# Using Subversion (SVN)

Overview

Subversion (SVN) is a version control system that allows you to track every change (or revision) made to different files, compare a file from one revision to another, and even revert to an old revision if necessary. In EECS150, benefits to using SVN are that:

1. Your data will be safe.
   a. SVN is hosted on the inst server.
   b. Anything you store in SVN will be accessible on any computer.
   c. You won't have to worry about moving files back to your U:/ drive manually (and losing data from forgetting to do this).
2. You can store all of your work in one place.
   a. Never again will you have to make different copies of your code when you want to keep a revision history of them. See below:
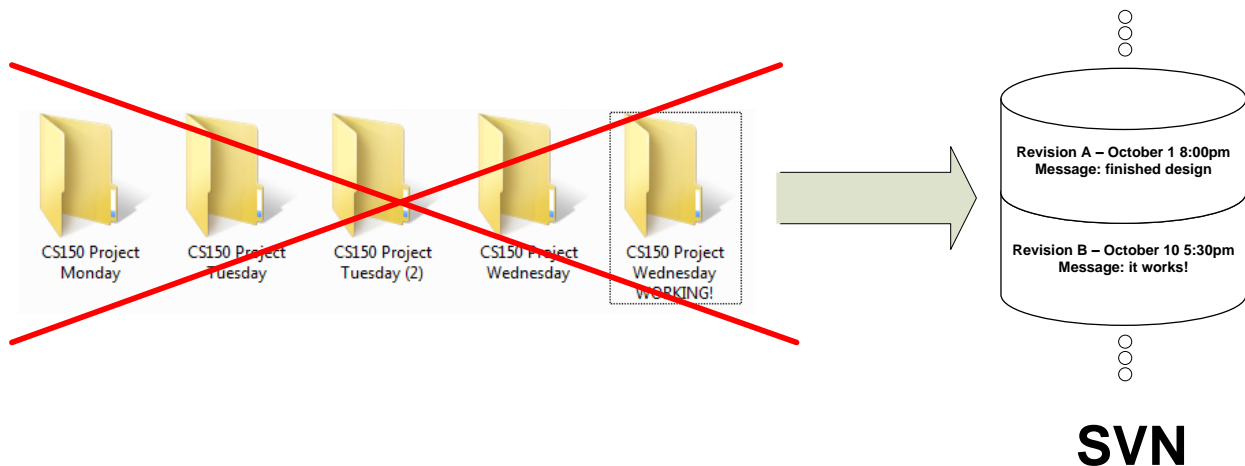
3. You will be able to safely work on the same code with your project partner (when the project actually starts…).
   a. If your partner breaks the project source, you will be able to see exactly what change introduced the bug (and eliminate it).
   b. Both you and your partner can modify the same file at the same time on different computers.
   c. If you decide that work done by either you or your partner made things worse, you can revert to any old version of whichever file at any earlier point in time.
4. You will get access to class library files
   a. Each project group directory in SVN has a copy of familiar files (`Counter.v`, `Register.v`, etc) inside them. **For Spring 2009:** this doesn't yet apply.

      b. As we distribute new material, you will have access to it locally and immediately, without having to search through the website.

This document covers the following:

1. Important concepts and jargon related to SVN.
2. Using these concepts with an SVN client (SmartSVN).
3. Getting access to your EECS150 SVN repository.

Important Concepts and Jargon Related to SVN

SVN stores your data in a **repository** (or database) which is situated either on your personal computer or on a server. In EECS150, repositories are hosted on the inst server. This allows you to **check out** (or download) your code (or whatever you have saved in the repository) onto any computer you wish.

The code that you check out is known as your **working copy**. Your working copy consists of all of the files that you checked out onto your computer. The files in your working copy behave exactly like any other files on your computer. You may modify them just like any other files. The only difference between files in your **working copy** and ordinary files is that you can **commit** changes made to files in your working copy back to the repository.

When you **commit** changes made to your working copy, you save a new revision of those files for all time. At any time in the future, you can look back on the revisions you made to a.) see what changes have occurred from then to now, and b.) **revert** (or move back to) to an old revision if necessary. When you commit a change back to the repository, you can write a **commit message** describing the change you made. Whenever you get a module working, you should commit that module and type the message: "I got it working!" or similar. This allows you to easily see what revision to revert to in case you introduce a bug at some later time.

Before you commit changes made to your working copy back to the repository, the repository only has your data as it was last committed. This means that if both you and your partner have different working copies (which is normal as each of you will need to check out your own copy), you will potentially have different versions of the same file. To keep both working copies as in-sync as possible, you will **svn update** as soon as you start working on your working copy, every time you start work on your working copy. **Again, the first thing you should do whenever you sit down to work on SVNed files is run SVN UPDATE.** If you forget to run svn update and modify a file that your partner was also modifying, you might run into a **conflict** when you and your partner both commit your changes back to the repository. You can resolve conflicts my looking though the file and deciding which pieces of your work and which pieces of your partner's work you really want to keep. Run **svn update** often to avoid conflicts; know, however, that none of your work will be lost if you have a conflict.

There are countless other features associated with SVN. The above are most important. Ask your TA if you have any questions.

Using an SVN Client: SmartSVN

Without a client program, you must work with SVN through the command line. This is extremely unwieldy and time-consuming, so we will now discuss how to use **SmartSVN**, an SVN client installed on all machines in 125 Cory.

1. From your desktop, open **Start→Programs→SmartSVN.**
    a. SmartSVN might be labeled with its version number (for example: "SmartSVN 4").
   b. The desktop icon for SmartSVN looks like this:

2. Open SmartSVN. You should see the following screen:
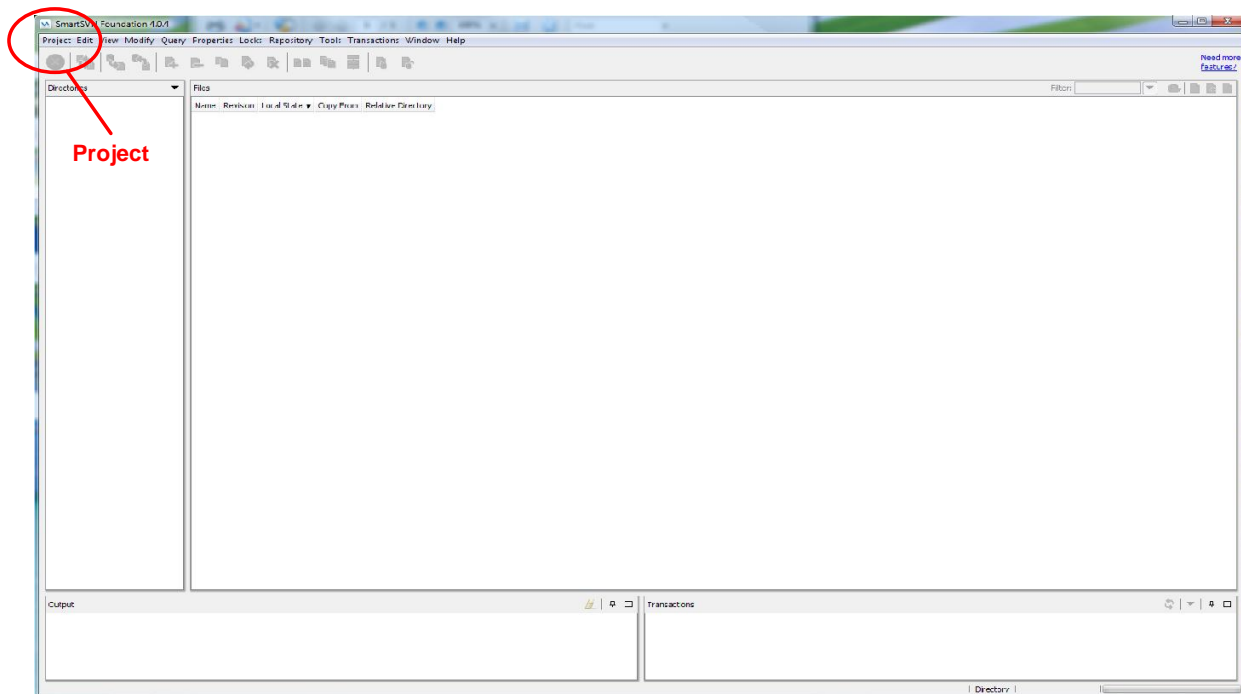
3. First, you must **check out** your part of the repository onto your local machine (this will be your **working copy**).
    a. Open **Project→Check Out…→Quick Checkout**
4. You should now be prompted with the **Check Out Project** dialog:
    a. For **URL** type: **https://isvn.eecs.berkeley.edu/cs150/sp09/<cs150-XY>**

          i.   You will replace **"<cs150-XY>"** with your actual login name. For example, a valid URL could be:
https://isvn.eecs.berkeley.edu/cs150/sp09/cs150-ab.

    b.   For **Local Directory**, browse to the directory on your local machine where you would like to store your working copy.

    c.   Click **Next.**

5.   You will be prompted for your **UNIX** username (**cs150-XY**) and password.

    a.   **If you did not change your UNIX password at the beginning of the semester (and instead only changed your WINDOWS password), please email Ilia.**

    b.   Click **Next.**
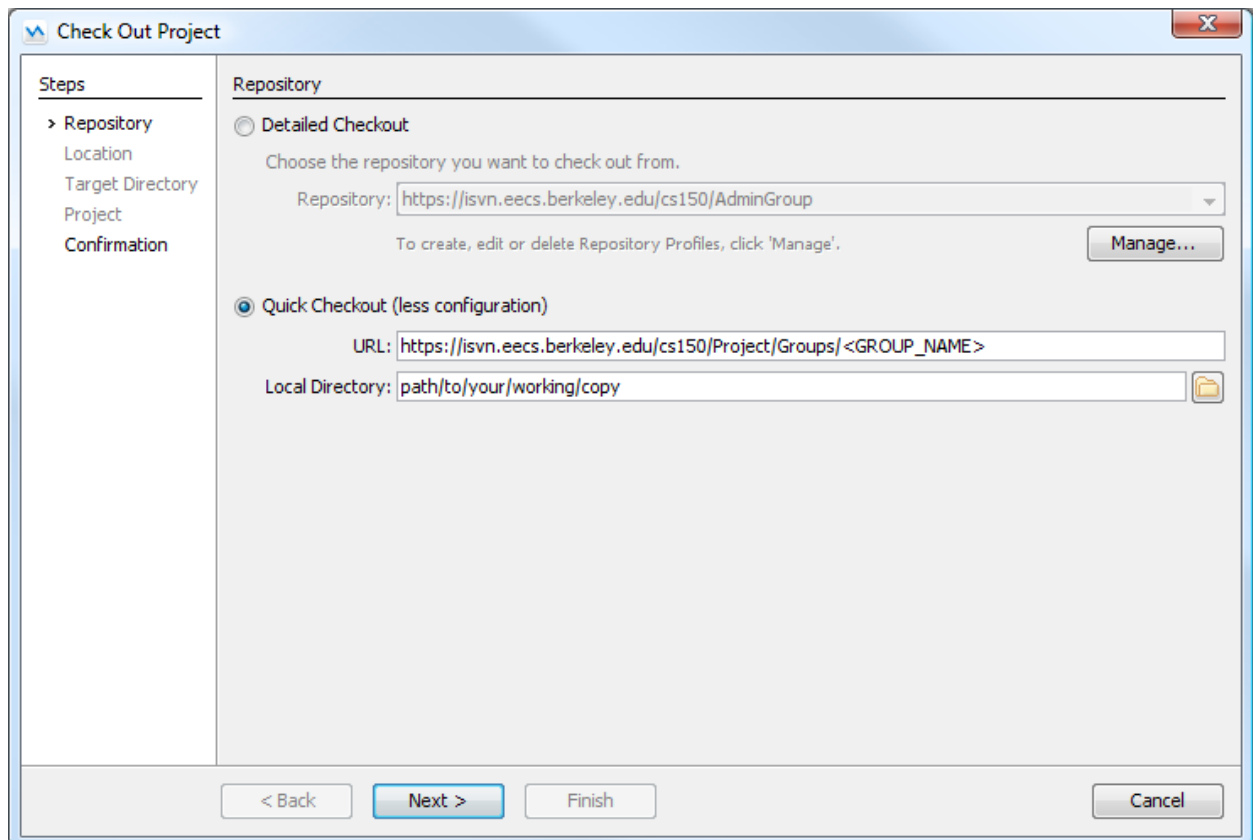
6. You should now have successfully **checked out** your code from the SVN repository. You should now see the following SmartSVN window, showing your working copy:
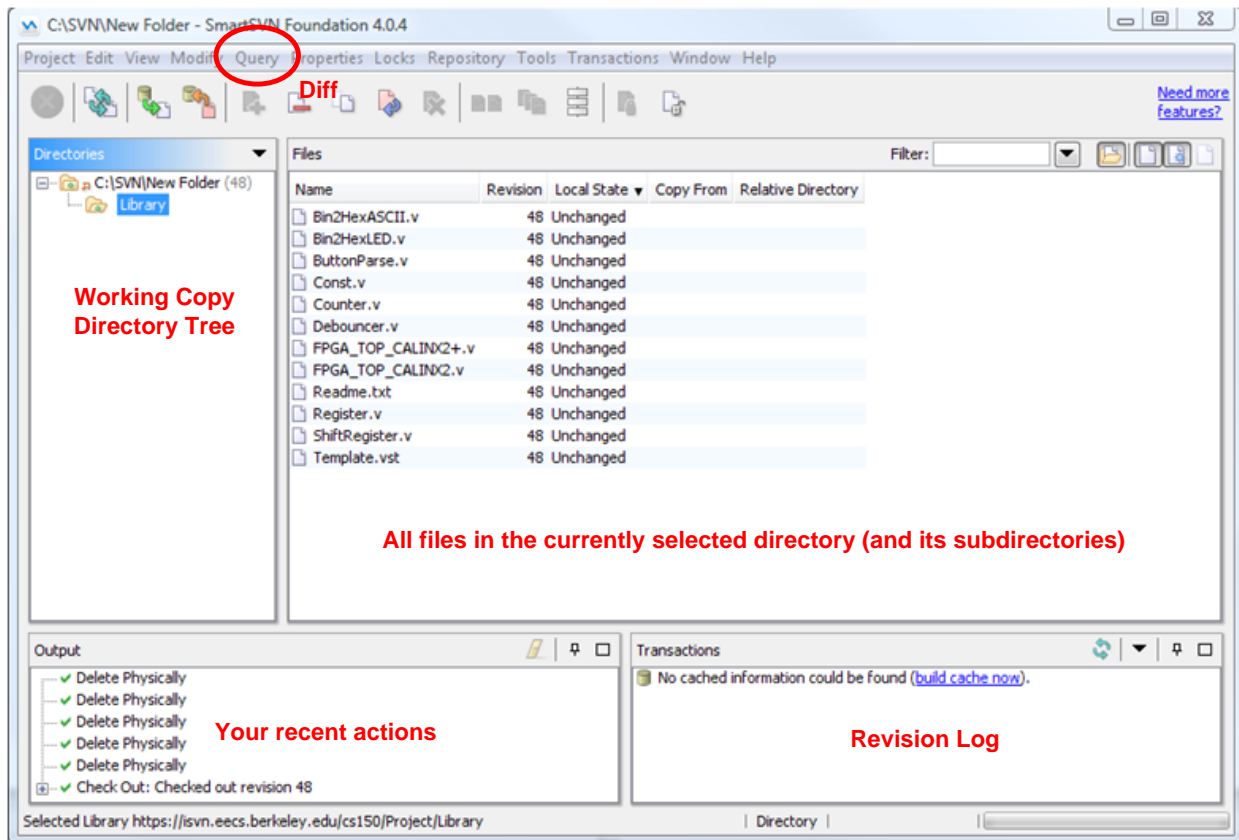
The following icons are also worth noting:

Perform the **SVN update** command on the currently selected file/directory tree.
  i. Select whether you want to update to **HEAD** (the current revision saved in the repository) or to a specific revision.
 ii. Click **Update.**

Run **SVN commit** on the currently selected file/directory tree.
  i.   Leave **Recurse into subdirectories** checked (unless you only want to commit a specific file).
  ii.   Type a **Commit Message** describing the changes you made (in English!).
  iii. Click **Commit.**

If you would like to compare the differences between your working copy and the most recent revision saved to the repository:

1. Select the file in the SmartSVN window that you would like to compare against the copy in the repository.
2. Open **Query→Show Changes.**
   a. You will see the **Pristine Copy** (the most recent revision committed to the repository) on the **left** and your working copy on the **right.**
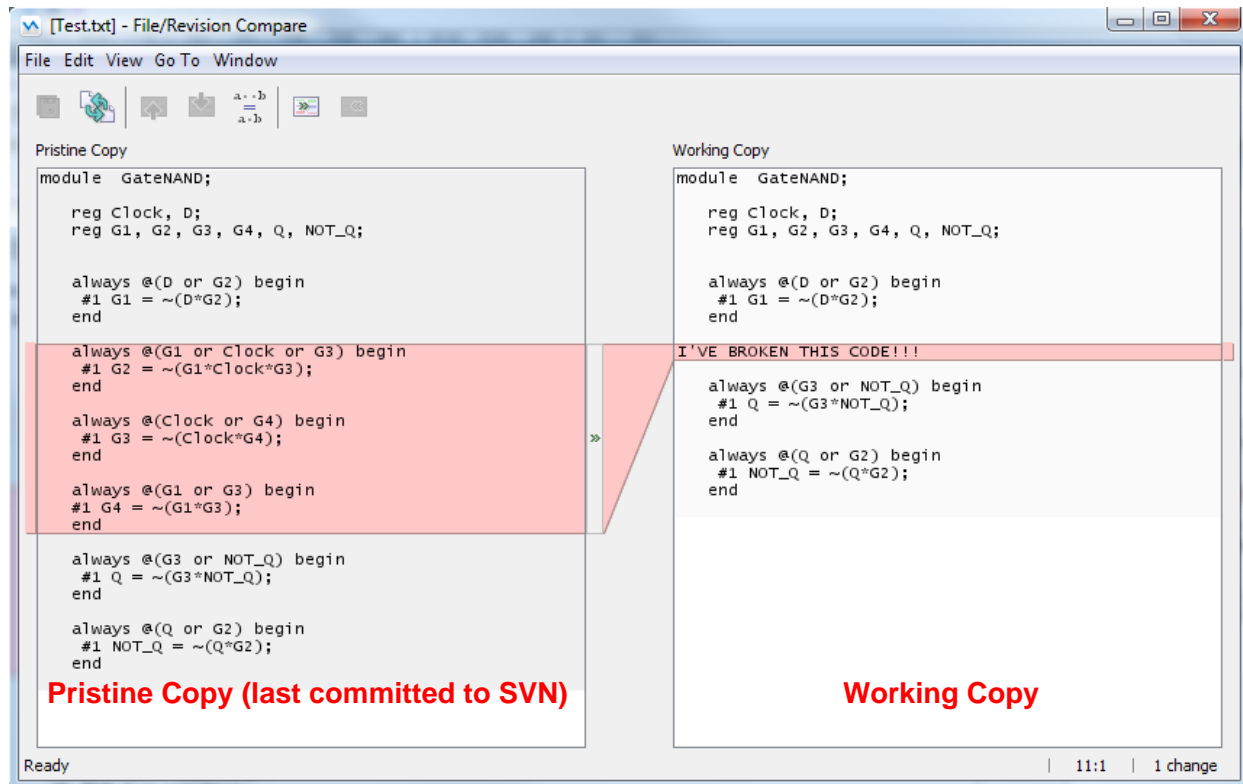
The above examples detail the most commonly used operations in SVN and SmartSVN. Obviously there are many others. The above should serve to get you started using SVN effectively as soon as possible. Feel free to experiment. You will likely use SVN (or another form of version control) for the rest of your time on this planet.

Getting Access to your EECS150 SVN Repository

As was partially detailed above, you need the following to get access to your EECS150 SVN repository:

1. Your computer account (cs150-XY) name
   a. If you don't have one by now, you are in trouble in other ways than not being able to access SVN. Please email the staff immediately if this is the case.
2. Your UNIX password.
   a. Again, this is **not** your windows password.
   b. If you did not reset your **UNIX password** through **ssh update** when you first got your **cs150-XY**, please email Ilia.

Know that project check-off and homework submissions (circa Homework 3) will only be conducted through the code/files you svn commit. **It is your responsibility to get all the SVN account information that you need to start using SVN.** We will **NOT** accept code for final checkoff, or homework files in general, in any other form except through SVN.

Start using SVN early. Version control is the most useful tool you have at your disposal when working on practically any project. Getting used to SVN now will save you much time and headache down the road.

| RevB – 02/07/09 | Chris Fletcher | Made edits relevant to CS150 Spring 2009. |
| RevA – 10/15/08 | Chris Fletcher | Wrote new tutorial. |