-Today’s lecture:

- Subversion
- Assignment-2 questions

Subversion

Revision keywords:

- Revision, a snapshot version of the file
- HEAD, the latest revision in the repository
- BASE, the backup copy (last revision before changes) kept in the .svn directory on your local computer

Basic work cycle

1. Update your working copy
2. Make your changes
3. Examine your changes
4. Merge other’s changes
5. Commit your changes

Update:

$ svn update
U foo.c
U bar.c
Updated to revision 2.

Letter codes:

U : updated file
A : added file/directory
D : deleted file/directory
R : replaced file
G : file received changes, but same changes already made locally
C: file in conflict, - cannot update file

Make changes to your working copy

- *file changes* -> no need to notify Subversion

- *tree changes* -> ask Subversion to mark files/directories for scheduled removal, addition, copying or moving

File changes:

- Use your favorite editor or whatever program to change a file and subversion will automatically detect that the file has changed.

- These changes will be part of the next commit

Three changes:

- svn add foo
  - the file/directory will be added to the repository on the next commit
- svn delete foo
  - the file/directory will be deleted on the next commit
- svn copy foo bar
  - create a new copy of foo that will be added on the next commit
- svn move foo bar
  - svn copy foo bar
  - svn delete foo

Examine your changes:

- $**svn status**

Checks for local changes since the last update (in the current directory and all child directories). Use svn status ‘path’ to check only a specific child directory + subdirectories

Codes:

-**A,D,C** as on update

- **M** – file is modified

- **?** file/directory is not part of revision control (you have added a file/directory without the “svn add” command)

- **!** the file/directory was under version control, but has been removed without the “svn delete” command

$svn diff

-To see *how* you modified the files

(use
$svn diff 'filename'
to only see modifications to a specific file)

$svn revert

-Cancel your changes and go back to the base copy (from the .svn area)

Resolve conflicts

- when you get a C on update you have a conflict
  - the changes you have made to a file is not compatible with the changes of another user
  - the conflict must be *resolved* before you are allowed to do a commit
  - Subversion provides three temporary files to help you resolving the conflict

Resolving conflicts

- $ svn update
C sandwich.txt
Updated to revision 2.

$ ls -1
sandwich.txt
sandwich.txt.mine
sandwich.txt.r1
sandwich.txt.r2

$ svn commit --message "Add a few more things"
svn: Commit failed (details follow):
svn: Aborting commit: '/home/sally/svn-work/sandwich.txt' remains in conflict

Resolving conflicts

-Three possible solutions

Solution 1:

Merge the changes by hand by editing the file sandwich.txt and tell Subversion you resolved the conflict ($ svn resolved sandwich.txt)

The working copy has conflict markers showing the conflict:

$ cat sandwich.txt

Tomato

Provolone

<<<<<<< .mine

Salami

Mortadella

=======

Sauerkraut

Grilled Chicken

>>>>>>> .r2

Bottom piece of bread

Solution 2:

Copy any of the three temp files on top of the working file sandwich.txt

- NB! This should be communicated with the other developer.

3 possible options:

- Use your changed file (sandwich.txt.mine)
- Use the other developers changed file (sandwich.txt.r2)
- Use the previous revision before any of you changed the file (sandwich.txt.r1)

Example:

```
$ ls sandwich.*
sandwich.txt sandwich.txt.mine sandwich.txt.r2 sandwich.txt.r1
$ cp sandwich.txt.r2 sandwich.txt
$ svn resolved sandwich.txt
```

Solution 3

Revert back to the BASE copy (.svn) and throw away all your local changes.

$ svn revert sandwich.txt

Reverted 'sandwich.txt'

$ ls sandwich.*

sandwich.txt

-‘svn resolved’ is not necessary

-svn resolved

- When you have resolved the conflict using alt.1 (merge by hand) or 2. (use one of the temp files)
- This removes the temp files
- This allows a commit
- Subversion doesn’t check if you resolved it or not, so be careful!!
- Must be used with an argument:
- $svn resolved sandwich.txt

Commit your changes

$svn commit

- you must attach a log message explaining your changes

-$ svn commit --message “made some changes..“

- if you don’t use –message (or --file) Subversion opens your defined ($EDITOR) editor so that you can write the message there.

-Example using a pre-written message file:

$ svn commit --file logmsg

Sending sandwich

Transmitting file data .

Committed revision 4.

Commit

When you haven’t updated your local copy and checked for changes on the repository, your local copy might be out of date:

```
$ svn commit --message "Add another rule"
Sending rules.txt
svn: Commit failed (details follow):
svn: Out of date: 'rules.txt' in transaction 'g'
```

Then you must do an update and resolve possible conflicts upon a new commit.