SENG 371
Software Evolution

Version Control Systems

Prepared by
Pratik Jain

1

Lab Outline

- Introduction to Version Control Systems
- Subversion
- Git and Github

2

Introduction to VCS

- A *version control system* (also known as a Revision Control System) is a repository of files, often the files for the source code of computer programs, with monitored access. Every change made to the source is tracked, along with who made the change, why they made it, and references to problems fixed, or enhancements introduced, by the change.

3

- Our own Version Control System, when we save file using different names to remember what changes we have done.

For ex – pratik_old.c
pratik_new.c
pratik_change.c

If we are smart enough then pratik_V1.c
pratik_V2.c

4

Why Version Control System(VCS)? I

Version control systems are essential for any form of distributed, collaborative development.

- **Backup and Restore.** Files are saved as they are edited, and you can jump to any moment in time.
- **Synchronization.** Lets people share files and stay up-to-date with the latest version.
- **Short-term undo.** Playing with your files is easy, Throw away your changes and go back to the "last known good" version in the database.
- **Long-term undo.** You made a change year ago, and realize later it had a bug. Jump back to the old version.

5

Why Version Control System(VCS)? II

- **Track Changes**. Leave comment every time file is updated explaining why the change happened.
- **Track Ownership.** Name (or id) of each person who made the change is tagged.
- **Sandboxing**. While making a big change, make temporary changes in an isolated area. Test before "checking in" your changes.
- **Branching and merging**. A larger sandbox. You can **branch** a copy of your code into a separate area and modify it in isolation (tracking changes separately). Later, you can **merge** your work back into the common area.

6

Basic Setup

Most version control systems involve the following concepts, though the labels may be different.

- **Repository** : The database storing the files.
- **Server**: The computer storing the repository.
- **Client**: The computer connecting to the repository.
- **Working Set/Working Copy**: Your local directory of files, where you make changes.
- **Trunk/Main**: The primary location for code in the repository. Think of code as a family tree — the trunk is the main line.

Basic Actions I

- **Add**: Put a file into the repository for the first time, i.e. begin tracking it with Version Control.
- **Revision**: What version a file is on (v1, v2, v3, etc.).
- **Head**: The latest revision in the repository.
- **Check out**: Download a file from the repository.
- **Check in**: Upload a file to the repository (if it has changed). The file gets a new revision number, and people can "check out" the latest one.

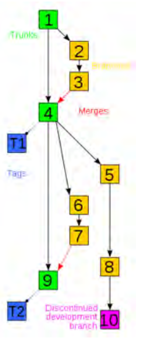
Basic Actions II

- **Checkin Message**: A short message describing what was changed.
- **Changelog/History**: A list of changes made to a file since it was created.
- **Update/Sync**: Synchronize your files with the latest from the repository. This will give you grab latest revisions of all files.
- **Revert**: Throw away your local changes and reload the latest version from the repository.

Advanced Actions I

- **Branch**: Create a separate copy of a file/folder for private use like bug fixing or testing.
- **Diff/Change/Delta**: Finding the differences between two files. Useful for seeing what changed between revisions.
- **Merge (or patch)**: Apply the changes from one file to another, to bring it up-to-date. For example, you can merge features from one branch into another.
- **Conflict**: When pending changes to a file contradict each other (both changes cannot be applied).

Graph Of Revision Controlled Project



Advanced Actions II

- **Resolve**: Fixing the changes that contradict each other and checking in the correct version.
- **Locking**: Taking control of a file so nobody else can edit it until you unlock it. Some version control systems use this to avoid conflicts.
- **Breaking the lock**: Forcibly unlocking a file so you can edit it. It may be needed if someone locks a file and goes on vacation or is not available.
- **Check out for edit**: Checking out an "editable" version of a file. Some VCS have editable files by default, others require an explicit command.

SUMMARY(SO FAR……)

software carpentry

Version Control

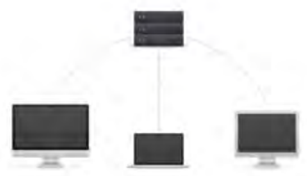
Introduction

Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information

13

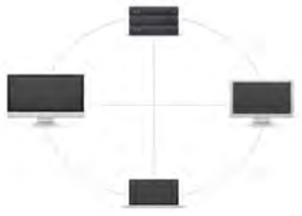
TYPES OF VCS

Client-server model or Centralized- Developers use a shared single repository.



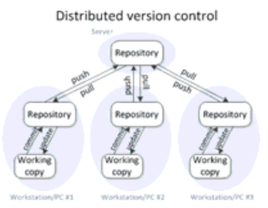
14

- **Distributed version control systems(DVCS) or Decentralized -** Developer works directly with his or her own local repository, and changes are shared between repositories as a separate step.



15

DISTRIBUTED VCS(BETTER EXPLAINATION)



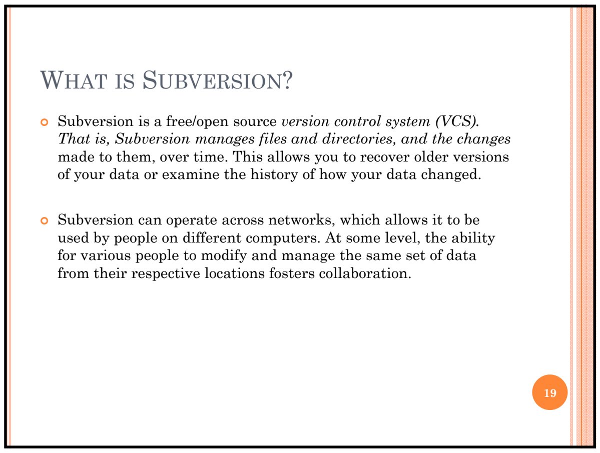
16

Client Server VCS Tools	Distributed VCS Tools
Concurrent Version System(CVS)	Git
Subversion	Mercurial
ClearCase(by IBM)	Bazaar
Visual Source Safe(by microsoft	SVK
	BitKeeper

17

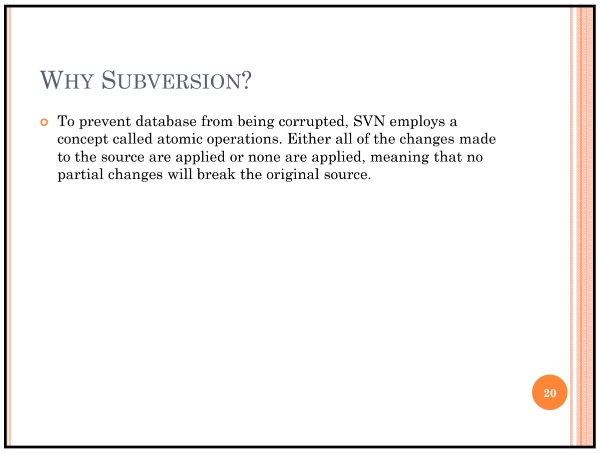
- **CVS :** First Come, first Serve System where users have to publish changes, otherwise there can be a conflict between code of two programmers. The CVS server runs on Unix-like systems, with client software that runs on multiple operating systems.
  - There is no atomic operation support.
  - Moving or renaming files does not include version update.
  - Branch operations are expensive as it is not designed for long-term branching
- **Mercurial:** It's a distributed version control system, Originally made to compete with Git for linux development. Mercurial is implemented in python as opposed to C. Its easier to learn than Git because of some common features of SVN.

18



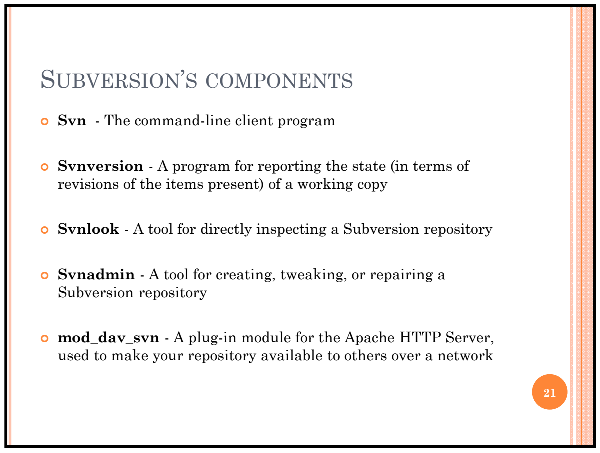
WHAT IS SUBVERSION?

- Subversion is a free/open source *version control system (VCS). That is, Subversion manages files and directories, and the changes* made to them, over time. This allows you to recover older versions of your data or examine the history of how your data changed.
- Subversion can operate across networks, which allows it to be used by people on different computers. At some level, the ability for various people to modify and manage the same set of data from their respective locations fosters collaboration.



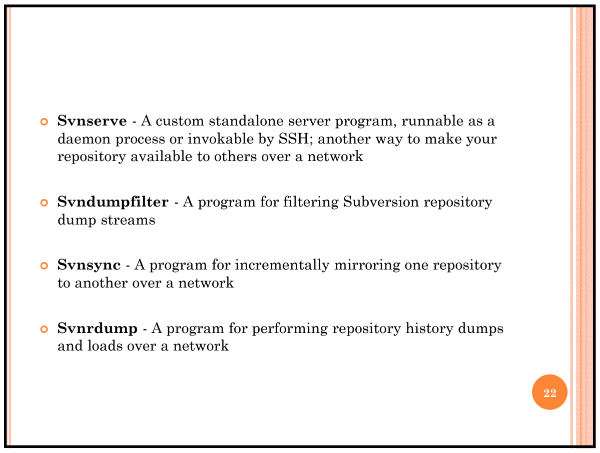
WHY SUBVERSION?

- To prevent database from being corrupted, SVN employs a concept called atomic operations. Either all of the changes made to the source are applied or none are applied, meaning that no partial changes will break the original source.

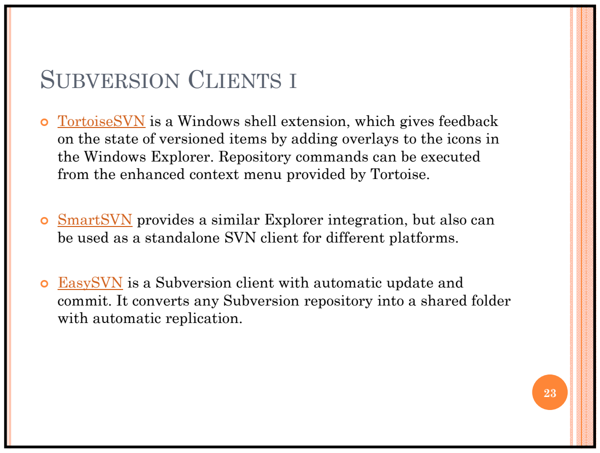


SUBVERSION'S COMPONENTS

- **Svn** - The command-line client program
- **Svnversion** - A program for reporting the state (in terms of revisions of the items present) of a working copy
- **Svnlook** - A tool for directly inspecting a Subversion repository
- **Svnadmin** - A tool for creating, tweaking, or repairing a Subversion repository
- **mod_dav_svn** - A plug-in module for the Apache HTTP Server, used to make your repository available to others over a network

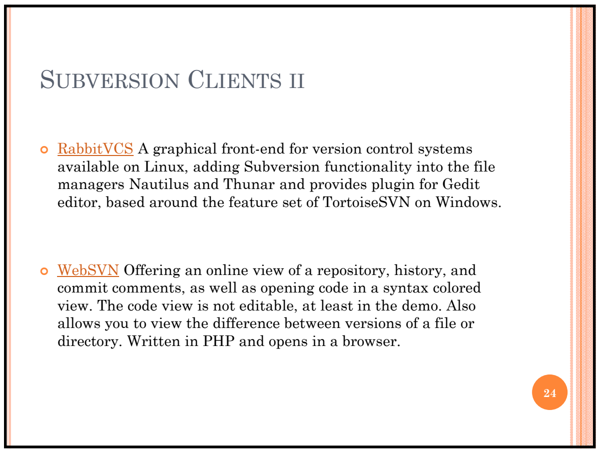


- **Svnserve** - A custom standalone server program, runnable as a daemon process or invokable by SSH; another way to make your repository available to others over a network
- **Svndumpfilter** - A program for filtering Subversion repository dump streams
- **Svnsync** - A program for incrementally mirroring one repository to another over a network
- **Svnrdump** - A program for performing repository history dumps and loads over a network



SUBVERSION CLIENTS I

- TortoiseSVN is a Windows shell extension, which gives feedback on the state of versioned items by adding overlays to the icons in the Windows Explorer. Repository commands can be executed from the enhanced context menu provided by Tortoise.
- SmartSVN provides a similar Explorer integration, but also can be used as a standalone SVN client for different platforms.
- EasySVN is a Subversion client with automatic update and commit. It converts any Subversion repository into a shared folder with automatic replication.



SUBVERSION CLIENTS II

- RabbitVCS A graphical front-end for version control systems available on Linux, adding Subversion functionality into the file managers Nautilus and Thunar and provides plugin for Gedit editor, based around the feature set of TortoiseSVN on Windows.
- WebSVN Offering an online view of a repository, history, and commit comments, as well as opening code in a syntax colored view. The code view is not editable, at least in the demo. Also allows you to view the difference between versions of a file or directory. Written in PHP and opens in a browser.

Summary

software carpentry

Version Control

Basic Operation

25

SVN & GIT Hosting

- http://unfuddle.com/- Widget in Mac OS, popular among software development teams. Free for small project of 2 people.
- http://www.sliksvn.com/- Another option for svn and git hosting but not as good as unfuddle.
- http://beanstalkapp.com/ – Secure, private, reliable, and good interface. Costs $15/month.
- http://code.google.com/hosting/- Best open source hosting, good for small/large team working together.

26

Lab Exercise

- Make a team of 4 people.
- Create a Google code project(Give permissions to other members).
- Select Subversion as a repository.
- Connect to Google code repository.
- Need Access to some SVN client to access repository.
- Use tortoise SVN client to "Checkout" your repository.

27

- Whatever client we are using, we need to enter URL of repository along with username and password in client.
- URL looks like this :- *https://svn<servernumber>.hostname.com/svn/<repository>*
- Right click on root folder, checkout the files from repo. Now create a working folder. You can add files to this folder.
- After checking out the project files, you can edit them from the directory you've created on your local computer.
- After editing files, to commit them, click the check In button in the toolbar to update the changes.

28