Subversion

CSC207 Winter 2015

Computer Science
UNIVERSITY OF TORONTO

Version Control Systems

Version control is used to manage the changes to code or other documents over time.

In this course we will use Subversion (`svn`).

There are other version control systems: CVS, Perforce, Microsoft Visual SourceSafe, etc.

And newer, better, more complex ones, such as Git and Mercurial.

The first company you end up working will likely use a version control system.

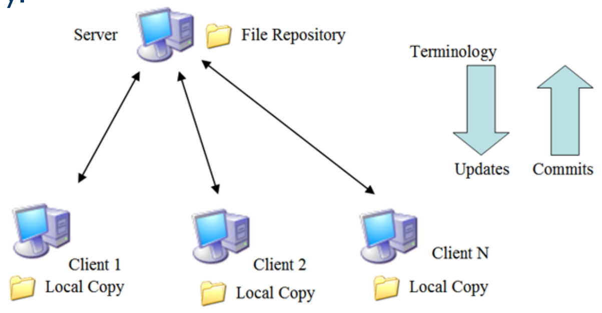
Other CSC courses (e.g., CSC209, CSC301, CSC369) use version control for assignment submission.

Recap: Centralized Version Control

Keep code in a central location, called a repository. This is the "master copy".

Create local copies of the repository in your account at school, on your machine at home, on your laptop...

When the local copy changes, "commit" the changes to the repository.



Version control — Managing Concurrency

What if two (or more) people want to edit the same file at the same time?

- Option 1: prevent it
  - Pessimistic concurrency
  - Only allow one writeable copy of the file
  - e.g., Microsoft Visual SourceSafe
- Option 2: patch up afterward
  - Optimistic concurrency
  - "Easier to get forgiveness than permission"
  - e.g., Subversion, CVS, Perforce

Unix and SVN Demo

Version control — Storage Scheme

Storing entire copy of a file on every change would require an enormous amount of disk space and would make synchronization slow.

Version control systems store incremental differences.

The incremental differences allow the system to reconstruct previous versions.

Where is my repo?

You need to know the URL so you can checkout your repository:

```
svn checkout URL
```

For most students, their repository name will be the same as their CDF username, so the URL will be:

https://markus.cdf.toronto.edu/svn/csc207-2015-01/your_cdf_username

During the lab on Monday, you will complete an exercise using your Subversion repository.

Exercise

Consider the following sequence of events:

- Alice and Bob share files `A.txt` and `B.txt` in the directory `D` using `svn`.
- Suppose Alice creates a file `C.txt` and modifies the file `A.txt`.
- Also suppose Bob modifies the file `B.txt`.

1. Provide a sequence of `svn` commands that will ensure that both Alice and Bob have the most recent versions of the files.

2. What if Bob also modified file `A.txt`? Provide a sequence of `svn` commands that will ensure that both Alice and Bob have the most recent versions of the files.

Subversion — some common commands

- **svn checkout [url]** Get initial copy
  - Do this anywhere to get a local copy of a repository.
- **svn add [filenames]** Add new files
  - Notifies Subversion that you want it to track the new files
  - add will almost always be followed by a commit because add doesn't actually modify the repository.
- **svn status [filenames]** See what's changed

Subversion — some common commands

- **svn update [filenames]** Synchronize with repository
  - Copies from the master repository to your local copy
  - Any commits made by another person or commits done by you from another local copy will be updated in your local copy.
  - Does not change the repository
  - Watch the messages closely.
- **svn commit [filenames]** Commit local changes
  - Copy changes to the repository
  - Will only be allowed if the local copy is up to date

Subversion — some common commands

- **svn remove [filenames]** Remove files from repository
  - Must first remove the file from the local copy
  - It is not easy to remove directories
  - Need to commit, just like for add
- **svn diff [filenames]** Show diffs between local & repo
  - Handy to see what changed between versions
- **svn log [filenames]** Show history of files
  - Shows log message, timestamps and who made the revisions