



A Brief Overview of CVS

Introduction

CVS is a system that allows people to work simultaneously on groups of files (for instance program sources), and keeps track of revision histories for these files. “CVS” stands for “Concurrent Versioning System”, and it has a long, rich history dating back to the late 1980s.

CVS works by holding a central “repository” of the most recent version of the files. You may at any time create a personal copy of these files by “checking out” the files from the repository into one of your directories. If at a later date newer versions of the files are put in the repository, you can “update” your copy. You may edit your copy of the files freely. If new versions of the files have been put in the repository in the meantime, doing an update merges the changes in the central copy into your copy. When you are satisfied with the changes you have made in your copy of the files, you can “commit” them into the central repository. Figure 1 summarizes the CVS layout:

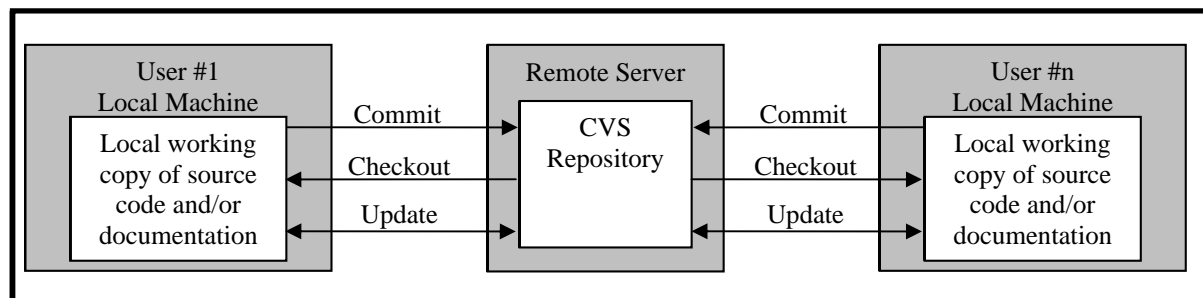


Figure 1: CVS Layout

Benefits of CVS

Benefits of using CVS (or any version control system) include:

- Continuous recorded history of software development: any version of software can be reproduced with little effort and any version can be used as a baseline for development of new branches. Furthermore, differences between revisions can be directly compared.
- Management of system configurations: hardware and software parameters are tracked over time, and differences between subsystems are captured and maintained (for example, hardware-specific parameters of multiple machines).
- Detailed defect tracking: a recorded history is maintained for the identification and resolution of specific problems.
- Support for multiple developers: software development between many developers and across many software versions is automatically handled.
- System test and verification results: CVS can be applied at the system test and verification stages of a project to provide detailed recorded histories of system performance and operational status of specific system revisions.

CVS Terminology and Commands

Below is a list of basic terminology and commands associated with CVS. This is by no means a full list of CVS's features, but it does provide a basic introduction to the more common CVS commands.

- Repository: The directory storing the master copies of the files. The main or master repository is a tree of directories.
- Module: A specific directory (or mini-tree of directories) in the main repository.
- Revision: A numerical or alpha-numerical tag identifying the version of a file.
- Add: Use a CVS add when you want to add a new file to the repository. After performing a CVS add, you must then execute a CVS commit (see below).
- Commit: When your files are ready to be merged back into the repository for the rest of your developers to see, execute CVS commit. You will be able to create a message that describes the



changes that you have made (for future reference). Your changes will then be added to the repository. When you do a commit, if you haven't updated to the most recent version of the files, CVS tells you this; then you have to first update, resolve any possible clashes, and then redo the commit.

- Checkout: CVS checkout pulls a copy of a file from its repository that can be worked on or examined. Checking out files will create a sub-directory module and check-out the files from the repository into the sub-directory for you to work on.
- Update: CVS update refreshes your copy of a module with any changes from the central repository. This action will tell you which files have been updated, and which have been modified by you and not yet committed. It is possible that the changes in the central copy clash with changes you have made in your own local copy. CVS will warn you of any files that contain clashes and these clashes will be highlighted.
- Log: You can execute a CVS log command to get a full description history about any set of files.
- Diff: You can use CVS diff to compare different revisions of files to one another.
- Tag: One of the more powerful features of CVS is its ability to mark all the files in a module at once with a symbolic name. You can create meaningful tag names such as "This tag is for Rev B of the hardware". Because a tag is independent of file history, it allows you to capture a "snapshot" of the entire code base that can later be easily retrieved.

Using CVS

Developers and team members interact with CVS by using a CVS client application which runs on their local machines. There are many different CVS clients to choose from, but any developer can use the CVS client of his or her preference—all CVS clients fundamentally conform to the CVS architecture and will interact correctly with any established CVS repository. Two popular CVS clients are WinCVS and TortoiseCVS, both of which are available free of charge through the GNU license (www.wincvs.org and www.tortoise cvs.org). A third client, SmartCVS (www.syntevo.com), provides an enhanced interface with features that are highly valuable to software developers: remote status, built-in SSH management, and a highly intuitive compare utility (among other features).

WinCVS provides a stand-alone, Explorer-type of interface for interacting with the CVS repository, as shown in Figure 2:

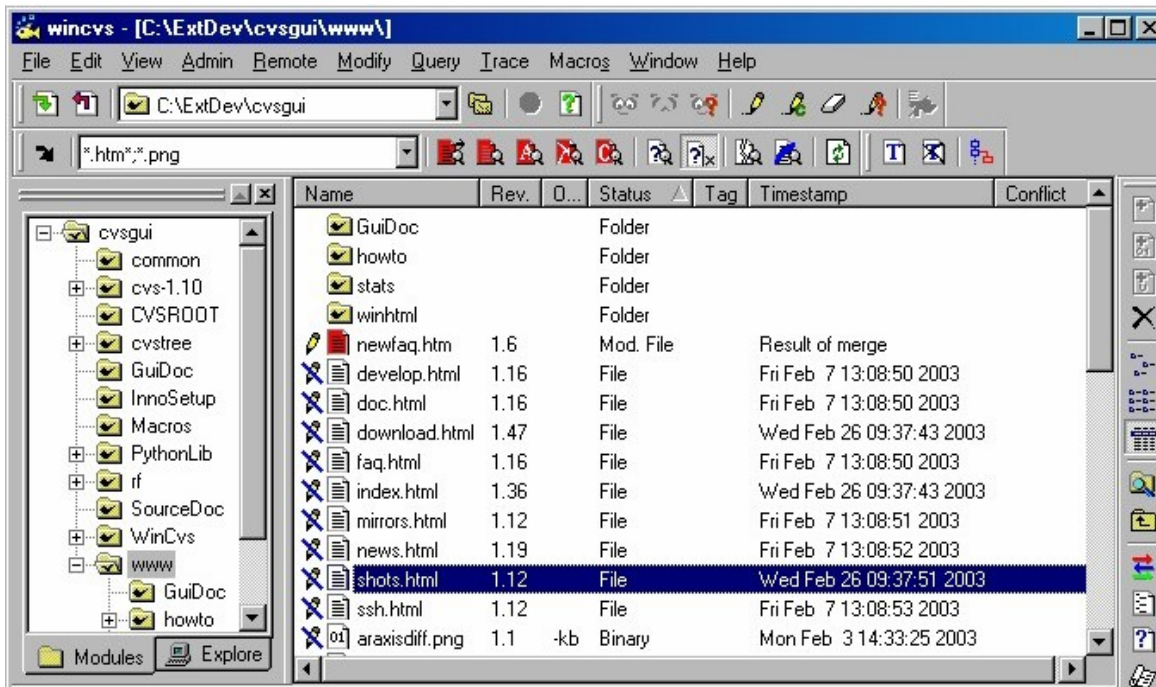


Figure 2: WinCVS Screenshot



TortoiseCVS provides an interface that is integrated with the right-click menus of Windows so that the CVS repository can be accessed through the standard Windows Explorer. A screenshot of TortoiseCVS is shown in Figure 3:

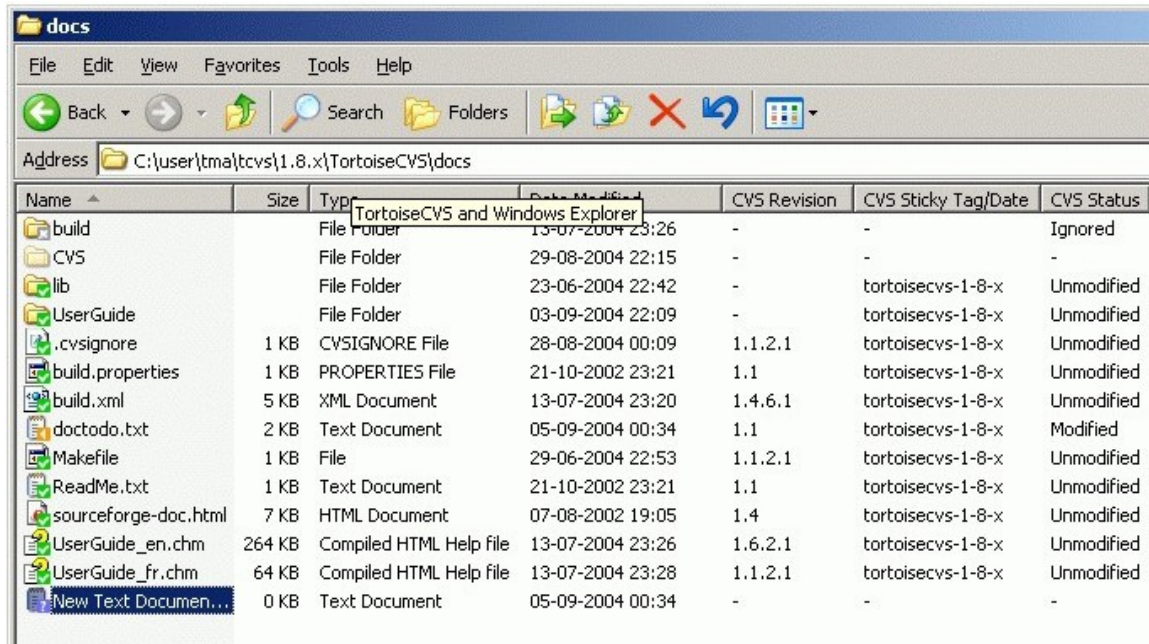


Figure 3: TortoiseCVS Screenshot

Finally, any CVS repository can be accessed through a secure, encrypted connection using Secure Shell (SSH). This method of access ensures proper user authentication and provides network traffic encryption to ensure that sensitive data stored in the repository remains absolutely private and secure.

Summary

Although CVS has traditionally been used to track and maintain software source code, CVS can also be applied to track much more information about any project as a whole, including system test information, system hardware configurations, and documentation. The automatic tracking of revision histories provides a valuable log of software and system development. The ability to retrieve any revision of a project's files provides an easy method for troubleshooting problems and tracking system improvements over time. Finally, the multitude of CVS clients available to the end user means that daily use of CVS can be fairly straightforward.