

## MMA Computing Memo #3

# CVS Tutorial

B.E. Glendenning

*1999-May-10*

## 1 Introduction

This document supplies a very brief set of instructions on how CVS can be used for revision control of source code, documents, and other files needed for MMA software. It is in no way intended to be a general introduction to CVS or revision control in general. For that, you should go to <http://www.cyclic.com>. You should also have a printed copy of the manual, “Version Management with CVS” for version 1.9 (or greater).

You can use CVS both from the command line and from a GUI “client”, under both Windows and Unix. The command line version only is described at present – the GUI has the same functionality but is more difficult to describe! At present, WinCVS appears to be the best GUI client for Windows.

## 2 Getting Started

The CVSROOT variable is used to tell CVS where the master repository is located.

On Unix in Socorro you can do this merely by giving it a path, *i.e.*, you would set the environment variable to:

```
/home/filehost/mmaswgrp/cvs
```

This is the directory that contains the “master” CVS repository.

From Windows, or over the network, we need to supply additional information – who to log in as, what computer has the archive, and where the files are. While there are a variety of ways to do this, for the MMA we have adopted the following:

```
:pserver:bglenden@mma.nrao.edu:/home/filehost/mmaswgrp/cvs
```

The only thing you should change in the above is your username, i.e. replace `bglenden` with your login name. If you have used this `pserver` method you must now login to CVS:

```
bash-2.02$ cvs login
(Logging in to bglenden@mma.nrao.edu)
CVS password: (Enter password, it will not echo to the screen)
```

If you see an error message, please contact your CVS administrator<sup>1</sup>. Note in particular that if you change your password under Unix, the administrator will have to manually change the CVS password to match it.

CVS works by maintaining a local “slave” copy of the files in the master. The first time you start CVS you probably want to get a copy of the whole archive. You can do so as follows:

```
cd /some/work/directory
cvs checkout .
```

If you weren’t interested in all files in the directory, you could check out, for example, just the “doc” directory via:

```
cvs checkout doc
```

In general, CVS commands work on single files, many files, or recursively on whole directory trees.

### 3 Adding files and directories to CVS

If you want to add a directory or file to CVS, you first need to edit or otherwise create the file where you want it in your “slave” copy. You do this in two stages. First you add the file. This notifies CVS about the

---

<sup>1</sup> At present, Brian Glendenning.

existence of the file, but doesn't otherwise copy it into the "master." When you are ready to copy the files to the master, you then `commit` them. Generally you `commit` the files immediately after adding them, but there is no requirement to do so. For example, suppose we created new `.h` and `.c` files to implement some functionality. We could add them to the repository as follows:

```
cv$ add foo.h foo.c
cv$ commit foo.h foo.c
```

You will be asked to enter a message describing the files when you `commit` them.

A slight complication is that CVS will automatically convert the end of line characters to correspond to the different Unix and Windows conventions. Of course you only want to do this for text files, not for binary files (*e.g.*, Microsoft Word files). You indicate that a file is binary when you add it:

```
add -kb somefile.doc
```

One other thing to keep in mind is that when you add a new top level directory to the CVS repository, you should also add it to the text file `CVSROOT/modules`. This will allow others to more readily find out the existence of the new top-level directory (for example, with the WinCVS "list modules on the server" macro).

## 4 Changing files

If you want to change a file, you should probably first make sure that your "slave" copy is up to date:

```
cv$ update myfile.c
```

Besides updating a particular file, you can update a whole directory tree by typing the name of the directory after `update`.

You then indicate to CVS that you will be changing the file by telling CVS that you will `edit` it. This makes the file writable, and will inform other interested parties that you might be changing the file.

```
cv$ edit myfile.c
```

When you have made and tested all your changes to the file, you can copy the files back into CVS with the `commit` command:

```
cv$ commit myfile.c
```

If you decided you want to abandon your changes, instead of `commit` you `unedit` your changes.

If someone else has made modifications to your files, CVS will lead you through the process of merging your changes with the other changes. This is straightforward with text files, but is generally not possible with binary files. The next section outlines a couple of strategies for dealing with binary files.

## 5 Locking and Watching files

As described above, one generally does not want more than one person modifying a binary file at the same time. The most straightforward way of ensuring this is to lock the file. This is a little bit obscure<sup>2</sup> with CVS:

```
cv$ admin -l somefile.doc
```

When you `commit` your changes the lock will be released. If you decide to abandon your changes, you can unlock it:

```
cv$ admin -u somefile.doc
```

If you have a file locked, nobody else can lock it or commit any changes. It is a good practice to always lock binary files before working on them.

Another option is to be notified (by email) when anyone is working on a file (binary or not). This can be implemented by:

```
cv$ watch add somefile.doc
```

When you are no longer interested in noticing changes to files, you can indicate this with “`watch remove`”.

---

<sup>2</sup> However it is easy with WinCVS. On the button bar it has a key and broken key icon for locking/unlocking a given file.

## **6 WinCVS**

WinCVS is a GUI interface to CVS available from the cyclic web-site (<http://www.cyclic.com>). All the CVS functionality is available in the client. Some of the capabilities are shown on the figure.

Note that when you check out a module, the default behavior of WinCVS (but not CVS) is to remove empty directories. You can fix this behavior by unchecking the “remove (prune) empty directories” preference in the

