CISC 275: Introduction to Software Engineering

Lab 5:
Introduction to Revision Control with

SUBVERSION

Charlie Greenbacker
University of Delaware
Fall 2011

Overview

- Revision Control Systems in general
- Subversion at a Glance
- Compare with “another” revision control system
- Main Features
- Typical Use Case
- Basic Commands
- Lab Exercise

Source Code Management

- "Too many cooks spoil the broth"
- Large projects = large teams = lots of source code
  - How do you keep everything straight?
- Tracking changes, keeping logs
- Juggling many multiple versions
- Provide "sandbox" for developers
- Integrating different components of project
- Sharing files & tracking w/spreadsheets won't cut it!

Key Services Provided

- Store source code in a repository to work from
- Allow programmers to check-out a working copy
- Modify & commit changes back to repository
  - Or release and start over from fresh copy
- Maintain extensive information about revisions
- Enable updating of (sync) changes since checkout
- Merge different sets of changes, resolve conflicts
- Roll-back changes if problems develop

Historical Context

- 1972: Source Code Control System (SCCS)
  - Introduced "weaving" for file content revisions
- Early 1980s: Revision Control System (RCS)
  - Stored revisions with diff utility, RCS file format
  - Operated on individual files, not entire project
- 1986: Concurrent Versions System (CVS)
  - Extended manager RCS file sets (a project)
  - Client-Server model with central repository
  - Allows several developers to work concurrently

Subversion at a Glance

- Designed to fix shortcomings & "misfeatures"
  - Goal: (mostly) compatible successor to CVS
- More logical version numbering method
- Better file & directory management
- Web-based repository access
- Machine-readable (parseable XML) output
- Better support for binary files
- Used by Apache, GNU, KDE, Google, SF.net
- Free Open Source Software, Apache License

Comparison with CVS

CVS

- RCS plain file storage
- Works... slowly
- File revision numbers
- Diff file contents only
- Prefers text files
- Interrupted commits!
- Rollback can remove bad commits
- No new features

Subversion

- Relational database
- Noticeably faster
- Universal numbering
- Diff meta data, dirs, etc.
- Unicode, binaries, ok
- Atomic, “all or nothing”
- New code overwrites, bad code persists
- Still actively developed

Brief Note...

- Distributed Version Control
  - Peer-to-peer vs client-server (SVN, CVS)
  - No “canonical copy” in central repository
  - Synchronization via exchanging patches
  - Most popular: Git & Mercurial
- *My take: great for open source & startup firms, not for traditional enterprise development*

Main Features

- "Meant to be a better CVS"
  - Similar interface, repository migration tools
- Revision numbers assigned per commit, not file
- Improved revision history management
  - Directories, renames, meta-data all versioned
  - Not just file existence & contents
- Apache server for WWW repository browsing
- XML-based log output, automated scriptability
- Efficient storage of project branches & binaries

Typical Use Case

- Check-out working copy from repository
- Modify small part of code (add feature, fix bug, etc.)
  - Compile & test extensively outside repository
- If modification is successful:
  - Are you sure it really works? Test it again!
  - If so, add comment & commit to repository
  - Now others can get update their w/ your changes
- Else, drop changes and start over
- Rule of Thumb: Commit Early & Commit Often!

Basic Commands

- Checking out project source code:
  `svn checkout URL [revision] [path]`
  - Creates a working copy of the project
  - Keeps your actions separate from the baseline
  - Allows you to tinker with your copy without risk of messing up the entire project
  - Think of it as a 'sandbox'
    - Nothing you do is permanent until you say it is
    - You can check-out multiple copies to try different things out

Basic Commands

- Add new file or directory
  `svn add path`
  - Schedules item for addition to repository
    - Gets added next time you commit
  - Use `svn add *` to add everything you've created

- Delete item
  `svn delete path`
  - Schedules item for removal from repository
  - Again, no lasting changes until you commit

Basic Commands

- Update your working copy
  `svn update [path]`
  - Grabs any new changes from the repository since your most recent update or checkout
    - Keeps your work in sync with latest revision
  - For each updated item, prints line w/ status code
    - A = Added, D = Deleted, U = Updated, C = Conflicted, G = Merged
    - C means your edits overlap new changes
    - M if local edits were compatible with updates

Basic Commands

- Commit local changes
  `svn commit [path]`
  - Uploads changes from local copy to repository
    - This is how you share your code with the team
  - Run `svn update` first to check compatibility
  - Add comments with `--file` or `--message` option, or editor will be launched for comment
    - Use descriptive commit messages
    - "Latest Revision" is not descriptive enough!

Basic Commands

- View commit log messages
  `svn log [path]`
  - Shows messages from repository
  - Without specific argument, displays for everything

- Display in-line author & revision information
  `svn blame path`
  - Prints code line by line, with name of programmer and revision number for latest change

Basic Commands

- Display status of working copy files
  `svn status [path]`
  - Find out what modifications you've made so far
  - Provides output similar to `update`

- Show differences between items
  `svn diff ...` (lots of ways to use it)
  - Compares working copy to repository, between separate revisions, etc.

Basic Commands

- Discard local changes to a file
  `svn revert path`
  - Lets you start over with a fresh copy
  - Use `--recursive` flag to revert everything
  - Also removes scheduled operations (e.g. `Add`)
  - **Caution:** no way to undo this!

Lab Exercise

- Install SVN on your system (if you need it)
- Make sure you can connect to repository:
  - https://shuebox.nss.udel.edu/cisc275/shared/
  - https://shuebox.nss.udel.edu/cisc275/group0/
    ...
    https://shuebox.nss.udel.edu/cisc275/group7/

Next Lab

- More with Subversion
- Lab exercise:
  - Merging & dealing with conflicts