Version Control with Subversion

Matthew Plough
mplough@princeton.edu

November 29, 2006

This document originally was written by Matthew Plough for the COS 333 course. It was edited substantially by Robert Dondero on November 2, 2011 to make it specific to Assignment 4 of the COS 217 course.

1. Introduction

Subversion is a version control system. Well written, extensive documentation can be found in the Subversion book at http://svnbook.red-bean.com/. This document presents only a brief overview of the concept of version control and how to use Subversion.

2. Version Control

The COS 217 course requires you to use a version control system for Assignment 4. It does so for a number of reasons. Without one, you'll have problems sharing code; you would need to find a place that all of your team members can access when you aren't all present, and it will be frustrating for all of you to work on the code at the same time.

In the past, you probably deleted what they need to know critical files inadvertently, and frantically sought to retrieve them. While trying to find a bug, you have probably added so many changes to a file that you can no longer remember what the code originally looked like, so you may have to rewrite the code completely.

Version control systems address those problems, allowing you to write code more efficiently.

Chapter 2 of the Subversion Book gives an excellent overview of how version control works. If you have not used a version control system before, it's essential reading; and it's not too long.

2.1. The Repository

All versions of your work are stored in a central location called a *repository*; any of them can be retrieved at any time.

The repository acts as a "vault" for a project; it changes how you work with your code. Instead of working on the only copy of your project, you check out a copy of the project, make changes, and commit the changes back to the repository. This gives a lot of flexibility; if you make changes that prove useless, you can quickly revert back to a known-good version of your work. Similarly, if you accidentally delete a file, you restore the last version from the repository.

Repositories are generally quite small; only the minimum amount of information needed to construct a given version of a file is stored. If a file does not change between versions, a new copy is not written.

Princeton's Office of Information Technology (OIT), at my request, has created an empty repository for your Assignment 4 team.

2.2. The Working Copy

Any member of your team can check out files from the repository; this local copy of the code is called the *working copy*. Each team member can make changes to his or her own working copy, and once satisfied with the changes, commit them back into the repository for others to access.

3. Working with Subversion

This section contains enough information to get you up and running on hats.

3.1. Getting Ready

Edit the .bashrc file in your home directory, adding this line at the end:

```
export EDITOR="emacs -nw"
```

That line tells Subversion to launch emacs when you upload files to your team's repository. More on that below. If you use the "X Window Server" environment (as described near the beginning of the course), then add this line instead:

```
export EDITOR="emacs"
```

Recall that Bash executes the `.bash_profile` file, which in turn causes Bash to execute the `.bashrc` file, when you log into hats. So you should log out of hats and then log back in for the change to take effect.

3.2. Creating a Working Copy of your Team's Repository

Issue these commands on hats:

`cd` *`cos217`*

> Change directories to the one where you store your files for the COS 217 course. I'll assume it's named *cos217*.

`svn checkout https://svn.princeton.edu/cos217fall11/`*`teamX`*

> where *teamX* is one of these as appropriate: team11, team12, team21, team22, team31, team32, team41, team42, team51, team52, team5a1, team5a2, team6, or team7.

Subversion creates a directory named *teamX* subordinate to your cos217 directory. The *teamX* directory contains a directory named .svn. The .svn directory contains Subversion-internal information. Don't delete it or change it. The *teamX* directory also contains all files that are in your team's repository.

IMPORTANT: If your team's repository already contains .c files (some of which you should not download), then you should issue this command instead of the previous one:

```
svn checkout https://svn.princeton.edu/cos217fall11/teamX --depth=empty
```

The `--depth=empty` option commands Subversion to create a working copy, but to download no files to it.

3.3. Seeing the Names of All Files in your Team's Repository

Issue these command on hats:

```
cd cos217

cd teamX

svn list -r HEAD
```

Subversion writes to stdout the names of all files that reside in your team's repository.

3.4. Downloading a File from your Team's Repository

Issue these commands on hats:

```
cd cos217

cd teamX

svn update somefile.c
```

Subversion copies *somefile.c* from your team's repository to your working copy iff that file has been changed since your last update of that file, that is, iff *somefile.c* doesn't exist or is out-of-date in your working copy.

IMPORTANT: Do not issue this command:

```
svn update
```

That command tells Subversion to copy *all* files from your team's repository to your working copy. But recall that you're not allowed to copy some files from your team's repository to your working copy. Please see the Assignment 4 specification for details.

3.5. Uploading a New File to your Team's Repository

Issue these commands on hats:

```
cd cos217
```

```
cd teamX
```

```
emacs somefile.c
```

Use emacs (or any editor) to create *somefile.c*.

```
svn add somefile.c
```

Subversion adds *somefile.c* to the list of files in your working copy that it is managing.

```
svn commit
```

Subversion launches emacs to create a temporary file, thereby allowing you to enter a message indicating the nature of the change. At this point "Initial checkin." would be an appropriate message. Then Subversion copies all files that have been added or changed since your last commit to your team's repository.

3.6. Uploading a Changed File to your Team's Repository

```
cd cos217
```

```
cd teamX
```

```
svn update somefile.c
```

Subversion copies *somefile.c* from your team's repository to your working copy iff that file has been changed since your last update of that file, that is, iff *somefile.c* doesn't exist or is out-of-date in your working copy. Thereby you've made sure that you will edit the most recent version of *somefile.c*.

```
emacs somefile.c
```

Use emacs (or any editor) to change *somefile.c*.

```
svn commit
```

Subversion launches emacs to create a temporary file, thereby allowing you to enter a message indicating the nature of the change. At this point you should type some descriptive message. Then Subversion copies all files that have been added or changed since your last commit to your team's repository. In particular, it copies *somefile.c* to your team's repository.