Eine Einführung in Versionverwaltungsysteme

Wissenswertes für GIT-Anwender

bernd.wunder@leb.eei.uni-erlangen.de

Lehrstuhl für Elektronische Bauelemente
Friedrich-Alexander-Universität Erlangen-Nürnberg

28. Januar 2012

Inhaltsverzeichnis

Versionsverwaltungsysteme

> *Eine Versionsverwaltung ist ein System, das zur Erfassung von Änderungen an Dokumenten oder Dateien verwendet wird. Alle Versionen werden in einem Archiv mit Zeitstempel und Benutzerkennung gesichert und können später wiederhergestellt werden.* [1]

Aufgaben

- Protokollierungen der Änderungen
- Wiederherstellung von alten Ständen
- Archivierung der einzelnen Stände eines Projektes
- Kooperative Entwicklung (Entwicklungsteams)
- gleichzeitige Entwicklung mehrerer Entwicklungszweige (branches)

[1] Quelle: Wikipedia

Versionsverwaltungsysteme

engl. Bezeichnungen: *Version Control System (VCS)*, *Software Configuration Management (SCM)*, *Revision Control System (RCS)*

unvollständige Übersicht einiger VCSe:

Revision Control System (RCS)

Entwicklung:	1980 bis 2004
Einteilung:	zentrales, dateibasiertes VCS
Probleme:	Binärdateien, Verzeichnisse, locks, merge, keine Atomic commits, keine Metadaten, umbenennen
Prüfsumme:	—
Compression:	—
Lizenz:	GPL2
Betriebsysteme:	UNIX, WIN95

Entwickelt für die Versionsverwaltung von Text-Dateien auf einem Computer! RCS ist im Wesentlichen mit ersten VCS, dem *Source Code Control System (SCCS)*, vergleichbar.

Versionsverwaltungsysteme

Concurrent Versions System (CVS)

basierend auf:	RCP (nutzt gleiches Speicherformat)
Entwicklung:	1989 bis 2008
Einteilung:	zentrales VCS
Probleme:	Binärdateien, Verzeichnisse, locks, merge, keine Atomic commits, keine Metadaten, umbenennen
Prüfsumme:	—
Compression:	—
Lizenz:	GPL2
Betriebsysteme:	UNIX, WINDOWS, Mac OS X

CVS wird nicht mehr aktiv weiterentwickelt. Die offizielle Webseite wird nicht mehr weiter betreut. [a]

[a] Quelle: Wikipedia

Versionsverwaltungsysteme

Subversion (SVN)

basierend auf:	CVS (Nachfolger)
Entwicklung:	seit 2000
	Ziele CVS Probleme (siehe oben) zu beseitigen
Einteilung:	zentrales VCS
Probleme:	Binärdateien, Verzeichnisse, locks, merge
Prüfsumme:	—
Compression:	—
Lizenz:	Apache License
Betriebsysteme:	UNIX, WINDOWS, Mac OS X

CVS wird nicht mehr aktiv weiterentwickelt. Die offizielle Webseite wird nicht mehr weiter betreut. [a]

[a] Quelle: Wikipedia

Grundbegriffe

Repository

Datenbank in dem jeder Dateistand eines Projektes über die Zeit hinweg gespeichert ist.

Working Tree

Arbeitsverzeichnis in dem die Modifikationen durchgeführt werden.

Commit

beinhaltet alle Veränderungen bzw. spiegelt den aktuellen Zustand der in das VCS aufgenommen werden soll wieder. Enthält neben den Änderungen zusätzliche Metadaten (Commit Message, Autor, Datum, Signatur, ...)

HEAD

zeigt auf die neueste Version *Kopf* im aktuellen Zweig (Branch)
Achtung: Unterschide zwischen GIT und SVN,CVS

Grundbegriffe

Secure Hash Algorithm (SHA-1)

ist eine eindeutige, 160 Bit (40 hexadezimale Zeichen) lange Prüfsumme für beliebige digitale Informationen.

Beispiel:

mit dem GNU/Linux Programm *sha1sum* wird die Prüfsumme für den Text *"Isabella und Lilly Wunder"* berechnet werden:

```
bernd@Power:~$ echo "Isabella und Lilly Wunder" | sha1sum
25989877d4888b5a4f41850069a7c53ac2c8e3ff  -
```

Grundbegriffe (GIT)

Branch

bezeichnet einen parallelen Entwicklungszweig. Der Hauptzweig in einem Versionsverwaltungssystem hat meistens einen speziellen Namen. (z.B in SVN->*trunk* und in GIT->*master*)

Objektmodell

Git-Objekte (blob, tree, commit, tag) sind in einer Objektdatenbank gespeichert und über SHA1-Summen identifizierbar. Die History eines Repository lässt sich als Graph von Objekten modellieren.

Index

Der *Index* ist ein lokaler Zwischenspeicher. Alle Änderungen werden zuerst in den index geschrieben. Anschießend wird der Index durch einen *commit* in das Repository eingecheckt.

Grundbegriffe (GIT)

Clone

ist eine Kopie eines Repositories mit der gesamten History der Entwicklung.

Tag

Ein Tag ist ein symbolischer Name für schwer zu merkende SHA-1 Summen. So können spezielle *Commit* einen Namen, also ein *Tag* erhalten.

Warum Versionsverwaltung

1. viele gleichzeitige laufende Projekte / Features (z.B. Sicherheitsupdates, neue Funktionstests, verschiedene Versionen)
2. Um des Chaos Herr zu werden

Warum Versionsverwaltung

1. viele gleichzeitige laufende Projekte / Features (z.B. Sicherheitsupdates, neue Funktionstests, verschiedene Versionen)
2. Um des Chaos Herr zu werden
3. Überblick über die gesamte Entwicklung behalten
4. kolaboratives Arbeiten in einem Team

Branches, Merges und Tags

Branch Parralle Entwicklung in Teams erfordert oft mehrere Zweige

Tag Name für eine bestimmte Version

Merge Führt den Parallelen Zweig in den Hauptzweig zurück

Trunk Name in SVN für den Hauptentwicklungszweig

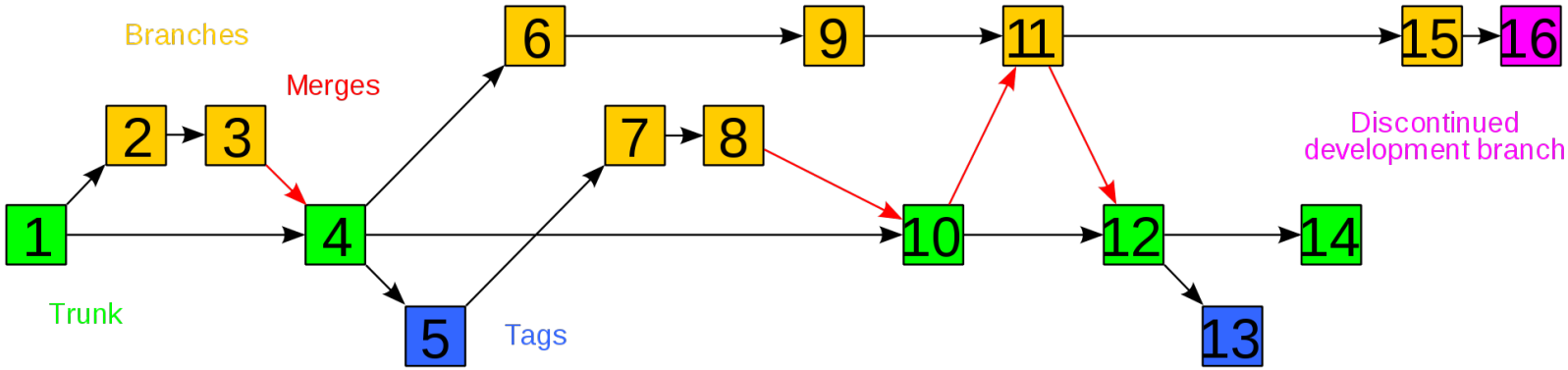

Abbildung: Branches, Merges und Tags [2]

[2] Quelle: Wikipedia

Repository Tool: *gitk*

gitk

In Tcl programmiertes grafisches Frontend zur Anzeige des Repositories.
Ist im Git Standard Umfang enthalten und somit **immer** vorhanden.
Ermöglicht einen schnellen Überblick über die History, Commits, Diffs, Tags und die Struktur des Repositories.

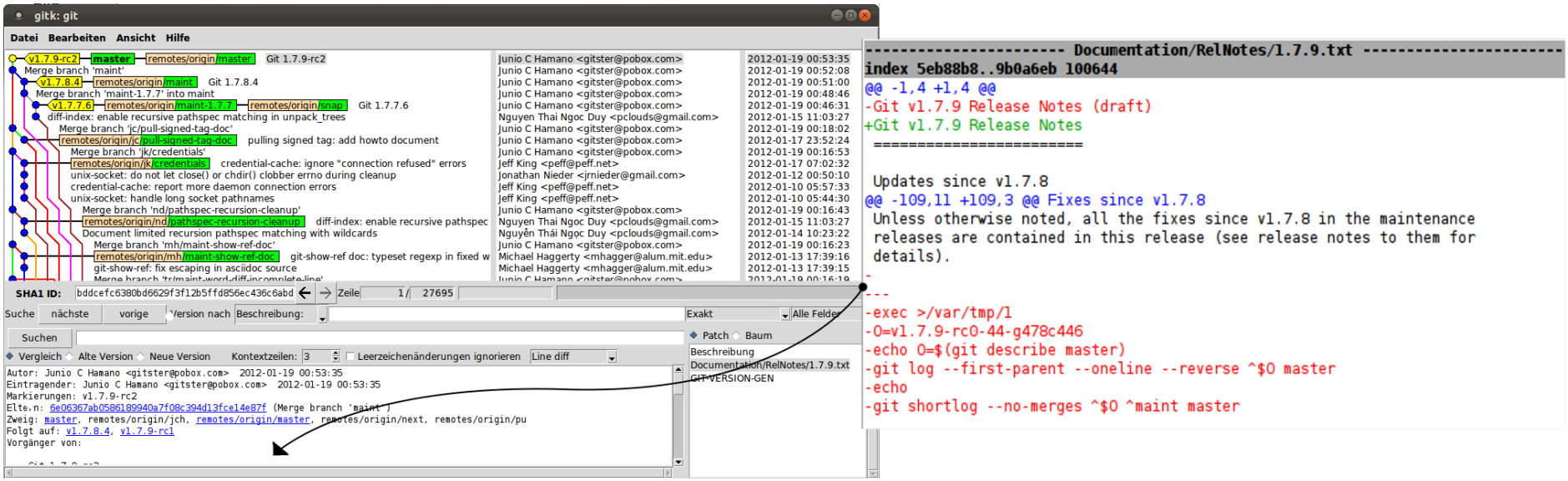

Abbildung: *gitk*: einfach, übersichtlich und immer da!

Commit Tool: *git-gui*

git-gui

Einfaches grafisches Programm um Änderungen bereitzustellen und einen Commits zu erstellen.

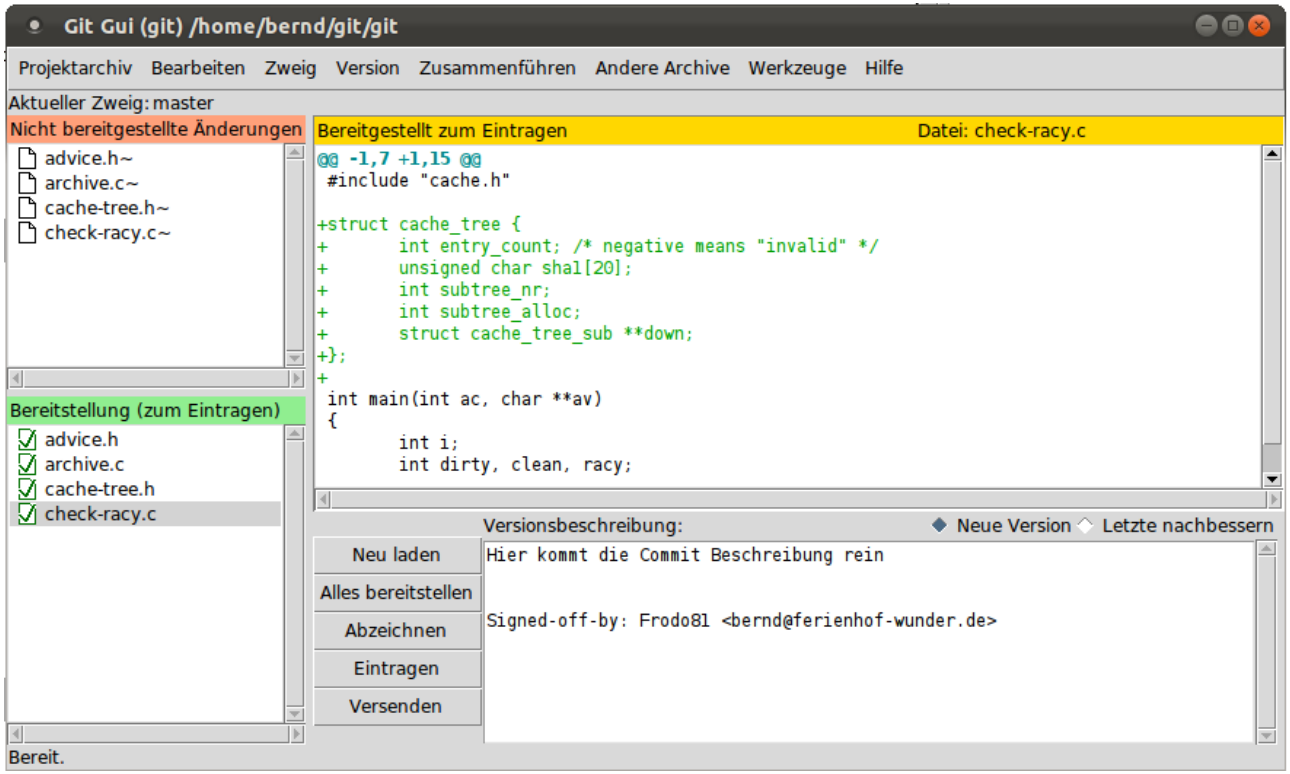

Abbildung: *git-gui*: Frontend für die Erstellung eines Commits. Ist wie *gitk* im Standardumfang enthalten.

Diff-Tool: meld

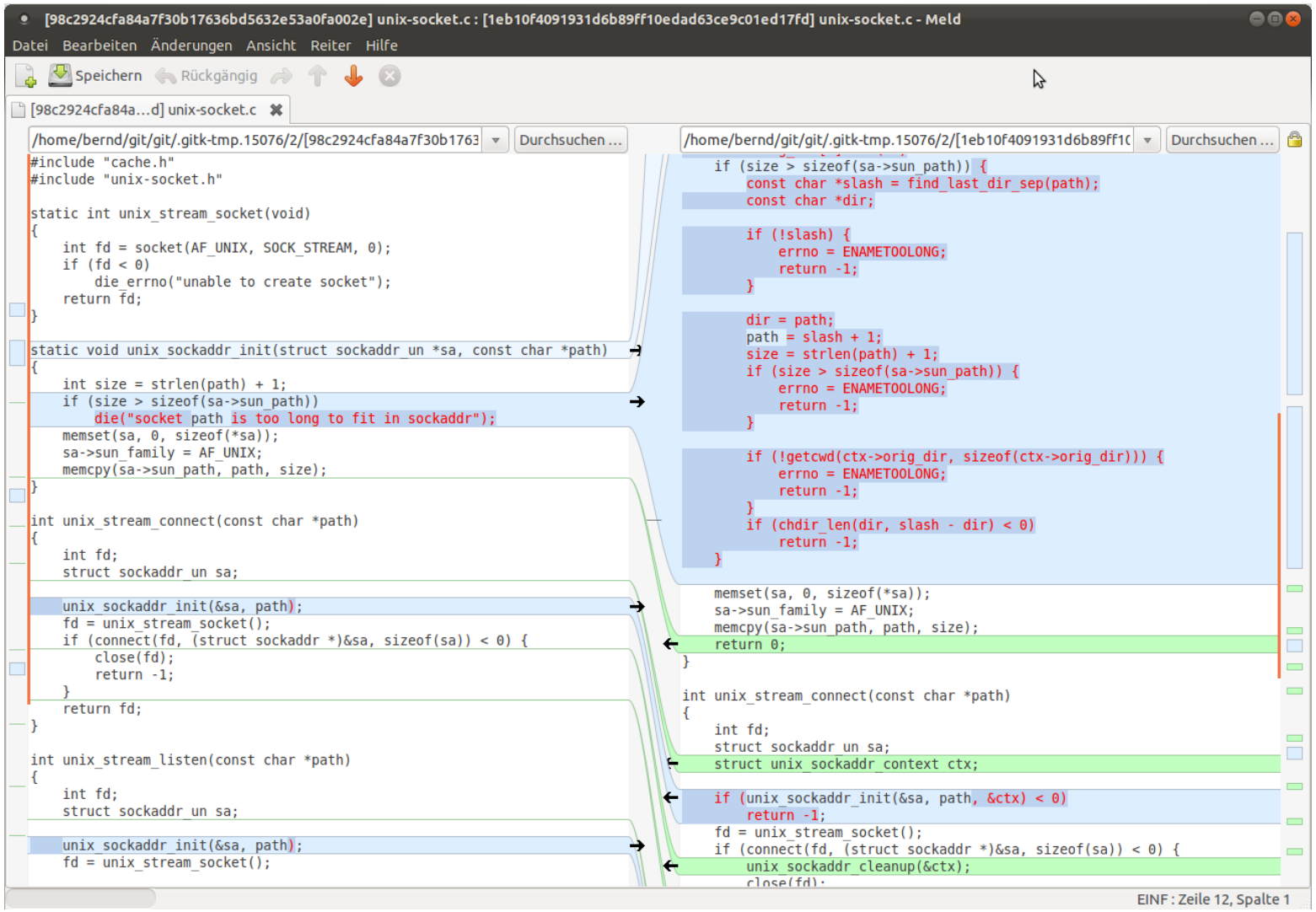

Abbildung: *meld*: einfach, übersichtlich und immer da!

Befehle

Getting Started

neues Repository im aktuellen Verzeichnis anlegen:

```
git init
```

eine Datei dem Repository hinzufügen:

```
git add /Pfad/Zur/Datei
```

alle Dateien im aktuellen Verzeichnis dem aktuellen Repository hinzufügen:

```
git add .
```

einen Commit ausführen:

```
git commit -m "Initial Commit"
```

Befehle 2

Remotes

Unter einem Remote versteht man eine entfernte Quelle. Der **git remote** Befehl dient zum Verwalten der Remotes:

```
bernd@power:~$ git remote
origin
power
```

oder ausführlicher mit:

```
bernd@power:~$ git remote -v
origin   /media/Transcend/Versionsverwaltung_mit_GIT/ (fetch)
origin   /media/Transcend/Versionsverwaltung_mit_GIT/ (push)
```

einen neuen Alias auf einen entfernten Remote hinzufügen:

```
git remote add newName https://www.weitWeg.de/Repository.git
```

Befehle 3

branch

Für die Erstellung eines neuen Zweiges (branch):

```
git branch lilly
```

Eine ausführliche Übersicht über die vorhandenen Branches:

```
bernd@Power:~$ git branch -v
* lilly   98a74d9 Some further commands
  master  e2694e7 Some explaination about commands
```

Umbenenen des aktuellen Zweiges nach isabella:

```
git branch –m isabella
```

In einen anderen Zweig wechseln und auschecken (hier in master):

```
git checkout master
```

Quellen & Literatur


Beamer Paket
http://latex-beamer.sourceforge.net/98a74d912af7ebf23902ee03aa9be4cb


User's Guide to the Beamer


DANTE e.V. http://www.dante.de