Tabs aus. Wichtig z.B. bei meinem Shellskript viw:

```
vim -c "tabdo nmap <F1> gqap | \  
      set tw=72" -p "$@"
```

(setzt Textbreite und Mapping für alle Tabs =  
Textumbruch mit vim)

- ◆ gegenüber dem alten `":n"`, `":e file"`, `":arg"` und vor allem `":rew"` eine Erholung!
- ◆ Hilfe unter `":help tabpage"`

## Wort-Vervollständigung

- ◆ Schon unter **vim6** mit `CTRL/P` (past) und `CTRL/N` (next), äußerst nützlich. Im Unterschied zu Textverarbeitung keine Suche in Wörterbuch, sondern nächstgelegene Matches - höhere Trefferwahrscheinlichkeit! (Kommandos evtl. wiederholen).
- ◆ **vim7**: Menüs zur Auswahl. **Beispiel**
  - ◆ Erstes Angebot kann man mit `CTRL/Y` oder druckbarem Zeichen (das dann erscheint) annehmen
  - ◆ oder per Cursortaste bzw. erneutem `CTRL/P` bzw. `CTRL/N` auswählen und mit `ENTER` annehmen
  - ◆ oder per Backspace und/oder weiteres Zeichen Auswahl einschränken
  - ◆ Bei nur einer Auswahlmöglichkeit erscheint kein Menü.
- ◆ Näheres unter `":help ins-completion-menu"`.

## Syntax-Vervollständigung

- ◆ Mit `CTRL/X+CTRL/O` (schon sehr emacs-artige Fingergymnastik!)

- ◆ Laut Hilfe für C, CSS, (X)HTML, Javascript, PHP, Ruby, SQL und XML (mit DTD), laut Announcement auch für Python

- ◆ für C muss ein Tag-File mit `ctags` erzeugt werden -

### Beispiel

Nur globale Namen, keine block-lokalen Variable?

- ◆ Für Python nur Klassen- und Funktionennamen

- ◆ Ist noch in der Entwicklung, einfach ausprobieren

- ◆ scratch-Fenster (wozu?) mit `CTRL/W`, Cursor hoch, `:q` schließen (Mehrfenster-Navigation)

- ◆ Hilfe unter `:help compl-omni` (noch nicht schön)

## Archive und Directories direkt editieren

- ◆ Einfach tar-, tgz- oder zip-Archiv editieren - Kommentar überflüssig. **Beispiel**

- ◆ Directories analog

- ◆ Komprimierte Files (.gz) gingen sowieso schon unter vim6

## Remote editieren

- ◆ Einfach mit `vim ftp://hostname/path/file`;

weitere Protokolle: `rcp`, `sftp`, `scp`, `fetch`, `http` (`fetch`, `wget`), `rsync`, `dav` (?)

- ◆ Einzelheiten unter `:help ftp` (viel!)

## Spellcheck

- ◆ Bisher skriptbasiert, sehr schönes 1800-Zeilen-vimscript von Clabaudt
- ◆ Nachteil: Performance, Nebeneffekte (Cursorhuppen).
- ◆ Jetzt integriert, mit genauerer Prüfung (Großschreibung, seltene Wörter) und Vorschlägen: Für Englisch nur `:set spell` setzen
- ◆ Für Deutsch Download von <http://ftp.vim.org/pub/vim/runtime/spell/> (oder Mirror); Files `de.latin1.spl` bzw. `de.utf-8.spl` (Wörterbücher, 2.4MB) und `de.latin1.sug/de.utf-8.sug` (Vorschläge, 8 MB). Unterbringen in `$VIM/vim70/spell/`.
- ◆ Mein vim-Skript:

```
nmap <Esc>A  :set spell<CR>
nmap <Esc>?  z=
nmap <Esc>i   zg
nmap <Esc>q   :set nospell<CR>
nmap <F1>    gqap
set comments=
set tw=72
set spelllang=de
hi SpellBad  ctermfg=Red  ctermbg=Gray \
             cterm=underline
hi SpellCap  ctermfg=blue ctermbg=Gray \
             cterm=underline
let spellst = ["de", "en"]
let langcnt = 0
```

```
function Sel_lang()
  let g:langcnt = (g:langcnt+1) % \
    len(g:spellst)
  let lang = g:spellst[g:langcnt]
  echo "language " . lang . " selected"
  exe "set spelllang=" . lang
endfunction
```

```
nmap <Esc>l :call Sel_lang()<CR>
```

### ◆ Probleme:

- ◆ Beim Update in `.vim/doc/` File *tag* löschen, sonst leitet Online-Hilfe u.U. auf alte Hilfe (Cache); richtig in jedem Fall `:"help spell.txt"`.
- ◆ Einbuchstabige Wörter werden angemerkert: Neues Wörterbuch selbst erstellen (ist in Online-Hilfe genau beschrieben). Minimallänge von Wörtern kann nicht eingestellt werden.
- ◆ Schneller Wechsel zwischen Sprachen erst mit obigem Skript möglich, sonst umständlich mit `:"set spelllang=..."`

### **vimscript wird Python-artig**

- ◆ Überschrift verspricht zuviel: Es gibt Listen und Dictionaries genau wie bei Python, Benutzung sehr ähnlich
- ◆ natürlich keine Objektorientierung, d.h. keine Methoden, nur Funktionen

- ◆ deshalb auch neu: "Funktionsreferenz" (in Python ist eine Funktion automatisch ein Objekt)
- ◆ Hilfe unter `:"help List"`, `:"help Dictionary"`
- ◆ vimscript wird damit deutlich mächtiger, aber nicht schön.
- ◆ Vorteil: vimscript läuft überall, eingebetteter Python-Interpreter muss beim Übersetzen konfiguriert werden (und Python muss installiert sein)
- ◆ es hängt also von Einsatzzweck ab, was besser ist

## Nette Gimmicks

- ◆ `:"set cursorline"`, `:"set cursorcolumn"` (kurz: `:"set cul"`, `:"set cuc"`): **Beispiel**
- ◆ Option `:"shelltemp"` standardmäßig unter UNIX/Linux aktiv, d.h. externe Kommandos werden über Pipelines verbunden und nicht mehr über Tempfiles. Wichtig für Editieren chiffrierter Files (Download meiner Skripte: s. Ende)
- ◆ vim konnte schon immer binär editieren; Suche nach verstümmelten Zeichen schwierig. Jetzt auch Muster der Form `:\%d123` oder `:\%x7b` möglich. Wichtig für "Schrottzeichen" in unsinnig formatierten Mailtexten, die weiterverarbeitet werden sollen.



- ◆ Unicode-Zeichen jetzt bis zu 6 Byte lang; Kommando "8g8" sucht nach fehlerhaften UTF-8-Zeichen. Vgl. ":help new-more-unicode".
- ◆ Cursor kann bis hinter das Zeilenende bewegt werden (noch nicht probiert)
- ◆ Mehr Neuigkeiten unter ":help new-7" (viel!)

## Installation, Konfiguration

### Portabilität:

UNIX, Linux, Mac, OS/2, Cygwin, Amiga, Sharp, Zaurus, Open VMS ...

### Selbst übersetzen:

- ◆ Download von [www.vim.org](http://www.vim.org) (am besten als .bz2; 6.5 MB).
- ◆ Standardfall: configure, make, make install (Letzteres als root)
- ◆ Als Nichtroot installieren:  
`configure --prefix=$HOME/vim7`  
(z.B.) - `$HOME/vim7/bin` dann in PATH bringen (an Anfang)
- ◆ Beim configure-Ruf optional anzugeben:  
`--with-features=small`

(oder: medium, big, huge) - Funktionsumfang selbst vorgeben (vgl. `:"help install"`)

◆ Zusätzlich mit Python:

```
#!/bin/bash
# vim63/vim7 compile with python under
# SuSE Linux 9.0
```

```
UNIX=yes
SHELL=bash
CFLAGS=-O2 LDFLAGS="-s \
-L /usr/lib/python/config \
-lpthread" \
    configure --with-features=big \
--enable-pythoninterp \
    --with-python-config \
-dir=/usr/lib/python/config
```

## Konfiguration

Datei `.vimrc` in `$HOME` anlegen und zumindest

```
colorscheme peachpuff
syntax enable
:hi link helpbar Identifier
```

unterbringen - Syntaxfarben sind unglaublich hilfreich (Farbschemata durchprobieren: Skript `colorscheme.vim` in meinem Archiv). Letzte Zeile korrigiert Fehler in manchen Colorschemata (Begrenzer der Hilfe-Links schlecht sichtbar)

## Literatur:

**[1]** [www.vim.org](http://www.vim.org) - Home

**[2]** <http://ftp.vim.org/pub/vim/runtime/spell/> - Spellcheck-Dateien

**[3]** R.Wobst, vim ge-packt, mitp Verlag 2005, ISBN 3-8266-1589-1 (vim6.2)

**[4]** <http://home.wtal.de/rwobst/vim> - Errata, Skripte und Ergänzungen zu [3]

**[5]** S.Oualline, Vi IMproved - Vim, New Riders 2001, ISBN 0-7357-1001-5; (vim5.7, auch online erhältlich)

Artikel in iX: geplant Heft 9/06

**SENKJU!**