Version Control with tkCVS

Gerald Brandt
Manitoba HVDC Research Centre

March 24, 1998

Contents

1 - Introduction

Version control software is used to track and manage changes in text based files. In the case of the Manitoba HVDC Research Centre, these files consist of source code (Fortran, C, and C++), and HTML Web pages. In this document, the word *source* is used to describe all text files.

In an *uncontrolled* site where multiple developers have access to edit and contribute to a common source pool, the potential for conflict and problems arises. For example, you may spend a day working on a file, after you've made and saved your changes, another developer may save their changes, and completely overwrite yours.

With the same source pool under version control, the second developer will be will be alerted if a file that they are submitting has been changed by someone else. They will then be asked to merge their work with the work done by the first developer before they re-submit the file.

The version control software that we will be using is called *CVS* (Concurrent Versions Systems). It is a UNIX text based software package that does complete multi-user versioning of text based files. On top of CVS, we will be running *tkCVS*, a graphical front-end to CVS, where you will be doing all of your work.

Your System Administrator will add access to tkCVS for you. From there, you use tkCVS to transfer a working copy of the source repository to your desktop computer. Whenever your work is ready to commit back to the repository, you can do so by simply using the mouse to point and click. If there are conflicts between your work and another developers, you must resolve them before re-committing your work. If there are no conflicts, then CVS will merge the changes you made with what is in the repository.

2 - Starting tkCVS

There are different ways to start tkCVS, depending on what type of computer is on your desktop, and the software installed.

Unix

If your have a Unix computer on your desk, and plan on doing all of your version control from there, the process for starting tkCVS is quite simple. First, rlogin to *everest*. At the prompt type:

```
% xhost + everest
% rlogin everest
% ./startcvs
```

This will execute a shell script in your home directory that will setup the environment variables needed by CVS.

Windows 95 or NT

If you have a Windows computer on your desk, and plan on doing all of your version control from there, then you have two methods of starting tkCVS, depending on the software installed on your computer.

Exceed

If you have the Exceed X Window software from Hummingbird, you can add a 'Version Control' icon to your desktop. This will automatically launch tkCVS and display it on your screen. If you do not have an icon for version control, see your System Administrator.

VNC

If you have the VNC software installed, you will have noticed a small window asking you for your password when you logged into Windows (this is after the login, and does not refer to the Windows login screen). After typing in your password, the taskbar at the bottom of the Windows screen will show a box labeled VNCViewer. Click on this box to open its window. In this window you will see a simple XWindow screen, with a version control icon on it. Click on this icon to start tkCVS.

3 - The Main tkCVS Screen

After starting tkCVS, you will see a window similar to the one shown below (figure 1). The Current Directory will be your Unix home directory (for example /home/gbr). From here, you can simply follow

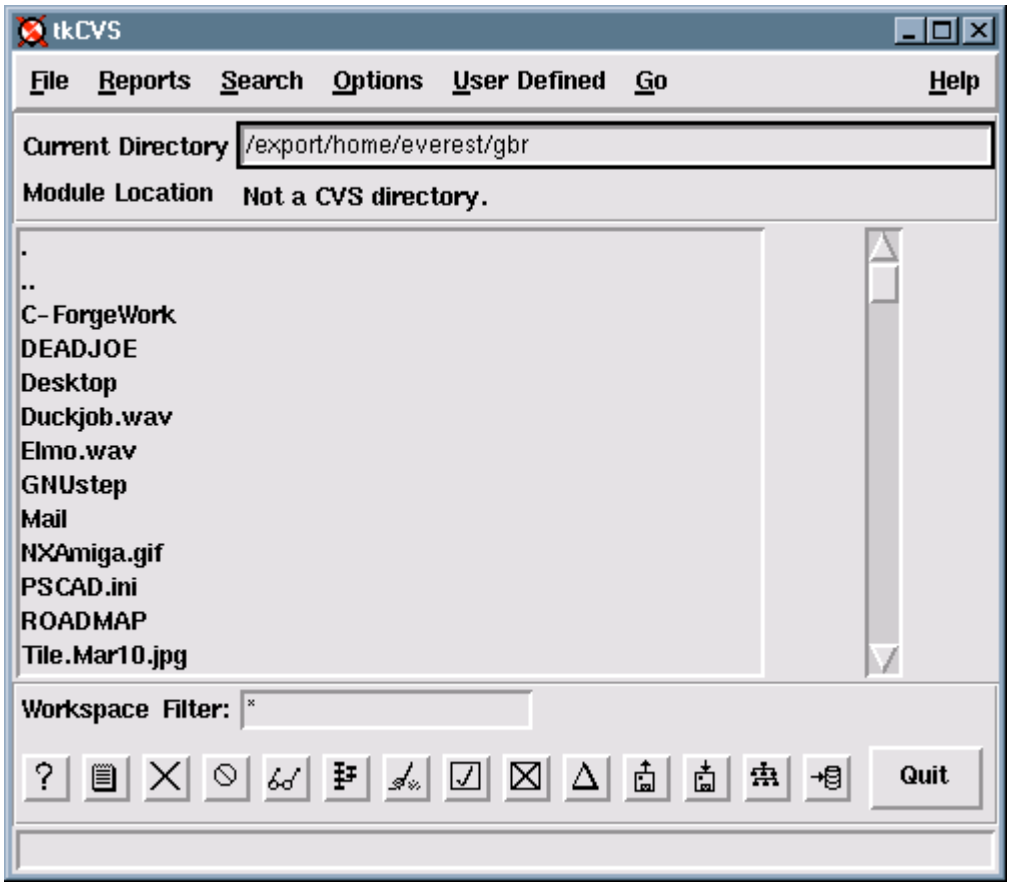

Figure 1. Main tkCVS Screen

the directory listing to where your source files are located. If you are in a directory that is not under CVS control, tkCVS will report that fact, and will only display the names of files and directories that are in the Current Directory. Once you are in a version controlled directory, the tkCVS display will change to indicate exactly what each file is, and where the module is located (figure 2). Taking a look at figure two, we see a second column added to the display. This column list the status of the file as compared to what is in the current repository. The following is a list of what could be displayed in the second column.

< dir >	This is a sub-directory
Ok	the file in your source directory is exactly the same as the one in the CVS controlled repository.
{Needs Patch}	the file in your source directory is older then the one in the repository, and needs to be updated.
{Needs Merge}	The file in your source directory has been modified by you, and the file in the source repository has been changed by another developer. CVS will automatically attempt to merge the two files together.
{Locally Modified}	The file in your source directory has been modified by you, and needs to be placed back into the repository.
{File had Conflicts}	This is a *secondary* state. The file in your source directory was modified, as was the one in the repository. CVS attempted to do the merge, but was unable to resolve some conflicts. You must resolve the conflicts before re-submitting the file.
????	CVS has no knowledge of this file, it is NOT under CVS control. You will usually see this on backup files and object modules, which should not be under version control anyway.
{Locally Added}	You added a new file from your directory into the repository, but have not yet checked it in.
{Locally Removed}	You deleted a file from the repository, but have not yet checked in your deletion.

TkCVS has a small quirk (I can't decide if it's a bug or a feature). That is, if a file has conflicts, then all files listed below it in the tkCVS main window will have a status of ***????***. Once the conflicts are resolved, the display will correct itself.

4 - Editing With Version Controlled Files

Once you have a directory that is under CVS control, there is nothing special that you need to do or change about your working habits—just edit and create files as you normally would. Use whatever editor and creation tools that you are comfortable with. There are a few precautions you must observe to make sure that the repository is usable on all platforms.

CVS Folders: CVS creates a new directory called 'CVS' in every directory that is under its control. This directory contains important version control information and is automatically generated. You should *never* remove or edit the directories, or the files contained in them.

No Symbolic Links: Unix allows users to create symbolic or soft-links between files and directories. CVS *cannot* handle this. Do not create symbolic links in your working directory. If you do, and then attempt to put them under CVS control, they will be silently ignored.

Naming Conventions: Since the files may be shared across multiple platforms, file naming conventions must be limited to to a format that all platforms understand. Since Windows is not case sensitive (ie: 'a' == 'A'), you should attempt to keep all file and directory names in lower case. Also, some older versions of Unix do not allow very large file names, so keeping the names under 32 characters is a good idea.

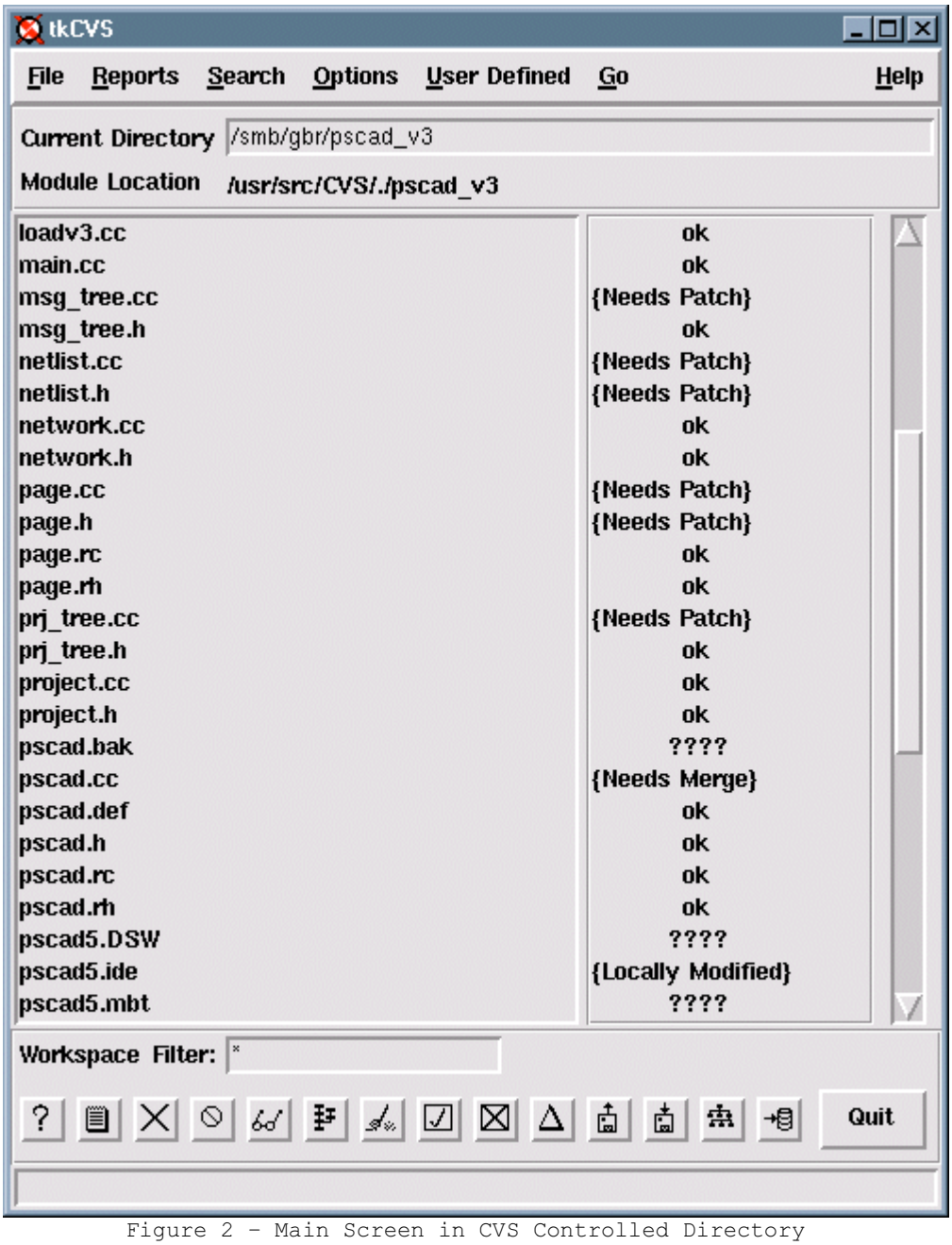

Figure 2 - Main Screen in CVS Controlled Directory

5 - Checking Your Work into the Repository

When you are finished editing, ***test your work by compiling and running*** before you place your work back in the repository. The repository should only contain complete and tested code. Code that is incomplete or does not compile does not belong in the repository.

Figure 3 - Toolbar Check In Button

Figure 3 shows the toolbar button that performs check-ins. If a file has been ***locally modified, locally added,*** or ***locally removed*** then this button does the check-in.

Individual Files

Simply highlight the file to check in by clicking on its name once, and then click on the Check-In button. TkCVS will display the Commit Changes window (see figure 4). The Bug Tracking # field is used to update the Bug Tracking database system. The bug tracking software has not yet been implemented, so this field should be filled with the text 'no-bug' (less quotes). The comment field will hold your comments on what changes were done to the file, and why. It is important that this field truly indicates what changes were made.

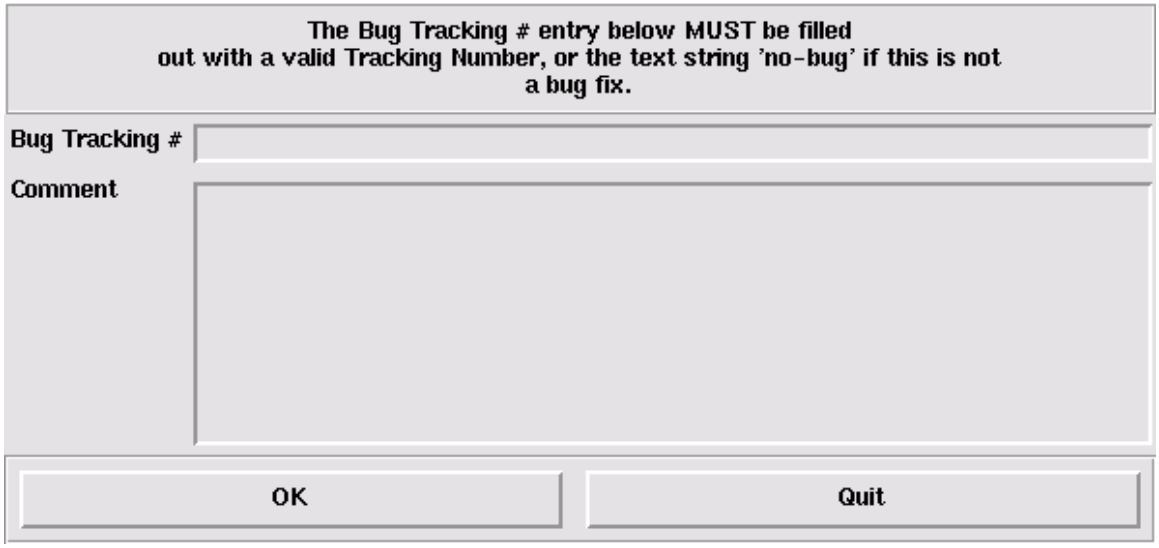

Figure 4 - Commit Changes Window

Multiple Files

Highlighting multiple files for check in is a simple matter of holding down the 'control' or 'shift' keys while clicking on a file name. Holding down the 'control' key will allow you to select multiple individual files, while holding down the 'shift' key will allow you to select a range of files. ***Note: all files selected will be stamped with the same bug-id number and comment. Make sure that this is what you want to do!***

All Modified Files

When you first enter a directory, no files are highlighted. If you click on the Check In button when no files are highlighted, **ALL** files that have changed will be checked in, this includes all sub-directories. TkCVS will confirm that this is what you want to do. ***Note: all files selected will be stamped with the same bug-id number and comment. Make sure that this is what you want to do!***

After you have entered all of your comments, press the okay key to update the file in the repository. TkCVS will confirm your request, and then proceed with the check in. After the check in is complete, a CVS Commit window will open displaying what actions were taken, and new file revision numbers.

6 - Updating Your Files from the Repository

When you need to update your local files from the repository, click on the Update button (figure 6) on the

toolbar.

Figure 6 – ToolBar Update Button

If a file is marked as ***needs patch*** or ***needs merge***, then this is the button that does the job.

Individual Files

To update an individual file with a new version from the repository, simply click on the filename of the file, and then click on the Update button. TkCVS will confirm that you want to update this file, and then perform the update if you select OK. A 'CVS Update Output' window will then open and display the results of the update. If the file was updated cleanly, the window will display an uppercase U followed by the filename. If the update resulted in a merge, the window will display the merge operation, and the results of the merge. See the section on merging.

Multiple Files

Multiple files can be selected in the same fashion as described in in the Check In procedure. TkCVS will again confirm that this is what you want, listing all of the file names in the window. After the update has been perfromed, the 'CVS Update Output' windows will open, showing the results of the update on all of the files selected.

All Files

If you would like to update all of the files in your source directory, including all sub-directories, follow the procedures listed in the Check In section, and then press the update button. Again, a confirmation and an Output window will me displayed.

7 - Merging Your Files with the Repositories

The merging of files is performed in your source directory. The repository is never changed until the merge is performed cleanly. In order to merge a file you must first update the file from the repository (see previous section). This will attempt to merge your local changes with the changes to the file in the repository. If the merge is successful, the status of the file will change from ***needs merge*** to ***locally modified.***

File had Conflicts

If the merge was unsuccessful, the status will change from ***needs merge*** to ***file had conflicts***. You must now manually resolve the conflicts before you can check in the file. There are two methods to resolve the conflicts. One is text based, and performed by you in the editor, while the other attempts to help with the merge by showing you the conflicts and performing the merge depending on your suggestions.

Manual Text based Merge

When CVS cannot merge two files together by itself, manual intervention is needed. CVS helps with the merge by creating a new copy of the file in conflict. For example, if you had a file named 'class.cpp', and it needed manual merging, CVS would create a special version of 'class.cpp' with both versions of the file in it. If you open the file in your editor, you will find special markers placed in the file to show you where the conflicts are. The beginning of the conflict (in our example file 'class.cpp') would look like this

<<<<<<<< class.cpp

All of the code following this line contains your changes to the code that were in conflict with the code in the repository. After your code, you will see another marker made up of eight equal signs.

========

Below this marker is the code from the repository file. The end-of-conflict marker then follows this code, and also displays the version number of the file in the repository.

>>>>>>>> 1.2

You must edit this file until your changes, and any changes made by other developers all work smoothly together.

Partially Automated Merge

If the conflicts created are not overly complicated, you can attempt to merge them using a graphical interface. To start the merge, select the file that has conflicts. Then go to the 'User Defined' menu and select merge. A window will open displaying the file as it is in the repository on the left hand side, your version of the file on the right hand side, and what the file will look like after the merge on the bottom (see figure 6). The text in blue in the upper windows shows the conflict, while the text in yellow in the bottom window shows the results of your change request. On the bottom of the window are several buttons. These buttons are:

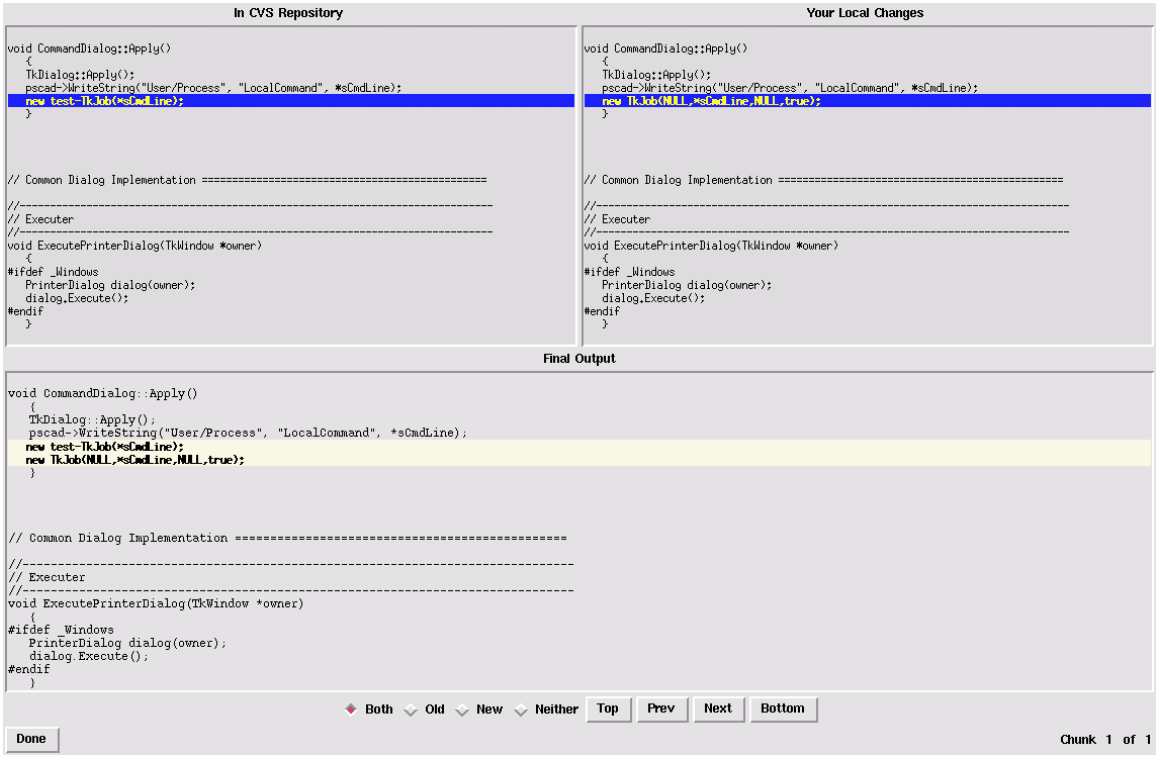

Figure 7 – tkMerge Window

- Done closes the window and save any changes that you made.
- Both keep both sets of changes in the file. This also keeps the special markers in the file.
- Old keeps the changes in the repository, and discards yours.
- New keeps your changes and discards those in the repository.
- Neither throws away both changes
- Top, Prev, Next, Bottom move between the conflicts

The text in the bottom left hand corner indicates how many conflicts there are, and which one you are looking at.

8 - Viewing Differences Between Your File and the Repository

To view differences between your files and the files in the repository, simply highlight the file that you want to compare and click on the 'see differences' button on the tool bar (figure 8).

Figure 8 – See Differences Button

This will bring up the tkDiff window (see figure 9). This window is fairly self-explanatory. The right view displays the current file in the repository, and the left one displays your file. The display at the bottom shows how many differences there are in the files, and the buttons on the right allow you to move between the differences. No editing is allowed from this window.

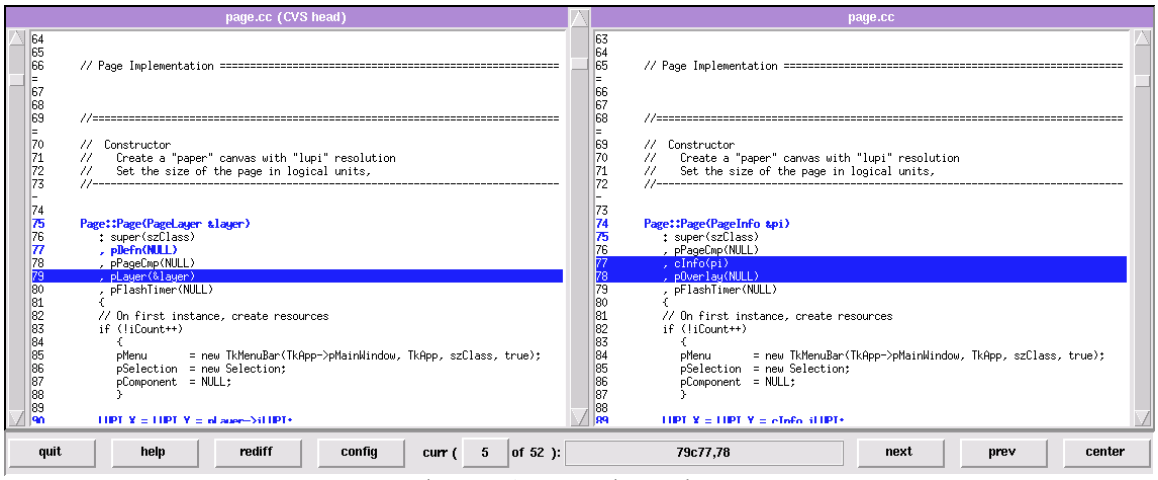

Figure 9 - tkDiff Window

9 – Adding and Removing Files in the Repository

As the project continues, new files are usually added to the repository. Selecting the file that needs to be added, and then clicking on the Add button in the toolbar (figure 10) accomplishes this. The file status will change from ***????*** to ***file locally added***, and must then be checked in to complete the process. Removing a file is done is a similar fashion, select the file, and then click on the delete button (figure 10). The status of the file will change to ***file locally deleted***, and must then be checked in to complete the process.

Figure 10 - the Add (top) and Remove (bottom) buttons

10 – Viewing the Repository

There are times when you may want to see the revisions that a particular file has gone through. To do this, you highlight the file to view, and then click on the Revision Log button in the toolbar (figure 11).

Figure 11 – Revision Log Button

This will bring up the Log Browser window (figure 12). This window has three sections in it. The top one, labeled Version A, displays information on a single revision of the file. The file to display is selected by clicking with the right mouse button on a revision in the lower box.

The second one, labeled Version B, displays information on a single revision of the file. The file to display is selected by clicking with the left mouse button on a revision in the lower box.

For the everyday user of tkCVS, the buttons in the bottom of the window are straightforward. Normal users of tkCVS will NOT be using the 'Merge Branch to head' or the 'Merge Changes to Head' buttons. These are for use by the CVS repository Administrator (see Policy section).

11 – Summary

At times you may want tkCVS to re-scan the current directory. This can be accomplished by clicking on the 'Re-read current directory' button in the toolbar.

This covers the basic functions of tkCVS for everyday use. There are many functions that were not covered in this manual, mainly due their obscure status. You should feel free to experiment, as long as you stay away from sensitive areas. These are areas which only the repository administrator should go near. Areas that a normal user should stay away from are:

- Anything to do with importing
- Anything to do with checking out entire modules
- Anything to do with tags
- Anything that works on entire modules instead of files
- The menu 'User Defined' → 'Update_to_Head'
- The two toolbar buttons just to the left of the quit button.

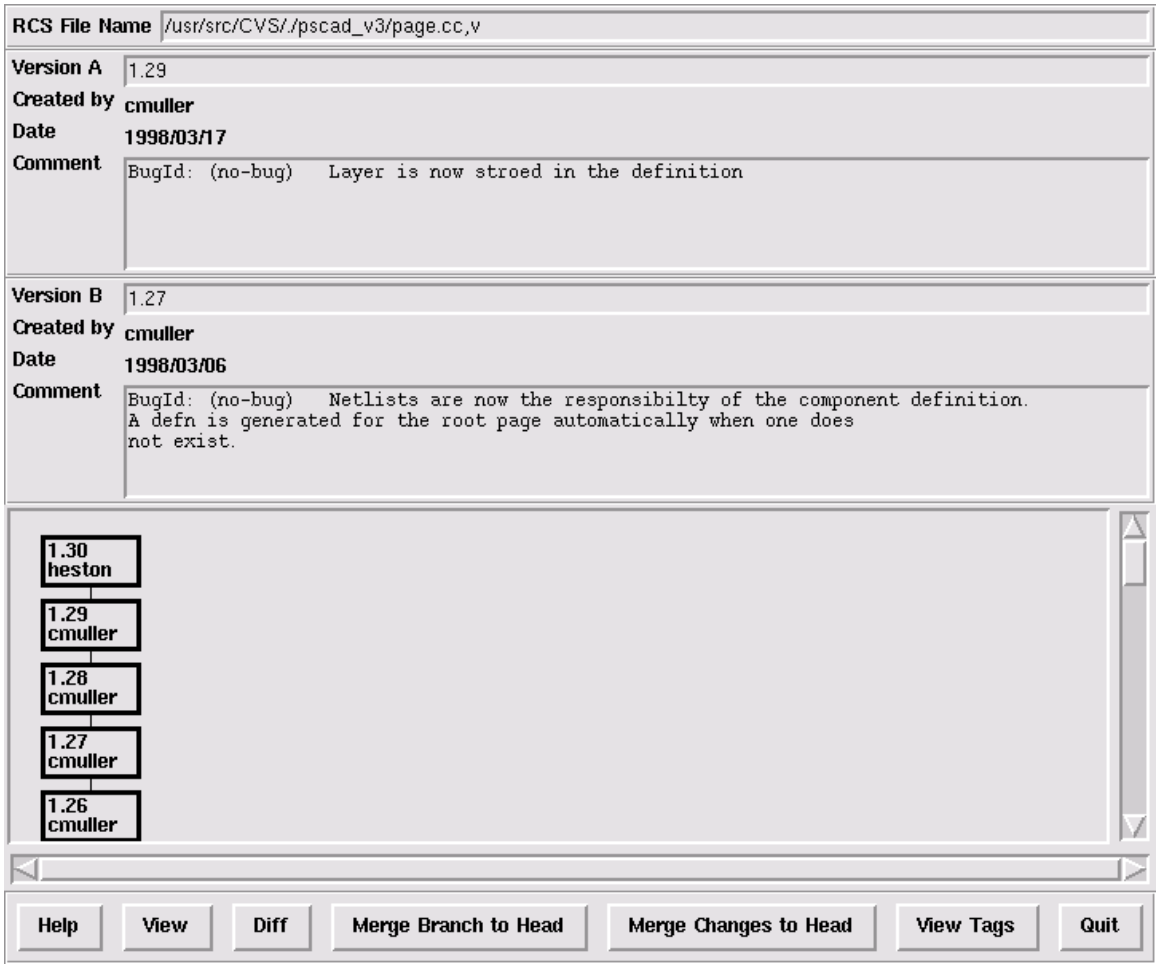

Figure 12 - Log Browser

12 – Policy

CVS is a Version Control software program, not a Code Management program. What this means is that CVS will maintain and give access to all versions of the source files it has control over. It does not insist that everything in the repository must compile, or know that simultaneous changes to various source files may conflict with each other logically. CVS's concept of a conflict is purely textual based. Because of this, there are several policies which should be followed to help CVS keep track of everything.

1. One person from the development group should be made the Repository Administrator. This person will have extra responsibilities in order to maintain the repository. As far as tkCVS is concerned, one user is the same as another, so it is up to the development group to maintain the Administrator concept.
2. The entire project should be tested and compiled cleanly **BEFORE** it is checked into the repository. Once a clean compile and run is achieved, a check in can occur. This is a process that may take several iterations. For example, if you update your files with all the changes in the repository, fix all conflicts, and get a clean compile and run, you MUST repeat the update process to verify that another developer has not checked in their version of the files. There is no chance of losing the other developers work, but if your work conflicts with theirs, you may have files in the repository that do not compile and run.
3. CVS does not replace developer communication. Some conflicts may need to be resolved with cooperation from other developers.

4. CVS does not have change control. Change control can refer to a number of things. It can mean *bug-tracking*, that is keeping a database of reported bugs and the status of each one in a database. The HVDC version of tkCVS asks for a Bug-Id number during check in, but this ties into a separate bug tracking system, which is not yet functional.
   Another aspect of change control is keeping track of the fact that changes to several files were in fact changed as one logical change. If you submit changes to multiple files in one check in, CVS promptly forgets that that those files were checked in together. The only thing that that ties them together is the fact that all the files will have the same log message.

Repository Administrator

The administrator has several duties, including creating the releases for software in the repository. Releases are essentially completed code that is available to members outside of the development team for the purposes of bug testing or final releases of products. These releases must be made in a special way, and the repository should be tagged to indicate where a release was made. This document does not cover these tasks.

13 – Windows NT Peculiarities

The file sharing between Linux and Windows NT has some flaws. Everytime you access your local files (on your NT system) through tkCVS, the date and time stamps on your files will change to either some future, or past data (i.e.: 2014 or 1942). So, it is best to stay out of tkCVS until you are ready to do all of your version controlling stuff. Users with Borland C have a 'touch' command to bring the dates back in sync. Visual C users will need to get a version of 'touch' from Gerald or Garth.

14 - References

CVS Version Control for Inter@active Consulting Group W³ Site Projects
Sean Dreilinger
December 29, 1997
http://www.interactive.com

Version management with CVS for CVS 1.9.16
Per Cederqvist et al
Distributed with CVS