

# Subversion

- <http://subversion.tigris.org/>
- Subversion is a free/open-source version control system
- It stores a tree of files in a central repository. The repository is much like a database server, except that it does not use SQL to access the data. Instead, it uses its own command syntax.
- Subversion can access its repository across networks, which makes it a good choice for free/open source projects.

## Subversion Features

- Directory versioning
- True version history
- Atomic commits
- Versioned metadata
- Choice of network layers
- Consistent data handling
- Efficient branching and tagging
- Hackability

## Subversion Requirements

- › APR (the Apache Portable Runtime library)
- › Apache HTTP Server (optional)
  - svnservice: standalone server
- › Berkeley DB (optional, but strongly recommended)
- › Subversion

# Revisions

- Subversion's revision numbers apply to entire trees, not individual files
- A new Subversion repository begins its life at revision zero and each successive commit increases the revision number by one
- Several commands use revision as an argument  
    `svn -revision`
- Revision Keywords:
  - ◆ HEAD: The latest revision in the repository.
  - ◆ BASE: The “pristine” revision of an item in a working copy.
  - ◆ COMMITTED: The last revision in which an item changed before (or at) BASE.
  - ◆ PREV: The revision just before the last revision in which an item changed (Technically, COMMITTED – 1.)
- Revision dates:
  - ◆ Enclosed in braces "`{ }`"
  - ◆ In order of: year month day hour minute second but not all fields required

## Repository operations

- Recommended directory structure for each project directory is to have the following subdirectories:
  - ◆ trunk, branches, tags
  
- Checkout creates working directory under current directory  
`svn co svn:/path_to_repository/project/trunk my_working_copy`
  
- Add new files edit and save `my_filename.c`  
`svn add my_filename.c`
  
- Commit changes specific files or entire tree below current directory  
(NOTE: Must be in directory where file is located)  
`svn commit my_filename.c -m 'Fixed the foo bug'`  
`svn commit -m 'Fixed the foo bug'`  
`svn commit --non-recursive -m 'Fixed the foo bug'`
  
- Update working copy from repository  
`svn update`  
`svn update -r revision_number`

# Repository History

- View the log of changes according to your working copy  
(must update to match repository)  
svn log
- View changes to a file  
svn diff
- View file at a particular revision  
svn cat
- View directory listing at a particular revision  
svn ls
- View history of a path in the repository  
svnlook history
- View differences between revisions  
svnlook diff

# Project Branches

- › Reasons
  - Freeze a release for only fixes, no new features
  - Parallel code path (ie. GPL and commercial)
  
- › Copy trunk to a branch

```
cd /working_copy_of_project
svn copy svn:/path_to_repository/project/trunk \
svn:/path_to_repository/project/branches/Rev-1.5 \
-m 'Freezing Rev. 1.5'
```
  
- › To work on branch, create a working directory under current directory

```
svn co svn:/path_to_repository/project/branches/Rev-1.5 my_Rev15
```
  
- › Perform repository operations just like in trunk

## Merging branches

- Reasons
  - Fixes in branch should be included in trunk
  - Branch is used for major change that is ultimately to be part of trunk
- View differences between files in trunk and branch  
`svn diff reva:revb`
- Merge changes from trunk into working copy of branch  
`svn merge reva:revb svn:path_to_repository/project/trunk/my_filename.c`
- Commit changes to branch  
NOTE: CWD = directory of branch working copy  
`svn commit -m 'Merged changes from trunk'`

# Administration

- Create repository  
    svnadmin create
  
- Recover from DB errors  
    svnadmin recover
  
- GOTCHA: Multiple users are running locally or sharing the standalone server, permissions may cause DB problems. The workaround is to frontend svn or svnservice command with shell script to set umask.  
    #!/bin/sh  
    umask 002  
    svn.orig
  
- Backups
  - Berkeley DB: Sleepycat describes procedure for full backup
  - Incremental backups
    - svnadmin hotcopy
    - hot-backup.py: Python wrapper for svnadmin hotcopy

# Subversion Server

- May use Apache if wide ranging project or fine tuned access required
- Standalone server sufficient for smaller projects  
    `svnserve -d -r /path_to_repository` (-r argument allows users to shorten svn command)
- User file specified in `svnserve.conf`  

```
[general]
password-db = our_pw_file
```
- Users defined in password file (NOTE: Unencrypted text)  

```
[users]
harry = foopassword
sally = barpassword
me:
```
- Username followed by ":" allows access without password, but assigns username to revision commits

## Command overview

- › svnadmin
  - › create
  - › recover
  - › help
- › svn
  - › import
  - › checkout (co)
  - › add, copy, merge, update
  - › commit
  - › list (ls), log
  - › help
- › svnlook
  - › cat, diff
  - › history
  - › help