Maik Zemann

# CVS

(Concurrent Versions System)

The open standard for version control

# Gliederung

- Was ist CVS? Motivation?
- Konzept von CVS
- Die wichtigsten Befehle
- Merging
- Logging im Quelltext
- Tags, Branches und Releases
- CVS über SSH
- Links und weiterführende Information

# Was ist CVS?

- Versions Kontroll System
- Ermöglicht den Verlauf eines Projektes zu verwalten
- Jede Version jeder Datei jederzeit zugreifbar
- Einsatz zum Beispiel bei

# Warum Versionskontrolle?

- Multi-User tauglich
- Hilfe bei Bugfixes und Dokumentation
- Änderungen in Dateien können rückgängig gemacht werden
- Überschreiben von Dateien wird verhindert
- Speicherplatzsparende Sicherung aller Dateiversionen
- Änderungsverlauf wird nachvollziehbar

# Gliederung

- Was ist CVS? Motivation?
- Konzept von CVS
- Die wichtigsten Befehle
- Merging
- Logging im Quelltext
- Tags, Branches und Releases
- CVS über SSH
- Links und weiterführende Information

# Die Grundlegenden Konzepte (1)

- Inkrementelle Natur
- CVS speichert nur Unterschiede zwischen den Versionen einer Datei
- Jeder Benutzer arbeitet in einem eigenen Verzeichnis mit lokalen Kopien
- Mehrere Benutzer können zeitgleich konkurrierend an derselben Datei arbeiten
- Gewährleistet Unabhängigkeit einzelner Benutzer

# Die Grundlegenden Konzepte (2)

- Keine Locking-Mechanismen
- CVS behandelt ein Projekt als einfachen Dateibaum mit den dazugehörigen Dateien
- Im *Repository* (cvsroot) werden sämtliche Fassungen der Dateien inkl. der Änderungsprotokolle gespeichert
- Üblicherweise werden CVS nur *Quellen* unterstellt
- CVS protokolliert beim Einstellen Benutzer, Datum, etc.
- Achtung: CVS kann Änderungen nur syntaktisch auf Kollisionen überprüfen

# Gliederung

- Was ist CVS? Motivation?
- Konzept von CVS
- Die wichtigsten Befehle
- Merging
- Logging im Quelltext
- Tags, Branches und Releases
- CVS über SSH
- Links und weiterführende Information

# Vorbereitungen

- Umgebungsvariablen in **`.bashrc`**

1. **Repository:**

   **`export CVSROOT=/cvs`**

1. **Benutzerrechte (Gruppe):**

   **`export CVSUMASK=003`**

1. **Editor:**

   **`export CVSEDITOR=´emacs´`**

# Befehls-Syntax

- Alle CVS-Kommandos werden als Parameter des Befehls **`cvs`** übergeben:

```
$ cvs [opt.] kommando [opt.] [<filename>]
```

# Initialisieren

- Einrichten eines globalen CVS-Verzeichnisses nach setzen der Umgebungsvariablen:

```
$ cvs init
```

- Alternativ kann mit der Option **-d** der Pfad gesetzt werden:

```
$ cvs –d /cvs init
```

- Dadurch wird im Verzeichnis **/cvs** ein Unterverzeichnis **CVSROOT** angelegt

# CVS ein Projekt unterstellen

- Zur Erinnnerung: Projekt = Verzeichnisbaum
- CVS kann beliebig viele Projekte verwalten
- Wechsel in die oberste Verzeichnisebene des Projektes

```
$ cvs import <modulname> <vendor-tag> <release-tag>
```

**<modulname>** = Name des Wurzelverzeichnisses des Projektes im Repository

**<v-tag> und <r-tag>** = Können auf Dummy-Werte gesetzt werden

# Auschecken

- Zum bearbeiten der Dateien muss man sich eine lokale Kopie des Projektes erzeugen

```
$ cvs checkout <modulname>
```

- Bestimmte Versionen wiederherstellen

```
$ cvs checkout -r  <revision> <modulname>
```

- Dadurch wird im aktuellen Verzeichnis ein Unterverzeichnis mit dem Modulnamen erzeugt

# Einchecken

- Nach dem Bearbeiten einer oder mehrerer Dateien im lokalen Arbeitsverzeichnis müssen die Änderungen CVS mitgeteilt werden

```
$ cvs commit <filename>
```

- Dadurch wird ein Editor geöffnet in dem Änderungen dokumentiert werden können

- Alternativ kann durch die Option **`–m ``Nachricht```** eine Mitteilung per Kommandozeile übergeben werden

# Dateien / Verzeichnisse hinzufügen

- Nachdem im lokalen Arbeitsverzeichnis eine neue Datei oder ein neues Unterverzeichnis erzeugt wurde merkt man diese zum hinzufügen zu CVS vor

```
$ cvs add <filename>
```

Oder

```
$ cvs add <verzeichnisname>
```

- Anschließend müssen diese Dateien mit **`cvs commit <filename>`** an CVS übermittelt werden

# Dateien / Verzeichnisse löschen

- Zuerst müssen die zu löschenden Objekte lokal gelöscht werden mittels **`rm`**
- Anschließend die Operation vormerken durch

```
$ cvs delete <filename>
```

- Anschließend müssen die Änderungen mit **`cvs commit <filename>`** an CVS übermittelt werden um global im Repository gelöscht zu werden
- Zur Sicherheit werden die gelöschten Dateien von CVS aufbewahrt

# Weitere nützliche Befehle

```
$ cvs log <filename>
```

- gibt eine Liste aller log-Nachrichten aus

```
$ cvs diff –r<revision> -r<revision> <filename>
```

- listet Unterschiede zwischen zwei Versionen auf

```
$ cvs commit –r <version>
```

- legt Versionsnummern fest

# Ein Beispiel (1)

Im Verzeichnis `~/` liegt eine Datei `test` mit dem Inhalt ``Hello``

*user@host:~/$* `cvs init` --initialisiert Repository

*user@host:~/$* `cvs import project d1 d2` --unterstellt `test` dem Modul `project`

Editor wird geöffnet – Kommentar eingeben – speichern und beenden

```
N project/test
No conflicts created by this import
```

*user@host:~/$* `cvs checkout project` --checkt Modul `project` aus

```
cvs checkout: Updating project
U project/test
```

Neues Unterverzeichnis `./project` mit der Datei `test` und dem Unterverzeichnis `CVS`

Inhalt der Datei `test` in `./project` wird auf ``Hello World`` geändert

*user@host:~/project$* `cvs commit test` --übergibt CVS die Änderungen

Editor wird geöffnet – Kommentar eingeben – speichern und beenden

```
checking in test;
/cvs/project/test,v <-- test
new revision:1.2;previous revision 1.1
done
```

# Ein Beispiel (2)

```
user@host:~/project$ cvs diff –r1.1 –r1.2 test
```
--zeigt Unterschiede zwischen Versionen

```
Index: test
===================================
RCS file: /cvs/project/test,v
retrieving revision 1.1
retrieving revision 1.2
diff –r1.1 –r1.2
1c1
< Hello
---
> Hello World
user@host:~/project$ rm test
```
--löscht `test` lokal

```
user@host:~/project$ cvs delete test
```
--merkt `test` zum löschen vor

```
Cvs remove: scheduling `test` for removal
Cvs remove: use `cvs commit` to remove file permanently
```

# Ein Beispiel (3)

```
user@host:~/project$ cvs commmit
```
--überträgt Änderungen an CVS

```
Cvs commit: Examining .
```

Editor wird geöffnet – Kommentar eingeben – speichern und beenden

```
RCS file: /cvs/project/test,v
removing test
/cvs/project/test,v --> test
done
user@host:~/project$
```

# Gliederung

- Was ist CVS? Motivation?
- Konzept von CVS
- Die wichtigsten Befehle
- Merging
- Logging im Quelltext
- Tags, Branches und Releases
- CVS über SSH
- Links und weiterführende Information

# Merging (1)

- Da mehrere Benutzer zeitgleich Änderungen am Repository vornehmen empfiehlt es sich vor jedem **`cvs commit`** ein **`cvs update`** durchzuführen
- Dies führt zur lokalen Synchronisation mit dem Repository, ohne Verlust der lokalen Änderungen
- Wurden Dateien von unterschiedlichen Benutzern an gleicher Stelle verändert, kommt es zu Konflikten
- Beide Versionen werden dann in die Datei aufgenommen und eine Warnmeldung ausgegeben

# Merging (2)

```
...
<<<<<<< filename
 ... lokaler Text ...
=======
 ... globaler Text ...
>>>>>>> 1.3
 ...
```

--Ausgabe in der Datei

```
cvs update: Updating .
RCS file: /home/aj/cvsroot/Test/filename,v
retrieving revision 1.2
retrieving revision 1.3
Merging differences between 1.2 and 1.3 into filename
        rcsmerge: warning: conflicts during merge
        cvs update: conflicts found in filename
```

--Fehlermeldung

# Gliederung

- Was ist CVS? Motivation?
- Konzept von CVS
- Die wichtigsten Befehle
- Merging
- Logging im Quelltext
- Tags, Branches und Releases
- CVS über SSH
- Links und weiterführende Information

# Logging im Quelltext

- Gestattet Versionsinformationen in der Datei explizit zu machen durch Schlüsselwörter:
  - $Author$
  - $Date$
  - $Revision$
  - $Logs$
  - $Id$
- Schlüsselwörter werden von CVS beim ein- bzw. auschecken mit der entsprechenden Information ersetzt

# Gliederung

- Was ist CVS? Motivation?
- Konzept von CVS
- Die wichtigsten Befehle
- Merging
- Logging im Quelltext
- Tags, Branches und Releases
- CVS über SSH
- Links und weiterführende Information

# Tags

- Tags ermöglichen die symbolische Benennung einzelner Versionen
- z.B. zur zeitlichen Zusammenfassung mehrerer Dateien
- Jederzeit zusammenhängend zugreifbar

# Releases

- Snapshot des derzeitigen Standes der Entwicklung
- Erzeugung durch

```
$ cvs tag <release>
```

- Zugriff jederzeit durch

```
$ cvs checkout –r <release> <modulname>
```

# Branches

- Äste, die unabhängig vom Haupt-Baum im Repository bearbeitet werden können
- Sie können eigenständig bleiben, oder später dem Haupt-Baum hinzugefügt werden
- Erzeugung durch

```
$ cvs tag –b <branch>
```

- Zugriff durch

```
$ cvs checkout –r <branch> <modul>
```

# Gliederung

- Was ist CVS? Motivation?
- Konzept von CVS
- Die wichtigsten Befehle
- Merging
- Logging im Quelltext
- Tags, Branches und Releases
- CVS über SSH
- Links und weiterführende Information

# CVS über SSH

- Ausgehend von einem bestehenden Repository auf dem Server
- Folgende Umgebungsvariablen müssen auf dem Client gesetzt werden

```
$ export CVSROOT=":ext:user@remotehost:/pathto/repository"
$ export CVS_RSH="ssh"
```

- Alle Befehle wie gewohnt verwenden

# Gliederung

- Was ist CVS? Motivation?
- Konzept von CVS
- Die wichtigsten Befehle
- Merging
- Logging im Quelltext
- Tags, Branches und Releases
- CVS über SSH
- Links und weiterführende Information

# Links und weiterführende Information

- http://www.suse.de/de/private/support/online_help/howto/cvs/cvs.html
- http://www.suse.de/de/private/support/online_help/howto/cvs/cvs2.html
- http://www.techfak.uni-bielefeld.de/ags/ni/lectures/internstuff/howto/howto-CVS.html
- http://www.abiody.com/jfipa/publications/CVSGuide/index.php?lang=de --(ssh)
- http://www.cvshome.org