

Programmiermethoden und -werkzeuge 1

Woche 5 - Vim

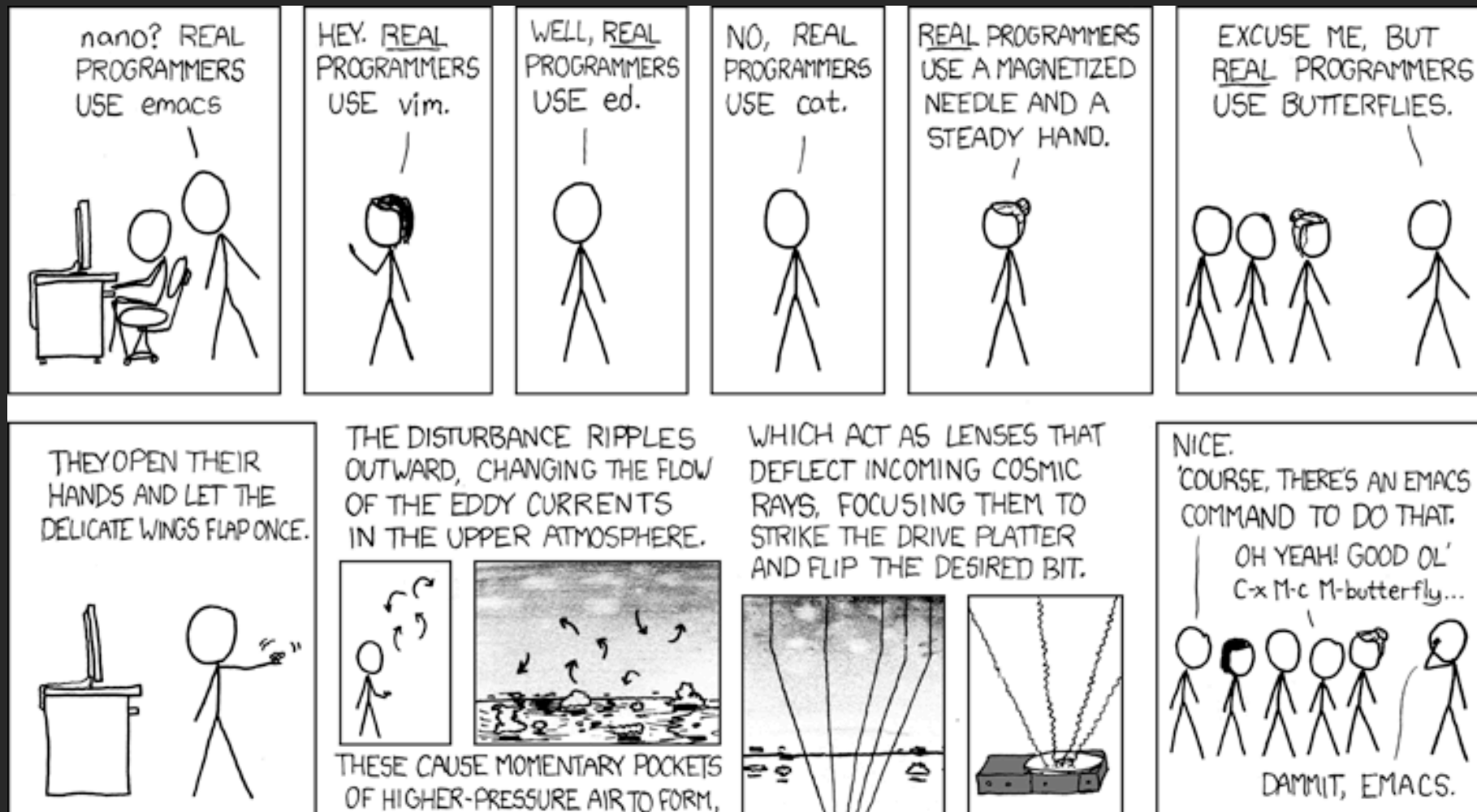
Jochen Hosenfeld

jochen.hosenfeld@informatik.hs-fulda.de

Fachbereich Angewandte Informatik

November 21, 2025

xkcd: Real programmers



Text-Editoren im Terminal

- **nano**
 - simpel
 - benutzerfreundlich
- **Emacs**
 - hochgradig konfigurierbar
 - viele Erweiterungen
 - steile Lernkurve
- **Vim**
 - viele Erweiterungen
 - lange Lernkurve

Historie von Vim

- 1991 von Bram Moolenaar veröffentlicht
- Weiterentwicklung von Vi (1976)
- **Vi improved**
- “Editor war” mit Emacs
- Beide gehören zu den am längsten bestehenden Anwendungsprogrammen aller Zeiten

xkcd: Hottest editors

HOTTEST EDITORS

1995 — [EMACS-VIM]
2000 — [EDITOR WAR]
2005 — VIM
2010 — NOTEPAD ++
2015 — SUBLIME TEXT
2020 — CRISPR
2025 — CRISPR (VIM
KEYBINDINGS)

Warum Vim?

- Fast auf jedem Unix-/Linux-System vorinstalliert
 - ideal für Remote-Zugriff (SSH)
- Läuft direkt im Terminal
 - keine grafische Oberfläche nötig
- Schnell & ressourcenschonend
 - funktioniert auch auf schwacher Hardware
- Mächtig bei Routineaufgaben
 - effizientes Editieren, Navigieren und Suchen

Warum Vim?

Philosophie: Programmieren ist mehr als nur Schreiben

Partneraufgabe: Was könnte damit gemeint sein?



Warum Vim?

Philosophie: Programmieren ist mehr als nur Schreiben

- Lesen
- Navigieren
- kleine Änderungen
- zwischen verschiedenen Files wechseln
- längeren Code schreiben

Die primäre Aufgabe ist nicht das Erstellen, sondern das Bearbeiten von Code.

Ein “idealer” Editor

Einzelaufgabe: *Wenn Sie einen perfekten Editor bauen könnten: Wie sollte dieser idealerweise funktionieren?*



Ein “idealer” Editor

Partneraufgabe: *Tauschen Sie Ihre Ideen mit Ihrem Nachbarn aus.*



Syntax von Vim

Verb + Substantiv

```
1 d      delete
2 w      word
3
4 dw     "delete word"
```

Syntax von Vim

Verb + Substantiv

- *Operationen* und *Text*, auf den diese ausgeführt werden
- *Commands*
 - *repeat* (.)
 - *undo* (u und U)
- *Undo*
 - letzte Änderung, aber Sinne einer abgeschlossenen *Operation*, nicht einzelne Buchstaben

Verben

```
1 d    delete
2 c    change (delete and enters insert mode)
3 >    indent
4 v    visually select
5 y    yank (copy)
```

Modal Editor

modal => mode (Modus, Zustände)

- *Normal*
 - Navigieren, kleine Änderungen, Kopieren
 - *(wird noch genauer betrachtet)*
- *Insert*
 - Text schreiben
- *Visual*
 - Bereiche selektieren und bearbeiten
- *Command-line*
 - Am unteren Ende für Befehle wie Filtern, Suchen und Ersetzen

Normal Mode

- Navigieren, kleine Änderungen, Kopieren
- Schnelle und effiziente Navigation
- Wechsel in andere Modes und wieder zurück
- Arbeiten alleine über Tastatur, Maus ist “zu langsam”
- Eigene Programmiersprache als Schnittstelle

Wechsel zwischen Modes

```
1 i      Normal => Insert
2 v      Normal => Visual
3 :      Normal => Command-line
4 Esc    back to Normal
```

Belegung der Tasten

Die **Esc**-Taste ist schwer erreichbar, daher wird ihre Funktionalität oft auf die **Caps-Lock**-Taste umgelegt.

Normal Mode - Motions

1	<code>h</code>	<code>left</code>
2	<code>j</code>	<code>down</code>
3	<code>k</code>	<code>up</code>
4	<code>l</code>	<code>right</code>

`h j k l`

Historisch hat die Belegung mehr Sinn ergeben, da das Layout des Keyboards des Entwicklers anders aussah als heute üblich.

`h j k l` vs. Arrow Keys

Standardmäßig funktionieren die Pfeiltasten, für deren Bedienung muss die Hand aber “zu weit” bewegt werden!

Normal Mode - Motions

1	w	Move forwards by one word
2	b	Move backwards by one word
3	e	Move to end of a word
4	\$	Go to end of text on current line
5	0	Go to beginning of current line
6	^	Go to beginning of text on current line
7		
8	2w	Go two words forward
9	3e	Go to end of third word ahead

Belegung der Tasten

Manche Befehle sind mnemonisch (e,b,w), manche von Regular Expressions (0, \$) übernommen, manche sind freie Tasten.

Wiederholungsfaktor (Counts oder Repeat Counts)

Counts vor einem Befehl geben an, wie oft dieser ausgeführt wird.
Gilt für viele Operatoren und Bewegungen.

Command-line Mode

1	:q	Close file
2	:q!	Close file, don't save changes
3	:w	Save changes to file
4	:wq or :x or ZZ	Save changes and close file
5	:e sun.rb	Open file "sun.rb"
6	:help w	Get help for "w" command
7	:help :w	Get help for ":w" command
8		
9	:!ls	Run shell command ls
10	:w play.rb	Save current file as "play.rb"
11	:r hat.rb	Read in file "hat.rb"

Changes

1	x	Delete character at cursor
2	r	Replace character under cursor
3	i	Insert at cursor
4	I	Insert at beginning of line
5	a	Append at cursor
6	A	Append at end of line
7		
8	o	Open new line below
9	O	Open new line above

Grundlegende Funktionen

Öffnen, navigieren, in Insert Mode wechseln, editieren, speichern, schließen

Delete

- | | | |
|---|------------------------------------|-----------------------|
| 1 | <code>dw</code> | Delete word |
| 2 | <code>d\$</code> or <code>D</code> | Delete to end of line |
| 3 | <code>d2w</code> | Delete two words |
| 4 | <code>dd</code> | Delete entire line |
| 5 | <code>2dd</code> | Delete two lines |

Change

1	<code>cw</code>	Change word, ...
2	<code>c\$</code> or <code>C</code>	Change to end of line, ...
3	<code>c2w</code>	Change two words, ...
4	<code>cc</code>	Change entire line, ...
5	<code>2cc</code>	Change two lines, ...
6		... and enters insert mode

Besonderheiten von Delete und Change

d und **c** alleine löschen noch nicht. Beide müssen mit einem Movement kombiniert werden.

Pattern

Wird ein Key doppelt gedrückt, wird der Effekt auf der ganzen Zeile ausgeführt.

Undo und Redo

- | | | |
|---|---------------------|-----------------------------|
| 1 | <code>u</code> | Undo last change |
| 2 | <code>U</code> | Undo changes on entire line |
| 3 | <code>ctrl+r</code> | Redo changes |

Reminder

Undo macht eine komplette Aktion rückgängig. Undo und Redo können mehrmals ausgeführt werden.

Yank (Copy) and Paste

- | | | |
|---|------------------|-------------------------------|
| 1 | <code>yw</code> | Yank word |
| 2 | <code>vwy</code> | Visual select word, then yank |
| 3 | <code>y\$</code> | Yank to end of current line |
| 4 | <code>p</code> | Paste after cursor |
| 5 | <code>P</code> | Paste before cursor |

Global Movement

1	<code>50G</code> or <code>:50</code>	Go to line <code>50</code>
2	<code>G</code>	Go to last line in file
3	<code>gg</code>	Go to first line in file
4		
5	<code>ctrl+u</code>	up
6	<code>ctrl+d</code>	down

Screen Movement

1	H	highest line
2	M	middle line
3	L	lowest line

Find

- 1 f "Find" the **next** character
- 2 F "Find" backwards
- 3 t "To" the **next** character, stop one character before
- 4 T "To" backwards, stop one character after

Reminder

Funktioniert nur in der eigenen Zeile. Trotzdem sehr praktisch.

Search

1	/waldo	Search for "waldo"
2	n	Go to next search result
3	N	Go to previous search result
4	?carmen	Search backwards for "carmen"
5	ctrl+o	Jump to previous location (jump back)
6	ctrl+i	Jump to next location (jump forward)
7	%	Go to matching parentheses or brackets
8		
9	set ic	Change search settings to ignore case
10	set noic	Change search settings to use case

Search and Replace

- 1 `:%s/bad/good` Replace bad with good **in** current line
- 2 `:%s/hi/bye/g` Replace hi with bye **in** entire file
- 3 `:%s/x/y/gc` Replace x with y **in** entire file, prompt **for** changes

Visual Mode

1	<code>v</code>	Open visual mode
2	<code>vw</code>	Visual select word
3	<code>vwd</code> or <code>vwX</code>	Visual select word, then delete word
4		
5	<code>V</code>	Open visual line mode
6	<code>Ctrl+V</code>	Open visual block mode

Best Practices

1	.	"Dot" - Repeat last action
2	diw	Delete inner word
3	diq	Delete inner quotes
4	di{	Delete inner block
5	dip	Delete inner paragraph
6		
7	cit	Change inner tag

Repeatability

Mit Vim kann oft der gleiche Effekt über verschiedene Wege erreicht werden. Der "beste" Weg ist der, der die **Wiederholbarkeit** der Aktion ermöglicht. Oftmals ist deswegen ein generelleres Textobjekt wie bei *diw* besser als bei *dw* oder eine **Motion**.

Split-Screen

1	<code>:split</code>	Split current window horizontally
2	<code>:vsplit</code>	Split current window vertically
3		
4	<code>Ctrl+w h/j/k/l</code>	Switch windows
5	<code>:q</code>	Quit current window
6	<code>:qa</code>	Quit all windows

Buffers

Anders als z. B. bei VSCode werden Files bei Vim **Buffers** genannt. Derselbe **Buffer** kann in verschiedenen **Windows** angezeigt werden. Somit kann dieselbe Datei an verschiedenen Stellen betrachtet werden.

Konfiguration mit vimrc

Statusinformation am unteren Rand

- “Relative Line Numbers”
 - Abstand zur aktuellen Zeile
 - Aktuelle Zeile behält die globale Nummer bei
 - praktisch für Movements wie “6j” und “2k”

Vim Features in anderen Editoren

- VSCode
- IntelliJ IDEA
- Jupyter Notebooks
- Obsidian
- ...

Hilfe und Tutorials

- VimTutor
- VimGenius
- Cheat Sheet
- Vim Adventures