

Introduction to CVS

UVic SEng 265

Daniel M. German
Department of Computer Science
University of Victoria

3-1 Introduction to CVS

dmgerman@uvic.ca

Concurrent Versions System (CVS)

- ❖ Motivation
 - ❖ We need to synchronize our work with others
 - ❖ We need to know who changes what and when
 - ❖ Source code files constantly change
 - ❖ Obtaining previous versions of software
 - ❖ Constructing patches
- ❖ Unix already has many tools in place to help: diff patch rcs
- ❖ But we would prefer a more integrated tool
- ❖ **Remember:** Unix enables the construction of power programs and commands based on the composition of smaller ones
- ❖ www.cvshome.org

3-2 Introduction to CVS

dmgerman@uvic.ca

CVS

- ❖ A type of configuration management system
- ❖ Configuration management:
 - ❖ The application of technical and administrative direction to the lifecycle of software
- ❖ CVS maintains a collection of files and the history of their modification.
- ❖ For each file it manages it:
 - ❖ maintains of series of changes
 - ❖ it stamps each change with the time it was made, along with userid of the person who made it

3-3 Introduction to CVS

dmgerman@uvic.ca

Why is CVS important?

- ❖ Allows multiple people to work on the same piece of software at the same time
- ❖ Keeps a running log of all changes made to the source tree
- ❖ With this log, reverting back to previous versions is possible
- ❖ Handles synchronization between developers

3-4 Introduction to CVS

dmgerman@uvic.ca

Who uses CVS

- ❖ Used for many open-source projects
- ❖ Enables many and widely distributed programmers to collaborate on code projects
- ❖ Handles changes to code
- ❖ Provides a storage space for important versions and releases of software source
- ❖ Also handles conflict: *what happens when two programmers make changes to the same source file?*
- ❖ For now, we are only interested in:
 - ❖ CVS as storage mechanism
 - ❖ CVS as handler of code change

Module

- ❖ A CVS module is a collection of directories and files under CVS management
 - ❖ CVS adds extra bookkeeping information in the form of additional files amongst the original module directories,
 - ❖ This information is used by CVS to decide what to do when you issue a CVS command
 - ❖ May correspond to a single program, or perhaps many programs
 - ❖ Each module can contain many subdirectories containing source files, documentation, scripts, test files

Modules

- ❖ Because we may have many projects, sometimes it is suitable to have each project correspond to a single module
 - ❖ modules for different CSC/SENG courses
 - ❖ modules for different projects in the same course (when the project is large)
 - ❖ modules for different private projects
- ❖ in CVS, there is no requirement that projects be of a certain minimum size
- ❖ And there is no requirement of what you can put into a module

Working copies

- ❖ When CVS is used to manage a module, there are always, at least, two copies of that module: the **working copy** and the **repository copy**
- ❖ The **working copy** is located somewhere in your home directory
 - ❖ all of your work (editing, compilation, testing, re-editing, etc.) is performed on this working copy
- ❖ The **repository copy** is kept in a safe location (\$CVSROOT)
 - ❖ like a safety deposit box: you keep valuable documents in such a box because they are safer there than in your home
 - ❖ in your home, they could be damaged or destroyed
 - ❖ source code is similar: CVS keeps a copy of module away from our working copy

Using CVS

- ❖ CVS commands are the tools used to move information between the working copy and the CVS repository copy
 - ❖ `cvs add <file or dir>`
CVS begins keeping track of files(s) (files may be added at any time, but we usually need only use add once for a managed file)
 - ❖ `cvs commit <file or dir>`
CVS updates the repository with the changes made to the working copy of <file> (at this point, both copies of <file> are identical)
 - ❖ `cvs update <file or dir>`
CVS reports a summary of the relationship between the working copy of <file> and the repository copy of <file>
 - ❖ `cvs status <file or dir>`
CVS reports more detail of the relationship between the working copy of <file> and the repository copy of <file>

Repository

- ❖ Where the managed copies of the modules are kept
- ❖ A module can be accessed by those who have the proper rights to do it
- ❖ For the purpose of this course, and in the SENG265 repository:
 - ❖ Each student in the class has one module (his/her username)
 - ❖ It can store as many files and directories as the user wants
 - ❖ We are going to use the repository as the medium to submit your assignments

Using CVS

- ❖ In the beginning we usually do not have a working copy of the module
- ❖ The repository copy of the module may already exist
 - ❖ This managed copy must have been created by someone first
 - ❖ The managed copy may also have been maintained by another user
- ❖ If we need to use the module, then:
 - ❖ We ask CVS to checkout the module for us (`cvs checkout <module>`)
 - ❖ This creates a working copy of the module in our directory
- ❖ It is not the same as checking out a book from the library
- ❖ The word checkout is used for historical reasons

CVS...

- ❖ We may now do whatever we want on the working copy (edit source code, re-compile, run tests, etc. using the working copy, create new files, erase them)
- ❖ All of these changes to the working copy do not affect the repository copy until we use CVS commands
- ❖ Examples:
 - ❖ You will checkout and work on a copy of your seng265 module
 - ❖ When working on a large open-source project, you might wish to checkout the source code for the project, which makes a copy of the files in your account

Using CVS



❖ Key concepts:

- ❖ The working copy and repository copy are in different locations (they don't even need to be in the same machine)
- ❖ CVS commands are used to synchronize files amongst these copies

❖ Later in the term we will explore:

- ❖ How to use CVS to report differences between different versions of the same file
- ❖ How to recover earlier versions
- ❖ How to use CVS to annotate versions with information on the changes