Vim für Einsteiger

Agenda

- Historie
- Features
- Bewegen in und Editieren von Dateien
- Einführung in das Hilfesystem
- Makros, Mappings, Marks
- Visual mode
- noch Fragen?

Persönliches

- Toni Zimmer
- SysAdmin seit ca. 14 Jahren

Aus dem Leben eines SysAdmins

- Konfigurationsdateien editieren
  → vim
- Log-Dateien durchstöbern
  → vim -R
- Dokumentation schreiben
  → Wiki → firefox → textern → (g)vim
- Python/Bash-Scripte
  → vim
- Mailing
  → mutt → vim

Historisches

- 1966: Ken Thompson uses **QED** and rewrites it

  for another OS
- 1969: Thompson and Dennis Ritchie create Unix, including **ed** (wq, modal) – line editor!
- 1975: George Coulouris writes **em**
- 1976: Coulouris shows em to Bill Joy, who rebuilds it for BSD: **ex** (including visual mode)
- 1979: executable **vi** was introduced (opening ex in visual mode)
- 1987: STEVIE (open source version of vi)
- 1991: **Vi Imitation** by Bram Moolenaar

Features

- Umfangreiche Doku/Hilfe
- Ist überall verfügbar
- Ist erweiterbar
- CLI
- ...

Bewegen und Editieren (1)

- vim filename
- i – zum Einfuegen (insert mode)
- <Esc> bzw. <C-c> für „normal“ mode
- :wq – zum Speichern und Schliessen (write/quit)

Zwischeninfo: Modi

- normal/command mode
- insert mode
  - i bzw. I
  - a bzw. A (append)
- visual mode
  - v/V
  - <C-v>

Bewegen und Editieren (2)

- w/e – zum Anfang/Ende des naechsten Wortes bewegen
- 0,$ – zum Anfang/Ende der aktuellen Zeile bewegen
- 2w – zum Anfang des übernächsten Wortes bewegen
- 3e – zum Ende des dritten Wortes bewegen
- dw – ein Wort loeschen
- d$ – bis zum Ende einer Zeile loeschen
- d2w – lösche die folgenden zwei Wörter
- dd – lösche aktuelle Zeile
- 2dd – lösche zwei Zeilen
- u – mache Änderung rückgängig
- <C>-r – stelle Änderung wieder her

Bewegen und Editieren (3)

- x – lösche ein Zeichen
- p/P – hänge zuvor gelöschten Text nach/vor dem Cursor an (put)
- r – ersetze (replace) Zeichen unter Cursor
- ce – ändere (change) bis zum Ende des aktuellen Wortes
- c$ – ändere Zeile bis zum Zeilenende

- <C-g> – zeige Dateiposition/-status an
- G – gehe zum Dateiende
- gg – gehe zum Dateianfang
- 3G – gehe an den Anfang der dritten Zeile

Bewegen und Editieren (4)

- / – suche vorwärts
- ? – suche rückwärts
- n – gehe zum nächsten Treffer der Suche
- N – gehe zum vorherigen Treffer der Suche
- <C-o> – gehe zum vorherigen "Standort" des Cursors
- <C-I> – gehe wieder „vor“
- % – gehe zur zugehörigen Klammer ({[

Bewegen und Editieren (5)

- :%s/alt/neu/gc – ersetze jedes "alt" durch "neu" in allen Zeilen, frage aber vorher nach
- :!ls – fuehrt den Befehl "ls" aus
- :w name – speichert aktuelle Datei unter "name" ab (ggf. nur per visual mode ausgewaehlten Bereich)
- :r name – liest Inhalt der Datei "name" in aktuelle Datei ein
- :r!ls – schreibt Ausgabe des Befehls "ls" in aktuelle Datei

Bewegen und Editieren (6)

- o – fuege neue Zeile unter aktueller Position ein
- O – fuege neue Zeile ueber aktueller Position ein
- y – kopiere ("yank") Text
- R – ueberschreibe Text bis zum Beenden des Insert-Mode
- :set bla setze Option bla
- :set nobla deaktiviere Option bla (Bsp: ic, is, hls)

Help! (1)

- :help bzw. :h bzw. <F1> (wenn das Terminal das erlaubt)
- :h command/option/key
- jump via tags: <C-]> (zurueck mit <C-o>)
- :h help-summary
- :h i_CTRL-R (listet Hilfe zu <C-r> im Insert-Mode auf
- :help hls (mit Tab durch die Liste wandern)

Help! (2 - windows)

- :help windows
- <C-w><C-w> lässt Hilfe offen und springt in ursprüngliches Fenster
- <C-w>_ maximiert aktuelles Fenster
- <C-w>= bringt beide Fenster auf gleiche Größe
- :q schliesst aktuelles Fenster wieder

Help! (3) - Übung

- Hilfe zum Thema „help“ Starten
- zu einem anderen |Thema| navigieren
- Window auf „fullscreen“ vergrößern
- :help! → was bedeutet die Fehlermeldung?
- Hilfe schließen

Zwischeninfo: vim != vi

- (nearly) compat mode
- vim nicht auf allen Plattformen verfügbar
- kein Syntax highlighting, keine Hilfe, kein undo-Stack, kein visual mode, keine Macros
- :h vi_diff

Bewegen (1)

- w/e/b vs. W/E/B
- H/M/L (Cursor positionieren)
- */# (zum nächsten/vorherigen String unter aktueller Cursor-Position springen)

Bewegen (2) - scrolling

- <C-e> / <C-y>
- zt (top)
- zb (bottom)
- zz bzw. z.
- <C-u> / <C-d>
- :set scroll

Bewegen (3) - marker

- mx - setze Marke mit dem "Namen" x
- 'x - gehe in die Zeile, in der die Marke x gesetzt wurde
- `x - (Backtick + x) gehe direkt an die Stelle, an der die Marke gesetzt wurde
- `` - springe zwischen aktueller Marke und letzter Postion hin und her
- `. - springe zur letzten Aenderung
- :marks

Editieren (1) – copy'n'paste

- y / yy
- ywP
- yyp
- x
- xp
- P
- "ayy (speichere Zeile (yy) in Register a)
- "ap (kopiere Inhalt aus Register a an aktuelle Stelle)
- :reg

Editieren (2) – text objects

- aw / iw
- a“ / i“
- a' / i'
- a(/ i(
- at / it
- ap / ip
- as / is
- :help text-objects

Editieren (3) – text object combos

- d + w
- d + iw
- d + aw
- c + w
- c + iw
- ...

Editieren (4) – misc

- <C-p> bzw. <C-n> – Wort vervollständigen
- <C-x><C-l> – Zeile vervollständigen
- <C-a> / <C-x> – hoch-/runterzaehlen
  - ggf. Obacht bei fuehrender Null!
- :!%
- :set paste

Zwischeninfo: .vimrc

- Option in vim-Session vs. Option in configfile:
  - :set hlsearch → set hlsearch
  - :set number → set number

- vimscript
- Kommentar beginnt mit “

Zwischeninfo: .viminfo

- „history“ (Suchbegriffe, editierte Dateien, „command line“, Register-Inhalte, Markierungen, Makros...)

Abkürzungen

- :ab fa Feierabend
  - wirkt im Insert-Mode UND in ex-command line
- :iab fa Feierabend
  - wirkt nur im Insert-Mode
- wirkt auch bei copy'n'paste via MMB!
- :iab HAllo Hallo

Mappings

- :map <F4> ddp
- :map <F5> ddkP
- :map V IViele Grüße<CR><CR><Esc>
  - V ist aber eigentlich belegt, daher:
- :map <Leader>V IViele Grüße<CR><CR><Esc>
- Leader Key ist per default der backslash
- :let mapleader=“,“
- :nmap/vmap/imap

Abk./Mappings – Übung

- Schreibe folgende „Mail“ und lasse die Abk. durch den „richtigen“ Text ersetzen:

  Hallo,

  hab Dich tel. nicht erreicht, d. h. b. a. W. ist der WLAN-AP für die Telko nv.

  MfG

Makros

- Arbeitsschritte zur Wdhlg. aufzeichnen
- :reg
- qx – beginne mit Aufzeichnung und speichere in Register 'x'
- <Esc> ggf. zum Beenden Insert-Mode
- q – beende Aufzeichnung
- @x – „Abspielen“ des Makros 'x'
- Q → :vi

Makros – Übung

- erzeuge munin.conf:
  - 10 Rechnernamen in folgendes Schema pressen:

    [workstation;hostname1]

    address hostname1

    use_node_name yes

Makros – Auflösung (1)

- vim munin.conf
- folgenden Text einmal „manuell“ eingeben:

```
[workstation;hostname1]
        address hostname1
         use_node_name yes
```

- <Esc> um in normal mode zu wechseln
- dann Makro definieren (Keys genauso eingeben):
  - qxGkk3yyGpeee<C-a>je<C-a>G
- Makro aufrufen: @x

Makros – Auflösung (2)

- Makro erklärt:
  - qx - starte Aufzeichnung und speichere in Register „x“
  - Gkk - geh ans Ende der Datei, dann zwei Zeilen hoch
  - 3yyG - kopiere drei Zeilen, springe wieder ans Ende
  - peee - füge kopierte Zeilen ein, springe ans Ende des dritten „Wortes“ (sollte die „1“ in „rechner1“ sein)
  - <C-a> - zähle um eins hoch
  - je<C-a> - geh eine Zeile tiefer, auf die nächste „1“ und zähle wieder hoch
  - q - beende Makro-Aufzeichnung

Visual mode

- v – starte visual mode (s. Status-Zeile)
- V – starte visual mode zeilenweise
- <Esc> – beende visual mode
- gv – markiere letzten Bereich
- :help visual-operators
- :help Visual

Visual block mode

- <C-v> – starte visual block mode (s. Status-Zeile)
- beliebige Bewegung um Block zu markieren
- o – gehe an das andere Ende des Blocks
- O – gehe an das andere Ende des Blocks auf gleicher Zeile
- Zeilenende ($) wird "intelligent" erkannt/angepasst

Vielen Dank für die Aufmerksamkeit!