

Subversion

An Open-Source Version Control
System

Presentation by: Eric Hagen

What are Subversion's Features?

- Directory Versioning
 - Subversion tracks changes to whole directory trees
- True Version History
 - Add, delete, copy, and rename both files and directories
 - Each file retains its own individual history

What are Subversion's Features?

- Atomic Commits
 - Modifications commit completely or not at all
- Choice of Network Layers
- Consistent Data Handling

Revision Keywords

- HEAD
 - Youngest version in the repository
- BASE
 - Current revision of your working copy
- COMMITTED
 - Most recent revision prior to or equal to BASE
- PREV

Initial Checkout

- Checkout command – `svn checkout`
 - Creates a working copy on your machine of the HEAD or most recent revision
 - Calling example
 - `svn checkout <Repository URL> [<directory>]`
 - `svn checkout https://svn.sourceforge.net/svnroot/gpstk`

Updating your working copy

- Update command – svn update
 - Updates your current working version to the HEAD
 - Calling example
 - svn update

```
bash-3.00$ svn update
At revision 19.
bash-3.00$ █
```

Make Changes to your Working Copy

- Add command – `svn add`
 - Schedule a file, directory, or symbolic link to be added to the repository
 - Calling example
 - `svn add <FILE>`
 - `svn add ETests.pl`



Applications

Actions



File

Edit

View

Terminal

Tabs

Help

```
bash-3.00$ cd svn
```

```
bash-3.00$ cd gpstk
```

```
bash-3.00$ ls
```

```
main
```

```
bash-3.00$ cd main
```

```
bash-3.00$ ls
```

```
apps          ChangeLog      config.sub      COPYING        Doxy
```

```
AUTHORS      config.guess   configure.ac    depcomp        exam
```

```
bash-3.00$ cd tests
```

```
bash-3.00$ emacs ETests.pl&
```

```
[1] 7349
```

```
bash-3.00$ svn add ETests.pl
```

```
A          ETests.pl
```

```
[1]+  Done
```

```
emacs ETests.pl
```

```
bash-3.00$ █
```

Make Changes to your Working Copy

- Remove command – `svn remove`
 - Schedule a file, directory, or symbolic link to be removed from the repository
 - Calling example
 - `svn remove <FILE>`
 - `svn remove ETests.pl`

```
bash-3.00$ svn remove tests/ETests.pl
D      tests/ETests.pl
```

Make Changes to your Working Copy

- Copy command – `svn copy`
 - Create a new item as a duplicate of another item. The new item is then scheduled for addition to the repository
 - Calling example
 - `svn copy <FILE1> <FILE2>`
 - `svn copy ETests.pl ETests2.pl`

```
bash-3.00$ svn copy tests/ETests.pl tests/ETests2.pl
```

```
A      tests/ETests2.pl
```

```
bash-3.00$ █
```

Make Changes to your Working Copy

- Move command – `svn move`
 - The same functionality as `svn copy` except the original file is scheduled for removal from the repository
 - Calling example
 - `svn move <FILE1> <FILE2>`
 - `svn move ETests1.pl ETests2.pl`

```
bash-3.00$ svn move tests/ETests.pl tests/ETests3.pl
```

```
A      tests/ETests3.pl
```

```
D      tests/ETests.pl
```

```
bash-3.00$ █
```

Checking your Changes

- Status command – `svn status`
 - Detects all the changes you have made to your working copy from the BASE
 - Calling example
 - `svn status`

```
bash-3.00$ svn status
```

```
A      ETests.pl
```

```
bash-3.00$ █
```

Some Status Codes

- A
 - The file, directory, or symbolic link is scheduled for addition
- C
 - The file, directory, or symbolic link is in a state of conflict
- D
 - The file, directory, or symbolic link is scheduled for removal

Some Status Codes

- M
 - The file, directory, or symbolic link has been modified in some way
- R
 - The file, directory, or symbolic link is scheduled to replace an item
- ?
 - The file, directory, or symbolic link is not under version control

Difference

- Difference command – `svn diff`
 - You can find out exactly how you have changed a file by comparing it to a cached version in a `.svn` file
 - Calling example
 - `svn diff`
 - Prints out all changes to all files

```
bash-3.00$ emacs ETests.pl&
[2] 7501
bash-3.00$ svn status
M      ETests.pl
[2]+  Done                  emacs ETests.pl
bash-3.00$ svn diff
Index: ETests.pl
=====
--- ETests.pl      (revision 19)
+++ ETests.pl      (working copy)
@@ -1,2 @@
   #This is only a test!
+#Change!
bash-3.00$ █
```

Revert

- Revert command – `svn revert`
 - Reverts a file to its premodified state
 - Essentially the same as removing the item then `svn updating` that file
 - Calling example
 - `svn revert <FILE>`
 - `svn revert ETests.pl`

```
bash-3.00$ svn revert ETests.pl
Reverted 'ETests.pl'
bash-3.00$ cat ETests.pl
#This is only a test!
bash-3.00$ █
```

Update the Repository with Commit

- Commit command – `svn commit`
 - Sends all of your changes to the repository
 - Log message goes with commit to describe the change
 - Calling example
 - `svn commit -m "Message"`

```
bash-3.00$ svn status
```

```
A      main/tests/ETests.pl
```

```
bash-3.00$ svn commit -m "Eric testing more, adding ETests.pl"
```

```
Adding      main/tests/ETests.pl
```

```
Transmitting file data .
```

```
Committed revision 19.
```

```
bash-3.00$ █
```

Examining History with Log

- Log command – `svn log`
 - Displays the information about the history of a file, directory, or revision
 - Calling example
 - `svn log -r 15`
 - `svn log ETests.pl`

```
bash-3.00$ svn log -r 19 -v
```

```
r19 | ehagen | 2006-06-08 16:44:23 -0500 (Thu, 08 Jun 2006) | 1 line
```

```
Changed paths:
```

```
  A /main/tests/ETests.pl
```

```
Eric testing more, adding ETests.pl
```

```
bash-3.00$ █
```

Viewing with Cat

- View command – `svn cat`
 - Displays the contents of a file without actually downloading it off the repository
 - Calling example
 - `svn cat <FILE>`
 - `svn cat ETests.pl`

Viewing with List

- List command – `svn list`
 - Displays the files within a directory in the repository without actually downloading the directory
 - Calling example
 - `svn list <DIR>`
 - `svn list https://svn.sourceforge.net/svnroot/gpstk`

Conflicts

- The svn status symbol C next to a file denotes a conflict between your file and the latest version on the repository
- A conflict is defined as an overlap between changes already made to the repository and changes on your own working copy

Conflicts

- For every conflicted file Subversion finds, it places up to three extra, unversioned files in your working copy
- filename.mine – your file as it existed before you updated your working copy
- filename.rOLDREV – that file as it existed on the BASE revision, before you made changes to it on your working copy

Conflicts

- filename.rNEWREV – this is the file that you just received when you updated your working copy. This file corresponds to the HEAD.
- If one or more files is in a state of conflict, you cannot commit your changes to the repository until the conflict is resolved

Resolving Conflicts

- A conflict can be resolved in one of three ways
 - Merge the changes by hand
 - Copy one of the temporary files on top of your working file
 - `svn revert` to discard all of your local changes
- In order to commit, once one of these three things is done, run `svn resolved` to let Subversion know you have resolved the conflict

Merging Conflicts by Hand

- Lets say sandwich.txt is in a state of conflict.
- This is what sandwich.txt would look like

```
$ cat sandwich.txt
Top piece of bread
Mayonnaise
Lettuce
Tomato
Provolone
<<<<<< .mine
Salami
Mortadella
Prosciutto
=====
Sauerkraut
Grilled Chicken
>>>>>> .r2
Creole Mustard
Bottom piece of bread
```

Merging Conflicts by Hand

- The text between the <<< and === are changes you made in the conflicting area
- The text between the === and >>> were made by someone else in another revision

```
$ cat sandwich.txt
Top piece of bread
Mayonnaise
Lettuce
Tomato
Provolone
<<<<<< .mine
Salami
Mortadella
Prosciutto
=====
Sauerkraut
Grilled Chicken
>>>>>> .r2
Creole Mustard
Bottom piece of bread
```

Merging Conflict by Hand

- Now that the conflicting area is known, resolution can be made between the two parties to find out how the conflict should be handled
- After the conflict is resolved, run `svn resolved` and the working copy can be committed to the repository

Conclusions

- Subversion is easy to use for the beginning user
 - Commands are simple, not too numerous, and easy to learn
- Subversion is a powerful tool that preforms a necessary job