

SVN for Dr. Wartell's Courses

Authors: Professor Zachary Wartell

Revision: 9/12/2012 8:29:53 PM

Copyright August 2006, Zachary Wartell @ University of North Carolina at Charlotte
All Rights Reserved.

Table of Contents

1.	Prerequisites	1
2.	Overview	1
3.	Motivation.....	2
4.	Installing SVN.....	2
5.	Using SVN in this Class	3
5.1.	Read the SVN Manual.....	3
5.2.	Student Repository Location	3
5.2.1.	Verifying your Working Copy.....	5
5.3.	Project SVN Policies	7
5.3.1.	Submitting Projects	7
5.3.2.	Keep the repository clean!!!	7
5.3.3.	Double Check the Contents of Your Repository Directory.....	8
5.4.	Tips.....	8
5.5.	Common Error Messages	10
5.6.	Common Errors.....	11
6.	Citations	12

1. Prerequisites

This document assumes the reader has the following prerequisites:

- Basic knowledge of the Windows command-line interpreter (cmd.exe) or another command-line interpreter
 - See exercises 21 – 44 in The Command Shell section of the Windows Shell Tutorial [3].
 - Readers only familiar with Unix shells should also scan through the above exercises. cmd.exe —besides being very weak in comparison—also has various subtle differences.

2. Overview

If you are new to SVN read this entire document. If you are familiar with SVN read:

- 5.2 Student Repository Location
- 5.3 Project SVN Policies

3. Motivation

What is SVN?

SVN is a version control system. A version control system is software engineering tool. A version control system is like a common file server in that it stores an arbitrary set of directories and files that multiple people can access from their local computers. A version control system server stores the sharable directories and files in a specialized file system called a *repository*. Version control systems keep the entire historical record of all changes to the repository's files and directory structure along with information about which user made what changes to what files and when. Users can:

- retrieve the copies the repository files as the files existed at any particular point in time
- determine what changes other users have made to repository files
- merge changes to a file that has been independently modified by multiple users

Version control systems are necessary for large software projects developed by multiple programmers.

While course programming projects are often done individually, learning to use a version control system is an important software engineering skill. Students will use SVN to retrieve skeleton code and code examples and to submit all their programming assignments.

4. Installing SVN

SVN is a command-line program. GUI versions are also available. Tortoise SVN is the recommended SVN GUI. The instructions in this document focus on the command-line tool, but most of the commands are available from the GUI tools. Real world programmers learn to use both—for class purposes it is up to the student.

Computers in Woodward 335 have both the command-line SVN (hereafter just referred to as SVN) and Tortoise SVN installed. The command-line SVN should be run from the Windows command shell (cmd.exe). TortoiseSVN GUI is integrated into Windows Explorer.

Students choosing to work outside of Woodward 335 need to install SVN and/or TortoiseSVN from the following locations:

- 1) SVN – <http://subversion.tigris.org/> , the command-line tool
- 2) Tortoise SVN - <http://tortoisेस्वन.tigris.org/> , the recommended GUI tool
- 3) “Version Control with Subversion” - <http://svnbook.red-bean.com/> , the official manual

5. Using SVN in this Class

5.1. *Read the SVN Manual*

Read the chapters listed below of the SVN manual to understand how SVN works and how to use it. The SVN manual is freely available on line [2]. The most relevant chapters in the SVN manual are as follows:

Chapter 1: Fundamental Concepts: - read all

Chapter 2: Basic Usage

Help! – read all

Getting Data into your Repository – skip (4120 students won't use `svn import` since your repository has already been created).

Initial Checkout - read all

Basic Work Cycle – read all

Examining History – read all

Sometimes You Just Need to Cleanup – read all

Summary – read all

Chapter 9: Subversion Complete Reference – skim this section; this section simply

lists all the `svn` commands and options

Below the `svn` commands are categorized by the expected frequency of use in this course:

- Frequent: commit, add, update, help
- Occasional: checkout, list, status, delete, log, cleanup, info
- Rare: move, copy, diff, revert, mkdir
- More Rare: import, export, switch, lock, unlock, blame, propXXX, resolved, merge

The command-line `svn` built-in help is accessed via `svn help`. TortoiseSVN has a help file found by going from the Start Menu to All Programs:TortoiseSVN. TortoiseSVN works by integrating the SVN commands into Windows File Explorer pop-up menus. When you right-click in the Explorer window, the pop-menu will list SVN commands. (Other operating system versions are available).

While reading the manual, experiment with `svn` commands using your class repository subdirectory. I suggest you create a new repository subdirectory called `test` and experiment with the `svn commit`, `svn checkout`, etc. commands into this directory.

Only you and professor have access to your repository subdirectory. The location of your repository subdirectory is described below in Section 5.2.

5.2. *Student Repository Location*

Each student (and/or team) in class has already had a repository directory created for them. The URL of a student's SVN repository directory is as follows.

Assume your UNCC email address is:

johndoe@uncc.edu

and your first and last name as it appears in the UNCC BANNER system is:

John Doe

Then your SVN username is `johndoe` and your course SVN repository URL is:

`https://cci-subv.uncc.edu/svn/zwartell-public/ITCS XXXX/trunk/students/John Doe`

Throughout this document, the text “ITCS XXXX” would be replaced by appropriate string for the class you are taking. This will be given to you in class. Instructions for getting your SVN password will also be given in class. Note, if you already have a SVN account from a prior course on the server <https://cci-subv.uncc.edu svn> and your account name is the same as your UNCC email address, then you should be able to use this same account name and password for this course.

Note, if your UNCC BANNER name contains an initial, the initial is removed when creating the directory name of your course SVN repository URL. If your UNCC BANNER first or last name contains multiple words, the spaces are removed from between the multiple words in your first or last name when creating the directory name. For example, if your UNCC BANNER name is:

Smith, Jim B.

Then your SVN repository URL is:

`.../students/Jim Smith`

If your UNCC BANNER name is:

Van Winkle, Jim Bob

Then your SVN repository URL is:

`.../students/JimBob VanWinkle`

For team projects, all members of the team will be given access to a team directory with a name like:

`https://cci-subv.uncc.edu/svn/zwartell-public/ITCS XXXX/trunk/students/TeamName`

As an example, assume you are working on a UNCC Novell machine and are storing your individual work on your H: drive. To retrieve a working copy of your repository directory, type the following at the cmd.exe shell prompt:

```
> svn checkout "https://cci-subv.uncc.edu/svn/zwartell-public/ITCS  
xxxx/trunk/students/John Doe" --username johndoe "H:\My Documents\ITCS xxxx"
```

You will be prompted for your password and SVN will checkout a working copy of your repository directory into the directory H : \My Documents\ITCS XXXX.

Alternatively, you can use the *SVN Checkout* pop-up menu item provided by TortoiseSVN within Windows File Explorer. Both the command-line and GUI SVN clients allow you save your password (called your ‘authorization’ information) so that you need not repeatedly enter it.

Note, you must specify the full URL path name to access your individual or team repository directory. For example, you will not be able to access the upper-level directory <https://cci-subv.uncc.edu/svn/zwartell-public/ITCS xxxx/trunk/students>.

5.2.1. Verifying your Working Copy

Exploring your Working Copy

The above command will create a working directory tree similar to the following depending on the course you are taking:

- H:\My Documents\ITCS XXXX\Examples
- H:\My Documents\ITCS XXXX\Project 1
- H:\My Documents\ITCS XXXX\Project 2
-
- H:\My Documents\ITCS XXXX\Third Party Libraries

SVN refers to any such local copy as a *working copy*—which is distinguished from the copy stored on SVN server’s repository known as the *repository copy*. Verify that your working copy looks like the above. Also use `svn info` to verify the following.

The working copy directories:

H :\My Documents\ITCS XXXX\Project X

are associated with repository directories:

<https://cci-subv.uncc.edu/svn/zwartell-public/ITCS XXXX/trunk/students/John Doe/Project X>

The working copy directory

H :\My Documents\ITCS XXXX\Examples

is associated with a repository directory similar to

`https://cci-subv.uncc.edu/svn/zwartell-public/ITCS
XXXX/trunk/public/examples`

Further, the directories

`H:\My Documents\ITCS XXXX\Third Party Libraries\X`

are associated with repository directories such as

`https://cci-subv.uncc.edu/svn/zwartell-
public/vendor/Y/X/local/precompiled`

Note, the Examples and Third Party directories are associated with a common repository directory shared by all students in this course. (In contrast, your other working copy directories are mapped to your private repository directory). For the shared directories, all students only have read access to these directories.

Hidden directory .svn

The top level directory of a working copy contains a hidden subdirectory used internally by the SVN client to tell SVN what repository directory that working copy directory comes from. The hidden directory is named `.svn` or `_svn`, depending on the operating system at SVN client. Use the cmd shell or Windows File Explorer to briefly examine the contents of this hidden directory.

Generally, you can ignore the detailed contents of `.svn`. However, it is important to understand that this hidden directory is required for the SVN client to work. For example:

- If you accidentally delete `.svn`, all association between the local working copy directory and the repository directory is lost!
- If you create a new local directory with new files in your working copy, you must first `svn add + commit` the new local directory first before you can `svn add` the new files. (This updates the `.svn` in the working copy top directory).
- If you copy/move a working copy sub-directory using OS file operations to another working copy sub-directory you still have to explicitly inform SVN of the changes using SVN commands ([see Tip 6]).
- Unfortunately unlike all the 100's of other operating systems, traditionally Microsoft operating systems disallowed prefixing a file name with a `.` symbol. Although, new MS OS's allow `.` prefix, this historic limitation caused many Windows tools to use `_` prefix instead. With command-line and GUI SVN clients different versions and distributions may use either `.svn` or `_svn`. If you happen to use versions of SVN that use the opposite prefix, the two SVN clients may not recognize a working copy created by the other client. Note, some SVN clients will allow you to change which convention the client will use; see your SVN client manual for details.

Exploring your repository directory

You can directly explore your repository directory (i.e. on the server as opposed to your local working copy) in several ways.

Once you have a working copy, from the shell:

```
> cd "H:\My Documents\ITCS xxxx"  
> svn list -R
```

or from the shell without a working copy:

```
> svn list -R "https://cci-subv.uncc.edu/svn/zwartell-public/ITCS  
xxxx/trunk/students/John Doe"
```

For the TortoiseSVN pop-up menu, select *Repo-browser* menu item which will pop-up the *Repository Browser* dialog box. Note, if you svn commit changes to the repository while keeping the *Repository Browser* open, you may need to manually select the *Refresh* pop-up menu item to refresh the dialog box.

5.3. Project SVN Policies

5.3.1. Submitting Projects

Submit all your assignments by using svn add and svn commit to upload your code to your SVN repository subdirectory.

Your projects will be graded based on the version of your code that is svn commit'ed to your SVN repository subdirectory at the time of the due date for the project.

Follow the following policies which are elaborated in the subsections below:

- **5.3.2 Keep the repository clean!!!**
- **5.3.3 Double Check the Contents of Your Repository Directory**

5.3.2. Keep the repository clean!!!

It is standard practice when using version control software to not commit intermediate or output files generated by the compiler into the repository. Intermediate or output compilation files are all regenerated when someone else checkout's the source code and recompiles it. Like all compilers MSVS 20xx generates lots of these (*.ncb, *.ilk, *.exe,

*.obj, etc.).

Do not commit these auto-generated files to your SVN repository!

Only add the following files to the repository:

- source code files (.cpp, .h, etc.)
- compilation scripts or project files
 - Under MSVS 2010:
.vcxproj, .sln, .vcxproj.user
 - Under earlier MSVS versions:
.vcproj, .sln, .vcproj.<username>.<machinename>.user
- a .txt file or .doc file describing what parts of the project you completed or left incomplete, etc.
- any input files such as image files or other data files required by your program
- any subdirectories containing the above files

When adding subdirectories, be careful not to blindly submit their entire contents because often they contain additional automatically generated files (See Tip 4).

Reasoning:

Adding intermediate auto-generated files to the repository is wasteful, messy and can create subtle problems. Many intermediate files will create compilation problems if they are copied between different computers. Putting them in the repository is equivalent to such copying.

5.3.3. Double Check the Contents of Your Repository Directory

After committing changes to the repository, double check that the repository successfully received the changes by examining the repository directory as described in Section 5.2.1 Exploring your repository directory.

As a final check, do an SVN checkout into a temporary directory and make sure you code compiles and runs using the checked out directory contents

5.4. Tips

1. Don't forget to svn commit – svn adds, svn deletes, svn renames, etc. must be followed by svn commit to collect all the changes you made into a single svn transaction and send them to the repository with a single log message.
2. Don't forget -m with svn commit – you should always use the -m option to give a log message when you svn commit

3. Confusing svn add versus commit – if you alter a file in your working copy and that file already exists in the repository, you only need to commit the file. `svn add` is only used when you have created a new file or directory in your working copy which does not exist in the repository.
4. Avoiding adding auto-generated files –
 - a. When you `svn add` a new directory:
 - i. `svn add -N` the directory
 - ii. `svn commit -N` the directory
 - iii. `svn add` and commit the only appropriate files inside that directory.
 - b. If you accidentally add and commit junk files use `svn delete` or the *TortoiseSVN→Repo-browser* from the TortoiseSVN pop-up menu to delete the files.
5. TortoiseSVN: login – check the “Save Authentication” checkbox to avoid having to retype your password
6. Moving/Copying working copy directory to other sub-directories in your working copy (the effect of the .svn directory) –

When using OS file copy operations to copy sub-directory in your working copy from one location in your working copy to another location in your working copy, you still have to inform SVN of the directory changes using `svn add` and `commit`.

Alternatively, you can copy files using `svn copy` which does a copy in the repository (server side) and then do an `svn update` to update your working copy. Also, in the TortoiseSVN GUI this can be done using the Repo Browser found in the right-click menu. (But you'd still follow up this server side operation by doing an `svn update` on your working copy).

If you need to copy subdirectories in a working copy and you insist on using OS file copy operations, then you should follow the following guidelines. As an example, assume the following:

- a. You checked out your ITCS XXXX repository directory to a working copy directory 'H:\ITCS XXXX'.
- b. 'H:\ITCS XXXX\Examples\List' is associated with repository directory "<https://cci-subv.uncc.edu/svn/zwartell-public/ITCS XXXX/trunk/examples>List>"
- c. 'H:\ITCS XXXX\Project 2' is associated with repository directory "<https://cci-subv.uncc.edu/svn/zwartell-public/ITCS XXXX/trunk/students/John Doe>"

If you OS file copy

"H:\ ITCS XXXX \Examples\List"

in its entirety onto

"H:\ITCS XXXX \Project 2\List"

"H:\ITCS XXXX \Project 2\List\" must be svn add and commit'd to the Project 2 repository.

Extra Unusual Circumstance:

Note, if you do an OS file copy of a top level directory of working copy, A, (which contains its own hidden .svn sub-directory), into a different working copy B, SVN will get confused because the copy of A, A', will have its own .svn directory linking it to the original repository of A but these files are now sitting underneath working copy B that is linked to B's repository. In this unusual circumstance, you generally want to delete the hidden .svn directory from A', the copy of A, underneath the working copy B and then svn add and commit those subdirectories to B's repository.

5.5. Common Error Messages

Command-line SVN:

1. svn: Authentication Failed

If svn gives this message without even prompting you for a password: Fix: add the --password option with a random password. This usually forces svn to start prompting you for your password again. If svn is prompting you for a password, try to login once more (double checking the procedure in Section 5.2) and then email the TA.

2. svn: '.' is not a working copy

In a working copy directory svn creates a hidden directory (called _svn or .svn) that contains info on where the working copy's associated repository is located. If the directory you are in does not contain this .svn sub-directory, svn doesn't know what repository to connect to. This error may occur:

- a. if you are trying to svn add a new directory but its parent directory is not already in the repository. Fix: 'svn add -N' and 'svn commit -N' the parent directory first. You may have to do this recursively starting from the parent's parent as well if the parent's parent isn't in the repository.
- b. if you accidentally delete the .svn directory from a working copy subdirectory. Fix: See 7.b
- c. if you do a lot of moving directories in your working copy or copying directories between different computers. Fix: See 7.b
- d. if you mix versions of SVN clients that use a different hidden file name prefix. Unfortunately unlike all the 100's of other operating systems, traditionally Microsoft operating systems disallowed prefixing a file name with a .symbol . Although, new MS OS's allow . prefix, this historic limitation caused many Windows tools to use _ prefix instead. With command-line and GUI SVN clients different versions and distributions

may use either .svn or _svn. If you happen to use versions of SVN that use the opposite prefix, the two SVN clients may not recognize a working copy created by the other client. Note, some SVN clients will allow you to change which convention the client will use; see your SVN client manual for details.

3. svn: Please run svn cleanup

Fix: use the svn cleanup command from the directory generating the error.

4. svn: Commit failed (details follow):

svn: system('... svn-commit.tmp') returned 1

Fix: When committing svn requires a log message. Use the -m option.

5. svn: Commit failed (details follow):

svn: '<directory path>' is not under version control and is not part of the commit, yet its child '...' is part of the commit

Fix: 'svn add -N' and 'svn commit -N' the directory <directory path> before trying to commit files inside that directory. See also 2.

6. svn: Commit failed (details follow):

svn: Can't open file '/coit/T/COIT/MYDEPT/repos/zwartell-public/db/transactions/7140-24.txn/node.nfd.0': Permission denied –

This may indicate the SVN server is down. Fix: Post to the class discussion board /email the TA.

7. Other errors:

a. Fix: Post to the class discussion board /email the TA.

b. Fix: Recheckout

i. rename your "ITCS 4120" directory to "ITCS 4120.bak"

ii. redo svn checkout into "ITCS 4120"

iii. copy files your trying to add from "ITCS 4120.bak" to the corresponding location in "ITCS 4120"

TortoiseSVN:

1. The pop-up menu only lists the checkout command

TortoiseSVN pop-up menu only lists the full set of commands if the current directory is a valid working copy. See Command-line SVN: 2.

5.6. Common Errors

1. HTML escape character (e.g. %20) is not the cmd.exe escape character

Be careful with escape sequences in strings representing file names and paths. For example the HTML standard for URLs uses the % symbol as an escape character, so

`http:\\www.no%20spaces%20here.com` is translated as

`http:\\www.no spaces here.com`

However, general file systems (Windows/Mac/Unix) and command-line shells for these OS do not use % as an escape sequence when creating files names or directory names. So if you did the following at the Windows cmd.exe prompt:

```
c:\ mkdir "no%20space%20here"
```

you will actually get a directory with a nutty looking name including the %'s, and the file system commands,command-line tools, and MSVS compiler components will probably fail in some fashion when processing that directory name.

6. Citations

- [1] Zachary Wartell. ITCS 4120 Website.
<http://www.cs.uncc.edu/~zwartell/ITCS%204120/current/ITCS%204120%20-%20Wartell.html> .
- [2] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. Version Control with Subversion: For Subversion 1.5. [<http://svnbook.red-bean.com/>]
- [3] Cay S. Horstmann, Windows Shell Tutorial.
<http://www.horstmann.com/bigj/help/windows/tutorial.html> circa 10/7/2011