Tools

Seminar: Open-Source-Softwareentwicklung

Hagen Hübner

18. Oktober 2014

HS 2014

Betreut von Malte Helmert

Universität Basel

Übersicht

1. Leitideen
2. Kommunikation
3. Programmiertoolchain
   - Editoren
   - Versionskontrolle
   - Compiler / Interpreter
   - Debugger / Testing / Profiler
   - Sonstiges
4. IDE's vs. Unix Toolchain
5. Grenzen Open-Source - Proprietäre Systeme

Leitideen

1. Grosse Vielfalt an Tools für die allermeisten Aufgaben (mit gewissen Ausnahmen)
2. Toolkompetenzen unabdingbar für Mitarbeit an einem FLOSS-Projekt
3. Projekt schränkt Toolauswahl ein.

Kommunikation

- Riesige Auswahl an Tools per Protokoll (mit gewissen propietären Ausnahmen)
- Toolkompetenz unabdingbar für reibungslose Kommunikation und Mitarbeit an einem FLOSS-Projekt

Programmiertoolchain

- Editoren
- Versionskontrolle
- Compiler / Interpreter
- Debugger / Testing / Profiler
- Sonstiges

Editoren

Wenig Zusatzfunktion			Viele Zusatzfunktionen
ed	gedit	Sublime Text	emacs
nano		vim	jedit
MS-Editor		notepad++	
TextEdit		Adobe Brackets	
Leafpad		kate	

Unterschiede: Syntaxhighlighting - Autovervollständigung - Code folding-Plugins - Unterstützung für mehrere Sprachen - Erstellbare shortcuts - Neue Funktionen per plugin - Compiler Einbindung

Versionskontrolle

- Nicht alle Versionierungssysteme sind Open-Source.
- Nicht alle können mittels Open-Source Software verwendet werden.

Beispiel (Linux-Kernel)

Entwickler Abkehr von BitKeeper welche zur Entwicklung von Git anregte.

Compiler / Interpreter

- Grosse Unterschiede zwischen den verfügbaren Compilern
  - Proprietär/Floss - Gui/CLI
- Die meisten Sprachen bieten eine grosse Auswahl an freien/proprietären Programmen
- Build-Automation mit unterschiedlichsten Interfaces

Debugger / Testing / Profiler

- Ähnliche Situation wie bei Compilern
- Oft von den gleichen Firmen/Communities entwickelt

Beispiel (GNU Projekt)

Compiler - Profiler - Debugger
gcc - gprof - gdb

Sonstiges

- Virtuelle Arbeitsumgebungen, Sandboxing
- Emulatoren
- Dokumentation
- Packet/Distributionserstellung

IDE's vs. Unix Toolchain

- IDE's vereinen viele unterschiedliche Werkzeuge unter einer Oberfläche
- Lernkurve | Erweiterbarkeit | Benutzerfreundlichkeit | Universalität
- Stehen im Widerspruch mit der Unix Philosophie des: „*Make each program do one thing well*“

Unix Philosophie

„The many ways Unix provides to glue together programs mean that components of its basic toolkit can be combined to produce useful effects that the designers of the individual toolkit parts never anticipated“ (The Art of Unix Programming, Chapter 1. Eric Steven Raymond)

Beispiel (Viele Einzelprogramme anstatt IDE)

vim <-> diff <-> sed <-> gcc <-> gdb <-> gprof <-> make

Grenzen Open-Source - Proprietäre Systeme

Beispiel (Entwicklung für IOS)

Betriebsystembindung: IOS-Entwicklungs-Kit nur für Mac OS X erhältlich. (Oder per Virtualisierung)

Kosten, Tool-Beschränkungen: 99$ pro Jahr um Programme auf Iphone und in den Appstore zu laden

Toolkit nicht Open-Source