

An Introduction to Subversion

Scott Martin

Department of Linguistics
The Ohio State University
<http://www.ling.osu.edu/~scott/>

LCC Tutorial
May 14, 2007



Why Subversion?

History All committed versions of a document are maintained forever

Availability Documents are securely accessible in a single place

Sharing Several people can contribute to a document

History

- Changes to a document are committed under a new revision number each time
- These incremental changes reflect the evolution of a document over time

Availability

- Versions of a document are tracked in a single place, the repository
- Documents are accessible over the internet via SSH tunnel
- No extra configuration needed, just a Unix account

Sharing

- The date and time of a new revision is maintained along with the user who committed it
- Additions, deletions, and changes to a document are tracked on a line-by-line basis
- Changes made to the same document by different users can be merged automatically or manually

Creating a Repository

In user's home directory,

```
$ svnadmin create repository
```

creates a new repository in the directory repository/.

Importing Project Files

In the directory where project files are located:

```
$ svn import . [URL]
```

imports a project into the new repository.

The Repository URL

The URL from the previous slide is in the format

```
svn+ssh://user@host/home/user/repository/project
```

user The username

host The host where the repository resides

project The name of the project in the repository



Checking Out a Working Copy

A working copy of a project can be checked out of the repository with:

```
$ svn checkout [URL]
```

This command gets the latest versions files contained in the repository associated with URL (as described in the previous slide). To make sure that the repository contains everything in the imported project, use:

```
$ diff project_dir checkout_dir
```

on Unix.

Updating

To update the working copy files to the latest revisions in the repository:

```
$ svn update
```

All files in the current directory are updated. To update a single file, simply use:

```
$ svn update [file]
```

This command can also fetch a revision different than the latest revision with the `-r` flag:

```
$ svn update -r n [file]
```

where n is the desired revision number.

Status Check

The status of the files in a working copy is available using:

```
$ svn status
```

This command gives information about which files are changed or new.

Adding and Removing Files

New files are added or removed with:

```
$ svn add [file]
```

```
$ svn delete [file]
```

Note that files are not actually added or deleted to the repository until committing.

Committing Changes to Existing Files

After editing files, changes are committed from the project directory:

```
$ svn commit --message="a commit message"
```

Without the `--message`, Subversion starts an editor to ask for a commit message.

Viewing Commit Messages

Commit messages are available for all revisions with:

```
$ svn log [file]
```

Messages are printed in chronological order along with the associated revision number.

Annotating Changes

To see which users made which changes:

```
$ svn annotate [file]
```

Changes are shown line by line with the revision number and associated user name.

Help

Help on any command is available:

```
$ svn --help [command]
```

Where `command` is one of `import`, `checkout`, `commit`, `update`, `export`, `status`, `log`, etc.

Clients

Some front-end clients for Subversion are available:

[TortoiseSVN](#) For Windows

[Subclipse](#) For Eclipse IDE

[Netbeans](#) contains a Subversion integration module

These clients take care of the commands in the background. Most provide a graphical user interface.



Subversion Reference

- Ben Collins-Sussman, Brian W. Fitzpatrick and C. Michael Pilato. *Version Control with Subversion*.
<http://svnbook.red-bean.com/>
- List of some Subversion front-end clients:
<http://subversion.tigris.org/links.html>