Software versioning and revision control systems

CSE219, Computer Science III

Stony Brook University

http://www.cs.stonybrook.edu/~cse219

Software versioning and revision control systems

- Revision control (also known as *version control*, *source control*, and *source code management*) is the management of changes to documents, computer programs, large web sites, and other collections of information.
  - A system for managing changes to files
  - Used by individuals and teams to keep:
    - History of changes,
    - Share and distribute common source code.
  - Think of it as a file database

Version Control System Services

- Backup and Restore
- Synchronization
- Short-term undo
- Long-term undo
- Track Changes
- Track Ownership
- Sandboxing
- Branching and merging

Backup and Restore

- Files are saved as they are edited
- One can jump to any moment in time
- Need that file as it was on August 23, 2014?
  - no problem, just ask the VCS for it

Synchronization

- Lets developers:
  - share files
  - stay up-to-date with the latest version
- Even while developers are working simultaneously.

Short-term Undo

- Editing a file and messed it up?
- Throw away your changes and go back to the “last known good” version in the database

Long-term Undo

- For particularly bad mistakes
- Suppose you made a change a year ago, and it had a bug
- Jump back to the old version, and see what change was made that day

Track Changes

- As files are updated, you can leave messages explaining why the change happened
  - stored in the VCS, not the file
- This makes it easy to see how a file is evolving over time, and why
- Developers should document every change

Track Ownership

- A VCS tags every change with:
  - the name of the person who made it
  - date/time of change
- Helpful for blamestorming

Sandboxing

- Insurance against yourself
- Making a big change?
- You can make temporary changes in an isolated area
    - test and work out the kinks before "checking in" your changes

Branching and Merging

- A larger sandbox
- You can branch a copy of your code into a separate area and modify it in isolation
  - tracking changes separately
- Later, you can merge your work back into the common area.

Setup Terms

- Repository (repo): The database storing the files.
  - Server: The computer storing the repository
  - Client: The computer connecting to the repository
- Working Set/Working Copy: Your local directory of files, where you make changes.
- Trunk/Main: The "primary" location for code in the repository
  - Think of code as a family tree — the "trunk" is the main line.

Basic Actions

- Add: Put a file into the repository for the first time, i.e. begin tracking it with Version Control
- Revision: What version a file is on (v1, v2, etc.)
- Head: The latest revision in the repository
- Check out: Download a file from the repository
- Check in: Upload a file to the repository (if it has changed).
  - the file gets a new revision number, and people can “check out” the latest one

Basic Actions

- Checkin Message: A short message describing what was changed
- Changelog/History: A list of changes made to a file since it was created
- Update/Sync: Synchronize your files with the latest from the repository
  - this lets you grab the latest revisions of all files
- Revert: Throw away your local changes and reload the latest version from the repository

Advanced Actions

- Branch: Create a separate copy of a file/folder for private use (bug fixing, testing, etc)
  - Branch is both a verb ("branch the code") and a noun ("Which branch is it in?")
- Diff/Change/Delta: Finding the differences between two files
  - useful for seeing what changed between revisions.

Advanced Actions

- Merge (or patch): Apply the changes from one file to another, to bring it up-to-date
  - For example, you can merge features from one branch into another
- Conflict: When pending changes to a file contradict each other
  - both changes cannot be applied
- Resolve: Fixing the changes that contradict each other and checking in the correct version

Advanced Actions

- Locking: “Taking control” of a file so nobody else can edit it until you unlock it.
  - some VCSs use this to avoid conflicts.
- Breaking the lock: Forcibly unlocking a file so you can edit it.
  - may be needed if someone locks a file and leaves
- Check out for edit: Checking out an “editable” version of a file
  - some VCSes have editable files by default, others require an explicit command.

Types of VCSs

- Revision Control System (RCS)
  - dead as a stand-alone system
- Concurrent Versioning System (CVS)
  - dying
- Subversion (SVN)
  - killing CVS
  - open source under the Apache license
  - http://subversion.apache.org/
- Distributed/decentralized revision control:
  - Git
  - Mercurial
- GNU Bazaar
- BitKeeper

- keeps track of software revisions
- allows many developers to work on a given project without requiring that they maintain a connection to a common network.

git

- Git:
  - GNU license
  - Free download: http://git-scm.com
  - Clients: http://www.sourcetreeapp.com, http://www.syntevo.com/smartgit
  - Repositories: GitHub, BitBucket (private repos. for <=5 users)
- Used by Linux kernel (original author Linus Torvalds)
- Used by permanent software development (report by itjobswatch.co.uk):
  - 20.32% git
  - 16.14% Subversion
  - 10.80% Microsoft Team Foundation Server
  - 1.39% Mercurial

git Common operations

- Setting Up a Git Repository:
  - git init: initializes a new Git repository.
    - If you want to place a project under revision control, this is the first command you need to learn.
  - git clone ?location: creates a copy of an existing Git repository.
    - Cloning is the most common way for developers to obtain a working copy of a central repository.
    - Example: **git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds**/
  - git add ?file: moves changes from the working directory to the staging area.
  - git commit: takes the staged snapshot and commits it to the project history.
  - git pull: downloads a branch from a remote repository, then immediately merges it into the current branch.
  - git push: move a local branch to another repository.

Apache Subversion (SVN)

- Developed by the Apache Software Foundation
- Distributed under Apache License (an open source license)
- Used by:
  - Apache Software Foundation,
  - Google Code,
  - FreeBSD,
  - GCC,
  - Mono,
  - SourceForge.
- Server-client model: Native SVN server or Apache HTTP Server.

SVN Common operations

- Import: is the act of copying a local directory tree (that is not currently a working copy) into the repository for the first time.
- Checkout: is to create a local working copy from the repository. A user may specify a specific revision or obtain the latest.
- Commit (check in or ci): is to write or merge the changes made in the working copy back to the repository.
- Update (or sync): merges changes made in the repository (by other people or by the same person on another machine) into the local working copy.
- Merge: is an operation in which two sets of changes are applied to a file or set of files: updates or syncs the user working copy with changes made and checked into the repository by other users + check in files + incorporate branches into a unified trunk.

Apache Subversion

- How to run SVN?
  - Command line: svn executable

    ```
    svn commit a.txt
    svn update
    ```

  - SVN Clients: TortoiseSVN, Netbeans SVN plugin, Eclipse Subclipse, etc.

Homework 1 Help

- Getting the Software:
  - NetBeans IDE
  - Java SE Development Kit 8.X
  - Git and a git client