SubversionTutorialPart1

From Erh-itproject

This tutorial is designed to show a basic work flow using Subversion and to demonstrate the use of some of the svn commands. Please keep the SVN Book handy for reference, especially Chapter 2, for more details and descriptions of how to interpret the output of the commands.

Part 1

Part 1 covers the basics of creating a repository, creating a new application in the repository, and the following basic svn subcommands:

- info
- status
- checkout
- commit
- diff
- rename
- delete
- revert

This tutorial presents the commands you should run but in general does not show the results (to encourage you to actually perform the commands). A sample log of actually performing this tutorial is at SubversionTutorialLogPart1. Commands you should enter are prefaced with ==> and are in bold.

Create directories in your home directory for a repository and a working copy:

```
==> cd $HOME
==> mkdir svnrepo svnwork
```

Create a repository named test

```
==> svnadmin create $HOME/svnrepo/test
```

Check out the repository to get a working copy:

```
==> cd svnwork
==> svn checkout file://$HOME/svnrepo/test
```

List information about and status of the working copy:

```
==> cd test
==> svn info
==> svn status -uv
```

The svn info command is just to demonstrate how to determine what repository the working copy is from. The svn status is used to determine if your working copy is out of date or has modifications pending. The status and update commands should be used frequently.

Next, initialize the repository layout; we will use standard convention of trunk, tags, branches directories at the top level of the repository:

```
==> mkdir trunk tags branches
==> svn status -uv
==> svn add trunk tags branches
==> svn status -uv
==> svn commit -m "Initial repository layout"
```

The repository has its initial layout. We will delete the working copy and start over working in the trunk branch (trunk is usually where main development is done). The key here is we are just cleaning up the working copy and not making any changes to the the repository:

```
==> cd $HOME/svnwork
==> rm -rf test
==> svn checkout file://$HOME/svnrepo/test/trunk test
==> cd test
==> svn info
```

Now create a new application named myApp. At the end of this tutorial, myApp will have a program file, a config file and documentation. We will start following a normal development cycle where we add stuff incrementally. First check out trunk and create the myApp directory.

```
==> mkdir myApp
```

At this point, we could add the myApp directory to the repository, but since this is a new application, we will make all the application's files first.

```
==> cd myApp
```

Create the following two files:

myscript.sh:

```
#!/bin/sh
# Source in configuration file
. myApp.config
echo "My config=$site"
```

myApp.config:

```
# Site configuration
site=RNK
```

Make myscript.sh executable, and test running it. Edit the program to get it to run; it should just print the value of site.

```
==> chmod +x myscript.sh
==> ./myscript.sh
```

Place the new files under subversion control, and commit:

```
==> cd ..
==> svn add myApp
==> svn commit -m "Created myApp"
==> cd myApp
```

Note that the svn add and commit commands both worked recursively by default recognizing all the updated files. See the svn commit documentation for how to commit specific files (svn help commit).

Now lets make a change. For simple changes like changing the content of files, just edit as you normally would, review and test changes, and commit.

Edit myscript.sh to as below to add some simple error checking.

myscript.sh:

```
#!/bin/sh
# Source in configuration file
if [ -f myApp.config ]
then
  . myApp.config
else
  echo "Resource file myApp.config missing"
fi
echo "My config=$site"
```

Make sure to test running the program again. Once done editing, run the diff command to review your changes:

```
==> svn diff myscript.sh
```

Note in the output that the diff is done on the working copy against the latest version in the repository. (I find diffs very useful for reviewing changes but use a graphical diff program like tkdiff). Commmit the changes. It is a good practice to do an svn status to double check what has been changed before doing the commit. In this case, only myscript.sh should be tagged as modified:

```
==> svn status -uv
==> svn commit -m "Added check for config file."
```

You can also undo changes. The most common is to undo changes that didn't work. Edit myscript.sh again and make a change like add a new comment. Use svn revert to restore to the file to its latest version in the repository.

```
==> vi myscript.sh
==> svn diff myscript.sh
==> svn revert myscript.sh
==> svn diff myscript.sh
```

svn revert also works with uncommitted changes made with svn copy, delete, rename commands.

Now we will demonstrate how to change the directory structure which requires use of svn commands.

The overall layout of this application is not quite right for others to install. If they were to re-install the application, it would overwrite the configuration file which they may have customized. So lets make a config directory and move the configuration file to it:

```
==> mkdir config
==> svn add config
==> svn rename myApp.config config/myApp.config
==> svn status -uv
==> svn commit -m "Moved default configuration file to new config directory so as not to overwrite
```

The svn status command shows that config and config/myApp.config were added and myApp.config was deleted. We used the svn rename command to keep a history of where config/myApp.config originated from as we can see from the log entries. You can see that /trunk/myApp/config/myApp.config actually came from /trunk/myApp/myApp.config (revision 2):

```
==> svn log -v config/myApp.config
```

results in:

```
------------------------------------------------------------------------
r4 | awipsusr | 2008-12-10 17:13:27 +0000 (Wed, 10 Dec 2008) | 1 line
Changed paths:
   A /trunk/myApp/config
   A /trunk/myApp/config/myApp.config (from /trunk/myApp/myApp.config:2)
   D /trunk/myApp/myApp.config

Moved default configuration file to new config directory so as not to
overwrite any existing configuration when installing.
------------------------------------------------------------------------
r2 | awipsusr | 2008-12-10 16:55:30 +0000 (Wed, 10 Dec 2008) | 1 line
Changed paths:
   A /trunk/myApp
   A /trunk/myApp/myApp.config
   A /trunk/myApp/myscript.sh

Created myApp
------------------------------------------------------------------------
```

More importantly, the svn repository generally only saves differences between revisions to keep the database small. In this case, the whole file is not copied in the repository, only how to make the change. For very large files, this distinction can have a significant impact on the repository.

Now, lets go back and do it another way, where we use manual commands and don't keep the file's history. First we have to remove what we did, then redo it manually.

```
==> cp config/myApp.config .
==> svn delete config
==> svn add myApp.config
==> svn commit -m "Manually undoing previous change"
```

Now move the file using OS commands:

```
==> mkdir config
==> cp myApp.config config/myApp.config
==> svn add config
==> svn commit -m "Created config directory for sample config file."
==> svn delete myApp.config
==> svn commit -m "Deleted myApp.config so it does not overwrite on install"
==> svn log -v config/myApp.config
```

results in:

```
------------------------------------------------------------------------
r6 | awipsusr | 2008-12-10 17:28:03 +0000 (Wed, 10 Dec 2008) | 1 line
Changed paths:
   A /trunk/myApp/config
   A /trunk/myApp/config/myApp.config

Created config directory for sample config file.
------------------------------------------------------------------------
```

The log file shows that config/myApp.config has no history before revision 6. Therefore, the whole file was inserted as new. Now there are at least two complete copies (considered totally unrelated by Subversion) of myApp.config in the repository in different revisions. In this latest revision, we deleted myApp.config from the repository, but is not really gone. Once something is added to the repository, it is there forever; the space it consumes in the repository cannot be recovered. Deleted items are just no longer part of the latest (HEAD) revision as we can see by listing the repository at a previous revision and view the contents with svn cat:

```
==> ls myApp.config
(file doesn't exist in working copy)
==> svn list -r 5
==> svn cat file://$HOME/svnrepo/test/trunk/myApp/myApp.config@5
```

The first method for moving myApp.config using svn rename is the preferred way to make this type of change. It allows change history to be maintained and also makes better use of the storage efficiencies of the repository. Deleting and adding the same files back in only wastes space in the repository and doesn't preserve past relationships. There are better ways to manage this by restoring items from a previous revision using svn copy.

This completes Part 1 of the tutorial. Please leave the working copy and repository as is in your home directory for Part 2.