Versionskontrollsysteme

(CVS-Teil)

Uwe Berger
Markus Dahms

Inhalt

- CVS
- GUIs für CVS, SVN und git
- Weitere Versionskontrollsysteme
- Weiterführende Informationen

CVS allgemein

- CVS – Concurrent Versions System (1986), GPL
- CVS-Homepage: http://www.cvshome.org
- Weiterentwicklung von SCCS (Source Code Control System; AT&T, 1975) und RCS (Revision Control System; 1985)
  - Versionskontrolle bezog sich immer nur auf einzelne Dateien
- einige Neuerungen mit CVS:
  - unabhängiges Arbeiten an gleichen Dateien
  - zentrales Quellverzeichnis (Repository)
  - Client/Server-Architektur

CVS – einige Vor-/Nachteile

- Vorteile
  - 20 Jahre alt: damit stabiler Code, wenige (bekannte) Bugs, viele darauf aufsetzende Anwendungen, gut dokumentiert
  - geringer Platzbedarf des Repository
  - Netzwerkorientiert
- Nachteile
  - keine Binärdaten ohne weiteres ablegbar/versionierbar
  - keine "atomic commits"
  - fehlende Berechtigungssystem im Repository
  - kompilziertes Umbenennen von Dateien/Verzeichnissen

CVS - Zugriff

- Zugriffsmethoden auf das Quellverzeichnis (Repository):
  - lokal
  - :pserver (unverschlüsselt, eigene Benutzer-Datenbank)
  - rsh (unverschlüsselt, Serverbenutzerverwaltung)
  - ssh (verschlüsselt, Serverbenutzerverwaltung)
- Umgebungsvariable $CVSROOT ...
  - `export CVSROOT=/home/bergeruw/cvsrep`
  - `export CVSROOT=:pserver:uwe@esus:/home/cvs`
- oder Option -d (z.B. cvs -d /home/bergeruw/cvsrep befehl)

CVS-Repository anlegen

- Repository (auf dem Server) anlegen
  - `cvs init`
- Anmelden am CVS-Server
  - `cvs login`
- Initiales Anlegen eines Modules im CVS
  - `cd /in_das_Verzeichnis_des_Modules`
  - `cvs import -m '...' modulname ventor start`

CVS-Repository benutzen

- eine "Sandbox" anlegen
  - `mkdir sandbox; cd sandbox`
- Modul "auschecken":
  - `cvs checkout modul`
- ... Dateien bearbeiten... :-)
- Modul wieder "einchecken"
  - `cvs commit [dateien...]`

CVS - Nützliches

- CVS-Protokoll anzeigen
  - `cvs log [optionen]`
- Sandbox auf aktuellen Stand bringen (beidseitig)
  - `cvs update [optionen]`
- CVS-Informationen in verwaltete Dateien einbauen
  - Schlüsselwort-Ersetzungsfelder (siehe Manual)
- Unterschiede zwischen Dateiversionen anzeigen (z.B. relativ zur Sandbox)
  - `cvs diff -r Version Datei`

GUIs für CVS, SVN und git

- diverse IDEs (Integrierte Entwicklungsumgebungen)
  - anjuta (CVS), eclipse (CVS/SVN), kdevelop (CVS/SVN), netbeans (CVS)
- CVS
  - gCVS, tkcvs, lincvs, cvsweb, ...
- SVN
  - rapitsvn, supervision, gsvn, websvn, ...
- git
  - qgit, gitview

GUIs - Fortsetzung

- Diverse Plugins für Dateimanager, z.B.
  - Nautilus; Apotheke (CVS)
  - Konqueror; Ksvn (SVN)
- cvsfs: ein auf fuse basierendes Filesystem über einem CVS-Repository

Einige weitere VCS (1)

- GNU Arch: netzwerkorientiert, "atomic Commits", Repository-Replikation, Repository-Benutzerverwaltung
- BitKeeper: frei für Opensource-Projekte, netzwerkorientiert, Synchronisation untereinander, ehem. "Linux-VCS"
- Mercurial: aktive Entwicklergemeinde, in Python geschrieben und damit portabel, sicheres Netzwerkprotokoll
- Monotone: relativ "jung", Peer-to-Peer-Synchronisation, integierte Verschlüsselung, auf vielen Plattformen verfügbar
- Bazaar: einfacher Befehlssatz, intuitiv (einige Funktionen teilweise auf OS-Kommandos basierend)

Einige weitere VCS (2)

- Superversion: viele Plattformen (Java), grafisches Frontend integriert
- SVK: in Perl geschrieben, basiert auf SVN-Filesystem und kann es damit auch selbst verwenden, kann SVN-/CVS-/ Perforce-Repositorys spiegeln
- ... und viele andere ...

Weiterführende Informationen

- http://www.cvshome.org
- http://www.tigris.org
- http://better-scm.berlios.de/comparison/comparison.html
- http://linuxmafia.com/faq/Apps/scm.html
- http://fuse.sourceforge.net/wiki/
- G. Purdy; CVS kurz&gut (O'Reilly; ISBN 3-89721-265-X)