Version Control Systems: Motivation

- Alice, Bob, and Chuck are working on a large software system
  - Where should they keep their source code?
  - What if they want to work on their laptops from home disconnected from the network?
  - How should they manage concurrent modifications?
  - What if Bob needs to keep the code stable to give a demo while Chuck would like to try a new idea?
  - What if Chuck tries his new idea and breaks the code the day of the demo?

Solution: Use a version control system!
Version Control System

- Goal of a version control system
  - Handle simultaneous concurrent changes
  - Manage multiple versions of a system
- Can manage any files, not just source code
  - I use it for everything... including course materials
- Many version control systems exist
  - CVS, RCS, **Subversion**, SourceSafe, ClearCase
- Just like any other tool that we study
  - All these tools have similar goals and similar basic features (but different ways to use these features)
Basic Idea

Repository

- Holds master copy of all files
- Holds old versions of all files

Alice's local copy
Bob's local copy
Chuck's local copy

Developers should NOT modify the repository directly
Instead, each developer checks out and modifies a working copy
Basic Idea

Repository

Alice's local copy
- Modifies files
- Adds files
- Adds directories

Bob's local copy

Chuck's local copy

update

commit

update
Basic Idea Summary

- There exists one repository
  - Holds the master copy of all files for all projects
- Each software developer
  - Checks-out a local copy of the files for a project
  - Modifies the files in the local copy
  - Commits his/her changes periodically
  - Updates his/her local copy periodically
    - To see changes made by other developers
  - Adds new files that he/she creates
- Developers use a program (e.g., svn) to interact with the repository and perform the operations listed above
What Goes Into The Repository

- In general: keep in repository ONLY what you need to build the application
  - Never add files that are generated automatically
  - Yes: .cc, .c, .h, Makefile
  - No: .o files or executable
- Think before you add a file to the repository
  - Although you can always remove it later if you make a mistake or if you change your mind

When adding a directory, svn adds all the files inside it automatically, which may include all your executables, etc.!
Basic SVN Commands

- Set-up a repository (this is done only once)
  `svnadmin create path`

- Add a new project to the repository (once per project)
  `svn import projectname foldername`

- Working on a local copy (frequent commands)
  Create local copy: `svn co files`
  Commit changes: `svn ci [files]`
  Update local copy: `svn update [files]`
  Add a new file or directory: `svn add files`
<table>
<thead>
<tr>
<th>command</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>svn add files</code></td>
<td>schedule files to be added at next commit</td>
</tr>
<tr>
<td><code>svn ci [files]</code></td>
<td>commit / check in changed files</td>
</tr>
<tr>
<td><code>svn co files</code></td>
<td>check out</td>
</tr>
<tr>
<td><code>svn help [command]</code></td>
<td>get help info about a particular command</td>
</tr>
<tr>
<td><code>svn import directory</code></td>
<td>adds a directory into repo as a project</td>
</tr>
<tr>
<td><code>svn merge source path</code></td>
<td>merge changes</td>
</tr>
<tr>
<td><code>svn revert files</code></td>
<td>restore local copy to repo's version</td>
</tr>
<tr>
<td><code>svn resolve source path</code></td>
<td>resolve merging conflicts</td>
</tr>
<tr>
<td><code>svn update [files]</code></td>
<td>update local copy to latest version</td>
</tr>
</tbody>
</table>

**Others:** blame, changelist, cleanup, diff, export, ls/mv/rm/mkdir, lock/unlock, log, propset
Log Messages

- Commit messages are mandatory
  - -m "short message"
  - -F filename-with-long-message
  - Else an editor pops up
    - Write your message
    - Save and quit

- Specify editor with SVN_EDITOR
  - For example, add the following to your .bashrc
    
    ```bash
    EXPORT SVN_EDITOR=emacs
    ```

Possible to setup SVN to send out email (with the log message) after each commit
Example: Setting-up Repository

Assume that we are team “team0” working on attu

Each team will get a shared directory of the form

```
/projects/instr/10wi/cse303/team0/
```

- First, we need to create a repository in our shared directory
  
  `$ svnadmin create /projects/instr/10wi/cse303/team0/svn`
  
  (you can name it something else than “svn”)

- Second, give read/write permissions to your cse303 group
  
  `$ chgrp -R mycse303group repofoldername`
  
  `$ chmod -R g+rwx,o-rwx repofoldername`
Example: Starting a New Project

• Add initial files into the repo:

  Execute the following from your home directory on attu

  $ mkdir hw3
  $ svn import hw3 file:///projects/instr/10wi/cse303/team0/svn/hw3
  $ svn import projectname foldername

• Check-out a working copy of the new project

  On attu (move the old hw3 used for the import and then):
  svn co file:///projects/instr/10wi/cse303/team0/svn/hw3

  On your laptop
  svn co svn+ssh://attu.cs.washington.edu/projects/instr/10wi/cse303/team0/svn/hw3
Example: Typical Work Session

• Start by getting any updates from the repository
  
  Execute the following from within your working copy of the project
  
  $ svn update

• Edit the files

• If you add a file, add it also to the repository
  
  $ svn add filename

  *common error*: people forget to add files (won't compile for others)

• Finally, commit your changes (note: this is when files get added)
  
  $ svn ci
Conflicts

- When many people edit the same files at the same time, **conflicts can occur**
- **SVN tries to merge changes automatically**
  - Merging is **line-based for text files**
  - If cannot merge, **svn** will ask you to resolve conflict
  - One possible strategy is the following
  - When asked what to do, choose “**postpone**”
  - Edit file in your favorite editor to remove conflicts
    - Search for `<<<<` signs
  - Tell **svn** that you are done:
    - `svn resolve --accept working filename`
Summary

- Version control system such as SVN
  - One of the key software development tools
  - All companies use them!

- Advantages
  - Much better than manually emailing files, adding dates or version numbers to files, etc.
  - Handles concurrent changes
  - Manages multiple versions
  - Remembers old versions
  - Useful for software but works on any files!
Readings

- Online SVN documentation
  - http://subversion.apache.org/
  - http://svnbook.red-bean.com/