SVN primer

This course uses the Subversion version control system, commonly known as SVN, to keep track of lab files. This guide should help you get up and running with SVN so that you can start working on the labs.

Things to keep in mind

Many SVN commands have an optional path argument, and if this isn’t specified (which is usually the case), they only operate on the current directory and its subdirectories. For this reason, we have explicitly mentioned which directories certain commands should be run from.

Most SVN commands have abbreviated names. For clarity, we have used the full names below, but each command is linked to its description in the SVN book, which lists the abbreviated versions in parentheses after the command name. For example, `svn status` can be shortened to `stat`.

Checking out your files

Files are stored in a central repository on the EWS servers; the first step is checking out the repository on your machine to get a working copy. Each student gets their own directory within the repository, so to check your directory out, open a terminal window and enter the following command, replacing YourNetID with your actual NetID:

```
svn checkout https://subversion.ews.illinois.edu/svn/fa16-cs233/YourNetID ~/cs233
```

If all goes well, you might be asked for your Active Directory password, after which your lab files will be checked out in the directory `~/cs233`. If something goes wrong, see the troubleshooting subsection below.

**Important:** You only need to check out the repository once for each machine you work on. After you have a working copy checked out, you simply need to update it to get any new files, which is what the next section is about. (This may be different from what 225 makes you do, if you're taking that course concurrently. When in Rome, etc etc.)

Troubleshooting

If you get a “URL doesn't exist” or “403 Forbidden” error, we don't have a directory set up for you. Everyone who's registered for the course has a directory; if you aren't registered, you need to talk to Professor Zilles.

If you get a message asking for a GNOME keyring password and you have no idea what said password is, the easiest thing to do is recreate your keyrings. Run the following command:

```
rm -rf ~/.gnome2/keyrings
```

and then rerun the checkout command. This time, it should ask you for your AD password, after which you'll get asked to create a password for the default keyring. Remember the password you choose so you can use it on subsequent appearances of the message. You might also get the option to automatically unlock the keyring on login, which I would recommend selecting.

Please don't hesitate to ask your lab TAs for help if you need it.

Committing your code

After you've made changes to your files, you can commit them back to the repository using the command

```
svn commit -m "short message describing your changes"
```
You should run this command from the directory in which the changed files reside. The commit message can be left empty, but that is not good practice. If you want to only commit specific files, you can include those filenames as part of the command.

You must commit your code for it to get graded. Don’t let your hard work go to waste by failing to do so. We can only grade what’s in the repository, and it’s surprisingly common for people to forget to commit.

A common guideline for using version control systems is to commit early and commit often. This is especially useful when collaborating with others on a single repository, but it’s a good idea to do so even when you’ve working on your own, since it allows you to easily go back to a recent working version if you mess something up.

Getting new and updated files

Once you have a working copy checked out, you need to update it from time to time to get new and updated files from the repository. To do so, navigate to the root of your working copy; assuming you followed the directions above, this will be located at ~/cs233. Once in this directory, enter the command

```
svn update
```

to get new files from the repository.

Checking up on your files

`svn status` tells you the state of your working copy, i.e. what files have been changed since the last commit. It’s especially useful when you want to make sure you haven’t inadvertently modified any files you weren’t supposed to.

If you want to know the specific changes you made to a file, use `svn diff`. The most basic usage is

```
svn diff filename
```

which tells you what changes have been made to a file since the last commit. There are other ways to use this command, many of which are covered by the examples section of the linked page.

Finally, if you want to look at the commit history of your repository, use `svn log`. Two common usages are

```
svn log filename
```

to get the log for a particular file, and

```
svn log
```

to get the log for an entire directory - you might have to run `svn update` before this for it to work correctly. Again, other usages are covered on the linked page. This is particularly useful when you want to get the most recent commit time of a file, e.g. in order to know whether you made the deadline.

Undoing incorrect changes

The simplest case is when you want to go back to the last version that was committed. This operation is known as reverting, and the command is

```
svn revert filename(s)
```

If you want to revert an entire directory, use `-R .` for the `filename(s)` part.
If you've already committed erroneous changes, life is a bit more complicated. You need to perform a reverse `merge` to undo the commit. Use `svn log` to get the revision number you want to undo - call this `n` - and then use the command

```
svn merge -c -n filename(s)
```

to reverse the commit. If you want to undo the commit for all the files in a directory, use . for the `filename(s)` part.

### Fixing a locked working copy

If you get an error message along the lines of "working copy locked, run 'svn cleanup' to remove locks", you need to clean up your working copy, as the message suggests; run the command

```
svn cleanup
```

from the root of your working copy; again, assuming you followed the directions above, this will be located at `~/cs233`. If this fails to work, see [http://stackoverflow.com/questions/127932/svn-working-copy-xxx-locked-and-cleanup-failed](http://stackoverflow.com/questions/127932/svn-working-copy-xxx-locked-and-cleanup-failed) - the simplest approach is to

1. Copy edited items to another location.
2. Delete the folder containing the problem path.
3. Update the containing folder through Subversion.
4. Copy your files back.
5. Commit

### Additional resources

This just scratches the surface of SVN's capabilities - if you want to learn more, the SVN book is a great resource, as is Google.

SVN is an example of a centralized version control system - there is a clear distinction between the repository on the server and the working copy on your machine. Distributed version control systems, such as Git and Mercurial, remove this distinction, and are considered superior in certain ways. It's definitely worth acquainting yourself with them if you haven't already.