CVS−RCS− HOW−TO document for Linux (Source Code Control System)

Table of Contents

Table of Contents

CVS−RCS− HOW−TO document for Linux (Source Code Control System)

Al Dev (Alavoor Vasudevan) alavoor@yahoo.com

v19.0, 24 Jan 2001

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

---

8. Graphical Front ends

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

10. Security of CVS Repository

11. Multi−User CVS Remote Repository

12. RCS Shell scripts

13. Performance Tuning of CVS server

14. Problem Reporting System

15. Configuration Management System Tools

16. Related sites

17. SCCS v/s CVS−RCS

18. Other Formats of this Document

19. Copyright and License

20. sget

21. sedit

22. scommit

23. supdate

24. sunlock

25. slist

26. sinfo

27. slog

28. sdif

29. sadd

30. sdelete

31. sfreeze

---

1. Introduction

Source code control system is a MUST to manage the changes occuring to 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 application layer interface as a wrapper on top 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 ever 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 together to form a software release.

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

---

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 de−centralised an user checks out files/directories from the repostitory and has his own separate stable source directory tree.
- CVS can "STAMP" releases of 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 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 the "State of the Art" technology. The cvs software is very advanced and capable system developed over very long period of time (it took several years!!).
- 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, 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
To see the list of files installed do -
rpm -qpl cvs*.rpm | less
```

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

On other flavors of unix, you may need to download the RCS and CVS tar balls and follow 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, than you should add these to your local profile file /.bash_profile.

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

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'.

```
export CVSROOT=/home/cvsroot
mkdir $CVSROOT
chmod o-rwx $CVSROOT
chmod ug+rwx $CVSROOT
```

Now, change the group of $CVSROOT to group−name of users who want to use cvs sytem.

```
chgrp users $CVSROOT
```

To initialize the CVS and to put in source code files do −

```
cvs init

# Change directory is a must
cd $HOME/my_source_code_dir

# Must give vendor tag and revision tag
cvs import my_source_code_dir V1_0 R1_0
```

3.2 Migrate RCS to CVS

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

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

```
#!/bin/ksh

#############################################################
# Program to Migrate the existing source code in RCS to CVS
#
# Needs the korn shell RPM package  pdksh*.rpm from Linux
# contrib cdrom
#############################################################

#
# rcs2cvs - convert source tree from RCS to CVS
#

# project to convert
PROJECT='project'

# current RCS root
RCSROOT="$HOME/rcs"

if cd "$RCSROOT/$PROJECT"
then
        cd "$RCSROOT"
else
        echo >&2 "`basename "$0"`: can't change to RCS directory '$RCSROOT/$PROJECT'."
        exit 1
fi

# current CVS root
CVSROOT="$HOME/cvs"

# create new CVS directory for project 'project'
if mkdir "$CVSROOT/$PROJECT"
then
        :
else
        echo >&2 "`basename "$0"`: can't create CVS directory '$CVSROOT/$PROJECT'."
        exit 2
fi
```

```
# create CVS project tree from RCS tree
find "$PROJECT" -type d -name RCS -print |
while read RCS
do
        CVS="`dirname "$RCS"`"
        (if cd "$RCS"
        then
#               if find . -type f -name '*,v' -print | cpio -pdmv "$CVSROOT/$CVS"
                if find . -type f -print | cpio -pdmv "$CVSROOT/$CVS"
                then
                        :
                else
                        echo >&2 "`basename "$0"`: can't convert RCS subdirectory '$RCSROOT/$RCS' to CVS subdirectory '$CVSROOT/$CVS'."
                fi
        else
                echo >&2 "`basename "$0"`: can't change to RCS subdirectory '$RCSROOT/$RCS'."
        fi)
done
```

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 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)
```

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)
```

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... Use this command when you wish to ``publish'' your changes to other developers, by incorporating them in the source repository.

```
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 working directory and source repository, or between two revisions in source repository. (Does not change either reposi- tory 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 Emacs Editor

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

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" 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 untill you become very familiar with the CVS commands. The scripts are written for Korn shell since korn shell is always available on all flavors of unixes, but you can translate to bash or PERL if needed. You can customize these scrips to your taste. They are basically CVS commands but features are added to make it site specific. For example, sedit script provides locking so that users will know some−one is editing the file. Of course users can directly use the CVS commands to by−pass these scripts. These scripts demonstrate how CVS can be **customized** to a great extent.

NOTE: *The wrapper shell scripts assume that 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 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 sget
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 checkout. Ofcourse you can change the script to your requirement − make no locking, warning message or very strong locking. Click sedit
3. **scommit** [−r revision_number] <filename> To commit the changes you made to filename or entire directory. Upload your changes to CVS Click scommit
4. **supdate** <filename/directory> To update a filename or to update an entire directory by getting the latest files from CVS Click supdate
5. **sunlock** [−r revision_number] <filename> To unlock the file got by sedit. Will release the lock. Click sunlock
6. **slist** To see the list of files currently being edited by you. Does 'ls −l | grep | ...' command. Click slist . Note that there is also another unix command by 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 sinfo
8. **slog** <filename> To get the history of changes/revisions to a file from CVS Click slog
9. **sdif** <filename>

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

   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 sadd
11. **sdelete** <filename> To delete a file from CVS repository Click sdelete
12. **sfreeze** <revision name> <directory name> To freeze the code, that is make release of entire source tree. Click sfreeze

    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 with 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.

- 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 RIGHT Mouse button click on the text '1.3' and LEFT Mouse button click on text '1.1'. Than click on "Diff" button. This will display 2 pane−window!!
- Click on "Next" button to step thru more diffs. Click on "Center" to center the text.

There is also a Windows 95 client for CVS, and is called WinCVS 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 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 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 Organisation" is at http://www.cvshome.org/docs

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)

- http://www.zevils.com/doc/cvsbook/cvsbook_8.html
- http://www.zevils.com/doc/cvsbook/cvsbook_toc.html#SEC_Contents
- http://rcs.ee.washington.edu/spp/Projects/Manastash/Links/cvsbook_toc.html

8. Graphical Front ends

The following GUI front ends for CVS are available.

- 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 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 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 next section for PC X−Server.

---

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

It is **VERY STRONGLY recommended** that you use Samba(on cdrom samba*.rpm) and 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 local folder on MS Windows on the local disk. Install samba*.rpm on unix/linux server(which has CVS repository) and install the PC X−server on MS Windows 95/NT/2000 desktop. Using PC X−server you can easily log on to the unix box and check−out/check−in the files. And you can use the 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 thru 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
- Linux file server (cvs) is very robust, secure and reliable
- Only one UPS (uninterrupted power supply) battery is required for linux server.
- Linux file server (cvs) supports centralised backups via tools like Arkeia, Bru mirrors at http://aldev.webjump.com, angelfire, geocities, virtualave, bizland, theglobe, spree, infoseek, bcity, 50megs
- Linux file server (cvs) requires just one small server room which can air−contitioned and dust free. Small room keeps the cooling/heating costs down.
- Linux file server (cvs) provides security via unix groups and user id authentication

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−Server 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

Alternatives to X−servers:

- VNC (Virtual Network Computing) at http://www.uk.research.att.com/vnc VNC is not a X−server but can display the remote unix on Windows.

9.1 Windows 95/NT/2000 FTP Tools

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

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

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

Using Samba and 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 CVS repository via Samba and easily access from MS Windows.

9.3 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 CVS server and CVS repository secure do the following:

- Run CVS on 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. Multi−User CVS Remote 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 search box enter "ssh". Or visit http://www.redhat.com/apps/download and in search box enter "ssh". Download and install 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 6000 .rhosts
```

See also JA−SIG UPortal CVS repository

12. RCS Shell scripts

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

12.1 cotree.sh

```
#!/bin/ksh

# cotree.sh (Check Out Tree shell script)
# cotree.sh - Check out the entire RCS directory

# Usage :
# This will get the all the directories
#       unix> cotree.sh
#
# This will get just one single directory tree
#       unix> cotree.sh <directory name>

# See also cofiles.sh

###############################################################
# Setting up RCS (Revision Control System)
# Install the RCS programs - which gives command co, ci, rcslog
# Create a rcs home directory where you want to put all the
```

```
# source code repository. Call this $RCSDIR=/home/rcs_version_control
# Setup up an environment variable RCSDIR=/home/rcs_version_control
# in $HOME/.profile file. Like -
#               export RCSDIR=/home/rcs_version_control
# Create a directory structure under $RCSDIR and check in all your
# files using ci . See 'man ci'
# Now create a link from your home directory to your project
# under $RCSDIR
#               cd $HOME
#               mkdir $HOME/myproject
#               cd $HOME/myproject
# and run this script to get all the files and directory tree
#               cotree.sh
# This script will create the entire source-tree under user's
# home and also will have a soft link to RCS directories. Each
# user will run this script under his home directory.
###############################################################

check_out_directory()
{
        # Root directory of RCS (revision control system)
        # like RCSDIR=/home/rcs_version_control
        RCSDIR=$1
        DIRNAME=$2

        # The given directory name must exist in rcs root directory
        if [ "$DIRNAME" = "" -o  ! -d $RCSDIR/$DIRNAME ]; then
                print "\nDirectory DIRNAME=$DIRNAME does not exist!!"
                print "\nAborting the program ... and exiting...\n"
                exit
        fi

        mkdir -p $DIRNAME
        ln -s $RCSDIR/$DIRNAME/RCS $DIRNAME
        (
                cd $DIRNAME

                # This fails in case of filename=sample,vv
                # which inside RCS will be RCS/sample,vv,v
                # ls RCS | cut -d',' -f1 | xargs co
                # Use match to end of name $, as below -
                # Use ls RCS/* to avoid getting the names ./ and ../
                #ls RCS/* | cut -d'/' -f2 | sed -e's/,v$//g' | xargs co
                if [ -d RCS ]; then
                        ls RCS/* | cut -d'/' -f2 | sed -e's/,v$//g' | \
                        while read ii
                        do
                                #echo "ii is : $ii"
                                if [ -f "RCS/$ii,v" ]; then
                                        co $ii
                                fi
                        done
                fi
        )
}

# Root directory of RCS (revision control system)
# like RCSDIR=/home/rcs_version_control
if [ "$RCSDIR" = "" -o ! -d $RCSDIR ]; then
        print "\nDirectory RCSDIR=$RCSDIR does not exist!!"
        print "\nAborting the program ... and exiting...\n"
        exit
```

```
fi
#echo "rcsdir is : $RCSDIR"

# If a directory argument is passed, then check out all
# files for this directory only and exit.
if [ "$1" != "" ]; then
        (cd $RCSDIR; find $1 -type d -print ) |
        while read DIRNAME
        do
                #echo DIRNAME=$DIRNAME
                #DIRNAME=c_src
                # Send rcs root directory and dir name relative to rcs root dir
                tmpaa=` basename $DIRNAME `
                if [ "$tmpaa" != "RCS" ]; then
                        check_out_directory $RCSDIR $DIRNAME
                fi
        done
else
        (cd $RCSDIR; find * -type d -print ) |
        while read DIRNAME
        do
                echo DIRNAME=$DIRNAME
                #DIRNAME=c_src
                # Send rcs root directory and dir name relative to rcs root dir
                tmpaa=` basename $DIRNAME `
                if [ "$tmpaa" != "RCS" ]; then
                        check_out_directory $RCSDIR $DIRNAME
                fi
        done
fi
```

12.2 cofiles.sh

```
#!/bin/ksh

# cofiles.sh (Check Out files shell script)
# cofiles.sh - Check out all the files in current directory from RCS
# See also cotree.sh and 'man rcsclean'

if [ ! -d RCS ]; then
        print "\nDirectory RCS does not exist!!"
        print "\nAborting the program ... and exiting...\n"
        exit
fi

#echo "No. of args = " $# " and all args " $@

while true
do
        print -n "\n\nCheck-out all files in read-write mode? <y/n> [n]: "
        read ans
        if [ "$ans" = "" -o "$ans" = "n" -o "$ans" = "N" ]; then
                ans="N"
                break
        elif [ "$ans" = "y" -o "$ans" = "Y" ]; then
                ans="Y"
                break
        else
```

```
                print "\nWrong entry! Try again!!"
        fi
done
#echo "The ans is : " $ans

if [ $# -eq 0 ]; then
        # The 'ls RCS' fails in case of filename=sample,vv in RCS/sample,vv,v
        # ls RCS | cut -d',' -f1 | xargs co
        # Use match to end of name $, as below -
        if [ "$ans" = "Y" ]; then
                ls RCS | sed -e's/,v$//g' | xargs co -l
        else
                ls RCS | sed -e's/,v$//g' | xargs co
        fi
elif [ $# -eq 1 ]; then
        if [ -f "RCS/$1,v" ]; then
                # Here, in this case $1 will be like dbalter.sql
                # and not like db*.sql....
                #echo "One arg, no. of args = " $# " and all args " $@
                if [ "$ans" = "Y" ]; then
                        co -l "$1"
                else
                        co "$1"
                fi
        else
                # For case where $1=db*.sql and there is no db*.sql in
                # current directory
                #echo "No files... no. of args = " $# " and all args " $@
                tmpaa="RCS/$1,v"  # will be like RCS/db*.sql,v
                ls $tmpaa | \
                while read ii
                do
                        #echo "ii is : $ii"
                        if [ "$ans" = "Y" ]; then
                                co -l "$ii"
                        else
                                co "$ii"
                        fi
                done
        fi
else
        for ii in $@
        do
                #echo "ii is : $ii,v"
                if [ "$ans" = "Y" ]; then
                        co -l "$ii"
                else
                        co "$ii"
                fi
        done
fi
```

12.3 ciall.sh

```
#!/bin/ksh

# ciall.sh (Check in files shell script)
# ciall.sh - Check in all the files in current directory into RCS
```

```
# This script is very useful for checking in enmass large number
# of new files into RCS. Saves time by avoiding to type the
# 'description' for every file
# And for files already in RCS, it does regular check-in command


# To convert filenames to lower case filenames, use this
# technique - use 'tr', see 'man tr'
#ls * | \
#while read ii
#do
#       jj=`echo $ii | tr [A-Z] [a-z] `
#       echo "ii is : $ii"
#       echo "jj is : $jj"
#       mv $ii $jj
#done

if [ ! -d RCS ]; then
        print "\nDirectory RCS does not exist!!"
        print "\nWill be creating RCS directory now ...\n"
        mkdir RCS
fi

print "\n\nNOTE: This is not log message!"
print "Please enter description (will be used for"
print -n "all the files checked in) : "
read description

#find * -prune -type f |

# The number of args is zero or more....
if  [ $# -eq 0 ]; then
        listoffiles="*"
else
        listoffiles="$@"
fi

find $listoffiles -prune -type f |
while read ii
do
        #echo $ii
        if [ -f "RCS/$ii,v" ]; then
                #print "The file $ii already in RCS"
                ci -m"$description" $ii
        else
                #print "The file $ii is new file"
                ci $ii << EOF
$description
EOF
        fi
done
```

---

13. Performance Tuning of CVS server

For optimum performance 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 linux kernel to make it small and lean. Remove items which are not used. See 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 turning 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 X−window processes consume memory, CPU load and can 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 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 where in bugs are tracked and assigned to various developers. Visit the site http://www.stonekeep.com for Project tracking system.

15. Configuration Management System Tools

What is Configuration Management (CM) ?

There are a number of different interpretations. It is about tracking and control of software development and its activities. That is, the mangement of software development projects with respect to issues such as multiple developers working on the same code at the same time, targetting multiple platforms, supporting multiple versions, and controlling the status of code (for example beta test versus 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 http://www.aldev.8m.com and mirrors at webjump, angelfire, geocities, virtualave, bizland, theglobe, spree, infoseek, bcity, 50megs
- CVS Bubbles http://www.loria.fr/~molli/cvs−index.html
- CSSC (SCCS like system) ftp://alpha.gnu.org/pub/gnu/CSSC
- 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 recomended for new projects, 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 is 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 12 different formats namely − DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU−info, HTML(multi−pages), single HTML page, RTF(Rich Text Format), Plain−text, Unix man pages and SGML.

- You can get this HOWTO document as a single file tar ball in HTML, DVI, Postscript or SGML formats from − ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/other−formats/ and http://www.linuxdoc.org/docs.html#howto
- Plain text format is in: ftp://sunsite.unc.edu/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
- Translations to other languages like French, German, Spanish, Chinese, Japanese are in ftp://sunsite.unc.edu/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 CVS−HOWTO.sgml (to generate html file)
- sgml2rtf CVS−HOWTO.sgml (to generate RTF file)
- sgml2latex CVS−HOWTO.sgml (to generate latex file)

LaTeX documents may be converted into PDF files simply by producing a Postscript output using **sgml2latex** (and dvips) and running the output through the Acrobat **distill** (http://www.adobe.com) command as follows:

```
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 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.

This howto document is located at −

- http://sunsite.unc.edu/LDP/HOWTO/CVS−HOWTO.html

Also you can find this document at the following mirrors sites −

- http://www.caldera.com/LDP/HOWTO/CVS−HOWTO.html
- http://www.WGS.com/LDP/HOWTO/CVS−HOWTO.html
- http://www.cc.gatech.edu/linux/LDP/HOWTO/CVS−HOWTO.html
- http://www.redhat.com/linux−info/ldp/HOWTO/CVS−HOWTO.html
- Other mirror sites near you (network−address−wise) can be found at http://sunsite.unc.edu/LDP/hmirrors.html select a site and go to directory /LDP/HOWTO/CVS−HOWTO.html

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 dvi document give the command −

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

And resize the window with 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 postscript file using the program 'gv' (ghostview) or 'ghostscript'. The ghostscript program is in ghostscript*.rpm package and gv program is in 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 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 postscript document give the command −

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

You can read 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 a X−Windows front end to latex.

---

19. Copyright and License

---

20. sget

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program sget
# Program to check out the file from CVS read-only

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

Usage()
{
        print "\nUsage: $cmdname [-r revision_number/symbolic_tag_name] <file/directory name> "
        print "The options -r are optional "
        print "For example - "
        print " $cmdname -r 1.1 foo.cpp"
        print " $cmdname foo.cpp "
        print " $cmdname some_directory "
        print "Extract by symbolic revision tag like - "
        print " $cmdname -r REVISION_1 some_directory "
        print " "
        exit
}

# Command getopt will not supported in next major release.
# Use getopts instead.
while getopts r: ii
do
        case $ii in
        r) FLAG1=$ii; OARG1="$OPTARG";;
        ?) Usage; exit 2;;
        esac
done
shift ` expr $OPTIND - 1 `

#echo FLAG1 = $FLAG1 , OARG1 = $OARG1

if [ $# -lt 1 ]; then
        Usage
fi

bkextn=sget_bak

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

cur_dir=`pwd`
#echo $cur_dir

len=${#homedir}
len=$(($len + 2))
#echo $len

subdir=` echo $cur_dir | cut -b $len-2000 `
#echo "subdir is : " $subdir
tmpaa=`dirname $1`
if [ "$tmpaa" = "." ]; then
```

```
        fname=$1
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
        fname=`basename $1`
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
#echo "subdir is : " $subdir
#echo "fname is : " $fname

# Check if file already exists....
if [ -f "$HOME/$subdir/$fname" ]; then
        tmpaa="$HOME/$subdir/$fname"
        user_perms=" "
        group_perms=" "
        other_perms=" "
        user_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b3-3 `
        group_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b6-6 `
        other_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b9-9 `
        if [ "$user_perms" = "w" -o "$group_perms" = "w"  \
                        -o "$other_perms" = "w" ]; then
                print "\nError: The file is writable. Aborting $cmdname ......"
                print "       You should either backup, scommit or delete the file and"
                print "       try $cmdname again\n"
                exit
        fi
fi

# Move the file
mkdir -p "$HOME/$subdir"
touch "$HOME/$subdir/$fname" 2>/dev/null
\mv -f "$HOME/$subdir/$fname" "$HOME/$subdir/$fname.$bkextn"

# Create subshell
(
        cd $homedir

        # Use -A option to clear all sticky flags
        if [ "$FLAG1" = "" ]; then
                if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                        cvs -r checkout -A $fname
                else
                        cvs -r checkout -A "$subdir/$fname"
                fi
        else
                if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                        cvs -r checkout -A -$FLAG1 $OARG1 $fname
                else
                        cvs -r checkout -A -$FLAG1 $OARG1 "$subdir/$fname"
                fi
        fi
)
#pwd

if [ -f "$HOME/$subdir/$fname" ]; then
        print "\nREAD-ONLY copy of the file $subdir/$fname obtained."
        print "Done $cmdname"
```

```
        #print "\nTip (Usage): $cmdname <file/directory name> \n"
fi
```

21. sedit

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program sedit
# Program to check out the file from CVS read/write mode with locking

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double−quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

Usage()
{
#       print "\nUsage: $cmdname [−r revision_number] [−F] <filename>"
#       print "The options −r, −F are optional "
#       print "The option −F is FORCE edit even if file is "
#       print "locked by another developer"

        print "\nUsage: $cmdname [−r revision_number] <filename>"
        print "The options −r are optional "

        print "For example − "
        print " $cmdname −r 1.1 foo.cpp"
        print " $cmdname foo.cpp "
#       print " $cmdname −F foo.cpp "
        print " "
}

# Command getopt will not supported in next major release.
# Use getopts instead.
#while getopts r:F ii
while getopts r: ii
do
        case $ii in
        r) FLAG1=$ii; OARG1="$OPTARG";;
#       F) FLAG2=$ii; OARG2="$OPTARG";;
        ?) Usage; exit 2;;
        esac
done
```

```
shift ` expr $OPTIND - 1 `

#echo FLAG1 = $FLAG1 , OARG1 = $OARG1

if [ $# -lt 1 ]; then
        Usage
        exit
fi

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

bkextn=sedit_bak

cur_dir=`pwd`
#echo $cur_dir

len=${#homedir}
len=$(($len + 2))
#echo $len

subdir=` echo $cur_dir | cut -b $len-2000 `
tmpaa=`dirname $1`
if [ "$tmpaa" = "." ]; then
        fname=$1
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
        fname=`basename $1`
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
#echo "subdir is : " $subdir
#echo "fname is : " $fname

# If file is already checked out by another developer....
cvs_root=` echo $CVSROOT | cut -f1 -d' '  `
if [ "$cvs_root" = "" ]; then
        print "\nError: \$CVSROOT is not set!!\n"
        exit
fi
mkdir -p "$CVSROOT/$subdir/Locks" 2>/dev/null

if [ ! -e "$CVSROOT/$subdir/$fname,v" ]; then
        print "\nError: File $fname does not exist in CVS repository!!\n"
        exit
fi

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
                print "You must do a sget on `basename $subdir` directory. Give -"
```

```
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
        fi
fi

# Get the tip revision number of the file....
# Use tmpfile as the arg cannot be set inside the sub-shell
tmpfile=$homedir/sedit-lock.tmp
\rm -f $tmpfile 2>/dev/null
if [ "$FLAG1" = "" ]; then
        (
        cd $homedir
                if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                        cvs log $fname | head -6 | grep head: | awk '{print $2}' > $tmpfile
                else
                        cvs log "$subdir/$fname" | head -6 | grep head: | awk '{print $2}' > $tmpfile
                fi
        )
        OARG1=`cat $tmpfile`
        \rm -f $tmpfile 2>/dev/null
fi

lockfile="$CVSROOT/$subdir/Locks/$fname-$OARG1"
#echo "lockfile is : " $lockfile
#if [ -e $lockfile -a "$FLAG2" = "" ]; then
if [ -e $lockfile ]; then
        print "\nError: File $fname Revision $OARG1 already locked by another developer !!"
        aa=` ls -l $lockfile | awk '{print "Locking developers unix login name is = " $3}' `
        print $aa
        print "That developer should do scommit OR sunlock to release the lock"
        print " "
#       print "You can also use -F option to force edit the file even if"
#       print "the file is locked by another developer. But you must talk to"
#       print "other developer to work concurrently on this file."
#       print "For example - this option is useful if you work on a seperate"
#       print "C++ function in the file which does not interfere with other"
#       print "developer."
#       print " "
        exit
fi

# Get read-only copy now....
if [ ! -e "$HOME/$subdir/$fname" ]; then
        (
        cd $homedir
                if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                        cvs -r checkout $fname 1>/dev/null
                else
                        cvs -r checkout "$subdir/$fname" 1>/dev/null
                fi
        )
```

```
fi

# Check if file already exists....
tmpaa="$HOME/$subdir/$fname"
if [ -f $tmpaa ]; then
        user_perms=" "
        group_perms=" "
        other_perms=" "
        user_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b3-3 `
        group_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b6-6 `
        other_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b9-9 `
        if [ "$user_perms" = "w" -o "$group_perms" = "w"  \
                        -o "$other_perms" = "w" ]; then
                print "\nError: The file is writable. Aborting $cmdname ......"
                print "       You must backup, scommit or delete file and"
                print "       try $cmdname again\n"
                exit
        fi
        #print "\nNote: The file $tmpaa is read-only."
        #print "Hence I am moving it to $tmpaa.$bkextn ....\n"
        \mv -f $tmpaa $tmpaa.$bkextn
        chmod 444 $tmpaa.$bkextn
elif [ -d $tmpaa ]; then
        print "\nError: $tmpaa is a directory and NOT a file. Aborting $cmdname ....\n"
        exit
fi

# Create subshell
print "\nNow getting the file $fname from CVS repository ...\n"
(
        cd $homedir
        # Use -A option to clear the sticky tag and to get
        # the HEAD revision version
        if [ "$FLAG1" = "" ]; then
                if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                        cvs -w checkout -A $fname
                else
                        cvs -w checkout -A "$subdir/$fname"
                fi
        else
                if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                        cvs -w checkout -A -$FLAG1 $OARG1 $fname
                else
                        cvs -w checkout -A -$FLAG1 $OARG1 "$subdir/$fname"
                fi
        fi
)

if [ -e "$HOME/$subdir/$fname" ]; then
        # The lockfile is $CVSROOT/$subdir/Locks/$fname-$OARG1
        touch $lockfile
        if [ -e $lockfile ]; then
                print "\nDone $cmdname"
        else
                print "\nFatal Error: File $fname Revision $OARG1 not locked !!"
                print "\nCheck the reason for this failure.. before proceeding..."
        fi
fi

#pwd

#print "\nTip (Usage): $cmdname <filename> \n"
```

22. scommit

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program scommit
# Program to commit the changes and check in the file into CVS

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

Usage()
{
        print "\nUsage: $cmdname [-r revision_number] <filename>"
        print "The options -r are optional "
        print "For example - "
        print " $cmdname -r 1.1 foo.cpp"
        print " $cmdname foo.cpp "
        print " "
}

# Command getopt will not supported in next major release.
# Use getopts instead.
while getopts r: ii
do
        case $ii in
        r) FLAG1=$ii; OARG1="$OPTARG";;
        ?) Usage; exit 2;;
        esac
done
shift ` expr $OPTIND - 1 `

#echo FLAG1 = $FLAG1 , OARG1 = $OARG1

if [ $# -lt 1 ]; then
        Usage
        exit 2
fi

if [ -d $1 ]; then
        Usage
```

```
        exit 2
fi

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

# Find sub-directory
cur_dir=`pwd`
#echo $cur_dir
len=${#homedir}
len=$(($len + 2))
#echo $len

subdir=` echo $cur_dir | cut -b $len-2000 `
tmpaa=`dirname $1`
if [ "$tmpaa" = "." ]; then
        fname=$1
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
        fname=`basename $1`
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
# echo "subdir is : " $subdir
# echo "fname is : " $fname

# If file is already checked out by another user....
cvs_root=` echo $CVSROOT | cut -f1 -d' '  `
if [ "$cvs_root" = "" ]; then
        print "\nError: \$CVSROOT is not set!!\n"
        exit
fi
mkdir -p "$CVSROOT/$subdir/Locks" 2>/dev/null

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
                print "You must do a sget on `basename $subdir` directory. Give -"
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
```

```
        fi
fi

# Get the working revision number of the file....
# Use tmpfile as the arg cannot be set inside the sub-shell
tmpfile=$homedir/sedit-lock.tmp
\rm -f $tmpfile 2>/dev/null
if [ "$FLAG1" = "" ]; then
        (
        cd $homedir
                if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                        cvs status $fname 2>/dev/null | grep "Working revision:" | awk '{print $3}' >$tmpfile
                else
                        cvs status "$subdir/$fname" 2>/dev/null | grep "Working revision:" | awk '{print $3}' >$tmpfile
                fi
        )
        OARG1=`cat $tmpfile`
        \rm -f $tmpfile 2>/dev/null
fi

if [ "$OARG1" = "" ]; then
        print "The file $subdir/$fname is NEW, it is not in the CVS repository"
else
        lockfile="$CVSROOT/$subdir/Locks/$fname-$OARG1"

        if [ -e $lockfile ]; then
                # Check if this revision is owned by you...
                aa=` ls -l $lockfile | awk '{print $3}' `
                userid=`id | cut -d'(' -f2 | cut -d')' -f1 `
                if [ "$aa" != "$userid" ]; then
                        print " "
                        print "The file $subdir/$fname is NOT locked by you!!"
                        print "It is locked by unix user name $aa and your login name is $userid"
#                       print "If you are working concurrently with other developer"
#                       print "and you used -F option with sedit."
                        print "You need to wait untill other developer does scommit"
                        print "or sunlock"
                        print "Aborting the $cmdname ...."
                        print " "
                        exit 2
                fi
        else
                # The file must exist in cvs
                if [ -f "$CVSROOT/$subdir/$fname,v" ]; then
                        print "You did not lock the file $subdir/$fname with sedit!!"
                        print "Aborting the $cmdname ...."
                        exit 2
                else
                        print "\nThe file $subdir/$fname does not exist in CVS repository yet!!"
                        print "You should have done sadd on $subdir/$fname ...."
                        exit 2
                fi
        fi
fi

# Operate inside sub-shell - and operate from root directory
(
        cd $homedir

        # Do not allow directory commits for now ...
        #if [ -d "$subdir/$fname" ]; then
        #       cvs commit "$subdir/$fname"
```

```
        #fi

        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                cvs commit $fname
        else
                cvs commit "$subdir/$fname"
        fi
        exit_status=$?

        if [ $exit_status -eq 0 ]; then
                lockfile="$CVSROOT/$subdir/Locks/$fname-$OARG1"
                if [ -e $lockfile ]; then
                        \rm -f $lockfile
                fi

                # Must change the permissions on file in case
                # there are no changes to file
                chmod a-w "$HOME/$subdir/$fname"
                print "\nDone $cmdname. $cmdname successful"
                #print "\nTip (Usage): $cmdname <filename/directory name>\n"
        fi
)
```

23. supdate

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program supdate
# Program to update the file from CVS read/write mode

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

if [ $# -lt 1 ]; then
        print "\nUsage: $cmdname <filename>"
        exit
fi

# Put double quotes to protect spaces in $1
tmpaa="$1"
```

```
# Check if file already exists....
if [ $# -gt 0 -a  -f $tmpaa ]; then
        user_perms=" "
        group_perms=" "
        other_perms=" "
        user_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b3-3 `
        group_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b6-6 `
        other_perms=`ls -l $tmpaa | awk '{print $tmpaa }' | cut -b9-9 `
        if [ "$user_perms" = "w" -o "$group_perms" = "w"  \
                        -o "$other_perms" = "w" ]; then
                while :
                do
                        print "\n$cmdname will backup your working file "
                        print "$tmpaa to $tmpaa.supdate_bak before doing any merges."
                        print "Are you sure you want the merge the changes from"
                        print -n "CVS repository to your working file ? <y/n> [n]: "
                        read ans
                        if [ "$ans" = "y" -o "$ans" = "Y" ]; then
                                if [ -f $tmpaa.supdate_bak ]; then
                                        print "\nWarning : File $tmpaa.supdate_bak already exists!!"
                                        print "Please examine the file $tmpaa.supdate_bak and delete it"
                                        print "and than re-try this $cmdname "
                                        print "Aborting $cmdname ...."
                                        exit
                                else
                                        cp $tmpaa $tmpaa.supdate_bak
                                        break
                                fi
                        elif [ "$ans" = "n" -o "$ans" = "N" -o "$ans" = "" -o "$ans" = " " ]; then
                                exit
                        fi
                done
        fi
fi

if [ -d $tmpaa ]; then
        print "\nDirectory update is disabled because cvs update"
        print "merges the changes from repository to your working directory."
        print "Hence give the filename to update - as shown below: "
        print " Usage: $cmdname <filename>"
        exit
#       cvs update
else
        cvs update $tmpaa
fi

print "\nDone $cmdname. $cmdname successful"
print "\n\nThe original file is backed-up to $tmpaa.supdate_bak"
print "\nHence your original file is SAVED to $tmpaa.supdate_bak"
print "\n\n"
#print "\nTip (Usage): $cmdname <filename/directory name>\n"
```

24. sunlock

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program sunlock
# Program to unlock the file to release the lock done by sedit

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

Usage()
{
        print "\nUsage: $cmdname [-r revision_number] <filename>"
        print " The options -r is optional "
        print "For example - "
        print " $cmdname -r 1.1 foo.cpp"
        print " $cmdname foo.cpp "
        print " "
}

# Command getopt will not supported in next major release.
# Use getopts instead.
while getopts r: ii
do
        case $ii in
        r) FLAG1=$ii; OARG1="$OPTARG";;
        ?) Usage; exit 2;;
        esac
done
shift ` expr $OPTIND - 1 `

if [ $# -lt 1 ]; then
        Usage
        exit
fi

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

cur_dir=`pwd`
#echo $cur_dir

len=${#homedir}
len=$(($len + 2))
#echo $len

subdir=` echo $cur_dir | cut -b $len-2000 `
```

```
#echo "subdir is : " $subdir
tmpaa=`dirname $1`
if [ "$tmpaa" = "." ]; then
        fname=$1
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
        fname=`basename $1`
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
#echo "subdir is : " $subdir
#echo "fname is : " $fname

cvs_root=` echo $CVSROOT | cut -f1 -d' '  `
if [ "$cvs_root" = "" ]; then
        print "\nError: \$CVSROOT is not set!!\n"
        exit
fi

if [ ! -e "$CVSROOT/$subdir/$fname,v" ]; then
        print "\nError: File $fname does not exist in CVS repository!!\n"
        exit
fi

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
                print "You must do a sget on `basename $subdir` directory. Give -"
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
        fi
fi

# Get the tip revision number of the file....
# Use tmpfile as the arg cannot be set inside the sub-shell
tmpfile=$homedir/sunlock-lock.tmp
\rm -f $tmpfile 2>/dev/null
if [ "$FLAG1" = "" ]; then
        # Operate inside sub-shell - from root directory
        (
        cd $homedir
                if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                        cvs log $fname | head -6 | grep head: | awk '{print $2}' > $tmpfile
```

```
                else
                        cvs log "$subdir/$fname" | head -6 | grep head: | awk '{print $2}' > $tmpfile
                fi
        )
        OARG1=`cat $tmpfile`
        \rm -f $tmpfile 2>/dev/null
fi

lockfile="$CVSROOT/$subdir/Locks/$fname-$OARG1"
#echo lockfile is : $lockfile
if [ ! -e $lockfile ]; then
        print "\nFile $fname revision $OARG1 is NOT locked by anyone"
        print " "
        exit
fi

ans=""
while :
do
        print "\n\n***************************************************"
        print "WARNING: $cmdname will release lock and enable other"
        print "         developers to edit the file. It is advisable"
        print "         to save your changes with scommit command"
        print "***************************************************"
        print -n "\nAre you sure you want to unlock the file <y/n>? [n]: "
        read ans
        if [ "$ans" = "" -o "$ans" = " " -o "$ans" = "n" -o "$ans" = "N" ]; then
                print "\nAborting $cmdname ...."
                exit
        fi
        if [ "$ans" = "y" -o "$ans" = "Y" ]; then
                print "\n\n\n\n\n "
                print "CAUTION: You may lose all the changes made to file!!"
                print -n "Are you sure? Do you really want to unlock the file <y/n>? [n]: "
                read ans
                if [ "$ans" = "y" -o "$ans" = "Y" ]; then
                        break
                elif [ "$ans" = "" -o "$ans" = " " -o "$ans" = "n" -o "$ans" = "N" ]; then
                        exit
                else
                        print "\n\nWrong entry. Try again..."
                        sleep 1
                fi
        else
                print "\n\nWrong entry. Try again..."
                sleep 1
        fi
done

if [ -e $lockfile ]; then
        \rm -f $lockfile
        print "\nDone $cmdname"
else
        print "\nFile $fname is NOT locked by anyone"
        print " "
fi
```

25. slist

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

Save this file as text file and chmod a+rx on it.

Note that there is also another unix command by name slist (list available Netware servers), you should make sure cvs script slist comes before other in your PATH environment.

```
#!/bin/ksh

# CVS program slist
# Program to list all edited source files from CVS

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

# Usage:
#               $ slist      (All files and sub-directories)
#               $ slist *.*      (All files)
#               $ slist *      (All files and sub-directories)
#               $ slist ab*      (All files starting with ab wild-card)

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

cur_dir=`pwd`
#echo $cur_dir

len=${#homedir}
len=$(($len + 2))
#echo $len

subdir=` echo $cur_dir | cut -b $len-2000 `
#echo "subdir is : " $subdir

# If file is already checked out by another developer....
cvs_root=` echo $CVSROOT | cut -f1 -d' '  `
if [ "$cvs_root" = "" ]; then
        print "\nError: \$CVSROOT is not set!!\n"
        exit
fi

# If the current directory tree is not in cvs-root then exit
if [ ! -d $CVSROOT/$subdir ]; then
        print "\nThe directory $subdir does not exist in $CVSROOT"
```

```
        exit
fi

#echo "no of params : " $#
#echo "The arg $ 1 is : " $1
#echo "all args : " $@

if [ $# -eq 0 ]; then
        tmpbb=` find * -prune -type d `
elif [ $# -eq 1 ]; then
        if [ "$1" = "." ]; then
                tmpbb=` find * -prune -type d `
        else
                if [ -d $1 -a ! -d $CVSROOT/$subdir/$1 ]; then
                        print "\nThe directory $subdir/$1 does not exist in $CVSROOT"
                        exit
                fi
                tmpbb=$@
        fi
else
        tmpbb=$@
fi

#echo "The tmpbb is : " $tmpbb

# Now, remove all the directory names which are not in cvs-root
dirnames=""
for ii in $tmpbb ; do
        if [ -d $CVSROOT/$subdir/$ii ]; then
                dirnames="$dirnames $ii "
        fi
done
#echo "The dirnames is : " $dirnames

if [ "$dirnames" != "" ]; then
        find $dirnames  -type f |
        while read ii
        do
                # List only those files which are in cvs system
                if [ -f "$CVSROOT/$subdir/$ii,v" ]; then
                        #echo "ii is : " $ii
                        ls -l $ii | grep ^\-rw
                fi
        done;
fi

# Get all the files in the current directory
listfiles=`ls $tmpbb `
find * -prune -type f |
while read ii
do
        for jj in $listfiles ; do
                if [ "$jj" = "$ii" ]; then
                        # List only those files which are in cvs system
                        if [ -f "$CVSROOT/$subdir/$ii,v" ]; then
                                #echo "ii is : " $ii
                                ls -l $ii | grep ^\-rw
                        fi
                fi
        done
done;
```

26. sinfo

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program sinfo
# Program to get the status of files in working directory

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

if [ $# -lt 1 ]; then
        print "\nUsage: $cmdname [file/directory name] "
        print "For example - "
        print " $cmdname foo.cpp"
        print " $cmdname some_directory "
        print " "
        exit
fi

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

cur_dir=`pwd`
#echo $cur_dir

len=${#homedir}
len=$(($len + 2))
#echo $len

subdir=` echo $cur_dir | cut -b $len-2000 `
#echo "subdir is : " $subdir
tmpaa=`dirname $1`
if [ "$tmpaa" = "." ]; then
        fname=$1
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
```

```
        fname=`basename $1`
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
#echo "subdir is : " $subdir
#echo "fname is : " $fname

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
                print "You must do a sget on `basename $subdir` directory. Give -"
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
        fi
fi

# Create subshell
if [ -f $1 ]; then
        (
        cd $homedir
                clear
                print "\ncvs status is : "
        cvs status "$subdir/$fname"
        )
elif [ -d $1 ]; then
        (
                cd $homedir
                clear
                print "\ncvs status is : "
                tmpfile="$homedir/cvs_sinfo.tmp"
                rm -f $tmpfile
        echo "  " >> $tmpfile
        echo "  ****************************************" >> $tmpfile
        echo "        Overall Status of Directory" >> $tmpfile
        echo "  ****************************************" >> $tmpfile
        cvs release "$subdir/$fname" 1>>$tmpfile 2>>$tmpfile << EOF
N
EOF
        echo "\n   -------------------------------\n" >> $tmpfile

        aa=`cat $tmpfile | grep ^"M " | awk '{print $2}' `
        for ii in $aa
        do
                        jj="(cd $homedir; cvs status \"$subdir/$ii\" );"
                        echo $jj | /bin/sh  \
```

```
                                | grep -v Sticky | awk '{if (NF != 0) print $0}' \
                                1>>$tmpfile 2>>$tmpfile
        done

        cat $tmpfile | grep -v ^? | grep -v "Are you sure you want to release" \
        | less
        rm -f $tmpfile
        )
else
        print "\nArgument $1 if not a file or directory"
        exit
fi
```

27. slog

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program slog
# Program to list history of the file in CVS

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

if [ $# -lt 1 ]; then
        print "\nUsage: $cmdname <filename> \n"
        exit
fi

# Check if file does not exist....
if [ ! -f "$1" ]; then
        print "\nError: $1 is NOT a file. Aborting $cmdname ......"
        exit
fi

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

cur_dir=`pwd`
```

```
#echo $cur_dir

len=${#homedir}
len=$(($len + 2))
#echo $len

subdir=` echo $cur_dir | cut -b $len-2000 `
#echo "subdir is : " $subdir
tmpaa=`dirname $1`
if [ "$tmpaa" = "." ]; then
        fname="$1"
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
        fname=`basename $1`
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
#echo "subdir is : " $subdir
#echo "fname is : " $fname

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
                print "You must do a sget on `basename $subdir` directory. Give -"
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
        fi
fi

# Operate inside a sub-shell
(
        cd $homedir
        cvs log "$homedir/$subdir/$fname" | less
)

print "\nDone $cmdname. $cmdname successful"
#print "\nTip (Usage): $cmdname <filename>\n"
```

28. sdif

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program sdif
# Program to see difference of the working file with CVS copy

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

Usage()
{
        print "\nUsage: $cmdname <filename> "
        print "$cmdname -r<rev1> -r<rev2> <filename> \n"
        exit
}

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

FLAG1=""
FLAG2=""
OARG1=""
OARG2=""
# Command getopt will not supported in next major release.
# Use getopts instead.
while getopts r:r: ii
do
        case $ii in
        r)
                if [ "$FLAG1" = "" ]; then
                        FLAG1=$ii;
                        OARG1="$OPTARG"
                else
                        FLAG2=$ii;
                        OARG2="$OPTARG"
                fi
                ;;
        ?) Usage; exit 2;;
        esac
done
```

```
shift ` expr $OPTIND - 1 `

if [ "$FLAG2" = "" ]; then
        FLAG2=r
        OARG2=HEAD
fi

cur_dir=`pwd`
#echo $cur_dir

len=${#homedir}
len=$(($len + 2))
#echo $len

subdir=` echo $cur_dir | cut -b $len-2000 `
#echo "subdir is : " $subdir
tmpaa=`dirname $1`
if [ "$tmpaa" = "." ]; then
        fname="$1"
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
        fname=`basename $1`
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
#echo "subdir is : " $subdir
#echo "fname is : " $fname

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
                print "You must do a sget on `basename $subdir` directory. Give -"
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
        fi
fi

# Operate inside sub-shell
(
        cd $homedir
        if [ "$FLAG1" = "" ]; then
                cvs diff -r HEAD "$homedir/$subdir/$fname" | less
        else
```

```
                cvs diff -$FLAG1 $OARG1 -$FLAG2 $OARG2 "$homedir/$subdir/$fname" | less
        fi
)
```

29. sadd

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program sadd
# Program to add the file to CVS

# Every 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 we will have $CVSROOT/$subdir/$fname
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`
if [ $# -lt 1 ]; then
        print "\nUsage: $cmdname <filename/directory> \n"
        exit
fi

onearg="$1"
if [ ! -f "$onearg" -a ! -d "$onearg" ]; then
        print "\nArgument $onearg is not a file or a directory!"
        print "Usage: $cmdname <filename/directory> \n"
        exit
fi

# Argument is a directory name .....
homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

cvs_root=` echo $CVSROOT | cut -f1 -d' '  `
if [ "$cvs_root" = "" ]; then
        print "\nError: \$CVSROOT is not set!!\n"
        exit
fi

cur_dir=`pwd`
len=${#homedir}
len=$(($len + 2))
```

```
subdir=` echo $cur_dir | cut -b $len-2000 `
#echo "subdir is : " $subdir
tmpaa=`dirname "$onearg" `
if [ "$tmpaa" = "." ]; then
        fname="$onearg"
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
        fname=`basename "$onearg" `
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
#echo "subdir is : " $subdir
#echo "fname is : " $fname

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
                print "You must do a sget on `basename $subdir` directory. Give -"
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
        fi
fi

# Check if file exists ....
if [ $# -eq 1 ]; then
        if [ -f "$onearg" ]; then
                cvs add "$onearg"
                exit
        fi
elif [ $# -gt 1 ]; then
        print "\n\n\nAdding all the files in the current directory to CVS"
        print "Directories will not be added"
        print -n "Hit return to continue or CTRL+C to abort..."
        read ans
        for ii in $@
        do
                if [ -f "$ii" ]; then
                        cvs add "$ii"
                fi
        done;
        exit
fi
```

```
# When $subdir is "." then you are at the root directory
if [ "$subdir" = "." ]; then
        # The $onearg is a directory and not a file at this point...
        if [ -d "$CVSROOT/$onearg" ]; then
                print "\nDirectory $onearg already exists in CVSROOT"
                exit
        else
                # You are adding at root directory $CVSROOT
                if [ "$2" = "" -o "$3" = "" ]; then
                        print "\nUsage: $cmdname <directory> <vendor tag> <release tag>"
                        print "For example - "
                        print " $cmdname foo_directory V_1_0 R_1_0"
                        exit
                else
                        (
                                cd "$homedir/$subdir";
                                cvs import "$onearg" $2 $3
                        )
                fi
        fi
else
        # If current directory exists in CVS...
        if [ -d "$CVSROOT/$subdir/$onearg" ]; then
                print "\nDirectory $onearg already in CVS repository!"
                exit
        else
                (
                        if [ -d "$homedir/$subdir/$onearg/CVS" ]; then
                                print "\nError: Directory $homedir/$subdir/$onearg/CVS exists!!"
                                print "\nAborting now ...."
                                exit
                        fi

                        # For import you MUST change to target directory
                        # and you MUST specify full-path starting with $subdir
                        cd "$homedir/$subdir/$onearg";
                        cvs import "$subdir/$onearg" Ver_1 Rel_1
                )
        fi
fi
```

30. sdelete

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program sdelete
# Program to delete the file from CVS

# Every 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 we will have $CVSROOT/$subdir/$fname
```

```
# In this program 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

# Caution: Put double-quotes to protect the variables having
#          spaces, like "$HOME/$subdir" if subdir is 'some foo.cpp'

cmdname=`basename $0`

if [ $# -lt 1 ]; then
        print "\nUsage: $cmdname <filename> \n"
        exit
fi

onearg="$1"

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

cur_dir=`pwd`
len=${#homedir}
len=$(($len + 2))
subdir=` echo $cur_dir | cut -b $len-2000 `
#echo "subdir is : " $subdir
tmpaa=`dirname "$onearg" `
if [ "$tmpaa" = "." ]; then
        fname="$onearg"
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        fi
else
        fname=`basename "$onearg" `
        if [ "$subdir" = "" ]; then
                subdir=$tmpaa
        else
                subdir="$subdir/$tmpaa"
        fi
fi
#echo "subdir is : " $subdir
#echo "fname is : " $fname

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
                print "You must do a sget on `basename $subdir` directory. Give -"
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
```

```
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
        fi
fi

# Operate inside a sub-shell ...
(
        cd $homedir

        # Check if file does not exist....
        if [ ! -f "$subdir/$fname" ]; then
                # Try to get the file from CVS
                sget "$subdir/$fname"
                if [ ! -f "$subdir/$fname" ]; then
                        print "\nError: $subdir/$fname does NOT exist in CVS repository."
                        print "\nAborting $cmdname ......"
                        exit
                fi
        fi

        bkextn=cvs_sdelete_safety_backup
        \mv -f "$subdir/$fname" "$subdir/$fname.$bkextn"

        cvs remove "$subdir/$fname"

        print "\nsdelete command removes the file from CVS repository"
        print "and archives the file in CVS Attic directory. In case"
        print "you need this file in future than contact your CVS administrator"
        print " "

        print "\nDone $cmdname. $cmdname successful"
        print "Run scommit on $homedir/$subdir/$fname to"
        print "make this change permanent"
        \mv -f "$subdir/$fname.$bkextn" "$subdir/$fname"
)
```

31. sfreeze

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

Save this file as text file and chmod a+rx on it.

```
#!/bin/ksh

# CVS program sfreeze
# Program to freeze and cut out the release of source tree from CVS

cmdname=`basename $0`

Usage()
{
        clear
        print "\nUsage: $cmdname symbolic_tag <directory name> "

        print "\nFor example :- "
```

```
        print "   cd \$HOME"
        print "   $cmdname  REVISION_1   myprojectsource_directory"
        print "To see the list of revisons do -"
        print "slog <filename> and see the symbolic name and do -"
        print "cvs history -T"

        print "\nTo create a branch off-shoot from main trunk, use"
        print "the -b and -r options which makes the tag a branch tag. This is"
        print "useful for creating a patch to previously released software"
        print "For example :- "
        print "   cd \$HOME"
        print "   cvs rtag -b -r REVISION_1   REVISION_1_1   myprojectsource_directory"
        print " "

#   print "\nTag info is located at \$CVSROOT/CVSROOT/taginfo,v"
#   print "You can do - cd $HOME; sget CVSROOT"
#   print "to see this file"
        exit
}

# Command getopt will not supported in next major release.
# Use getopts instead.
#while getopts r: ii
#do
#       case $ii in
#               r) FLAG1=$ii; OARG1="$OPTARG";;
#               ?) Usage; exit 2;;
#       esac
#done
#shift ` expr $OPTIND - 1 `

#echo FLAG1 = $FLAG1 , OARG1 = $OARG1

if [ $# -lt 2 ]; then
        Usage
fi

if [ ! -d $2 ]; then
        print "\nError: Second argument $2 is not a directory!"
        print "       Aborting $cmdname...."
        print " "
        exit
fi

homedir=` echo $HOME | cut -f1 -d' '  `
if [ "$homedir" = "" ]; then
        print "\nError: \$HOME is not set!!\n"
        exit
fi

cur_dir=`pwd`
len=${#homedir}
len=$(($len + 2))
subdir=` echo $cur_dir | cut -b $len-2000 `
#echo "subdir is : " $subdir

# CVS directory in your local directory is required for all commands..
if [ ! -d  "$homedir/$subdir/CVS" ]; then
        tmpaa=` (cd "$CVSROOT/$subdir"; find * -prune -type f -print | head -1 ) `
        tmpbb=`basename $tmpaa | cut -d',' -f1 `
        if [ "$tmpaa" = "" -o ! -f "$CVSROOT/$subdir/$tmpbb,v" ]; then
                print "\nThe directory $homedir/$subdir/CVS does not exist"
```

```
                print "You must do a sget on `basename $subdir` directory. Give -"
                print "       cd $homedir/`dirname $subdir` "
                print "       sget `basename $subdir` "
                exit
        else
                # Now try to create CVS in local dir by sget
                (
                        cd "$homedir"
                        if [ "$subdir" = "." ]; then  # don't use dot, will mess up cvs
                                cvs -r checkout -A $tmpbb
                        else
                                cvs -r checkout -A "$subdir/$tmpbb"
                        fi
                )
        fi
fi

if [ "$cur_dir" != "$homedir" ]; then
        print "\nYou are not in home directory $homedir!!"
        print "You must give the sfreeze command "
        print "from home directory $homedir"
        exit
fi

# cvs rtag symbolic_tag <directory name>
cvs rtag $1 $2

print "\nDone $cmdname. $cmdname successful"
```