# Quick intro to WinCVS

T.S. Norvell
Engr 7893. Memorial Unversity of Newfoundland

August 26, 2002

## 1 What are CVS and WinCVS

CVS is a revision control system. It can track changes to the files in your project and inform you of conflicts created when two developers make changes to the same file. We are running CVS on tera.mun.ca, where it maintains a "repository" containing all source code that you submit to it.

How does CVS help Consider the following scenarios.

- Akbar has just written a class. Jeff needs the this class before he can complete his coding. Akbar could email the code to Jeff and all the other members of the team. This results in a lot of email and a lot of duplicated work as all the team members try to keep their working copies in sync. With CVS, Akbar submits his completed class to the repository and all the other team members can easily synchronize their copies.
- Akbar and Jeff change different parts of file CSoccerDoc.cpp. Akbar could email his copy to Jeff who looks at the differences and produces a file reflecting both changes. Then Jeff emails this merged copy to all other team members. This requires Jeff and Akbar to be aware that they are both working on the same file at the same time. With CVS, Jeff and Akbar both submit their changes to the repository. CVS automatically creates a merged copy.
- Akbar and Jeff change the same part of the same file. In this case when the second one submits his code he will be informed of the conflict and asked to resolve it prior to submitting the code.
- Binky reports a bug in the released version. Akbar and Jeff are midway toward the next release. While they can fix the bug, it will be months before they have a version that is stable enough to release. With CVS they can recreate the source code of the last release version, make the fix there, and send a service pack to their customers. They can also merge the change to the release version into their current version.
- Akbar's hard-disk crashes. With CVS only changes made since the last time he submitted changes to the repository are lost.

WinCVS is a graphical user interface that makes the CVS client easier to use.

## 2 Common procedures

### 2.1 Installing WinCVS:

Skip this step if WinCVS is already installed.

- Obtain WinCVS 1.2 binaries from www.cvsgui.org.
- Unzip into a temporary directory. Execute Setup.exe and follow the instructions. Reboot.
- Optionally make a desktop shortcut and associate a short-cut key with it.

### 2.2 Logging in

- Start WinCVS.
- Select menu item Admin / Preferences...[1]
- In the WinCVS Preferences dialog set CVSROOT to

  *yourUserName*@tera.engr.mun.ca:/user/other/cvs/repos

  where *yourUserName* is replaced by your CCAE Unix login name.
- In the same dialog set Authentication to "passwd" file on the cvs server.
- In the same dialog select the WinCVS tab and set the HOME directory. At MUN, I suggest using M:\ as home. Click Ok to exit the dialog.
- Select menu item Admin / Login... .
- In the Password Authentication dialog, enter your CVS password (given to you by your prof.) and click Enter.

*If you have trouble, check in the log window to see if WinCVS is using the right user name. If it refuses to use the right name, then select menu item* Admin / Command Line*, enter the following command:*

```
cvs -d :pserver:yourUserName@tera.engr.mun.ca:/user/other/cvs/repos login
```

*and click* Ok.

The main effect of logging in is to record your password in the "home" directory you selected.

### 2.3 Logging out

Before shutting down WinCVS, you should log out using menu item Admin / Logout. The main effect of logging out is to erase your password from the client machine's disk.

[1]The first time you start WinCVS, this step is not needed.

## 2.4 Checking out a module

The tree of files on the server that makes up your project is called a "module" in CVS lingo. I have created a module for each student team. The names of the modules are red, green, blue, purple, and orange. Checking the module out makes a "working copy" of the module on the client machine. You only do this once for each working-copy.

- Select menu item Create / Checkout module...
- In the Checkout Settings dialog set the module name to your team's name.
- Select a local folder. M:\ will do.
- On the Globals tab *deselect* Checkout readonly.
- Click Ok to leave the dialog.
- Select menu item View / Refresh

## 2.5 Changes to the working copy

You can make changes to the files and directories now.

After you add any files or directories you should use CVS's Modify / Add selection command to tell CVS about the new files. To delete a file, it is best to use CVS's Modify / Remove selection command. The file will be deleted later. To delete a directory remove all its files. After the next Update, the directory should go away.

## 2.6 Updating

From time to time you will want any changes that others have made to the module reflected in your working copy. To do this you must *update* your local copy. This assumes that you have already checked the module out.

- Select the module in the Modules tab of WinCVS.
- Select menu item Modify / Update selection....
- In the Update Settings dialog, select Create missing directories that exist in the repository.
- Click Ok.

### 2.6.1 Conflicts

A *conflict* in CVS terminology happens when two developers modify the same part of the same file.

If you have made changes that cause a conflict, the update command will tell you. Watch for lines like:

```
C someFileName
```

in the log window. The C stands for conflict. In WinCVS's file window, the file affected will have a Status of Conflict. Editing the file will show you where your changes conflict with others. Using the Query / Log selection command you can find out who made the change. Edit the file to resolve the conflict.

Not all mutually incompatible changes are considered by CVS to be conflicts. For example suppose that a subroutine has been declared with declaration

```
int intpow( int exponent, int base )
```

Developer A and B start the day by updating their working copies. Both start with the same source. Developer A changes the order of the parameters in the declaration and definition. Developer B writes several calls to the subroutine using the original order. A updates (there is no change) and commits (see next section). Now B updates (getting A's changes) and commits. CVS will not notice any problem. However CVS's definition of a conflict is good enough for most purposes.

### 2.7 Committing

From time to time you will want to send changes that you have made to your fellow developers. It is a good idea to do this whenever you have made significant progress that they may benefit from. It is a bad idea to do this when your code does not compile.

- First update your own working copy. Resolve any conflicts.
- Select the module in the Modules tab of WinCVS.
- Select menu item Modify / Commit selection....
- Type in a log message describing all the work you have done since your last commit.
- Click Ok.

## 3 But Wait There's more

The above only scratches the surface of what you can do with CVS and WinCVS. You can inform others of your intention to edit; you can have CVS email you when someone has done a commit; you can recreate earlier versions of the module; you can create branches that represent parallel versions of the module; and so on.

For more information see the CVS manual at

http://www.cvshome.org/

A free book on CVS (but not WinCVS) is available at

http://cvsbook.red-bean.com/

A copy of this book will be on reserve in the library. There is a nice site on WinCVS at

http://www.computas.com/pub/wincvs-howto/

No revision control system will replace communication between developers and careful project organization. Make sure that you all know what the other developers are doing (or at least which parts of the source they are working on).