

# Developer's Toolkit - or things I wish someone had told me earlier...

LinuxDays Spotlight Course Fall 2016

Nils Leuzinger   Philip Stark



Oct 25, 2016

# Time Frame

- ▶ Block 1 (40 min)
  - ▶ Why CLI tools
  - ▶ Vim
  - ▶ Git
- ▶ Pause (5 min)
- ▶ Block 2 (40 min)
  - ▶ Build Tools (Java)
  - ▶ Misc
- ▶ Questions and Answers

# Overview

## Broad Topic

- ▶ Too much to talk about everything

## Managing Expectations

- ▶ What this course is
  - ▶ An Overview
- ▶ What this course won't be
  - ▶ In depth...

# Take Home Message

What we want you to take home from this course:

- ▶ It pays off to learn to use tools
- ▶ Automation is worth it

## If You're Bored

- ▶ Handout
- ▶ Slides and Demo Examples:

<https://github.com/hellerbarde/linuxdays-spotlight-fall2016>

# IDEs

## “Integrated Development Environments”

- ▶ Java: Eclipse, IntelliJ
- ▶ C/C++: Visual Studio, CLion
- ▶ C#: Visual Studio, MonoDevelop
- ▶ **Aside:** IDEs are good!

# Features

- ▶ Refactoring Tools
- ▶ Autocompletion
- ▶ **Build Tools**

# Build Tools

- ▶ From source code to releasable build
- ▶ Should be
  - ▶ Reproducible
  - ▶ Stable
  - ▶ Easy
  - ▶ Automated
- ▶ Why? (no, really, why? It's so much work...)

- ▶ Be in control
- ▶ Eliminate doubt



# No IDE Available

- ▶ Languages without dedicated IDE
  - ▶ Haskell
  - ▶ Erlang
  - ▶ Elixir

# Why?

- ▶ **Full** control
- ▶ Transparency
- ▶ Reproducibility
- ▶ Efficiency
- ▶ Consistency
- ▶ Wide application range

# Full Control

- ▶ Reduces the magic
- ▶ You're not dependent on an IDE

# Transparency

- ▶ Again, no magic.
- ▶ Greater learn effect

# Reproducibility

- ▶ Will your code work on **other** PCs?

# Efficiency

- ▶ Custom bash scripts
- ▶ Other shells (zsh, fish, ...)
- ▶ Great array of commands at your disposal

# Consistency

- Bash is tried and true

## Why use them?

```

                                MAKEVID.DOC
                                April 1989

The MAKEVID utility will let you build a SCREEN.VID driv
from separate .CSD, .GSD and .KBD files, without running
Word Setup program. These files are provided on the
Utilities disk 2. Setup normally creates a SCREEN.VID fi
when you install the Word program.

The following steps will create a DOS-only driver, that
not be valid under OS/2.

To create a VID file, type the following:

MAKEVID VIDNAME.VID -Lfr CSDNAME.CSD -Lfr GSDNAME.GSD -L
KBDNAME.KBD

.   VIDNAME.VID is the name of the resulting .VID file.
    Before using it with Word, this file has to be renam
    to SCREEN.VID and copied to the Word program directo
.   CSDNAME.CSD, and GSDNAME.GSD are screen drivers. Bot
    are required. These files are usually named after th
    display adapter or monitor they support, for example
    VGA.CSD, or VGA.GSD.
.   KBDNAME.KBD is a keyboard driver. The Word Program f

                                MAKEVID.DOC
                                CHARTEST.DOC

COMMAND: Copy Delete Format Gallery Help Insert Jump Library
Options Print Quit Replace Search Transfer Undo Window
Edit document or press Esc to use menu
Pg1 Co38      {}      ?      Microsoft Word

```



## Why use them?

```

the east end of the passage.

There is a small wicker cage discarded nearby.
The lamp

The lamp is now on.

We are crawling over cobbles in a low passage. There is a dim light
the east end of the passage.

There is a small wicker cage discarded nearby.
The cage

We are in a debris room filled with stuff washed in from the surface.
A wide passage with cobbles becomes plugged with mud and debris
, but an awkward canyon leads upward and west. A note on the wall
"Magic word XYZZY".

A three foot black rod with a rusty star on an end lies nearby.

```

```

the east end of the passage.

There is a small wicker cage nearby.
The lamp

The lamp is on now

We are crawling over cobbles in a low passage. There is a dim light
the east end of the passage.

There is a small wicker cage discarded nearby.
The cage

We are in a debris room filled with stuff washed from the surface.
A wide passage with cobbles becomes plugged with mud and debris
but an awkward canyon leads upward and west. A note on the wall
"Magic Word XYZZY".

A three foot black rod with a rusty star on an end lies nearby.

```

# Wide Application Range

```

12 ## Why use them?
13
14 ### Why?
15
16 - _Full_control
17 - Transparency
18 - Reproducibility
19 - Efficiency
20 - Consistency
21 - Wide application range
22
23 ### Full Control
24
25 - Reduces the magic
26 - You're not dependent on an IDE
27
28 ### Transparency
29
30 - Again, no magic.
31 - Greater learn effect
32
33 ### Reproducibility
34
35 - Will your code work on _other_ PCs?
36
37 ### Efficiency
38
39 - Custom bash scripts
40 - Other shells (zsh, fish, ...)
41 - Great array of commands at your disposal
42
43 ### Consistency
44
45 CLI is tried and true
46
47 ###
48 <!-- Picture of Word 30 years ago.-->
49 { width=50% }
50
51 ###
52 { height=200px }
53 { height=200px }
54
55
56 ### Wide Application Range
57 <!-- Mention that these slides were written as markdown, using git as a VCS, written in vim, with
58 Christian's theme. The Shell can bring all those things together really effortlessly -->
59
60 { width=70% }

```

NORMAL | master | 05-why-cli-tools.md[+]

## What Vim Is And Isn't

```

      vim - Vi Improved

      version 8.0.13
      by Bram Moolenaar et al.
      Vim is open source and freely distributable

      Sponsor Vim development!
      type :help sponsor<Enter> for information

      type :q<Enter>          to exit
      type :help<Enter> or <ft> for on-line help
      type :help version<Enter> for version info

      Running in Vi compatible mode
      type :set nocp<Enter> for Vim defaults
      type :help cp-default<Enter> for info on this
```

## What Vim Is And Isn't

```

10 ---
11 notes: |
12     FINISHED
13 ---
14 # Vim
15 # What Vim Is And Isn't
16 #
17 # Vim is an editor
18 #
19 # ...but probably unlike anything else you've ever used.
20 #
21 #
22 # Vimscript
23 #
24 # Remap keys
25 # Write whole plugins!
26 #
27 # Modal Editing
28 #
29 # Vim Modes
30 #
31 # Insert Mode
32 # Visual Mode
33 # Command-Line Mode
34 # Normal Mode
35 #
36 # Does what any other editor does
37 #
38 # Visual Mode
39 # Selecting text
40 #
41 # Command-Line Mode
42 # Execute built-in vim commands
43 # Execute any shell command!
44 #
45 # Normal Mode
46 # Magic and h&l.
47 #
48 #
49 # (resources/10vim/ADM-3A terminal.jpg){ width=50% }
50 #
51 # (resources/10vim/h&l_keyboard.jpg){ width=80% }
52 #
53 # Learning Vim
54 #
55 # How To "Groek" Vim

```

```

20 # The Developer's Toolkit or Things I wish someone had told me
21
22 We want to explain how to build software without the magic that most IDEs provide as part of their Run button.
23 Part of that is the tooling necessary to achieve some measure of efficiency in working inside the console or without
24 an IDE in general.
25
26 # Must:
27
28 - git
29 - vim (basics)
30 - konkrete Toolchain (Durchstech)
31
32 # Roter faden:
33
34 - Intro Smin (Phil)
35 - Only an overview. Not going into depth on any of the tools
36 - IDEs are not evil
37 - git, vim, ant, cmake
38 - Control over your environment is important
39 - remove second guessing yourself
40
41 # Why Command Line 10min (Nils)
42
43 - Full control
44 - transparency
45 - important to the understanding (what happens when i click build)
46 - reproducibility
47 - efficiency
48 - with custom bash scripts
49 - Consistency
50 - shell doesn't really change much over the years
51 - saw opera, firefox, thunderbird, chrome, android, iphone, youtube...
52
53 # Command line tools 10min (Phil)
54
55 - ssh (a sshA)
56 - rsync
57 - sshx
58 - For long builds
59 - for letting runs go over night
60 - history & tail -20
61 - Build your own shell script
62
63 - gdb
64 - Custom bash scripts
65 - example build.sh
66
67 # Vim 15-20min (Nils)
68
69 - Git 15-20min (Phil)
70
71 - Why not use the IDE integration?
72 - Better understanding
73 - more control
74
75 - PAUSE: 10min
76
77 # Java Build Tools 10min (Demo Nils, Theoria Phil)
78
79 - Why are there multiple ones?
80 - Example Rust
81
82 - Ant
83 - Gradle
84 - Maven
85
86 # Misc: Smin (Nils)
87
88 - C/C++ Build Tools
89 - Honorable Mentions

```

# Vim is an editor

- ▶ Extremely powerful
- ▶ Unlike anything you've ever used

# Configurability

## Themes

- ▶ Gruvbox, Molokai, Jellybeans, Badwolf, Solarized, Dracula, ...
- ▶ <http://vimcolors.com/>

## Vimscript

- ▶ Remap keys
- ▶ Write whole plugins!

# Vim Modes

- ▶ Insert Mode
- ▶ Visual Mode
- ▶ Command-Line Mode
- ▶ Normal Mode

# Insert Mode

- Does what any other editor does



# Visual Mode

- Selecting text

# Command-Line Mode

- ▶ Execute built-in vim commands
- ▶ Execute any shell command!

# Normal Mode

- Wizardry and hjkl





# How To “Grok” Vim

## Some basic vim terminology

- ▶ Verbs
- ▶ Motions
- ▶ Objects

## Vimtutor

# Productivity Tips

- ▶ Unmap the arrow keys.
- ▶ Learn something new from time to time.
- ▶ Actively try to improve your editing.
- ▶ Watch some talks online: "Write Code Faster: Expert-Level Vim"
- ▶ Book: Practical Vim
- ▶ Youtube: Thoughtbot
- ▶ <http://vimcasts.org>

# Vim Is not an IDE

- But it can be. (Almost)



# NERDTree

```

Press ? for help
.. (up a dir)
/home/nils/Gitstuff/linuxdays/
├─ bashcourse/
├─ documents/
│   └─ Hardware Recycle/
│       ├── installguide/
│       ├── Skript Advanced Courses/
│       └─ post-installation.sh*
├─ helpers_cheat_sheet/
├─ linux-spotlight_dev-toolbox/
│   ├── img/
│   ├── notes/
│   └─ resources/
│       ├── 05whyCLI/
│       │   ├── commandline_now.jpg
│       │   ├── commandline_then.jpg
│       │   ├── wide_application.jpg
│       │   └─ Word_1989.jpg
│       ├── 10vim/
│       ├── ant/
│       │   ├── build.xml
│       │   └─ images/
│       │       ├── gdb_animation.gif
│       │       ├── template.tex
│       │       ├── 00-introduction.md
│       │       ├── 05-why-cli-tools.md
│       │       ├── 07-cli-tools.md
│       │       ├── 10-vim-basics.md
│       │       ├── 20-git.md
│       │       ├── 25-pause.md
│       │       ├── 30-java-build-tools.md
│       │       ├── 40-misc.md
│       │       ├── 50-license.md
│       │       ├── 60-QA.md
│       │       ├── build.sh*
│       │       ├── handout.sh*
│       │       ├── linux-spotlight_hs16.pdf
│       │       ├── out.log
│       │       ├── showUnfinished.sh*
│       │       ├── spotlight_hs16.pdf
│       │       ├── spotlight_hs16.tex
│       │       ├── texbuild.sh*
│       │       └─ topic.md
│       └─ website/
│           └─ test.tex
└─ 22 # The Developer's Toolbox or things I wish someone had told me
    21
    20 We want to explain how to build software without the magic that most IDEs provide as part of their Run button.
    19 Part of that is the tooling necessary to achieve some measure of efficiency in working inside the console or without an IDE in general.
    18
    17 ## Must:
    16
    15 - git
    14 - vim (basics)
    13 - 1 konkrete Toolchain (Durchstich)
    12
    11 ## Roter Faden:
    10
    9
    8 - Intro 5min (Phil)
    7 - Only an overview. Not going into depth on any of the tools
    6 - IDEs are not evil
    5 - git, vim, ant, cmake
    4 - Control over your environment is important
    3 - remove second guessing yourself
    2
    1 - Why Command Line 10min (Nils)
    23 - Full control
    1 - transparency
    2 - important to the understanding (what happens when i click build)
    3 - reproducibility
    4 - efficiency
    5 - with custom bash scripts
    6 - Consistency
    7 - shell doesn't really change much over the years
    8 - see opera, firefox, thunderbird, chrome, android, iphone, youtube...
    9
    10 - Command line tools 15min (Phil)
    11 - ssh (& sshfs)
    12 - rsync
    13 - tmux
    14 - for long builds
    15 - for letting runs go over night
    16 - history | tail -20
    17 - Build your own shell script
    18 - gdb
    19 - Custom bash scripts
    20 - example build.sh
    21
    22 - Vim 15-20min (Nils)
    23
    24 - Git 15-20min (Phil)
    25 - Why not use the IDE integration?
    26 - Better understanding
    27

```

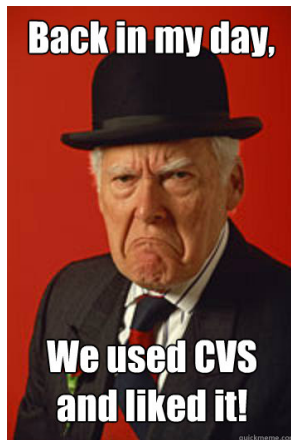
## Some more:

- ▶ Commenter
- ▶ Surround
- ▶ Easymotion
- ▶ <http://vimawesome.com/>

# Version Control Systems

- ▶ Version Control System (VCS)
  - ▶ Keeps old versions
  - ▶ Centralized “Master” copy
  - ▶ Ability to record a reason for a change
- ▶ Why?
  - ▶ See above!

# CVS



# SVN



## Git

# In case of fire



1. git commit



2. git push



3. leave building

# History

- ▶ First release 11 years ago
- ▶ Built for large codebases
  - ▶ Linux Kernel (**Millions** of lines)
- ▶ Works well for small projects too
  - ▶ Your programming exercises ;)
- ▶ **Stigma of complexity**

# Questions

- ▶ Who has heard of Git? (before today)
- ▶ Who used Git before?
- ▶ Who was forced to use Git?
- ▶ Who feels comfortable with it?
- ▶ Why not? 😊
- ▶ Who thinks it's black magic?



While we're on the topic...



# Complexity

- ▶ Single-user version control is hard...
  - ▶ Manage Versions
  - ▶ Record Changes

- ▶ Multi-user version control is really hard!
  - ▶ Conflict resolution
  - ▶ Branching
  - ▶ Sharing

# Stop! Story Time

- ▶ Demo our repository with
  - ▶ Command line
  - ▶ git cola

# Recommendations

- ▶ tig
- ▶ git cola

- ▶ Quick Break. Back in 5 minutes.
- ▶ Find the slides and the demo examples here:  
<https://github.com/hellerbarde/linuxdays-spotlight-fall2016>

# Overview of Build Tools

## Compile, Test and Run

- ▶ ant
- ▶ make
- ▶ Custom Shell Scripts

## Also find libraries

- ▶ cmake (C/C++)

## Also download missing libraries

- ▶ maven

# Common Ground

- ▶ Help **you** with building software



# Why Do You Need Automation

- ▶ Time
- ▶ Consistency
- ▶ Reliability

# Story Time

# Misc General Tools

- ▶ ssh (see Sandro's courses)
- ▶ rsync
- ▶ tmux
- ▶ “Terminal Multiplexer”
- ▶ For long builds
- ▶ pandoc with custom bash scripts!

```
# nils@Velociraptor: ~/Gitstuff/linuxdays/linux-spotlight_dev-toolbox <master x [*]>
❏ cat build.sh
#!/bin/bash

if [[ -n $1 ]]; then
    pandoc --template="resources/template.tex" -t beamer "$1" -o "${1%.md}.pdf"
else
    pandoc --template="resources/template.tex" -f markdown -t beamer [0-9][0-9]-*.md -o "linux-spotlight_hs16.pdf"
fi
```

# Misc Build tools

- ▶ C/C++ build tools:
  - ▶ autoconf
  - ▶ cmake
  - ▶ makefiles
  - ▶ unit test runner
- ▶ Debuggers:
  - ▶ gdb
  - ▶ Valgrind

- ▶ Slide template by Christian Horea, licensed under CC BY-SA 3.0
- ▶ Slides by Philip Stark, Nils Leuzinger, licensed under CC BY-SA 4.0
  - ▶ Find the slides and the demo examples here:  
<https://github.com/hellerbarde/linuxdays-spotlight-fall2016>
- ▶ Emoji provided free by <http://emojione.com>

## Questions?

- ▶ Is anything still unclear?
- ▶ Additional information needed?
- ▶ **Ask now!**

## Where to find these Slides

- ▶ Find the slides and the demo examples here:  
<https://github.com/hellerbarde/linuxdays-spotlight-fall2016>
- ▶ Slide template by Christian Horea, licensed under CC BY-SA 3.0
- ▶ Slides by Philip Stark, Nils Leuzinger, licensed under CC BY-SA 4.0
- ▶ Emoji provided free by <http://emojione.com>