Vi Improved

Der Editor Vim als Integrierte Entwicklungsumgebung

Fritz Mehner

31. Mai 2006
Linux User Group Iserlohn

Vi Improved

INHALT

- IDEs und Editoren
  Erwartungen, Leistungsmerkmale, Unterschiede
- Vim 7
- Einige Befehle und Möglichkeiten
- `ctags`, `preview`
- `quickfix` und `make`
- Projektverwaltung
- diff-Modus und Faltungen
- Benutzerkonfiguration
- Unterstützung für Bash, C, Perl, ...
- Vorzüge / Nachteile
- Weitere Informationen

- Demo / Fragen

IDEs - Integrierte Entwicklungsumgebungen

- alle notwendigen Werkzeuge in einer monolithischen Anwendung (Sicht des IDE-Entwicklers)
- projektbezogene Sicht
- Unterstützung für eine Programmiersprache (evtl. Familie)
- eigene Build-Werkzeuge (evtl. front-end für `make`)
- Klassenbrowser / Debugger / Versionskontrolle integriert
- evtl. GUI-Entwurfswerkzeuge
- schneller Einstieg für den Anfänger möglich
- Editorkomponente meist schwach
- Für jede Programmiersprache (mindestens) eine IDE

Programmiereditoren (UNIX)

- **programmiersprachenunabhängig**
  kein Wechsel der Entwicklungsumgebung beim Wechseln
  zu einer anderen Sprache / zu einem anderen Projekt (!)
- **Editorfunktionen für die Bedürfnisse des Programmierers ausgelegt**
- **Unterstützung unterschiedlicher Programmiersprachen durch plug-ins**
- **gute Integration von Betriebssystemwerkzeugen (Filter)**
- **eigene Skriptsprache (C, Lisp, Vim-Script, ...)**
- **Einbettung von Perl, Tcl, Python, ...**
- **Schnittstellen zur Integration externer Werkzeuge**
- **Betrieb ohne Maus und mit unterschiedlichsten Tastaturen möglich**
- **typische Vertreter: EMACS, Vim**
- **die umfangreichen Möglichkeiten erschweren eine schnellen Einstieg**
- **zur GUI-Erstellung nicht geeignet**

Erforderliche / erwünschte Leistungsmerkmale 1
IDE und Programmiereditor

- **Mappings**
  Tastenbelegungen und Tastenkürzel neu oder anders festlegbar
- **Syntaxerkennung**
  Einfärbung, Einrückung, Faltung
  eigene Syntaxerkennung kann erstellt werden
- **Faltung**
- **Blockbefehle (Bearbeitung rechteckiger Bereiche)**
- **Unterstützung der Formatierung**
  Ausrichtung
  Textumbruch (u.a. in Kommentaren)
  Beautifier (intern und extern)
- **Textvervollständigung**
- **Einfügen von Vorlagen (z.B. Dateikommentare)**
- **Einfügen von Programmierkonstrunkten (z.B. `switch`-Anweisung)**

Erforderliche / erwünschte Leistungsmerkmale 2
IDE und Programmiereditor

- Code-Navigation
  eingebaute Befehle
  Integration externer Werkzeuge (ctags, cscope, ...)
- Zugang zu externen Dokumentationen
  Unix-Handbücher, perldoc, ...
- Rechtschreibprüfung
- Dateivergleicher
- Kommandospeicher
- Unterstützung für die Einbindung externer Werkzeuge,
  wie z.B. Compiler, Style Checker (lint, podchecker, perlcritic, ...),
  anpaßbare Aufrufschnittstelle, Fehler-Parser
- Verwaltung von Code-Schnipseln
- Einbindung einer Versionskontrolle
- . . .

Vi Improved

Vim - Neu in Version 7

- **Rechtschreibprüfung**
- **Sprachbezogene Vervollständigungen**
  (C, HTML, PHP, Python, Ruby, SQL, XML, ...)
- **Verbesserte Dateinavigation (u.a. Favoriten)**
- **Verbesserte Unicode-Unterstützung**
- **Interner grep-Befehl**
- **Erweiterung der Skriptsprache**
- **Remote File Explorer (Editieren über das Netzwerk)**
- **Debugger und Profiler für die Skriptsprache**
- **viele Einzelverbesserungen**

allgemeine Form eines vi-Befehls
normal mode

```
<Befehl><Wiederholungsfaktor><Textobjekt>
oder
<Wiederholungsfaktor><Befehl><Textobjekt>
```

3dw	3 delete word
d'a	delete to mark a
2Y	2 yank (copy) line
3p	3 put (insert) copy-delete buffer

Vi Improved

Suchen, reguläre Ausdrücke

`/begriff` `?begriff`	'begriff' suchen
`n  N`	letzte Suche vorwärts / rückwärts wiederholen
`/^xxx` `/xxx$`	xxx am Zeilenanfang / Zeilenende suchen
`MK-\u\+-[1-9]\d*`	regulärer Ausdruck : Autonummern, Märkischen Kreis (MK) - mindestens ein Großbuchstabe in der Mitte - ein- oder mehrstellige Zahl am Ende (ohne führende Null)

Vi Improved

Makros, Marken

`qa`
... Editierbefehle ...
`q`
	Makroaufzeichnung in Register a

`@a`	**Makro in Register a anwenden**
`mb`	**Markierung b setzen**
`'b`	**Markierung b anspringen** **(b ist nur in der augenblicklich verwendeten Datei gültig)**
`mF`	**Markierung F setzen**
`'F`	**Markierung F anspringen** **(F ist von allen offenen Dateien aus erreichbar)**

Fenster und Dateipuffer

- Fenster sind beliebig teilbar
- Fensteranordnung kann rotiert werden
- Puffer können beliebig groß sein
- Zeilenlänge unbegrenzt

shell-Befehle ausführen

shell-Befehle sind auf den gesamten Puffer
oder auf markierte Bereiche anwendbar

```
:'<,'> ! sort -rn
```

markierten Bereich absteigend numerisch sortieren

```
:r ! ls -1 *.h
```

Namen aller h-Dateien einlesen
ein dateiname pro Zeile

```
:% ! uniq | grep '^[1-5]' | cut -f2
```

für die gesamte Datei (Bereich %):
- Dupletten beseitigen
- nur Zeilen behalten, die mit den Ziffern 1-5 beginnen
- jeweils nur die 2. Spalte behalten

Vim-Konfigurierung

- sehr leistungsfähiges Grundsystem
- 1500+ System-plug-ins und Einstellungsdateien bei www.vim.org
- benutzereigene Einstellungsdateien
- Erstellung eigener plug-ins und Integration externer Anwendungen möglich

- nicht alles, was von einer leistungsfähigen Entwicklungsumgebung erwartet wird, ist im Grundsystem verfügbar.
  Der Anwender muß sich ggf. um die Auswahl geeigneter Zusätze selbst bemühen

Vi Improved

Vim-Konfigurierung

Wichtige / nützliche Zusatzanwendungen

- **ctags** erzeugt eine Indexdatei von Sprachobjekten
- **aspell**, **ispell** zur Rechtschreibprüfung
- **cvscommand.vim** Schnittstelle zu einem CVS-Archiv
- **indent** C-Formatierer
- **project.vim** Projektverwaltung
- **manpageview.vim** Erweiterung des Befehls K
- **Debugger, Profiler, Code-Prüfer usw.**
  für unterschiedliche Programmiersprachen
  Einbindung über das quickfix-System (s.u.)

Vi Improved

EXUBERANT CTAGS

Vim-Konfigurierung

- erzeugt eine Indexdatei von Sprachobjekten
- 33 Programmiersprachen werden unterstützt
- die Erkennung ist vom Benutzer erweiterbar
- Quelle: http://ctags.sourceforge.net/
- das plug-in taglist.vim erschließt die Navigationsmöglichkeiten im Editor
- ctags wird von etwa 20 Editoren verwendet

ctags
preview window

`:'<,'> ! sort -rn`	markierten Bereich absteigend numerisch sortieren
`ctags`	ctags erzeugt eine Indexdatei mit Objekten : Klassen, Funktionsdefinitionen, Makros, ...
`ctrl-]`	Definition anspringen
`crtl-T`	zurückspringen
`:ptag`	preview window öffnen
`ctrl-W}`	Definition des Objektes unter der Schreibmarke im preview window zeigen
`:pclose`	preview window schließen

make und quickfix

- **Verwendung von `make` ist als Standard vorgesehen**
- **Das „`make`-Programm“ kann allerdings durch andere Kommandozeilenwerkzeuge ersetzt werden:**
  (z.B. Splint, podchecker, perlcritic)
- **Zum Parsen der Fehlermeldungen und zum Anspringen der Fehlerpositionen stehen die sog. quickfix-Befehle zur Verfügung**
  (Verwendung in unterschiedlichen plug-ins)

Projektverwaltung

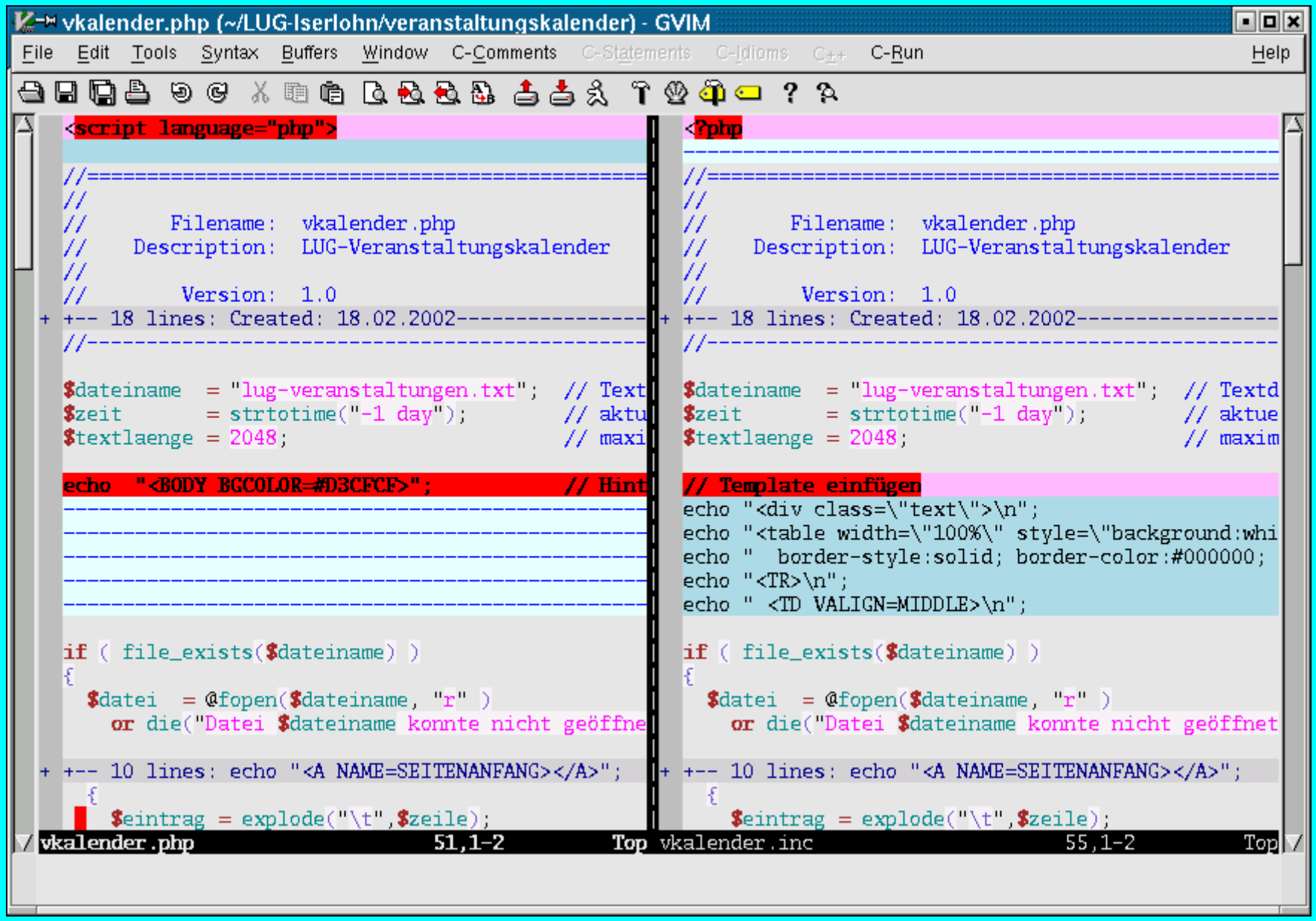
- **Vim-Script projekt (vim.sourceforge.net, script 69)**
  - Verwaltung baumartig organisierter Projekthierarchien
  - Laden aller Dateien eines Projektes
  - grep über alle Projektdateien
  - Verknüpfung von Anwendungen mit Projekteinträgen
  - vielfältig konfigurierbar

- **Vim : integrierte Sitzungsverwaltung**

- **Vim : Dateinavigation mit Favoriten usw.**

Vi Improved

diff-Modus und folding



Vi Improved

VIM konfigurieren und programmieren

```
    " Change to the directory the file in your current buffer is in
    "
    autocmd BufEnter * :lcd %:p:h

endif " has("autocmd")
"
"----------- eigene Ergänzungen -----------------
set tabstop=2
set shiftwidth=2
set autowrite
set incsearch
set visualbell
set nowrap
```

Datei .vimrc
(Ausschnitt)

```
" use font with clearly distinguishable brackets
set guifont=-b&h-lucidux\ mono-medium-r-normal-*-*-130-*-*-m-*-iso8859-9
"
" toggle insert mode <--> normal mode with the <RightMouse>-key
nmap    <RightMouse> <Insert>
imap    <RightMouse> <ESC>
"
" use of dictionaries
set dictionary=~/.vim/word.list
set complete+=k
"
" filename completion
set wildmenu
set wildignore=*.bak,*.o,*.e,*~
```

Datei .gvimrc
(Ausschnitt)

the 4 star editor

Erweiterung durch plugins

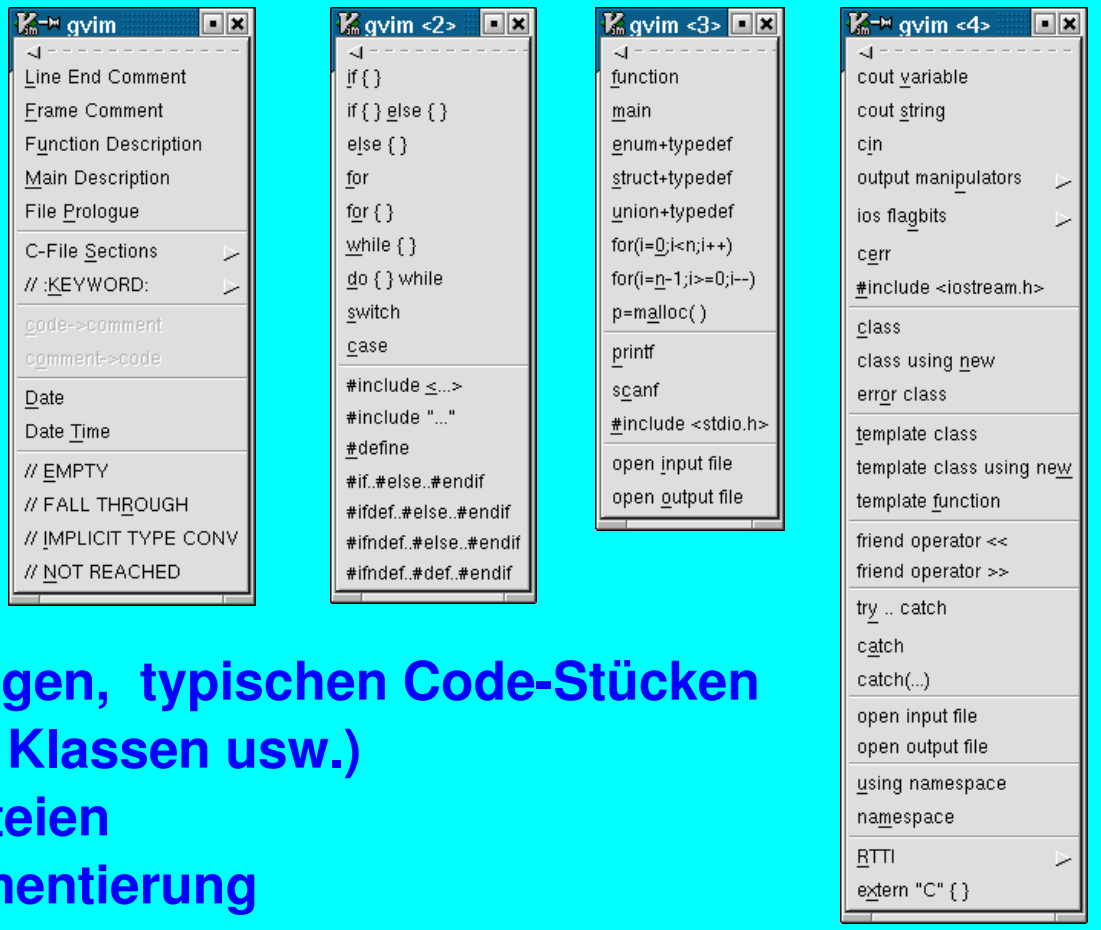
C / C++ -Unterstützung
durch Verwendung der VIM Script Language

```
"------------------------------------------------------------------------
"  Statements : #if .. #else .. #endif
"  Statements : #ifdef .. #else .. #endif
"  Statements : #ifndef .. #else .. #endif
"------------------------------------------------------------------------
function! CVIM_PPIfElse (keyword)
    let defaultcond = "CONDITION"
    let identifier=inputdialog("(uppercase) condition for #".a:keyword, defaultcond )
    if identifier != ""
        let @z=    "#".a:keyword."  ".identifier."\n\n\n"
        let @z= @z."#else      // ----- #".a:keyword." ".identifier."  -----\n\n\n"
        let @z= @z."#endif     // ----- #".a:keyword." ".identifier."  -----\n"
        put z
    endif
endfunction
```

Ausschnitt aus einem plugin

Vi Improved

C / C++ -Unterstützung (plugin)



- **Einsetzen von Anweisungen, typischen Code-Stücken (z.B. leeren Funktionen, Klassen usw.)**
- **Schablonen für neue Dateien**
- **Unterstützung der Kommentierung**
- **Unterstützung des Entwicklungszyklus (Syntaxprüfung, Compilieren, make starten, Ausführen, ...)**
- **Tastenkürzel ermöglichen schnelles Arbeiten**

- **ähnliche plug-ins für Bash, Perl, LaTeX, Ruby**

the 4 star editor

weitere Leistungsmerkmale (Auswahl)

- 2 x 26 benannte Kopierpuffer
- Tastaturmakros
- Syntax-Unterstützung für 330+ Programmiersprachen
- automatische Quelltextformatierung (konfigurierbar)
- Wortergänzung mitttels Wörterbüchern
- unbeschränktes undo / redo
- viele Fonts
- Unterstützung für einfache Textdateien
  (z.B. Umbruch, Wort-, Satz-, Abschnittserkennung)
- umfangreiche Sprachunterstützung (Arabisch, Farsi, Koreanisch, ...)
- Editieren von Binärdateien
- Editieren über das Netzwerk (netrw)
- große Anzahl unterstützter Plattformen
- . . .

Vorzüge (Auswahl)

- **stabil (!) und ausgereift**
  (aber ständige Weiterentwicklung/Fehlerbehebung;
  aktive Benutzergemeinde)
- ***ein* Werkzeug für viele Programmiersprachen**
- **Umfangreiche Editierfunktionen**
  einzigartige Befehlssyntax
- **Integration mit der shell (Filter)**
- **Unterstützung des Software-Erzeugungsprozesses**
  (make, ctags, diff, ...)
- **Volle Programmierbarkeit und Erweiterbarkeit**
- **Verbesserung des Programmierstils durch**
  Unterstützung von eigener Codierungsvorschriften
- **Verwendung ohne GUI möglich (Vim)**

Nachteile (Auswahl)

- **keine voll funktionsfähige Shell in einem Editorfenster**
  (erschwert u.a. die Einbindung von Debuggern)
- **graphisch unterstützte GUI-Erstellung nicht möglich**
- **deutlich höherer Lernaufwand als bei typischen IDEs**
  (dafür natürlich auch deutlich höherer Nutzen!)
- **für Gelegenheitsanwender weniger geeignet**

Weitere Informationen

http://www.vim.org/ - **Homepage** des Projektes; viele Verweise auf andere Seiten und auf Dokumentation. Vim-Skripte zur Erweiterung des Editors, Tips, Neuigkeiten, ...

Linda Lamb, Arnold Robbins
Learning the vi Editor , O'Reilly, Cambridge, MA, 1998, 6.Ed.

Steve Qualline
Vi Improved , new riders, Indianapolis, 2001

Reinhard Wobst
vim GE-PACKT, mitp-Verlag, Bonn, 2004

C editing with VIM HOWTO
Einen ersten Eindruck von der Unterstützung, die VIM für die C-Programmierung gibt, vermittelt das C editing with VIM HOWTO (Linux-Dokumentation).

VIM USER MANUAL
Das VIM USER MANUAL von Bram Moolenaar gibt eine gute und umfangreiche Einführung in die Benutzung von VIM

Fritz Mehner
gVim-Kurzanleitung, http:lug.fh-swf.de/vim

Johann Wolfgang von Goethe
über
UNIX, LINUX und VIM

**„Ein Mann, der recht zu wirken denkt,
muß auf das beste Werkzeug halten.“**

Faust, Vorspiel auf den Theater

Viele Dank für Ihre Aufmerksamkeit !