Duane's Incredibly Brief Intro to SVN

a working document for working students

Duane Bailey
(http://www.cs.williams.edu/~bailey/svn.pdf)

CONTENTS

I. Creating a Code Repository
II. Checking Out a Version of Your Project
III. Working on Files
IV. Resolving Conflicts with Your Partner(s) (Using SVN)
V. Packing Things Up

SVN (short for 'subversion') is a relatively simple system for keeping track of the various versions of the files that make up a project. The process of tracking the revisions of a project involves the following general steps:

1. Creating a repository for your project.
2. Importing the initial version of a project into your repository.
3. Cycling through the following life steps:
   a. Checking out a version of the project.
   b. Making changes to one or more files in the project
   c. Committing your changes back to the repository.

This sheet suggests one technique for setting up and maintaining a repository. It is suitable for keeping track of a small project, possibly between a couple of programmers. For more advanced approaches, you should refer to the reference manual at

http://svnbook.red-bean.com

(This book is also available for purchase from O'Reilly.)

N.B. Using SVN can simplify the process of working on a shared project, but it cannot resolve difficulties that stem of communication problems among team members. The mantra: communicate.

I. CREATING A CODE REPOSITORY

These steps are taken by the project owner, or by one member of a team designated as the code repository owner.

1. A repository is a small database that keeps track of all the changes you make to your various projects. We suggest you create the repository in a subdirectory of your home directory that is accessible to all members of your programming team. Commonly, this directory is called "svn". Create this with

```
mkdir ~/svn
```

2. If you are not working with others, please skip to step 3.

If you are working with a group of people on a project, you should make sure that the SVN directory is accessible to other people in your assigned group. (We will have provided you a group id, or you may ask Mary for one. To find out what groups you are a member of, type "groups" at the shell prompt.) In the following commands, we assume your group is "ourgroup":

```
chgrp ourgroup ~/svn
chmod go+x ~
chmod g+rwx ~/svn
```

The chmod instructions make sure that other members of the world can access files in your home directory (that you previously agree on), and that members of "ourgroup" can access and modify the SVN directory.

3. Log into the machine that serves your home directory. This is "eringer"

in our environment:

```
ssh eringer
```

(You can determine the owner of your home directory by typing

```
df ~/svn
```

which reports the amount of space free on your disk, and the machine the disk is mounted from. If no machine is volunteered, the disk is served by the local machine. It is vital that the repository be created from the machine that hosts the SVN directory)

4. Create the repository. Typically, repositories are named after the course you're working on. Here, we'll assume that's "cs237" and that your username is 10xyz

```
svnadmin create /home/cs-students/10xyz/svn/cs237
```

If this works, there will be no response. You may now log out of eringer.

5. We need to import your project into the repository. We'll assume that you've got the start of a project in a directory. Suppose that it is called ~/my237prj. You import it with the command:

```
svn import ~/my237prj svn+ssh://eringer/home/cs-students/10xyz/svn/cs237/prj
```

This will ask for an initial message for the project, probably using your favorite editor. This will ask for passwords much like ssh. Provide them. It should return with a response like

```
Committed revision 1.
```

It may also have indicated that any files already created in the project directory were added.

Note that the repository is referred to like a URL. The "svn+ssh" indicates that you want to use ssh to transmit your changes. The "//eringer/" indicates that the repository is hosted by the machine eringer. If you're off campus you may find that eringer.cs or eringer.cs.williams.edu is necessary to make contact.

Once the initial directory has been inserted into the repository, it may be removed without fear (though many wait until they've successfully checked out a version from the repository):

```
rm -rf ~/my237prj
```

II. CHECKING OUT A VERSION OF YOUR PROJECT

Once the project has been added to the repository, you (and your partners) are free to check it out on as many computers as you desire. To check out the project we created above you use the command:

```
svn checkout svn+ssh://eringer/home/cs-students/10xyz/svn/cs237/prj our237prj
```

and the latest version of the project is dropped into a subdirectory called "our237prj". The remainder of this document assumes that you are sitting in this project directory when you execute SVN commands. (If you aren't, SVN will politely point out that your current directory is not under source control.)

Typically, you only need to check out the project once, but if you get rid of the checked out version, you will need to check it out again.

III. WORKING ON FILES

1. If you work with others, you need to tell your partners that you're editing a file BEFORE you make your edits. If more that one person is editing a version of the file, you will run into conflicts when you check your files back in.

2. NEW FILES. (Skip to step 3 for "pre-existing" files.) If you create a new file that has not been part of a project, can add it to the repository with

```
svn add hello.c
```

This places the file in the repository, under source control. You now typically progress with the next step.

3. Let's assume that you want to edit the file "hello.c", which is maintained in the repository. You first need to make sure that you have the most recent version. Type

```
svn update hello.c
```

(As with all interactions with the repository, you should provide passwords as necessary.) The latest version of hello.c is checked out, if necessary, and you are free to edit it.

4. Once you have have made your changes to a file, you can commit to those changes, producing a new repository version.

```
svn commit hello.c
```

You will be asked for a quick message that describes those changes, probably in an editor, like emacs. This message is used in log file reports and helps you remember the sequence of edits. If you would like to specify that message on the command line, you may, with the -m option:

```
svn commit -m 'Added return type to main method.' hello.c
```

The quote marks are important protective delimiters.

The commit should result in output similar to the following:

```
Sending        hello.c
Transmitting file data ...
Committed revision 37.
```

IV. RESOLVING CONFLICTS WITH YOUR PARTNER(S) (USING SVN)

If you work with others, you will sometimes find that you have not worked together and that will be editing a file in parallel with another person. In the SVN system, several people may check out a file at a time and there is no obligation to commit to any changes.

If several people *edit* the same files at the same time and then try to commit, the first commit goes through and successive commits by other programmers do not--they are rejected. Often parallel edits to a single file are not actually conflicting logic, but SVN has no means of figuring this out. You must manually resolve the issue. Here is one way to do that.

First, move your version of the file to one side:

```
    cp hello.c hello.c.mine
```

You may now revert to your original version of the file

```
    svn revert hello.c
```

and then update that version to the latest release (the version your partner committed)

```
    svn update hello.c
```

At this point your version is in hello.c.mine, and your partner's version is in hello.c. You may compare the two versions with the Unix diff command:

```
    diff hello.c hello.c.mine
```

With these differences in hand it is frequently a simple task to incorporate your changes in your partner's version. Once the changes have been incorporated you should commit your version:

```
    svn commit hello.c
```

If your changes are complex, and you're determined, you may find the "patch" utility useful. It uses files generated by the diff command to generate a file of patches.

KEEPING TRACK OF WHERE YOU'RE AT (AND BACKING OUT!)

* You can find out how your file differs from the repository version with

```
   svn diff hello.c
```

The current version is compared with the repository version and the results are presented in a form similar to that presented by the diff command.

* If, at any point, you want to see the history of revisions for a file, you can get the log of changes with

```
   svn log hello.c
```

* If you are unsatisfied with the changes you made to your file, you may revert back to the version from the repository with

```
   svn revert hello.c
```

V. PACKING THINGS UP

When you're finished working on a project, you may decide to remove the working directory. Before you do that, you should make sure that everything in the directory may be recovered from the repository.

1. From within the working directory, you can type

```
   svn status
```

This provides a listing of files that are not found, in their present form, in the repository. A typical listing is:

```
   ?      web/index.html
   M      Makefile
```

This means that index.html, a file in the web subdirectory is not part of the database. You should consider adding the file to the database with

```
svn add web/index.html
```

(Emacs work files, binaries, and other files that can be generated from sources are typically not added to the repository. SVN can be configured to ignore these files in the status command; you can read about that in the SVN book.) Makefile, on the other hand, is part of the repository and has been modified, and the modifications have not be committed to the database. You should

```
svn commit Makefile
```

These commands should bring everything up-to-date. Another status check will find the directory is consistent with the database. You may now carefully remove this directory and its files.