Help Topic: The SVN Status Command

Understanding Your Tree Status and useful Aliases

Spring 2021

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

The SVN `status` Command

The `svn status` command informs you about what has changed in your local repository checkout compared to the moment you first checked it out or last did an `update`. It does not compare it to the contents of the svn server. For this reason `svn status` may be invoked offline. This is a tool I find valuable for confirming what exactly I may have done to my local checkout. This is important (a) right before committing changes, and (b) for confirming that I have the same checkout as a colleague working with the same revision number. More info on `svn status` can always be found by Googling "Subversion book" and reading the full PDF online free, or just typing `svn help status` anytime on the command line.

Basic usage of the svn `status` command

The basic syntax for the svn `status` command is:

```
$ svn status        (invoked somewhere within previously checked out tree)
```

The `svn status` does *not* require a network connection to the server and only indicates changes compared to your initial checkout or previous update.

Deciphering the output of the svn `status` command

The `svn status` output may look something like the below by way of example:

```
$ svn status
M       Colors.cpp
A       Fungo.cpp
A       Fungo.h
M       Items.cpp
D       Mango.h
?       Utils.h
?       Utils.cpp
```

Files that have not been changed, deleted, or added or are any way different compared to the last update are *not* reported. Otherwise each line reports the file name, and how it is changed with the single character on the left. In the example above, the output tells us:

- The files `Colors.cpp` and `Items.cpp` are already known to svn, but have been modified locally compared to the last update.
- The files `Fungo.h` and `Fungo.cpp` have been newly added by you. Your local svn tree knows about them, but they they are not known on the remote svn server. They have been *scheduled* for addition the next time you do a commit of your local changes to the server.
- The file `Mango.h` has been scheduled for deletion. It is gone locally, but is still part of the official repository on the server. Your next commit will remove it on the server.
- The files `Utils.h` and `Utils.cpp` are in your local file system but they are completely unknown to svn. They were not part of the checkout or last update, and no action will be taken on them during the next commit. Possibly they should be rightfully ignored, or perhaps you have forgot to `svn add` them.

Generally you will encounter files having one of the above modes. There are others however, and you can read about them in the svn documentation, or get a glimpse of them with `svn help status`.

Using the svn `status` command in quiet mode

The `svn status` command may be used with the optional argument, `--quiet`, or simply `-q`. This has the effect of suppressing unversioned items. In the example above, you would get the following output instead:

```
$ svn -q status
M       Colors.cpp
A       Fungo.cpp
A       Fungo.h
M       Items.cpp
D       Mango.h
```

The last two files, `Utils.h` and `Utils.cpp` are suppressed. The `--quiet` option is very useful when your local repository has many automatically generated files. This could include object files if you're working in C++, or other temporary files if you're working with LaTex for example. In these cases the number of such lines can overwhelm the output of the `svn status` command, rendering it hard to use.

Clever use of the svn `status` command in quiet mode

While the `--quiet` option is useful in suppressing unversioned files, sometimes those unversioned files are *supposed to be* under version control. The `--quiet` option may prevent you from noticing this. One common "gotcha" in groups of developers involves one developer forgetting to add and check in a critical piece of code. Things work fine on the first developer's machine, but are perhaps broken on others' machines due to that missing piece of code.

Here we describe a little hack using aliases, to get the best of both worlds, the succinctness of the `--quiet` command and the verbosity you need to catch critical files that are perhaps unintentionally unversioned.

The below couple of aliases are simply shortcuts for invoking the `svn status` command with and without the `--quiet` option. It's not hard to type out these few words, but this alias makes it

very handy:

```
alias svns    'svn status'
alias svnsq   'svn --quiet status'
```

The next alias is where things get fun. We describe how this works, but you can also just cut and paste this into your shell configuration file. The goal is to supress all unversioned files *except* certain types. In the case below, the goal is to not supress C++ header and class files:

```
alias svns_h    'svns | grep "? " | grep .h'
alias svns_cpp  'svns | grep "? " | grep .cpp'
alias svnsm     'svnsq; svns_h; svns_cpp'
```

The last alias, `svnsm`, is the only one I ever use. The previous two, `svns_h` and `svns_cpp`, just help build up the definition of the third. A few very common command line tricks are used here. You can read up on them separately if they aren't familiar to you:

- Piping output: The output of one command line invocation can be sent as input to another command line tool using the "pipe" utility. In the first line above, the output of "svns" is sent as input to the `grep` command, whose output is then sent as input to another invocation of `grep`.
- The `grep` utility will filter given input, to produce only lines that contain a given pattern. In the example above, `grep "?  "` only matches lines containing the 2-character string "? ", which should catch lines output from the `svn status` command for unversioned files. Of course we don't want *all* unversioned files so we further filter on two file types, the `.h` and `.cpp` files.
- The last little trick may be obvious. As on the last line, you can build up an alias by chaining them together with a semicolon.

The powerful command line tool `grep` is used to