CSC207: Unix Shell and Subversion

For this lab, and for the rest of the labs in this class, please make sure to submit by end of the lab class.

1 Overview

This week, you are going to work with Subversion using a UNIX command line shell. You will also practice submitting your work using subversion and MarkUs.

2 Choose a driver and a navigator

In these labs, you are encouraged to work in pairs. You and your partner together will be able tofigure out problems better than you would individually. Find yourself a partner. If you have trouble finding one, let your TA help you. This partnership is only for today's lab. We strongly advise you to form a partnership with a colleague who has a similar level of background. For example, if this is your first time working in a Linux environment, we suggest that you partner with someone who is also new to Linux. In all the labs, we will use the terms *driver* and *navigator*. Here are the definitions of the two roles:

- **Driver**: Types at the keyboard. Focuses on the immediate task at hand.
- **Navigator**: Thinks ahead and watches for mistakes.

In lab handouts, we'll often refer to you as `s1` and `s2`, and `s1` will be the first driver.

3 Log in and get things set up.

Use MarkUs to form a group with your lab partner. This will require `s1` to log in to MarkUs and invite `s2` to form a group. Then `s2` will need to log in and accept the invitation from `s1`. Take a note of the group MarkUs created for you. It will be of the form `group_nnnn`.

`s1` drives and `s2` navigates.

Now `s1` logs in and opens a new terminal window. You need to use some basic Unix commands in order to complete this lab. These commands are listed on the second handout. You will learn many more as you get comfortable working in a Linux environment throughout the term.

1. Change to `s1`'s home directory.
2. Create two directories named `local1` and `local2` in `s1`'s home directory (or in the directory of your choice). These two directories will be used later for checking out local copies of your repository.

4 Subversion (svn)

All repositories in this course will have a URL of the form:

`http://markus.utsc.utoronto.ca/svn/csc207-2016-05/repo-name`

It can be accessed by you, by the instructor, and by the TAs, but not by other students.

1. Change directories to `local1` and check out a working copy from `s1`'s repository. If `s1` has never checked out a repository from markus.utsc.utoronto.ca, s1's UTOR password will be required. At the end of this process, you should see something like "Checked out revision 1".

2. There will be a directory in your repository called `Lab1` that was created by your instructor. Change directory to the directory `Lab1` that is inside your local copy of your repository.

3. Create two directories, `dir1` and `dir2`, inside the directory `Lab1`, and add them to the repository. Recall that the Subversion's `add` command marks a file for addition to the repository, but does not modify the repository. Run the `commit` command too. Log messages should be meaningful. An empty message or a message like "did some changes" is not useful. A helpful message might be "Renamed method foo to bar", or "Removed the 3-argument constructor". The repository maintains version numbers on a per-commit basis. When a new version of a file (or several files) is committed, the version number of all files is incremented.
   Note: If you don't specify a log message, the editor pops up to allow you to create this log message.

4. Inside the directory `dir1`, create a file named `myfile.txt` with a text editor (e.g., by running `vi myfile.txt`), save it in the `Lab1` directory, and add it to the repository. Commit your changes.

5. Edit `myfile.txt`: put the names of `s1` and `s2` (and anyone else you are working with) in the file and *at least two blank lines* after the last name. Commit the changes to the repository.

   Switch roles: `s2` drives and `s1` navigates.

6. Open another terminal window. Go to `s1`'s home directory, change to the `local2` directory, and check out `s1`'s SVN project.

   Note: we are still in s1's account.

7. Make some changes to `myfile.txt` in `local2`, but leave the group member names alone, and commit the changes.

   Switch roles: `s1` drives and `s2` navigates.

8. Update the local copy in `local1`. Since you just changed `myfile.txt` in `local2` and committed the new version to the repository, you should see the changes in `local1`.

9. Now, you will create a set of conflicting changes. To do so, edit `myfile.txt` again in `local1`, changing the first line, and commit the changes.

10. Before updating `local2`, switch roles (`s2` drives and `s1` navigates) and have `s2` change the same line in `myfile.txt` inside `local2`. Make different changes this time, so that there will be a conflict.

11. Try to commit the changes. This should fail, since the file `myfile.txt` in `local2` has not been updated. Hint: take note of revision numbers that svn reports to you, and see how they are incremented with each commit.

12. Issue an `svn update command`. Read the feedback from the command carefully. Choose to edit the file `myfile.txt` to resolve the conflict and finish the update.

    Switch roles: `s1` drives and `s2` navigates.

13. In the terminal window containing `local2`, display the status of working copy of files and directories. With `--show-updates`, the `status` command adds working revision and server out-of-date information. With `--verbose`, it prints full revision information on every item.

14. Look at the revision history of `myfile.txt` using the `log` command.

15. What needs to be done now to make sure `local1` and `local2` both contain the latest version of `myfile.txt`? Complete the necessary steps.

5 Submitting your work

There is nothing else that you need to do to "submit" this lab. We will simply examine the history and content of your repository when marking.