CVS

Thomas Preuss
IN00

CVS - Inhalt

- **Einleitung/Motivation**
- Arbeiten mit CVS

  Begriffe

  Tutorial

CVS - Einleitung

- CVS = **C**oncurrent **V**ersioning/**V**ersions **S**ystem
  →Versionskontrollsystem
- verwaltet und archiviert Änderungen an Dateien
- Idee erstmals 1986 von Dick Grune in Newsgroup „comp.sources.unix“ gepostet
  (damals noch Sammlung von Shell-Skripten)
- 1989 von Brian Berliner entworfen und implementiert

CVS – Motivation (I)

- bei der Softwareentwicklung schleichen sich Fehler ein

  →Fehlersuche kann mitunter sehr aufwendig sein (wenn Änderung weit zurück liegt etc.)

→mit Hilfe von CVS hat man Zugriff auf gesamte Versionshistorie →Fehlersuche beschleunigt

CVS – Motivation (II)

- man könnte auch selbst jede „Entwicklungstufe“ speichern
  →sehr aufwendig
  →benötigt sehr viel Speicherplatz

- CVS kann einem diese Arbeit abnehmen und speichert dabei die Daten auch sehr effizient ab
  (nur Änderungen werden gespeichert)

CVS – Motivation (III)

- CVS ist auch hilfreich, wenn viele Leute am gleichen Projekt arbeiten

→denn CVS trennt die einzelnen Entwickler voneinander und mischt die jeweiligen Ergebnisse erst wenn jeder Entwickler fertig ist (siehe Konflikt Beispiel)

CVS – Zusammenfassung(I)

- „Vorteile“:
  - CVS ist dezentralisiert (Benutzer arbeitet auf Kopie)
  - effiziente Speicherung der Veränderungen (Dateiformat basiert auf dem des RCS)
  - Möglichkeit der Gruppenarbeit auf einem Satz von Dateien
  - effektives Zusammenführen von gleichzeitigen Änderungen an Textdateien

  - Client/Server fähig (Vorteil gegenüber RCS)
  - freie Software

CVS – Zusammenfassung(II)

- „Nachteile“:
  - kein Sperren (locking) von Dateien
  - kein „Build-System“, d.h. CVS ersetzt nicht make
  - hoher administrativer Aufwand (im Vgl. zu RCS)
  - lange Einarbeitungsphase (viele Befehle)

CVS - Inhalt

- Einleitung/Motivation
- **Arbeiten mit CVS**

  Begriffe

  Tutorial

CVS – Begriffe(I)

- **Repository / Datenbank** =
  Referenzkopie eines Quelltextes auf CVS-Server

- **Modul** = Verzeichnis im Repository

- **Workspace** = lokale Arbeitsversion eines Quelltextes aus Repository

- **Checkout** =
  Anfertigen einer lokalen Kopie der Dateien aus Repository

- **Update =**
  Lokale Kopie der Dateien wird mit aktueller Version aus Repository abgeglichen

CVS – Begriffe(II)

- **Diff =**
  Unterschiede zwischen zwei Versionen anzeigen

- **Commit** = **Checkin** =
  Übertragen von Änderungen aus lokaler Kopie ins Repository

- **Revision =**
  Versionsnummer einer Datei(für jede Datei individuell)

CVS – Begriffe(III)

- **Revisions & Versions:**
  - CVS unterscheidet zwischen Revisions (per-file-concept) und Versions (per-package-concept)

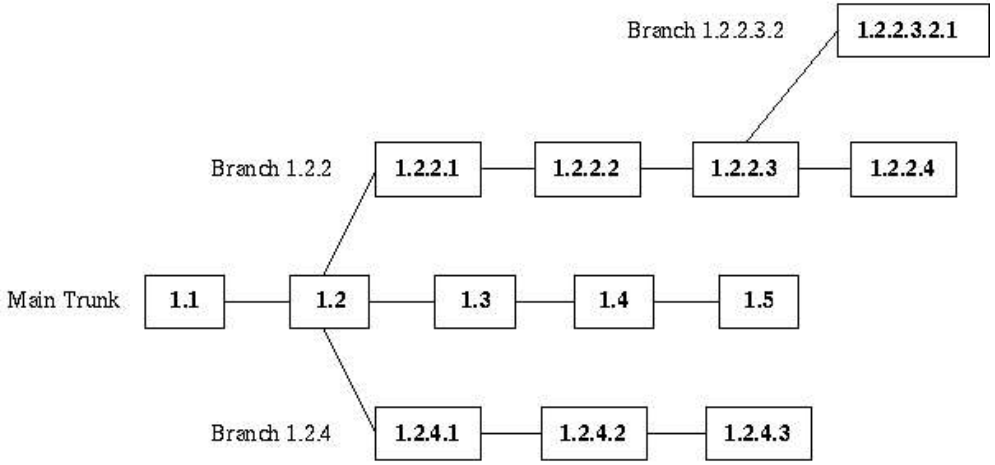
  Revisions:
  - jede Version einer Datei hat eine einzigartige *revision number (RN)*
  - solch eine *RN* hat immer eine gerade Anzahl von Zahlen, die durch Punkte getrennt sind
  - 1.1 ist die Initial-*RN* die eine Datei bekommt
  - folgende Versionen werden dann an der letzten Stelle durchnummeriert

CVS – Begriffe(IV)

Branches/Versions:

- jede Branch hat eine eigene *branch number* (*BN*)
- eine *BN* wird erstellt indem an die aktuelle *RN*, von der die neue Branch abgespalten wurde, eine Nummer angehängt wird
- Von einer Revision können mehrere Branches abgespalten werden

Beispiel:



CVS - Inhalt

- Einleitung/Motivation
- **Arbeiten mit CVS**

  Begriffe

  Tutorial

CVS - Tutorial

- **1. Verbindungsaufbau:**
  - Für Zugriff CVSRoot Variable setzen:

```
$ setenv CVSROOT /work/cvsroot
```

(:rsh/pserver/kerberos:joeuser@foo.bar.com:/path/to/cvsroot)

  - Mit CVS-Server verbinden:

```
$ cvs login
```

CVS - Tutorial

- **2. Modul erstellen:**

```
$ mkdir hello
$ cd hello
$ cvs import –m “first Project“ hello new start
```

→leeres Modul „hello“ wird in Repository erstellt

CVS - Tutorial

Verzeichnisstruktur im Repository:

```
/work
|
+--cvsroot
|    |
|    +--CVSROOT (administrative Dateien)
     |
     |
     +--hello
```

CVS - Tutorial

- **3. Lokale Kopie von CVS System holen:**

```
$ cvs checkout hello
```

→Ein lokales Verzeichnis mit dem Namen „hello“ wird erstellt und mit den Dateien aus dem Repository gefüllt

CVS - Tutorial

- **4. Dateien Hinzufügen/Entfernen:**

Hinzufügen:

```
$ cvs add hello.c
```

→Datei hello.c wird beim nächsten commit ins Repository eingefügt

Entfernen:

```
$ cvs remove hello.c
```

→Datei hello.c wird beim nächsten commit aus Repository entfernt

CVS - Tutorial

- **5. Änderungen ins Rep. übernehmen:**

$ **cvs commit** (–m “log message“)

→Änderungen bzw. neue Versionen werden mit angegebener “log message“ Vermerk ins Repository übertragen.

→wird –m Option weggelassen, so wird zur Eingabe eines Vermerks der unter der Umgebungsvariable $EDITOR angegebene Editor gestartet

CVS - Tutorial

- **6. Lokalen Workspace mit Rep. abgleichen:**

$ cvs update

→Dateien aus dem lokalen Checkout werden mit Repository verglichen und dabei mit einem vorangestellten Buchstabencode versehen...

- **U – updated** – Datei wurde im Repository aber nicht im Checkout verändert →Checkoutversion wurde mit Rep. abgeglichen
- **M – merge** – Diese Datei wurde im Checkout verändert, konnte aber problemlos mit Datei im Repository gemischt werden
- **C – conflict** – Die Datei im Checkout wurde verändert und kann nicht mit der im Repository gemischt werden (Kommunikation zw. Entwicklern)
- **? – unknown –** Diese Datei wurde ignoriert

CVS - Tutorial

- **Beispiel für Conflict Management(I)...**
  - Zwei Entwickler arbeiten auf Datei „hello.c“
  - Ursprünglich:

```
main(){
    puts(“Hello World“);
}
```

  - Entwickler verändert puts() zu printf() und comittet:

```
main(){
    printf(“Hello World“);
}
```

CVS - Tutorial

- **Beispiel für Conflict Management(II)...**
  - Konsolenausgaben:

$ cvs update

```
cvs update: Updating .
M hello.c
```

$cvs commit –m „printf is better“

```
cvs commit: Examining .
cvs commit: Committing .
Checking in hello.c;
/work/cvsroot/hello/hello.c,v <-- hello.c
new revision: 1.2; previous revision: 1.1
done
```

CVS - Tutorial

- **Beispiel für Conflict Management(III)...**
  - Währenddessen fügt ein weiterer Benutzer ein Komma und ein Ausrufezeichen hinzu:

```
main(){
      puts(“Hello, World!“);
}
```

CVS - Tutorial

- **Beispiel für Conflict Management(IV)...**
  - Dies führt zu Konflikt beim Update:

```
$ cvs update
cvs update: Updating .
RCS file: /work/cvsroot/hello/hello.c,v
retrieving revision 1.1.1.1
retrieving revision 1.2
Merging differences between 1.1.1.1 and 1.2 into hello.c
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in hello.c
C hello.c
```

CVS - Tutorial

- **Beispiel für Conflict Management(V)...**
  - Der Problembereich wird in Datei mit <<<<<<< und >>>>>>> markiert und muß von Hand gelöst werden

```
main(){
  <<<<<<< hello.c
          puts("Hello, World!");
  =======
          printf("Hello World");
  >>>>>>> 1.2
}
```

  - Solange sich nicht-abgeglichene Dateien im Checkout befinden, können die neuen Änderungen nicht eingepflegt(committed) werden

CVS - Tutorial

- **7. Eine neue Branch erstellen:**

$ **cvs rtag** –b –r *branchname module*

→eine neue Branch wird im CVS System erstellt

→diese muß jetzt ausgecheckt werden:

$ **cvs checkout** –r *branchname module*

→jetzt kann in dieser Branch gearbeitet werden und Änderungen betreffen nicht den Hauptentwicklungsstrang(main trunk)

CVS - Tutorial

- **8. Branch in akt. Checkout einfügen:**

$ **cvs update** –j *branchname*

→*branchname* wird in mit aktuellem Checkout „vermischt“

→Einfügen nur der Unterschiede aus verschiedenen Branches in aktuellen Checkout ist auch möglich:

$ **cvs update** -j *branchname –j branchname*

CVS - Tutorial

- **9. Keyword Expansion(I):**

  Bestimmte Schlüsselwörter können automatisch vom CVS- System durch entsprechende Informationen ersetzt werden...

  Id -- Dateiname, Revision, Datum, Autor, Status
  $Date$ -- Datum und Zeit des Eincheckens
  $Autor$ -- Autor der Datei
  $Revision$ -- Revision der Datei
  Log -- log Einträge werden hinzugefügt

CVS - Tutorial

- **9. Keyword Expansion(II):**
  - Beispiel:

```
/* $Id: hello.c,v 1.3 1996/11/15 22:14:05 swetland Exp $
* $Date: 1996/11/15 22:14:05 $
* $Author: swetland $
$Revision: 1.3 $ */

main(){
        printf(“Hello World“);
}
```

CVS - Tutorial

- **10. Tagging/Releases:**

```
$ cvs tag version_1_0
```

→damit kann man Dateien eine symbolische Bezeichnung geben

→später kann dann über diese symb. Bezeichnung eine bestimmte Version unabhängig von der aktuellen Dateiversion ausgecheckt werden

```
$ cvs checkout –r version_1_0 hello
```

CVS - Tutorial

- **11. Weitere nützliche Kommandos(I):**

$ **cvs status** *file*

Beispiel:

```
$ cvs status test.c
 ===================================================================
File: test.c Status: Needs Patch
Working revision: 1.10 Wed Jun 26 22:56:37 1996
Repository revision: 1.11 /X11/mosaic/cvsroot/xmosaic3/src/test.c,v
Sticky Tag: (none)
Sticky Date: (none)
Sticky Options: (none)
```

→gibt Statusinformationen über die jew. Datei an (working revision, repository revision)

CVS - Tutorial

- **11. Weitere nützliche Kommandos(II):**

$ cvs log *file*

Beispiel:
$ cvs log test.c **...**
description:
---
revision 1.11 date: 1996/11/13 17:10:33; author: spowers; state: Exp; lines: +9 –5
VMS is finally here!
---
revision 1.10 date: 1996/06/26 22:56:37; author: tpreilly; state: Exp; lines: +2 –0
Added inclusion of config.h
---------- **...**

→gibt Loginformationen über die jew. Datei an
(z.B.: alle Revisions + Datum + Autor +Kommentar)

CVS - Tutorial

- **11. Weitere nützliche Kommandos(III):**

$ **cvs annotate** *file*

→zeigt an wer was in der Datei verändert hat

CVS - Informationsquellen

- CVS-Homepage:
  - http://www.cvshome.org
- CVS-Manual:
  - http://www.cvshome.org/docs/manual/cvs.html
- CVS-HOWTo:
  - http://www.linuxdoc.org/HOWTO/CVS-RCS-HOWTO.html
- CVS-Einführung(dt.):
  - http://www.koehntopp.de/kris/artikel/cvs/
- CVS-Einführung(engl.):
  - http://www.cdt.luth.se/~peppar/presentations/cvs/