Version Control: Subversion

Colin Gordon
csgordon@cs.washington.edu

University of Washington

CSE333 Section 3, 4/14/11

Today's Topics

- Version Control: What and Why
- Intro to Subversion (a.k.a. `svn`)
- Good Version Control Practices

Ask questions any time!

Version Control

Version control is software that helps one or more people manage multiple versions of a set of files

- tracks snapshots of a whole set of files
- guides sharing and merging of changes for multiple people

What is Version Control Good For?

- Reverting to an earlier snapshot if you break your current version beyond repair
- Looking at old versions
  - See when a bug was introduced
  - See how to undo broken changes
- Track multiple directions of development together (e.g. stable and cutting-edge releases, a main branch and a separate branch for work on a new feature)

Version Control Terminology

- A saved copy of the files tracked by version control is called a *repository*.
- You can *check out* a copy of all the files in the repository.
- A checked out copy of files that you can change is called your *working copy*.
- When you make changes that you would like to have saved as a snapshot in the repository, you *check in* or *commit* your changes.
- Can *update* your working copy to get changes other people have committed to the repository since the last time you looked

Two Types of Version Control

Centralized

- One central repository
- When someone commits changes from a working copy, they go to the central repository *immediately*, visible to everyone upon sync

Distributed

- Many respositories; essentially each person gets their own local repository
- Commits save changes in the local repository.
- Can *push* and *pull* changes between different repositories; which one is the “official” repository is simply by convention, not enforced by the tool.
- Some terminology has slightly different meanings...

The History of Version Control Systems

- 1982: RCS
  - Only worked on individual files...
- 1990: CVS
  - Lots of strange, subtle, confusing behaviors... no renaming/moving...
- 2000: Subversion (SVN)
  - Fixed lots of CVS's problems, and added lots of new features
- 2005: *Distributed* version control becomes popular
  - Primarily because of Git and Mercurial

Subversion

In this class, we'll only talk about Subversion.

- Roughly speaking, the current standard
- Knowledge of using Subversion mostly carries over to other version control systems
- We won't require you to use it, but we cannot emphasize enough how strongly we recommend it.
- If you use version control, you don't have to use Subversion, but the chances of us helping you with it are lower if you use something else[1].
- Excellent online documentation: the "SVN Red Book"
  - A real, published book on using Subversion, free online
  - `http://svnbook.red-bean.com`

[1] Colin is also familiar with Mercurical

Creating a Subversion Repository

The main repository for Subversion is managed by a command called `svnadmin`.

```
[csgordon@monarch:~/cse333]$ svnadmin create svnrepo
[csgordon@monarch:~/cse333]$ ls
svnrepo
[csgordon@monarch:~/cse333]$
```

Unless something awful happens, you'll probably never need to do anything with `svnadmin` except create the repository. Don't ever directly change files inside the directory this creates; only the `svnadmin` command knows what they mean!

Checking Out a Repository (Locally)

To check out a working copy:

```
[csgordon@monarch:~/cse333]$ svn co \
    file:///homes/gws/csgordon/cse333/svnrepo/ local
Checked out revision 0.
[csgordon@monarch:~/cse333]$ cd local
[csgordon@monarch:~/cse333/local]$ ls
[csgordon@monarch:~/cse333/local]$
```

Your checkout actually contains a hidden directory called `.svn` that holds information about where the main repository is, and previous versions. Don't change any files in this directory!

Checking Out a Repository (Remotely)

You can do the same thing remotely, for example if you work at home but keep your main repository in your CSE home directory[2]:

```
[csgordon@home:~/cse333]$ svn co \
    svn+ssh://csgordon@ravenna.cs.washington.edu/homes/.../cse333/svnrepo/ \
    local
csgordon@ravenna.cs.washington.edu's password:
Checked out revision 0.
[csgordon@home:~/cse333]$ cd local
[csgordon@home:~/cse333/local]$ ls
[csgordon@home:~/cse333/local]$
```

[2] CSE home directories are backed up, so this is highly recommended!

Adding Files

You must explicitly tell Subversion which files to track.

```
[csgordon@monarch:~/cse333/local]$ echo 'hello!' > file.txt
[csgordon@monarch:~/cse333/local]$ cat file.txt
hello!
[csgordon@monarch:~/cse333/local]$ svn add file.txt
A         file.txt
[csgordon@monarch:~/cse333/local]$
```

At this point, your working copy tracks the file, but it is not yet in the main repository.

Checking In

To share your local changes, you must *commit* them:

```
[csgordon@monarch:~/cse333/local]$ svn commit -m "Added file.txt"
csgordon@ravenna.cs.washington.edu's password:
Adding         file.txt
Transmitting file data .
Committed revision 1.
[csgordon@monarch:~/cse333/local]$
```

You can omit the `-m` argument, the *commit message*, and some editor will pop up where you can type one.

Updating, Viewing the log

Maybe someone else has checked in a fix you need. Let's get it:

```
[csgordon@monarch:~/cse333/local]$ svn update
csgordon@ravenna.cs.washington.edu's password:
U    file.txt
Updated to revision 2.
[csgordon@monarch:~/cse333/local]$
```

What did they do? Let's find out:

```
[csgordon@monarch:~/cse333/local]$ svn log -r HEAD
csgordon@ravenna.cs.washington.edu's password:
------------------------------------------------------------------------
r2 | csgordon | 2011-04-13 13:41:08 -0700 (Wed, 13 Apr 2011) | 2 lines

modified file.txt

------------------------------------------------------------------------
[csgordon@monarch:~/cse333/local]$
```

Try `svn help log` to see how to view other parts of the log.

Checking Local Changes

Sometimes it's hard to remember what you've changed since last time you checked in:

```
[csgordon@monarch:~/cse333/local]$ vim file.txt
[csgordon@monarch:~/cse333/local]$ svn diff
Index: file.txt
===================================================================
--- file.txt (revision 2)
+++ file.txt (working copy)
@@ -1 +1,2 @@
 hello there!
+blah blah blah
[csgordon@monarch:~/cse333/local]$
```

Merge Conflicts

Sometimes you try to push, but someone else has changed your files!

```
[csgordon@monarch:~/cse333/local]$ svn commit
csgordon@ravenna.cs.washington.edu's password:
Sending        file.txt
Transmitting file data .svn: Commit failed (details follow):
svn: File '/file.txt' is out of date
svn: Your commit message was left in a temporary file:
svn:    '/homes/gws/csgordon/cse333/local/svn-commit.tmp'
[csgordon@monarch:~/cse333/local]$
```

Now we need to merge their changes with ours...

Merging

```
[csgordon@monarch:~/cse333/local]$ svn up
csgordon@ravenna.cs.washington.edu's password:
Conflict discovered in 'file.txt'.
Select: (p) postpone, (df) diff-full, (e) edit,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options:
```

Choosing edit will pop up an editor with a file like this:

```
hello there!
<<<<<<< .mine              (what follows are your changes)
blah blah blah             (your changes)
=======                    (divider)
ah-hah! I messed you up.   (the conflicting changes)
>>>>>>> .r3                (end diff, the conflicting revision)
```

Edit the file to a final version, save the file, and exit the editor.

Merging (Part 2)

Now subversion will ask if we resolved the file, want to try again, etc.

```
Select: (p) postpone, (df) diff-full, (e) edit, (r) resolved,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options: r
G    file.txt
Updated to revision 3.
[csgordon@monarch:~/cse333/local]$
```

If you later (e.g. after running tests) decide you goofed on the merge, you can fix it before checking in.

Reverting

Sometimes you'll mess up badly enough you want to go back to an old version:

```
[csgordon@monarch:~/cse333/local]$ svn revert file.txt
Reverted 'file.txt'
[csgordon@monarch:~/cse333/local]$
```

Now any local changes to file.txt are undone. So let's go back to nice, simple, revision 1:

```
[csgordon@monarch:~/cse333/local]$ svn up -r 1
csgordon@ravenna.cs.washington.edu's password:
U    file.txt
Updated to revision 1.
[csgordon@monarch:~/cse333/local]$ cat file.txt
hello!
[csgordon@monarch:~/cse333/local]$
```

Good Practices for Version Control

- Never check in broken code where someone else would get it
- Check in for one conceptual set of changes at a time
- Make sure your commit messages explain what was changed and why
  - This makes it much easier to find the revision you're looking for when you search the log!