VS−RCS−HOWTO Document for Linux (Source Code Control Sy

Table of Contents

Table of Contents

CVS−RCS−HOWTO Document for Linux (Source Code Control System)

Al Dev (Alavoor Vasudevan) alavoor[AT]yahoo.com

v22.9, 28 March 2003

This document is a "practical guide" to very quickly setup CVS/RCS source code control system. This document has custom shell scripts that are wrappers on top of CVS. These scripts provide an easy user interface for CVS. Several shell scripts are provided to make RCS easier to use. The information in this document applies to Linux and as well as to all other flavors of Unix like Solaris, HPUX, AIX, SCO, Sinix, BSD, SCO, Apple Macintosh (which is BSD unix) etc.. and BeOS.

20. Copyright and License

1. Introduction

(The latest version of this document is at http://www.milkywaygalaxy.freeservers.com. You may want to check there for changes).

A source code control system is a MUST to manage the changes occurring to a software project during development. Developers need a complete history of changes to backtrack to previous versions in case of any problems. Since source code is the most vital component of any software project and software development takes a huge amount of time and money, it is very important to spend some time in ***safe−guarding*** the source code by using source code control systems like CVS and RCS.

CVS (Concurrent Version Control System) is a powerful tool which allows concurrent development of software by multiple users. It uses RCS underneath and has an application layer interface as a wrapper on top of RCS.

CVS can record the history of your files (usually, but not always, source code). CVS only stores the differences between versions, instead of every version of every file you've created. CVS also keeps a log of who, when and why changes occurred, among other aspects.

CVS is very helpful for managing releases and controlling the concurrent editing of source files among multiple authors. Instead of providing version control for a collection of files in a single directory, CVS provides version control for a hierarchical collection of directories consisting of revision controlled files.

These directories and files can then be combined to form a software release.

CVS can be used for storing "C", "C++", Java, Perl, HTML and other files.

HISTORY of CVS: CVS is a very highly sophisticated and complex system. It is the ***"State of the Art"*** technology and is so called *"software miracle"*. The CVS software is a very advanced and capable system developed over a very long period of time. And it took several years to mature!! It took about 20 to 30 years of research to develop CVS algorithms and later it was coded into a software. And even today, it is still evolving!!

CVS algorithms actually started in Universities several decades ago and CVS implementation started out as a bunch of shell scripts written by Dick Grune, who posted it to the newsgroup comp.sources.unix in the volume 6 release of **December, 1986**. While no actual code from these shell scripts is present in the current version of CVS much of the CVS conflict resolution algorithms come from them. In April, 1989, Brian Berliner designed and coded CVS. Jeff Polk later helped Brian with the design of the CVS module and vendor branch support.

And today each and every major software development project in the world is written using CVS as the safe repository. As good old software hats say − *"You are in very safe hands, if you are using CVS !!!"*

2. Which One Is for Me? CVS or RCS

CVS actually uses RCS underneath. CVS is a lot more powerful tool and can control a complete source code tree. It is ***very strongly*** recommended that you use CVS, because you can greatly customize CVS with scripting languages like PERL, Korn and bash shells. See the sample korn shell scripts at Shell Scripts .

Advantages of CVS:

- CVS is decentralized so a user checks out files/directories from the repository and have his own separate stable source directory tree.
- CVS can "STAMP" releases of an entire project source tree.
- CVS can enable concurrent editing of files.
- CVS can be greatly customized to enable strong locking of files via shell scripts or PERL scripts. CVS supports weak locking with the command 'cvs watches' and also no locking permitting concurrent editing of files.

Disadvantages of CVS:

- Needs a little more administration than RCS.
- Very highly sophisticated and complex system. It is "State of the Art" technology. The cvs software is a very advanced and capable system developed over a very long period of time (it took several years!!). It took about 20 to 30 years of research to develop CVS and it is still evolving!!
- Has a large number of commands and command options, hence a steeper learning curve for beginners. The shell scripts at Shell Scripts can ease usage.

Advantages of RCS:

- RCS is very simple to setup, with less administrative work.
- RCS is used in a centralized area where everyone works.
- RCS is useful for simple systems.
- Very strong locking of files − concurrency eliminated.

Downside of RCS:

- Concurrent development by multiple developers is not possible due to file locking and being limited to a single working directory. Because of the single working directory limitation, changes to files by multiple developers can cause failure of the 'make' command.
- Cannot stamp releases of an entire software project.

This document also has shell scripts which provide simple commands to check−out, check−in, and commit files. See shell scripts at Shell Scripts

For RCS see the RCS mini−howto on the Linux cdrom:

```
cd /mnt/cdrom/Redhat/RPMS
ls −l howto−6.0−*.noarch.rpm
rpm −qpl howto−6* | grep −i rcs
```

or visit http://www.LinuxDoc.org/HOWTO/mini/RCS.html

See also the RCS shell scripts at rcs_scripts

3. Setting up CVS

First you need to install the CVS package. On Redhat Linux use:

```
cd /mnt/cdrom/Redhat/RPMS
rpm −i rcs*.rpm
rpm −i cvs*.rpm
rpm −i openssh*.rpm
To see the list of files installed do −
rpm −qpl cvs*.rpm | less
```

and browse the output using j,k, CTRL+f, CTRL+D, CTRL+B, CTRL+U or using arrow keys, page up/down keys. See 'man less'.

The Openssh is required if you want to use ssh (Secure Shell) with CVS.

On other flavors of Unix, you may need to download the RCS and CVS tar balls and follow the README, INSTALL files to setup CVS. Visit http://www.cyclic.com and http://www.loria.fr/~molli/cvs−index.html

3.1 Environment variables

The following environment variables need to be setup in /etc/profile − default values required for all users. If not set in /etc/profile, then you should add these to your local profile file /.bash_profile.

```
export EDITOR=/bin/vi
export CVSROOT=/home/cvsroot

# WARNING!! WARNING: If you set CVSREAD to yes, checkout and update will try hard to
# make the files in your working directory read−only.  When this is not set,
# the default behavior is to permit modification of your working files.
#export CVSREAD=yes
```

And of course, individual users can *override* the environment variables set in /etc/profile by resetting them in their local profile file /.bash_profile

```
# File ~/.bash_profile
# Overriding env variables by resetting
export EDITOR=/usr/bin/emacs
export CVSROOT=/home/someotherdir/java/cvsroot
```

Create a directory to store the source code repository and give read, write access to Unix group/user. Also make sure that the directory name of CVSROOT does not contain any blank spaces. For example CVSROOT should not be like '/home/my rootcvs'.

```
bash$ su − root
bash# export CVSROOT=/home/cvsroot
bash# groupadd −−help
bash# groupadd cvs
bash# useradd −−help
bash# useradd −g cvs −d $CVSROOT cvs
bash# mkdir $CVSROOT
```

```
bash# ls −ld $CVSROOT   ... (you should see the listing)
bash# chgrp −R cvs $CVSROOT
bash# chmod o−rwx $CVSROOT
bash# chmod ug+rwx $CVSROOT

# To initialize the CVS repository and to put in source code files
# do (but requires env CVSROOT to be set) :
bash# cvs init

# Add the unix users to the cvs group. Create supplementary groups for users.
# Note that you MUST not put any blank spaces after comma separating the
# group names in −G option.
# In example below user tom belongs to groups cvs, users and staff and user
# johnson belongs to group cvs only.
bash# usermod −−help
bash# usermod −G cvs some_unix_username
bash# usermod −G cvs,users,staff tom
bash# usermod −G cvs,users,staroffice billclinton
bash# usermod −G cvs johnson
bash# exit    .... (logout of root superuser mode)

# Login as a user and import files into cvs....
bash$ su − billclinton
bash$ export EDITOR=/bin/vi
bash$ export CVSROOT=/home/cvsroot

# WARNING! WARNING: If you set CVSREAD to yes, checkout and update will try hard to
# make the files in your working directory read−only.  When this is not set,
# the default behavior is to permit modification of your working files.
bash$ export CVSREAD=yes

# Change directory is a must (MANDATORY)
bash$ cd $HOME/somedir/anotherdir/directory/my_source_code_dir

# Must give vendor tag and revision tag
cvs import somedir/anotherdir/directory/my_source_code_dir vendor_1_0 rev_1_0

# Also note that it is very important to give the directory tree starting
# from the $HOME, that is, in above example starting from somedir.
# For example I did:
bash$ cd $HOME/howto/foobar
bash$ cvs import howto/foobar vendor_1_0 rev_1_0

# Another example is:
bash$ cd $HOME/javafilesdir
bash$ cvs import javafilesdir vendor_1_0 rev_1_0

# A sample testing and verification:
bash$ cd $HOME/howto/foobar
bash$ cvs checkout myfoo.java
```

TROUBLESHOOTING: When doing checkout it says module is unknown. It is a common mistake not to change directory while doing cvs import. You ***MUST change directory*** to the source−code−directory and then do cvs import. For example:

```
bash$ cd $HOME/somedirectory/foobardir
bash$ cvs import somedirectory/foobardir vendor_1_0 rev_1_0
```

3.2 Setup CVS on Client Box

On client boxes where you want to use the CVS, you should install cvs packages and ssh package (if you want to use ssh). Setup the environment variables:

```
bash$ export CVSROOT=":ext:developer@cvs_server_box.domain.com:/home/cvsroot"
bash$ export CVS_RSH="ssh"
```

The cvs_server_box.domain.com is the IP address of the remote CVS repository server and 'developer' is the user id. Another example using pserver is given below:

```
bash$ export CVSROOT=:pserver:username@cvs.tldp.org:/cvsroot
bash$ export CVS_RSH="ssh"
```

See also multiuser .

3.3 Migrate RCS to CVS

To migrate the existing RCS files to CVS, use the following script from downloadsoftware . Make sure that you installed the Korn shell package pdksh*.rpm from the Linux contrib cdrom.

NOTE : ***Get the Korn shell /bin/ksh by installing pdksh*.rpm from the Linux contrib cdrom***

Now the RCS is migrated to CVS as 'project'. You can start using the CVS commands on module 'project'.

4. Intro to CVS Commands

CVS provides a rich variety of commands (cvs_command in the Synopsis), each of which often has a wealth of options, to satisfy the many needs of source management in distributed environments. However, you don't have to master every detail to do useful work with CVS; in fact, five commands are sufficient to use (and contribute to) the source repository. The most commonly used CVS commands are: **checkout**, **update**, **add**, **remove**, **commit** and **diff**.

4.1 checkout

cvs checkout modules... A necessary preliminary for most CVS work: creates your private copy of the source for modules (named collections of source; you can also use a path relative to the source repository here). You can work with this copy without interfering with others' work. At least one subdirectory level is always created.

```
bash$ cvs −−help checkout
Usage:
  cvs checkout [−ANPRcflnps] [−r rev | −D date] [−d dir]
    [−j rev1] [−j rev2] [−k kopt] modules...
        −A      Reset any sticky tags/date/kopts.
        −N      Don't shorten module paths if −d specified.
        −P      Prune empty directories.
        −R      Process directories recursively.
        −c      "cat" the module database.
        −f      Force a head revision match if tag/date not found.
```

```
        −l      Local directory only, not recursive
        −n      Do not run module program (if any).
        −p      Check out files to standard output (avoids stickiness).
        −s      Like −c, but include module status.
        −r rev  Check out revision or tag. (implies −P) (is sticky)
        −D date Check out revisions as of date. (implies −P) (is sticky)
        −d dir  Check out into dir instead of module name.
        −k kopt Use RCS kopt −k option on checkout.
        −j rev  Merge in changes made between current revision and rev.
(Specify the −−help global option for a list of other help options)
```

4.2 Staying in sync with other developers − 'cvs update'

cvs update Execute this command from within your private source directory when you wish to update your copies of source files from changes that other developers have made to the source in the repository.

```
bash$ cvs −−help update
Usage: cvs update [−APdflRp] [−k kopt] [−r rev|−D date] [−j rev]
    [−I ign] [−W spec] [files...]
        −A      Reset any sticky tags/date/kopts.
        −P      Prune empty directories.
        −d      Build directories, like checkout does.
        −f      Force a head revision match if tag/date not found.
        −l      Local directory only, no recursion.
        −R      Process directories recursively.
        −p      Send updates to standard output (avoids stickiness).
        −k kopt Use RCS kopt −k option on checkout.
        −r rev  Update using specified revision/tag (is sticky).
        −D date Set date to update from (is sticky).
        −j rev  Merge in changes made between current revision and rev.
        −I ign  More files to ignore (! to reset).
        −W spec Wrappers specification line.
(Specify the −−help global option for a list of other help options)

In order to receive changes from the latest commits from your peer developers, do:
bash$ cvs update

If another developer has done bigger changes such as adding new directories etc. do:
bash$ cvs update −d
```

4.3 add

cvs add file... Use this command to enroll new files in CVS records of your working directory. The files will be added to the repository the next time you run `cvs commit'. Note: You should use the `cvs import' command to bootstrap new sources into the source repository. `cvs add' is only used for new files to an already checked−out module.

```
bash$ cvs −−help add
Usage: cvs add [−k rcs−kflag] [−m message] files...
        −k      Use "rcs−kflag" to add the file with the specified kflag.
        −m      Use "message" for the creation log.
(Specify the −−help global option for a list of other help options)

To add a new file to the repository do:
bash$ cvs add newFile
```

```
bash$ cvs commit

To add a new binary file to the repository do:
bash$ cvs add −kb newBinaryFile
bash$ cvs commit
(−kb specifies that file is binary)

To add a new directory to the repository do:
bash$ cvs add newDirectory
bash$ cvs commit

To remove an existing file from the repository do:
bash$ rm existingFile
bash$ cvs remove existingFile
bash$ cvs commit
```

4.4 remove

cvs remove file... Use this command (after erasing any files listed) to declare that you wish to eliminate files from the repository. The removal does not affect others until you run `cvs commit'.

```
bash$ cvs −−help remove
Usage: cvs remove [−flR] [files...]
        −f      Delete the file before removing it.
        −l      Process this directory only (not recursive).
        −R      Process directories recursively.
(Specify the −−help global option for a list of other help options)
```

4.5 commit

cvs commit file... To check in modifications (on existing files). Use this command when you wish to ``publish'' your changes to other developers, by incorporating them in the source repository.

NOTE : It's usually a very good idea to do 'cvs update' before committing changes.

```
bash$ cvs −−help commit
Usage: cvs commit [−nRlf] [−m msg | −F logfile] [−r rev] files...
        −n      Do not run the module program (if any).
        −R      Process directories recursively.
        −l      Local directory only (not recursive).
        −f      Force the file to be committed; disables recursion.
        −F file Read the log message from file.
        −m msg  Log message.
        −r rev  Commit to this branch or trunk revision.
(Specify the −−help global option for a list of other help options)
```

4.6 diff

cvs diff file... Show differences between files in the working directory and source repository, or between two revisions in the source repository. (Does not change either repository or working directory.)

```
bash$ cvs −−help diff
Usage: cvs diff [−lNR] [rcsdiff−options]
    [[−r rev1 | −D date1] [−r rev2 | −D date2]] [files...]
        −l      Local directory only, not recursive
        −R      Process directories recursively.
        −D d1   Diff revision for date against working file.
        −D d2   Diff rev1/date1 against date2.
        −N      include diffs for added and removed files.
        −r rev1 Diff revision for rev1 against working file.
        −r rev2 Diff rev1/date1 against rev2.
        −−ifdef=arg     Output diffs in ifdef format.
(consult the documentation for your diff program for rcsdiff−options.
The most popular is −c for context diffs but there are many more).
(Specify the −−help global option for a list of other help options)
```

4.7 Creating Releases

Since there usually are several files with different version numbers in a project, it's a good idea to "stamp" the files with a release tag for each release, this can be done like this (for version "v001"):

```
bash$ cvs tag −R "v001"
bash$ cvs commit

This release can be checked out with
bash$ cvs checkout −r "v001" YourProject
```

4.8 Emacs Editor

Emacs is a powerful editor and it supports CVS/RCS − especially for revision merging and comparing. The main Emacs site is at http://www.gnu.org/software/emacs/emacs.html.

5. Strong, Weak or No Locking

CVS is a powerful system and is highly customizable. CVS supports:

- Strong locking with "reserved checkouts" via **cvs admin −l** or Shell Scripts . Also read the Reserved checkouts. Here is a patch (http://www.cvshome.org/dev/patches/editf) from Eric Griswold for reserved checkouts.
- Weak locking via 'cvs watch' features. Also see "cvs edit" to give a warning(http://www.cvshome.org/dev/text2/res2) if someone else is already editing the file.
- No locking − the default permitting concurrent editing of files.

6. Shell Scripts

The following are wrappers around the basic CVS commands. These scripts give you initial **booster−push** into the CVS system and are useful until you become very familiar with the CVS commands. The scripts are written for Korn shell since it is always available on all flavors of Unix, but you can translate to bash or Perl if needed. You can customize these scripts to your taste. They are basically CVS commands, but features are added to make it site specific. For example, the sedit script provides locking so that users will know someone

is editing the file. Of course users can directly use the CVS commands to bypass these scripts. These scripts demonstrate how CVS can be **customized** to a great extent.

NOTE: *The wrapper shell scripts assume the user's home directory as the root and check out the tree from CVS to build the tree underneath user's home directory.*

TIP: *In these shell scripts, every target filename is composed of 3 parts − Home directory, sub−directory and the filename. The full−path is $HOME/$subdir/$fname And in CVS the same directory structure is maintained (by variable $subdir) therefore in cvs there will be something like $CVSROOT/$subdir/$fname. In all scripts, these 4 variables $HOME, $CVSROOT, $subdir and $fname play an important role. For example, sample values can be like HOME=/home/aldev, subdir=myproject/src, CVSROOT=/home/cvsroot, and fname=foo.cpp*

Copy these scripts to /usr/local/bin and this should be in the user's PATH environment.

1. **sget** [−r revision_number] <file/directory name> To get a file or entire directory from CVS in READ ONLY mode. Click downloadsoftware to get this.
2. **sedit** [−r revision_number] <filename> To edit a file in order to make changes to code. This will lock the file so that nobody else can check it out. Of course you can change the script to your requirement − make no locking, warning message, or very strong locking. Click downloadsoftware to get this.
3. **scommit** [−r revision_number] <filename> To commit the changes you made to filename or entire directory. Upload your changes to CVS. Click downloadsoftware to get this.
4. **supdate** <filename/directory> To update a filename or to update an entire directory by getting the latest files from CVS. Click downloadsoftware to get this.
5. **sunlock** [−r revision_number] <filename> To unlock the file got by sedit. Will release the lock. Click downloadsoftware to get this.
6. **slist** To see the list of files currently being edited by you. Does 'ls −l | grep | ...' command. Click downloadsoftware to get this. Note that there is also another Unix command by the name slist (list available Netware servers). You should make sure cvs script slist comes before other in your PATH environment.
7. **sinfo** <filename/directory> To get the information of changes/revisions to a file. Click downloadsoftware to get this.
8. **slog** <filename> To get the history of changes/revisions to a file from CVS. Click downloadsoftware to get this.
9. **sdif** <filename>

   sdif −r rev1 −r rev2 <filename> To get the diff of your file with CVS. Click downloadsoftware to get this.

   NOTE: sdif has only one 'f' because there is already another Unix command called 'sdiff'
10. **sadd** <filename> To add a new file to CVS repository. Click downloadsoftware to get this.
11. **sdelete** <filename> To delete a file from CVS repository. Click downloadsoftware to get this.
12. **sfreeze** <revision name> <directory name> To freeze the code, that is make a release of the entire source tree. Click downloadsoftware to get this.

    For example :

```
cd $HOME;
sfreeze REVISION_1_0  srctree
```

This will freeze code with tag REVISION_1_0 so that you can later checkout the entire tree by using the revision name.

7. CVS Documentation

At Unix prompt type:

1. cvs −−help
2. cvs −−help−options
3. cvs −−help−commands
4. cvs −H checkout
5. cvs −H commit
6. man cvs
7. man tkcvs
8. Visit http://www.cyclic.com
9. Visit http://www.loria.fr/~molli/cvs−index.html

The tkcvs http://www.tkcvs.org is the Tcl/Tk GUI interface to CVS. It also has online help. Try the following:

- cd $HOME/src/foo.cpp
- tkcvs
- Click on foo.cpp
- Click on 'Revision Log Icon' which is located next to 'spectacle' icon.
- This will display the branch TREE in the window. Now click the RIGHT Mouse button on the text '1.3' and click the LEFT Mouse button on text '1.1'. Then click on "Diff" button. This will display a two−pane window!!
- Click on the "Next" button to step thru more diffs. Click on "Center" to center the text.

There is also a Windows 95 client for CVS called WinCVS (see: http://www.wincvs.org and cyclicsite). WinCVS can be used along with Samba(on cdrom samba*.rpm) − http://www.samba.org

The essential command are:

- cvs checkout <filename >
- cvs update <filename>
- cvs add <file, ..>
- cvs remove <file, ..>
- cvs commit <file>
- cvs status <filename>
- cvs log <filename>
- cvs diff −r1.4 −r1.5 <filename> This gives a diff between version 1.4 and 1.5 on filename.

7.1 Online Documentation

On Linux systems, you can find the CVS documentation in postscript format at **/usr/doc/cvs*/*.ps**. Also there is an FAQ and other useful information.

```
bash# cd /usr/doc/cvs*
bash# gv cvs.ps
```

7.2 CVS Org Documentation

The documentation on CVS from "CVS Organization" is at http://www.cvshome.org/docs

The Official manual for CVS by Cederqvist is at http://www.cvshome.org/docs/manual/cvs.html

FAQ for CVS is at http://www.cs.utah.edu/dept/old/texinfo/cvs/FAQ.txt

7.3 CVS Training

- http://rpmfind.net/tools/CVS/training/cvstrain.html
- http://www.loria.fr/~molli/cvs/cvs−tut/cvs_tutorial_toc.html
- http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/tools/srt/
- http://durak.org/cvswebsites/
- http://www−users.informatik.rwth−aachen.de/~wge/tools/cvs/cvsclient/cvsclient_toc.html
- http://www−users.informatik.rwth−aachen.de/~wge/tools/cvs.html

General utilities for cvs (third party):

- The textbook "Open Source Development with CVS" by Karl Fogel at http://cvsbook.red−bean.com has third−party−tools and mirror sites at Zevils
- http://rcs.ee.washington.edu/spp/Projects/Manastash/Links/cvsbook_toc.html

7.4 CVS Online Textbook

The following CVS textbook is available online.

1. **"CVS Best Practices"** Version: 0.5 by Vivek Venugopalan, vivek@magic−cauldron.com at http://www.tldp.org/REF/CVS−BestPractices/html/index.html
2. Open Source Development with CVS by Karl Fogel http://cvsbook.red−bean.com
3. The Official manual for CVS by Cederqvist is at http://www.cvshome.org/docs/manual/cvs.html

---

8. Graphical Front Ends

The following GUI front ends for CVS are available:

- Popular CVS GUI front end http://cervisia.sourceforge.net, get RPM packages at Cervisia RPMs
- CVS home.org http://www.cvshome.org/dev/addons.html
- CVS Web for windows http://www.devguy.com/fp/cfgmgmt/cvs/cvs_admin_nt.htm#CVSWEBIIS and at http://stud.fh−heilbronn.de/~zeller/cgi/cvsweb.cgi
- TkCVS http://www.tkcvs.org is the Tcl/Tk GUI interface to CVS and at cyclicsite
- gCVS: A portable GUI for the non−technical CVS user http://www.arachne.org/software/gcvs
- jCVS is a CVS client package written entirely in Java http://www.jcvs.org And at cyclicsite
- WinCVS http://www.cvshome.org/cyclic/cvs/soft−maccvs.html and at cyclicsite
- Component soft Win CVS http://www.componentsoftware.com/cvs
- JA−SIG UPortal CVS http://www.mis3.udel.edu/~jlaker/development
- http://ppprs1.phy.tu−dresden.de/~trogisch/lincvs/lincvsen.html
- http://www.loria.fr/~molli/cvs/doc/cvs_toc.html

It is **very strongly recommended** that you use Samba(on cdrom samba*.rpm) and a PC X Server on MS Windows 95/NT. By using Samba the remote directory on Unix will look like local folder on MS Windows. See the next section for PC X Server.

For Apple Macintosh − Mac OS: See MacCvs at http://www.cvsgui.org and MacCvsPro at http://www.maccvs.org

---

9. CVS for MS Windows 95/98/NT/2000/XP

9.1 Method 1: Using VNC, Samba

It is **VERY STRONGLY recommended** that you use Samba(on cdrom samba*.rpm) and a VNC viewer (or PC X Server) on MS Windows 95/NT. With samba the Unix/Linux CVS server will be like a **file server**. By using Samba the remote directory on Unix will look like a local folder on MS Windows on the local disk. Install samba*.rpm on Unix/Linux server(which has the CVS repository) and install the VNC viewer (or PC X server) on MS Windows 95/NT/2000/XP desktop. Using a VNC (or PC X server) you can easily log on to the Unix box and check−out/check−in the files. And you can use tools like Java Visual Cafe or Java JBuilder on MS Windows to edit the files located in Unix/Linux folder(via samba). After editing, you can check−in the files to Unix through VNC or PC X−server.

Advantages of using CVS on Linux/Unix via MS Windows are:

- Only one single Linux File server (CVS server) can serve many MS Windows clients.
- A Linux file server (CVS) is very robust, secure and reliable
- Only one UPS (uninterrupted power supply) battery is required for a linux server.
- Linux can serve as MS Windows folder through Samba package.
- A Linux file server (CVS) supports centralized backups via tools like BRS, Arkeia, Bru mirrors at angelfire, geocities, virtualave, Fortunecity, Freewebsites, Tripod, 101xs, 50megs,
- A Linux file server (CVS) requires just one small server room which can air−conditioned and dust free. Small room keeps the cooling/heating costs down.
- A Linux file server (CVS) provides security via Unix groups and user id authentication

The best tool for remote access is VNC. The VNC is lightweight and is much better than the PC X servers. ***The VNC is very strongly recommended over PC X server***. The remote access methods available are:

- VNC (Virtual Network Computing) at http://www.uk.research.att.com/vnc VNC is not an X−server but can display the remote Unix on Windows. VNC is the best tool in the market for remote access, it is very lightweight and is a very powerful software.
- Get VNC rpms from rpmfind.
- The best Window manager for VNC is QVWM which is like MS Windows 98/NT/2000/XP interface, get it from http://www.qvwm.org.
- After starting vncserver, you can start the **vncviewer** program on clients like MS Windows, Mac or Linux.
- See also the List of X11 Windows Managers.

Compiling qvwm on Solaris : On Solaris you should install the following packages which you can get from http://sun.freeware.com − xpm, imlib, jpeg, libungif, giflib, libpng, tiff. And you can download the binary package for solaris from http://www.qvwm.org.

Or you can download the qvwm source for solaris from http://www.qvwm.org and compile it using gcc.

Troubleshooting compile: You should put unsigned long before arg in usleep() usleep((unsigned long) 10000)

The following PC X servers are available:

- Low cost, best and small size (3 MB) http://www.microimages.com and click on "X−Server (MI/X) for Windows"
- Humming bird eXceed 14 MB http://www.hummingbird.com
- Starnet 5.2 MB http://www.starnet.com

There are more than 2 dozen vendors for X servers for Windows:

- X−win pro 6.34 MB http://www.labf.com
- X−WinPro http://lab−pro.com
- X−Link http://www.xlink.com/x.htm
- Xoftware http://www.age.com

University resources:

- University listings http://www.et.byu.edu/support/pc/xterm.html
- Floppy based PC "X server" http://mirriwinni.cse.rmit.edu.au/~brad/co338/sem1/floppy.html

9.2 Method 2: Using Cygwin

You can install the Redhat Cygwin and install the CVS clients and SSH packages via Cygwin. With cygwin the Windows 95/NT/2000/XP will be like a Unix client. Bring up the cygwin bash shell prompt and you can access the remote CVS server. With cygwin the Windows 95/NT will be like any other Linux CVS client.

9.3 CVS exe for Windows 95/NT/2000/XP

You can install and run CVS on MS Windows directly. Download cvsnt from http://www.cvsnt.org. See the installation instructions and other documents of CVS on NT/2000 at http://www.devguy.com/fp/cfgmgmt/cvs/cvs_admin_nt.htm#install.

9.4 Windows 95/NT/2000/XP FTP Tools

You can also use the ftp tools on MS Windows to transfer files from a Unix/Linux (CVS repository) to windows:

- Go to Tucows and search "ftp tools" for MS Windows http://www.tucows.com

9.5 Visual Cafe(Java), JBuilder, MS Visual C++, HTML files

Using Samba and a PC X server it is possible to use CVS on MS Windows platform. And the tools like Symantec Visual Cafe (Java), Inprise JBuilder, MS Visual C++ and others are easily supported by CVS.

You can also store the HTML files on a CVS repository via Samba and easily access them from MS Windows.

9.6 Samba Admin tool

To administer samba use the admin tools from http://www.samba.org. Go here and click on "GUI Interfaces Tools".

---

10. Security of CVS Repository

To make a CVS server and CVS repository secure do the following:

- Run CVS on a stand−alone Linux/Unix box, see Performance Tuning.
- Remove unnecessary software packages from CVS linux box − to prevent external vandals running it. Just in case vandals break into the system, you do not want to give them a chance to run dangerous programs.
- Consider SSH as given in the chapter Multi−User Repository
- Consider Kerberos − install cvs−*−kerberos*.rpm package http://cvshome.org/dev/codelinux.html.
- Visit http://www.cvshome.org and post your security questions in the mailing list.

---

11. Remote, Multi−User CVS Repository

The Cederqvist manual at http://cvshome.org/docs/manual/cvs_2.html#SEC30 describes how to setup CVS for external access.

In order to use CVS for a group, one has to set up a permissions system to allow people to access the system from other machines. There are three ways to do this (:server:, :pserver:, and :ext:). The pserver mechanism and use of rsh are both insecure. Only the :ext: (with ssh) offers sufficient security protection.

If you set CVS_RSH to SSH or some other rsh replacement, the instructions **may be** similar to `.rhosts' but consult the documentation for your rsh replacement.

To get ssh visit http://rpmfind.net and in the search box enter "ssh". Or visit http://www.redhat.com/apps/download and in the search box enter "ssh". Download and install the ssh RPM and then configure CVS to use it. See also http://www.ssh.org.

Note: If you plan to configure CVS for use with rsh then you MUST do this critical step:

```
bash# chmod 600 .rhosts
```

See also JA−SIG UPortal CVS repository http://www.mis3.udel.edu/~jlaker/development.

11.1 SSH Authentication

If you're tired of entering passwords for each simple CVS command, then you can distribute your ssh−identity from the client to the server in order to allow automatic identification (i.e. no password needed!), this can be done by On the CVS server box do:

```
For ssh 1:
bash$ cd $HOME
bash$ ssh−keygen
```

```
For ssh 2:
bash$ cd $HOME
bash$ ssh−keygen −t rsa
```

You should be asked to save 'your identification' in /home/developer/.ssh/identity (ssh 1) or /home/developer/.ssh/id_rsa.pub (ssh 2) (or wherever $HOME points to), just hit enter. When asked for password and confirmation of password, continue hitting enter. Then copy your public key (identity.pub for ssh 1 or id_rsa.pub for ssh 2) to the server using secure copy (a part of ssh):

```
ssh 1:
clientbox$ scp .ssh/identity.pub developer@serverbox.domain.com:~/.ssh

ssh 2:
clientbox$ scp .ssh/id_rsa.pub developer@serverbox.domain.com:~/.ssh
```

Then log onto the server and fix the authorized_keys file.

```
ssh 1:
clientbox$ ssh developer@serverbox.domain.com
serverbox$ cd .ssh
serverbox$ cat identity.pub >> authorized_keys

ssh 2:
clientbox$ ssh developer@serverbox.domain.com
serverbox$ cd .ssh
serverbox$ cat id_rsa.pub >> authorized_keys2
serverbox$ chmod go−w authorized_keys2
```

You should now be able to ssh directly from the client to server without having to enter password, this can be tested with:

```
ssh 1 or ssh 2:
clientbox$ ssh developer@serverbox.domain.com
```

Version control from this point should not require you to enter the password.

Note: This enables anyone getting access to your client to continue into the server without knowing the password on the server.

Encrypted Disks

If you fear such a situation, this can (somewhat) be prevented by using a encrypted disk, e.g. PGPDisk holding the $HOME directory on the client. So when an intruder takes over your machine s/he needs to know the password for your encrypted disk in order to get further into the server. Another advantage of using a encrypted disk is that your (checked out) source code can reside on it.

11.2 Securing CVS by pserver Port Forwarding using an SSH Tunnel

From : http://www.cycom.co.uk/howto/cvssshtunnel.html

The Concurrent Versions System, in its "pserver" client/server mode, and secured by "ssh" encrypted tunnels,

can allow multiple authors to safely collaborate over the internet. CVS is a source file version control system optimised for wide area networks, concurrent editing, and reuse of 3rd party source libraries. Pserver is a protocol used for communication between CVS clients and servers. SSH is a tool to transparently encrypt TCP/IP network connections. MSWindows users can use a posix shell

The tools discussed here are most functional in a unix environment but windows users can obtain similar functions by using a posix shell available from http://www.cygwin.com/. which will include an openssh package in 'latest' and a cvs package in 'contrib'. Other native windows ports of the tools are available but may lack needed features such as SSH2 DSA support. Creating the CVS Repository on the repository server machine

If you are a contributing author, you don't need to know how the repository was created. You may skip this section. If you are the unix repository administrator, you would create a directory and run cvs init then adjust the control files in CVSROOT to suit the permitted users (writers, passwd). It is convenient to have all files owned by "cvsuser" Starting the repository service

If you are a contributing author, you don't need to know how the repository server is started. You may skip this section. If you are the unix repository administrator, you allow the server to be started by xinetd with security constraints that specify that only clients local to the server machine may connect. To do this, create a configuration file named "/etc/xinetd.d/cvspserver" with contents similar to:

```
service cvspserver
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = cvsuser
        server          = /usr/bin/cvs
        server_args     = −f −−allow−root=/cycomcvs pserver
        passenv         =
        log_on_failure  += USERID
        only_from               = 127.0.0.1
        bind                    = 127.0.0.1
}
```

then restart the xinetd super−service. Generating a public/private DSA keypair on the author's client machine

Contributing authors must perform this step only once. The "ssh" tools have various ways of authenticating users. The method chosen here is to use the DSA Digital Signature Algorithm. This is a public/private keypair algorithm which means that the secret private key need never be communicated to anyone and can stay safe on the clients hard disk (protected by a passphrase). The public key can be advertised to anyone with no loss of security. If you do not already have the ssh tools then you should obtain them from http://www.openssh.com/. They must support SSH2 as we use the DSA algorithm not RSA.

A unix client will generate the keypair using:− ssh−keygen −d This will result in the creation of a file " /.ssh/id_dsa.pub". You must send this public file to the unix repository administrator. Do not send any other file nor reveal any passphrase. Authorizing a client to tunnel to the repository server machine

If you are a contributing author, you don't need to know how authorization is allowed. You may skip this section. If you are the unix repository administrator, on receipt of a clients "id_dsa.pub" file, append the single line therein to the " cvsnobody/.ssh/authorized_keys2" file on the server. Creating the secure tunnel between author's client machine and the repository server machine.

CVS keeps a single copy of the master sources called a source repository. Remote authors access the repository using CVS client programs which talk to the repository service using a "pserver" protocol and connect using a registered TCP/IP port (port 2401).

The pserver protocol is insecure because passwords are transmitted unencrypted and there are often some hacked hosts on a network that are sniffing for passwords. The connection to be used for the pserver protocol therefore needs to be encrypted where it passes over any network. The "ssh" suite of programs provides such encrypted connections.

SSH will create a secure tunnel which makes the repository service appear to be local to your client machine. Similarly, your client machine will appear to be local to the repository service. Both client and server are fooled into thinking that they are on the same machine and that no traffic travels over any network.

The single line unix command to achieve this is:− /usr/bin/ssh −v −a −e none −N −o 'KeepAlive=yes' −o 'BatchMode=yes' −L 2401:localhost:2401 cvsnobody@j2ee.cycom.co.uk The fixed cvsnobody user is just for ssh tunneling purposes; it is not relevant to CVS. The j2ee.cycom.co.uk is the repository server machine name. This command will block. To destroy the tunnel, cntrl−c the command. If the tunnel collapses in use, reestablish it by repeating the command. Use another window to operate the CVS clients. Operating CVS clients on the author's client machine

Having established a tunnel, the remote CVS repository service now appears to be local to your client machine (i.e. at localhost). The CVS client programs obtain configuration data from a CVSROOT environment variable which should be set in unix (e.g in your ".bash_profile") using:−

```
CVSROOT=':pserver:jsharp@localhost:/cycomcvs'
export CVSROOT
```

The username "jsharp" and the repository root "/cycomcvs" will have been sent to you by the repository administrator.

The first CVS client command should always be:− cvs login You will be prompted for a password (again sent to you by the repository administrator).

To create a new cvs working directory and populate it from Honest John Car Rental Demo sources, use :−

```
mkdir mywork
cd mywork
cvs co hjvh
cvs co hjvhear
cvs co hjvhmodel
```

To freshen an existing working directory with updates from other authors, use:−

```
cd mywork/hjvh
cvs update
```

To publish the files that you have changed in an existing working directory, use:−

```
cd mywork/hjvh
cvs commit
```

To publish a newly created file in an existing working directory, use:−

```
cd mywork/hjvh
cd mywork/hjvh
cvs add mynewfile.txt
cvs commit
```

To import a new independent directory tree of sources into the repository, make sure all files in the tree are useful source and then use:−

```
cd projdir
cvs import projdir projV1_1 proj_V1_1
cd ..
mv projdir origprojsources
cvs co projdir
```

11.3 An Example − Access Remote CVS Server

To access a remote cvs server, here is an example:

```
# Set env variable
export EDITOR=/bin/vi
export CVSROOT=:pserver:yourname@cvs.tldp.org:/cvsroot

# Login to remote cvs server
cvs −d $CVSROOT login

# Goto some local directory
cd $HOME/<somedirectory>

# You MUST create a new directory, as below...
mkdir cvsroot   # Create the local cvs directory which has the same name as in CVSROOT

# Now get the files from remote CVS repository
cd cvsroot
cvs get LDP/howto   # Or you can do 'cvs get . ' which will get everything

# After you make changes some file and later check−in that do
cd $HOME/<somedirectory>/cvsroot
cvs ci −m "your update comments here" LDP/howto/somefilename.java
```

12. RCS Shell Scripts

If you want to use RCS instead of CVS then you can use the following shell scripts.

1. cotree.sh − Will check out the entire directory tree from RCS. Instead of doing 'co −l name' on each file one−by−one, you can give just one command. Very useful shell script.
2. cofiles.sh − Will check out all the files in the directory.
3. ciall.sh − Will check in all the files into RCS with just one shell command.

You can get these scripts from downloadsoftware .

13. Performance Tuning of a CVS Server

For optimum performance a CVS server must be running on a stand alone Linux/Unix box.

To get more bang for a given CPU processing power, do the following:

- Recompile the Linux kernel to make it small and lean. Remove items which are not used. See the kernel howto at http://www.linuxdoc.org/HOWTO/Kernel−HOWTO.html
- Turn off unneccessary Unix processes − on Linux/Unix systems run chkconfig.

```
bash$ su − root
bash# man chkconfig
bash# chkconfig −−help
bash# chkconfig −−list | grep on | less
From the above list, turn off the processes you do not want to start automatically −
bash# chkconfig −−level 0123456 <service name> off
Next time when the machine is booted these services will not be started.
Now, shutdown the services manually which you just turned off.
bash# cd /etc/rc.d/init.d
bash# ./<service name> stop
```

- Do not run any other application processes which are unnecessary.
- Do not leave X Window running unattended because its processes consume memory and contribute to CPU load. It can also be a serious security hole from outside attacks. The X Window managers generally used are KDE, GNOME, CDE, XDM and others. You must exit the X Window immediately after using and most of the time you should see a command line console login prompt on the CVS server machine.

14. Problem Reporting System

Along with CVS, you may want to use project tracking system or problem reporting system. Every software project needs a problem reporting system that track bugs and assigns them to various developers. See GNU gpl GNATS at http://www.gnu.org/software/gnats/gnats.html and http://dcl.sourceforge.net And commercial PRS at http://www.stonekeep.com look for a project tracking system.

15. Configuration Management System Tools

What is Configuration Management (CM) ?

There are a number of different interpretations. It is about the tracking and control of software development and its activities. That is, it concerns the management of software development projects with respect to issues such as multiple developers working on the same code at the same time, targeting multiple platforms, supporting multiple versions, and controlling the status of code (for example a beta test versus a real release). Even within that scope there are different schools of thought:

- Traditional Configuration Management − checkin/checkout control of sources (and sometimes binaries) and the ability to perform builds (or compiles) of the entities. Other functions may be included as well.

- Process Management − control of the software development activities. For example, it might check to ensure that a change request existed and had been approved for fixing and that the associated design, documentation, and review activities have been completed before allowing the code to be "checked in" again.

While process management and control are necessary for a repeatable, optimized development process, a solid configuration management foundation for that process is essential.

Visit the following links:

- FAQ on Configuration Management tools http://www.iac.honeywell.com/Pub/Tech/CM/CMFAQ.html
- Linux version control and configuration management tools http://linas.org/linux/cmvc.html
- Configuration Management systems http://www.cmtoday.com/yp/commercial.html
- Configuration Management Tools http://www.iac.honeywell.com/Pub/Tech/CM/CMTools.html
- DevGuy CVS config mgmt http://devguy.com/fp/cfgmgmt/cvs
- Yahoo category site
- Free config mgmt tool http://www.canb.auug.org.au/~millerp/aegis/aegis.html
- Free CM tools http://www.loria.fr/cgi−bin/molli/cm/wilma/fcmt
- Rational ClearCase tool http://www.rational.com/products/clearcase/prodinfo.jsp

---

16. Related Sites

Related URLs are at:

- Linux goodies main site is at http://24.221.230.253 and secondary site at http://www.milkywaygalaxy.freeservers.com Mirror sites are at − angelfire, geocities, virtualave, Fortunecity, Freewebsites, Tripod, 101xs, 50megs,
- CVS Bubbles http://www.loria.fr/~molli/cvs−index.html
- CSSC (SCCS like system) http://cssc.sourceforge.net and mirror−site
- SCCS for Linux http://www.bitmover.com/bitkeeper

---

17. SCCS v/s CVS−RCS

SCCS (Source Code Control System) is no longer being enhanced or improved. The general consensus has been that this tool is clumsy and not suited to large numbers of users working on one project. Actually, SCCS interleaves all the versions, but it can make new development get **progressively slower**. Hence, SCCS is NOT recommended for new projects; however, it is still there to support old code base in SCCS.

RCS (Revision Control System) is often considered to be better than SCCS. One reason for this is that RCS baselines the most recent version and keeps deltas for earlier ones, making new development faster. Additional discussions concerning SCCS vs RCS are at http://www.faqs.org/faqs/unix−faq/faq/part7

Note that RCS learned from the mistakes of SCCS...

CVS, which requires RCS, extends RCS to control concurrent editing of sources by several users working on releases built from a hierarchical set of directories. "RCS is [analogous to using] assembly language, while CVS is [like using] Pascal".

---

18. Other Formats of this Document

This document is published in 14 different formats namely: DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU−info, HTML, RTF(Rich Text Format), Plain−text, Unix man pages, single HTML file, SGML (Linuxdoc format), SGML (Docbook format), and MS WinHelp format.

This howto document is located at:

- http://www.linuxdoc.org and click on HOWTOs and search for the howto document name using CTRL+f or ALT+f within the web−browser.

You can also find this document at the following mirrors sites:

- http://www.caldera.com/LDP/HOWTO
- http://www.linux.ucla.edu/LDP
- http://www.cc.gatech.edu/linux/LDP
- http://www.redhat.com/mirrors/LDP
- Other mirror sites near you (network−address−wise) can be found at http://www.linuxdoc.org/mirrors.html select a site and go to directory /LDP/HOWTO/xxxxx−HOWTO.html

- You can get this HOWTO document as a single file tar ball in HTML, DVI, Postscript or SGML formats from − ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO/other−formats/ and http://www.linuxdoc.org/docs.html#howto
- Plain text format is in: ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO and http://www.linuxdoc.org/docs.html#howto
- Single HTML file format is in: http://www.linuxdoc.org/docs.html#howto

  A single HTML file can be created with the command (see man sgml2html) − sgml2html −split 0 xxxxhowto.sgml
- Translations to other languages like French, German, Spanish, Chinese, and Japanese are in ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO and http://www.linuxdoc.org/docs.html#howto Any help from you to translate to other languages is welcome.

The document is written using a tool called "SGML−Tools" which can be got from: http://www.sgmltools.org Compiling the source you will get the following commands like:

- sgml2html xxxxhowto.sgml (to generate html file)
- sgml2html −split 0 xxxxhowto.sgml (to generate a single page html file)
- sgml2rtf xxxxhowto.sgml (to generate RTF file)
- sgml2latex xxxxhowto.sgml (to generate latex file)

18.1 Acrobat PDF format

A PDF file can be generated from postscript file using either acrobat **distill** or **Ghostscript**. And a postscript file is generated from DVI which in turn is generated from a LaTex file. You can download distill software from http://www.adobe.com. Given below is a sample session:

```
bash$ man sgml2latex
bash$ sgml2latex filename.sgml
```

```
bash$ man dvips
bash$ dvips −o filename.ps filename.dvi
bash$ distill filename.ps
bash$ man ghostscript
bash$ man ps2pdf
bash$ ps2pdf input.ps output.pdf
bash$ acroread output.pdf &
```

Or you can use the Ghostscript command **ps2pdf**. ps2pdf is a work−alike for nearly all the functionality of Adobe's Acrobat Distiller product: it converts PostScript files to Portable Document Format (PDF) files. **ps2pdf** is implemented as a very small command script (batch file) that invokes Ghostscript, selecting a special "output device" called **pdfwrite**. In order to use ps2pdf, the pdfwrite device must be included in the makefile when Ghostscript was compiled; see the documentation on building Ghostscript for details.

18.2 Convert Linuxdoc to Docbook format

This document is written in linuxdoc SGML format. The Docbook SGML format supercedes the linuxdoc format and has a lot more features than linuxdoc. The linuxdoc is very simple and easy to use. To convert linuxdoc SGML file to Docbook SGML use the program **ld2db.sh** and some Perl scripts. The ld2db output is not 100% clean and you need to use the **clean_ld2db.pl** Perl script. You may need to manually correct a few lines in the document.

- Download the ld2db program from http://www.dcs.gla.ac.uk/~rrt/docbook.html or from Milkyway Galaxy site click on "Source code for C++ howto".
- Download the cleanup_ld2db.pl perl script from from Milkyway Galaxy site click on "Source code for C++ howto".

The ld2db.sh is not 100% clean, so you will get some errors when you run it.

```
bash$ ld2db.sh file−linuxdoc.sgml db.sgml
bash$ cleanup.pl db.sgml > db_clean.sgml
bash$ gvim db_clean.sgml
bash$ docbook2html db.sgml
```

And you may have to manually edit some of the minor errors after running the Perl script. For example you may need to put closing tag < /Para> for each < Listitem>

18.3 Convert to MS WinHelp format

You can convert the SGML howto document to a Microsoft Windows Help file, First convert the sgml to html using:

```
bash$ sgml2html xxxxhowto.sgml     (to generate html file)
bash$ sgml2html −split 0   xxxxhowto.sgml (to generate a single page html file)
```

Then use the tool HtmlToHlp. You can also use sgml2rtf and then use the RTF files for generating winhelp files.

18.4 Reading various formats

In order to view the document in dvi format, use the xdvi program. The xdvi program is located in tetex−xdvi*.rpm package in Redhat Linux which can be located through ControlPanel | Applications | Publishing | TeX menu buttons. To read a dvi document give the command:

```
xdvi −geometry 80x90 howto.dvi
man xdvi
```

And resize the window with the mouse. To navigate use Arrow keys, Page Up, Page Down keys, also you can use 'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n' letter keys to move up, down, center, next page, previous page etc. To turn off expert menu press 'x'.

You can read a postscript file using the program 'gv' (ghostview) or 'ghostscript'. The ghostscript program is in the ghostscript*.rpm package and the gv program is in the gv*.rpm package in Redhat Linux which can be located through ControlPanel | Applications | Graphics menu buttons. The gv program is much more user friendly than ghostscript. Also ghostscript and gv are available on other platforms like OS/2, Windows 95 and NT. You can view this document even on those platforms.

- Get ghostscript for Windows 95, OS/2, and for all OSes from http://www.cs.wisc.edu/~ghost

To read a postscript document give the command:

```
gv howto.ps
ghostscript howto.ps
```

You can read an HTML format document using Netscape Navigator, Microsoft Internet explorer, Redhat Baron Web browser or any of the 10 other web browsers.

You can read the latex, LyX output using LyX an X Window front end to LaTex.

---

19. CVS Shell Scripts

You can get the shell scripts for a nominal fee (which covers code maintenance and ISP charges) from Milkyway Galaxy site. These shell scripts are very useful.

---

20. Copyright and License
