

Eine Einführung in Versionverwaltungssysteme

Wissenswertes für GIT-Anwender

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

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

26. Januar 2012



Inhaltsverzeichnis

1 Einführung

- Versionsverwaltungssysteme
- Grundbegriffe

2 Grundlegende Konzepte

- Warum Versionsverwaltung
- Branches, Merges und Tags

3 Tools

- gitk
- git-gui
- Diff-Tools

4 Referezen

Versionsverwaltungssysteme

*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 Benutzererkennung 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)

¹Quelle: Wikipedia

Versionsverwaltungssysteme

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
Betriebssysteme:	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.

Versionsverwaltungssysteme

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
Betriebssysteme:	UNIX, WINDOWS, Mac OS X

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

^aQuelle: Wikipedia

Versionsverwaltungssysteme

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
Betriebssysteme:	UNIX, WINDOWS, Mac OS X

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

^aQuelle: 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: Unterschiede 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 1
25989877d4888b5a4f41850069a7c53ac2c8e3ff -
```


Grundbegriffe (GIT)

Branch

bezeichnet einen parallelen Entwicklungsweig. 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. Anschließend wird der Index durch einen *commit* in das Repository eingchecked.

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 Parallele 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

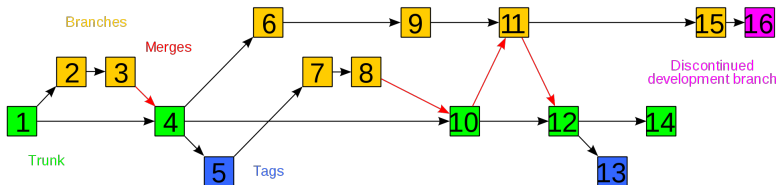


Abbildung: Branches, Merges und Tags²

²Quelle: 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.

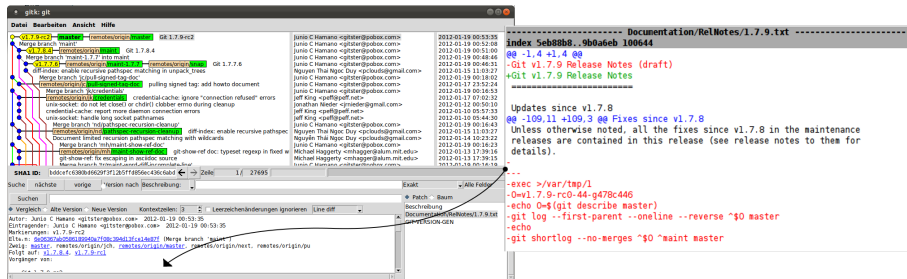


Abbildung: *gitk*: einfach, übersichtlich und immer da!



Commit Tool: git-gui

gitk

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

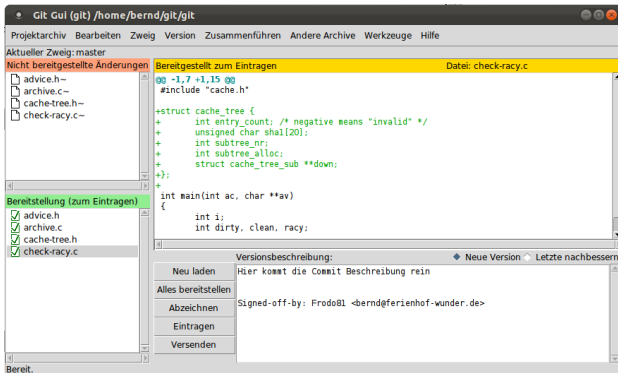


Abbildung: *git gui*: einfach, übersichtlich und immer da!



Diff-Tool: meld

```
[98c2924cfa84a7f30b17636bd5632e53a0fa002e] unix-socket.c: [1eb10f4091931d6b89ff10edad63ce9c01ed17fd] unix-socket.c - Meld
Datei Bearbeiten Änderungen Ansicht Reiter Hilfe
Speichern Rückgängig
/home/bernd/git/git/git-tmp.15076/2/[98c2924cfa84a7f30b1763] Durchsuchen... /home/bernd/git/git/git-tmp.15076/2/[1eb10f4091931d6b89ff1c] Durchsuchen...

#include "cache.h"
#include "unix-socket.h"

static int unix_stream_socket(void)
{
    int fd = socket(AF_UNIX, SOCK_STREAM, 0);
    if (fd < 0)
        die_erro("unable to create socket");
    return fd;
}

static void unix_sockaddr_init(struct sockaddr_un *sa, const char *path)
{
    int size = strlen(path) + 1;
    if (size > sizeof(sa->sun_path))
        die("socket path is too long to fit in sockaddr");
    memset(sa, 0, sizeof(*sa));
    sa->sun_family = AF_UNIX;
    memcpy(sa->sun_path, path, size);
}

int unix_stream_connect(const char *path)
{
    int fd;
    struct sockaddr_un sa;

    unix_sockaddr_init(&sa, path);
    fd = unix_stream_socket();
    if (connect(fd, (struct sockaddr *)&sa, sizeof(sa)) < 0) {
        close(fd);
        return -1;
    }
    return fd;
}

int unix_stream_listen(const char *path)
{
    int fd;
    struct sockaddr_un sa;

    unix_sockaddr_init(&sa, path);
    fd = unix_stream_socket();

    if (size > sizeof(sa->sun_path)) {
        const char *slash = find_last_dir_sep(path);
        const char *dir;

        if (!slash) {
            errno = ENAMETOOLONG;
            return -1;
        }

        dir = path;
        path = slash + 1;
        size = strlen(path) + 1;
        if (size > sizeof(sa->sun_path)) {
            errno = ENAMETOOLONG;
            return -1;
        }

        if (!getcwd(ctx->orig_dir, sizeof(ctx->orig_dir))) {
            errno = ENAMETOOLONG;
            return -1;
        }
        if (chdir_len(dir, slash - dir) < 0)
            return -1;
    }

    memset(sa, 0, sizeof(*sa));
    sa->sun_family = AF_UNIX;
    memcpy(sa->sun_path, path, size);
    return 0;
}

int unix_stream_connect(const char *path)
{
    int fd;
    struct sockaddr_un sa;
    struct unix_sockaddr_context ctx;

    if (unix_sockaddr_init(&sa, path, &ctx) < 0)
        return -1;
    fd = unix_stream_socket();
    if (connect(fd, (struct sockaddr *)&sa, sizeof(sa)) < 0) {
        unix_sockaddr_cleanup(&ctx);
        close(fd);
    }
}
```

EINF: Zeile 12, Spalte 1

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



Quellen & Literatur



Beamer Paket

<http://latex-beamer.sourceforge.net/>



User's Guide to the Beamer



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