CVS: Concurrent Versioning System

--Free version control system software--

Jack Ivy, Levi Throckmorton, and Stuart Vassey

History

- Developed by Dick Grune in the 1980s
  - From an earlier versioning-system called Revision Control System (RCS)
- Originally created by Grune in order to collaborate with students since they had totally different schedules
- Released under the GNU General Public License
- A group of volunteers maintain code today
- Subversion created to improve on CVS
  - (it's better)

The Basic Idea

- Software that keeps track of all work and all changes in a set of files
- Allows several developers to collaborate on a project
- Developers can work anywhere and any time they wish

Some Terminology

- **Repository** – the location that CVS stores and manages its modules
- **Module** – labels a single project (set of related files)
- **Check Out** – programmers receive copies of modules by checking them out
- **Working Copy** – what the checked out files make up

Terminology (con't)

- **Commit** – to submit changes made to the working copy (changes repository files)
  - Programmers cannot commit changes unless they have most recent copy of files (current version)
- **Current Version** – collectively, most up to date files in the repository
- **Update** – to acquire the changes in the repository and add them to the working copy

How It Works

- (typically) Client-server architecture: server stores current version and its history
- Clients connect in order to “check out” a copy of the project
- As clients work on and change their working copy, commits can be made to change current version
  - Version number automatically increments
- Server only accepts changes to most recent version

Features

- Use CVS to:
  - Compare versions
  - Get complete history of changes
  - Check out project as of given date or revision number
  - Update: only downloads changes
  - Maintain separate branches of project
    - Bug fixes
    - Under current development
    - Major changes

Different Uses

- Large software projects
  - Programmers collaborate all over space-time to create awesome programs for us!
- Paper collaboration
  - Professors at different Universities able to work together on papers
- Attempted to use with powerpoint, not much success there

How to Use

- In terminal
- Within eclipse
  - Already built in!

Eclipse

- Ideal for large software projects
- Already integrated!
- Easy to use
  - Don't have to remember command syntax

Terminal

- Useful for almost any project, large or small
- Any Unix machine works
- Syntax
  - Must remember the commands

DEMOS

DEMO 1

Terminal Demo

Invoking CVS in the Terminal

- Format (like other programs) is “cvs {command}”
- Before starting, need to tell CVS location of repository
  - Syntax: “cvs -d {nameOfRepos} {command}”
  - Can (and should) set CVSROOT variable if working with same repository over and over
- Process of creating a new project = importing
  - Syntax: “cvs import -m “msg” projname vendortag releasetag
    - -m signifies a log message – every commit (including the initial import) has to have this
    - Vendortag and releasetag largely unimportant

Various Basic Commands

- Checking out a project
  - “cvs checkout (OR co) {proj}”
- Changing a file
  - Simply open with file editor (“jedit {filename}”)
  - Save changes
- Updating your working copy
  - “cvs update {opt. Filename}”
  - Modified files will appear with an “M” next to their name
  - Can be restricted to certain files, usually is not

More Basic Commands

- Sending your modifications to repository
  - “cvs commit (OR ci) -m 'msg' {opt. Filename}”
  - Can commit certain files or all changed files
  - When file is committed, last portion of revision number is incremented by one
- Checking status of files
  - “cvs status {opt. Filename}”
  - If no file specified, shows status of all files
- Finding out who did what to a file
  - “cvs log {opt. Filename}”
  - You really want to specify a filename here

Advantages and Limitations

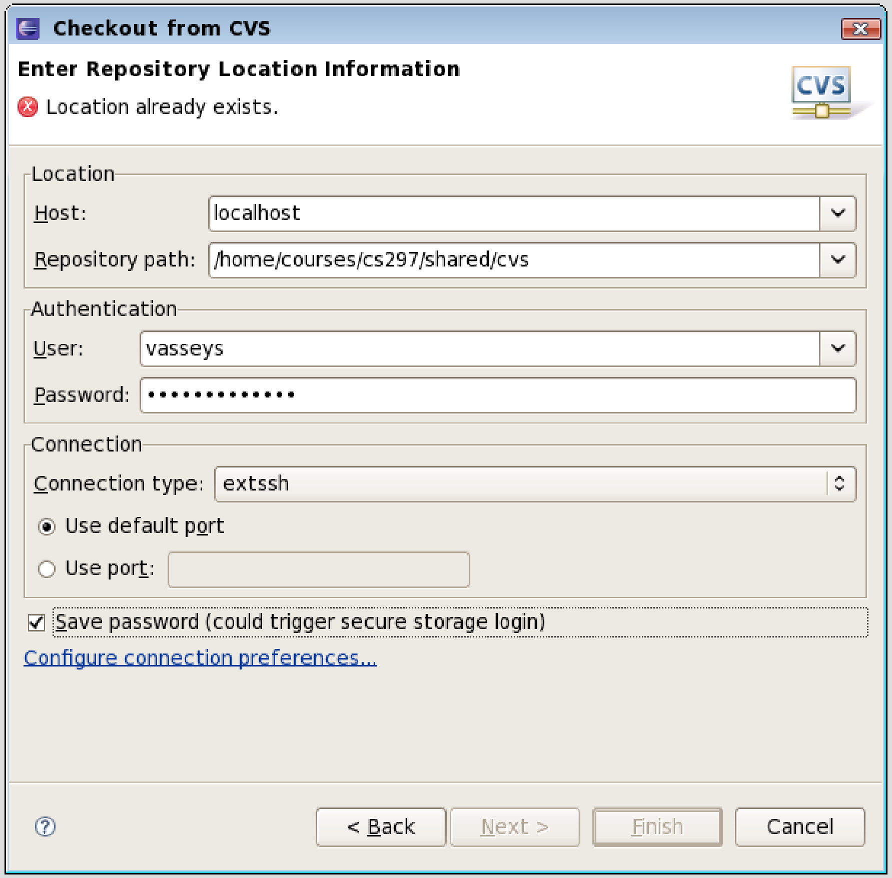
- Advantages
  - Quick and easy
  - All in the terminal
  - Simple language
  - Not many tasks necessary
- Limitations
  - You can't really see what you're doing
  - The UNIX bad parent thing
  - Potentially complex

DEMO 2

Eclipse Demo

Accessing a Repository

- Open eclipse in command line
- Go to **file>new>other**
- Select the cvs folder and **choose project from cvs**
- Select **create a new repository location**
- Use the following settings
- Next select **use an existing module** and choose **jbidwatcher**



Checkout from CVS

Enter Repository Location Information

Location already exists.

Location

Host: localhost

Repository path: /home/courses/cs297/shared/cvs

Authentication

User: vasseys

Password: ••••••••••••••

Connection

Connection type: extssh

[✓ Use default port]

Use port:

[✓] Save password (could trigger secure storage login)

Configure connection preferences...

Looking at the Project

- Can use the standard java perspective or cvs perspective
- Switch back and forth using buttons in top right corner
- Notice the version numbers next to the files
- When you modify a file a **>** will appear in front of the file name
- Right clicking on folders or files give you the cvs options. Many under **team**

Modify and Commit a File

- Everyone choose a java file from the **javazoom.jlme.decoder** folder
- Add in a ***//comment***
- Save the file
- Commit the file back to the repository by right clicking on the file and selecting **team>commit**
- You must have the current version of the repository to be able to commit
- If you get a commit error, then right click on the file and select **team>update**

View Modification History

- Update using right click, **team>update**
- Right click again **show in>history**
- Go to the history pane in the bottom of eclipse

Compare and Replace Your File to the Repository

- Modify a file of your choice with a comment
- Right click on the file **compare with>select HEAD**
- This opens up the file in a dual pane comparison view
- You can compare the differences
- Replace a file
- Right click on file or folder and select **replace>from HEAD**