CSE 390, Autumn 2012
Homework 9: Version Control (svn)
Due Friday, November 30, 2012, 5:00 PM

This assignment focuses on using Subversion for version control. Turn in a file named `homework9.txt` from the Homework section of the course web site, as well as checking in the various repository files described in the document.

Task 1 of 2: Version control scenario

Suppose that a pair of project partners, Alice and Bob, are working on a project together that involves the following files. Alice would like to set up a Subversion repository to manage these files. She has the following files in her local directory. The `.c` and `.h` files compile to produce various `.o` files and the executable `main`, using the `Makefile`. You can assume that any `.c` file depends on the corresponding `.h` file, and that the `main` files depend on all `.c` programs.

file1.h	file1.c	file1.o	file2.h	file2.c	file2.o	file3.h	file3.c	file3.o	main.c	main.o	main	Makefile

a) Which of the files should be checked in to the SVN repository? In your `homework9.txt` file, **write the names of all the files that should be checked in**. Then give a brief one-sentence explanation of why you did not include the other files (if you chose to omit any from the repository).

b) Suppose that the appropriate files above have been checked in to the repo, and then the following actions occur, in chronological order from top to bottom:

1. Alice checks out all files from the repo.
2. Alice modifies her `file1.c`.
3. Alice modifies her `file2.h`.
4. Alice commits `file2.h` to the repo.
5. Alice modifies line 11 of her `Makefile`, the rule that builds `main`.
6. Bob checks out all files from the repo.
7. Bob creates new files `file4.c` and `file4.h` on his local copy.
8. Bob adds a `Makefile` rule to build `file4.o` and modifies its line 11, the `main` rule, to include `file4.o`.
9. Bob modifies his `file2.h` so that it includes `file4.h`.
10. Bob commits `file2.h` to the repo.
11. Alice runs an SVN `update`.
12. Alice commits all uncommitted files she has modified (`file1.c` and `Makefile`).
13. Alice and Bob both attempt to build their local copies by running `make`.

After the above steps, some problems have been introduced both in the repo and in the partners' local copies. A "problem" would be a file that is out of date, missing from the repository or local copy, broken, having a merge conflict, etc. In your `homework9.txt` file, **briefly describe the problems introduced, at which step(s) they occur or will be noticed, and how Alice and Bob can repair them.** *(There are roughly 4-5 total problems total for you to find.)*

(If you're not sure of the answer, you could optionally experiment with what `svn` does by creating a small repository of your own. You can simulate 2 users using the repository by checking out copies to 2 different local directories, and modifying/checking in from each directory independently. Once again, this is *not required*.)

Task 2 of 2: Contribute to a shared svn repository

For this task you will check out and modify the contents of a shared svn repository on the department's `attu` server. The repository is located at the following path: **`/projects/instr/12au/cse390a/hw9/`**

Check out all contents from this repository, then perform the following actions, checking in files at each step as requested:

1. Examine the SVN change history for `ManyChanges.java`. Figure out the file/repo's **current version**, and the date/description of the **2 most recent changes** *to the file*. Include this information in your `homework9.txt` file.
2. Make changes to the file named `changeme_`***`YOUR_CSE_NETID`***`.txt`. For each student in the class we have already set up such a file in the repository. For example, if your name is John Smith and your CSE NetID is `jsmith`, there is a file already in the repo named `jsmith_changeme.txt`. So after you check out the contents of the repository, edit your local copy of the file and **check in your new revision of the file. Please use a descriptive check-in message when submitting your file to the repo.**
3. Create a new file named `message_`***`YOUR_CSE_NETID`***`.sh`. For example, if your name is John Smith and your CSE NetID is `jsmith`, create a file named `message_jsmith.sh`. Your file should contain a very short Bash shell script that echoes any message you like to the terminal. The message will be seen by the grader. The message should be at least 2 lines long but otherwise can be anything you want. Your shell script should not perform other actions such as deleting files, etc. The script should be executable and run without syntax errors. Once you have created and written this file successfully, **submit it to the repository so that it takes its place with the other files found there.** We will grade you on this task by looking at the repository and seeing that the file is there. Remember that adding a new file to the repository is slightly different than editing a file that was already there. **Please use a descriptive check-in message when submitting your file to the repo.**
4. Practice resolving a simple merge conflict on the file `Conflict_`***`YOUR_CSE_NETID`***`.java`. This file already exists in the repo. The first thing you should do is **edit the above file** and change the line that reads, "I have made 0 changes to the file so far." to say, "I have made 1 change to the file." Once you have done this, **run the script called `conflict.sh`** found in the repo. This script will execute a series of actions that lead to a concurrent change being made to the Java file in the repo. When you try to check in your copy of the file, you will receive a merge conflict. **Resolve the conflict** by editing the file to contain both your change and the other changes made by the `conflict.sh` script. (You can try to do an update before checking in and then postpone. (P)) Change the line that conflicts to say, "I have made both changes to the file," and retain any other lines that were added by `conflict.sh`. (Note: `conflict.sh` will only run from a Linux box, such as your own Linux/VirtualBox installation, a CSE basement lab Linux box, or a SSH connection to `attu`.) **Once you have resolved the conflict, check in your new revision of the file. Please use a descriptive check-in message when submitting your file to the repo.**
5. Make a few changes to the existing repo file `Shared.java`. This file contains a Java program that prints information about students' names and messages. You should make two changes to this file. The first change is to **modify line 15** of the file to add your name to the given array of names. The second change is to modify the block of lines beginning at **line 30** to add a line containing a one-line `println` message of your choice. Keep in mind that every student in the class will be trying to modify this same file, so as you are about to check it in, *you may find that you have a merge conflict*. If so, repair the conflict in such a way that your changes, as well as changes made by all other students, are preserved. Please do not submit a change to this file that damages or removes changes made by another student. If you are not sure, please ask a classmate or TA/instructor for help to make sure you do not damage others' work. The SVN repo will keep a record of all checkins, so we will be able to tell who has submitted new versions of the file and what changes were made by each student. **Once you have made your additions and resolved any conflicts, check in your revision of the file. Please use a descriptive check-in message when submitting your file to the repo.**

Installation Notes for SVN:

The `attu` server and CSE basement machines are already set up to allow you to use `svn`. If you plan to work on this phase on your own Linux box, you may need to install Subversion. In Ubuntu do this by typing the shell command:

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

In Fedora, click System->Administration->Add/Remove Software . Search for "subversion" and install the "A Modern Concurrent Version Control System" package. If prompted for "other packages have to be installed...", say yes.

If you need help with Subversion command syntax, see the `svn` examples in links on the web site and in the lecture notes. You can also look at the `man` page for `svn` or run `svn help` ***`command`*** to learn more.