CSE 303
Lecture 19

Version control and Subversion (svn)

slides created by Marty Stepp
http://www.cs.washington.edu/303/

Working in teams

- Whose computer stores the "official" copy of the project?
  - Can we store the project files in a neutral "official" location?
- Will we be able to read/write each other's changes?
  - Do we have the right file permissions?
- What happens if we both try to edit the same file?
- What happens if we make a mistake and corrupt an important file?
  - Is there a way to keep backups of our project files?
- How do I know what code each teammate is working on?

Recall: Groups and users

command	description
`chmod`	change permissions for a file
`umask`	set default permissions for new files
`groups`	list the groups to which a user belongs
`chgrp`	change the group associated with a file

- setting groups on files: `chgrp` ***`group filename`***

  `chgrp -R cse303k *` (set group of all to `cse303k`)

- permission codes: `chmod` ***`who`*`(+-)`*`what filename`**

  `chmod -R g+rwX *` (group can read/write all)

Version control

- **version control system**: Software that tracks and manages changes to a set of source code files and resources.
  - examples: CVS, **Subversion (SVN)**, Git, Monotone, BitKeeper, Perforce

- helps teams to work together on code projects
  - a shared copy of all code files that all users can access
  - keeps current versions of all files, and backups of past versions
  - can see what files others have modified and view the changes
  - manages conflicts when multiple users modify the same file

  - not particular to source code; can be used for papers, photos, etc.
    - but often works best with plain text/code files

Repositories

- **repository**: Central location storing a copy of all files.
  - **check in**: adding a new file to the repository
  - **check out**: downloading a file from the repo to edit it
    - you don't edit files directly in the repo; you edit a local **working copy**
    - once finished, the user checks in a new version of the file
  - **commit**: checking in a new version of a file(s) that were checked out
  - **revert**: undoing any changes to a file(s) that were checked out
  - **update**: downloading the latest versions of all files that have been recently committed by other users

Subversion

command	description
`svnadmin`	make administrative changes to an SVN repository
`svn`	interact with an SVN repository

- **Subversion**: created to repair problems with older CVS system
  - supports folders, better renaming, atomic commits, good branching
  - currently the most popular free open-source version control system

- installing in Ubuntu:

```
$ sudo apt-get install subversion
```

- creating a repository:

```
$ svnadmin create path
```

SVN commands

command	description
`svn add files`	schedule files to be added at next commit
`svn ci [files]`	commit / check in changed files
`svn co files`	check out
`svn help [command]`	get help info about a particular command
`svn import directory`	adds a directory into repo as a project
`svn merge source path`	merge changes
`svn revert files`	restore local copy to repo's version
`svn resolve source path`	resolve merging conflicts
`svn update [files]`	update local copy to latest version
others: `blame`, `changelist`, `cleanup`, `diff`, `export`, `ls/mv/rm/mkdir`, `lock/unlock`, `log`, `propset`	

Setting up a repo

- on `attu`, create the overall repository:
  - `$ svnadmin create path`
- from `attu`, add initial files into the repo (optional):
  - `$ svn import projectname foldername`
- give the repo read/write permissions to your `cse303` group
  - `$ chgrp -R mycse303group repofoldername`
  - `$ chmod -R g+rwX,o-rwx repofoldername`

Adding files to a repo

- on your computer, set up a local copy of the repo
  - **`$ svn co svn+ssh://attu.cs.washington.edu/`*`foldername`***
  - or, if you're setting up your local copy on `attu` as well:
    `$ svn co file:///homes/iws/`*`username/foldername`*
  - after checkout, your local copy "remembers" where the repo is

- now copy your own files into the repo's folder and add them:
  - **`$ svn add `*`filename`***
  - *common error*: people forget to add files (won't compile for others)

- added files are not really sent to server until commit
  - **`$ svn ci `*`filename`*` -m "`*`checkin message`*`"`**
  - put source code and resources into repo (no `.o` files, executables)

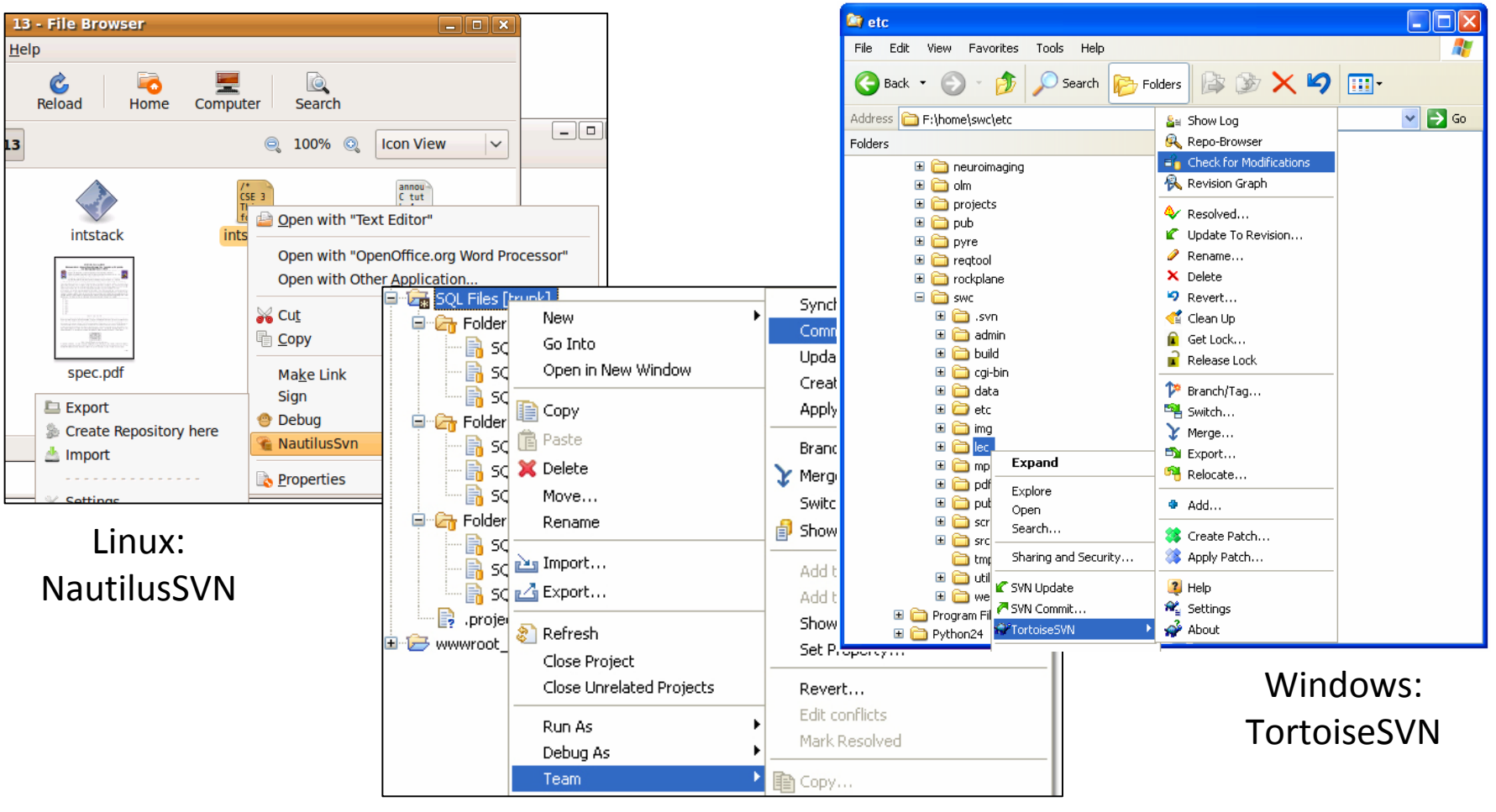
Committing changes

- updating (to retrieve any changes others have made):
  - `$ svn update`

- examining your changes before commit:
  - `$ svn status`
  - `$ svn diff` *`filename`*
  - `$ svn revert` *`filename`*

- committing your changes to the server:
  - `$ svn ci -m "added O(1) sorting feature"`

Shell/IDE integration



Linux:
NautilusSVN

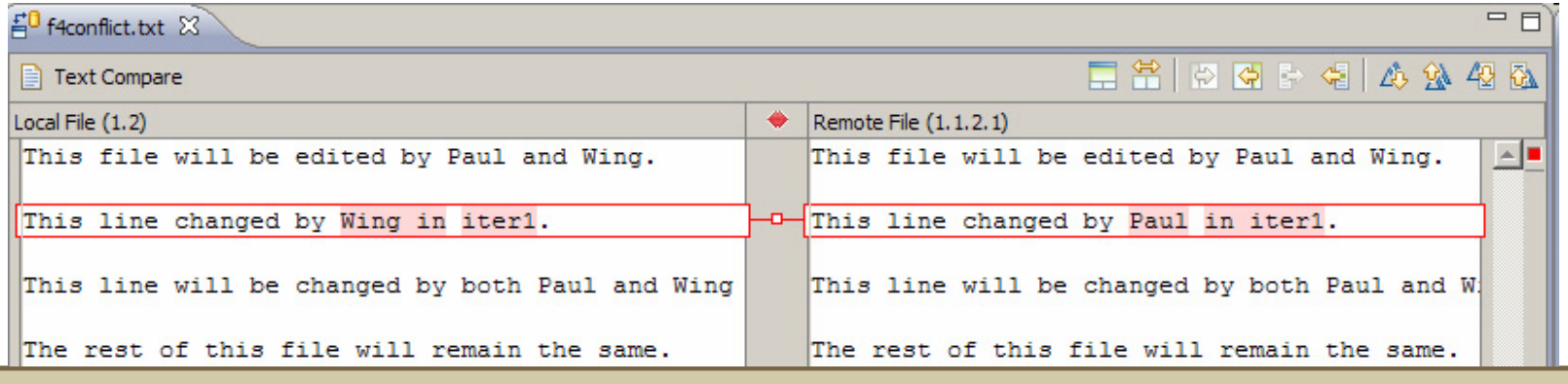
Windows:
TortoiseSVN

Eclipse:
Subclipse

Merging and conflicts

- **merge**: Two sets of changes applied at same time to same files
  - happens when two users check out same file(s), both change it, and:
    - both commit, or
    - one changes it and commits; the other changes it and does an *update*

- **conflict**: when the system is unable to reconcile merged changes
  - **resolve**: user intervention to repair a conflict. Possible ways:
    - combining the changes manually in some way
    - selecting one change in favor of the other
    - reverting both changes (less likely)



Branches

- **branch** (fork): A second copy of the files in a repository
  - the two copies may be developed in different ways independently
  - given its own version number in the version control system
  - eventually be merged
  - **trunk** (mainline, baseline): the main code copy, not part of any fork