

S-RCS- HOW-TO document for Linux (Source Code Control S

Table of Contents

<u>CVS–RCS– HOW–TO document for Linux (Source Code Control System)</u>	1
Al Dev (Alavoor Vasudevan) alavoor@yahoo.com	1
1. Introduction.....	1
2. Which one is for me? CVS or RCS.....	1
3. Setting up CVS.....	1
4. Shell Scripts.....	1
5. CVS Documentation.....	1
6. Emacs Editor.....	1
7. Problem Reporting System.....	2
8. Other Formats of this Document.....	2
9. Copyright and License.....	2
1. Introduction.....	2
2. Which one is for me? CVS or RCS.....	2
3. Setting up CVS.....	3
3.1 Environment variables.....	4
3.2 Migrate RCS to CVS.....	4
4. Shell Scripts.....	5
4.1 sget.....	6
4.2 sedit.....	8
4.3 scommit.....	11
4.4 supdate.....	13
4.5 sunlock.....	14
4.6 slist.....	17
4.7 sinfo.....	18
4.8 slog.....	19
4.9 sdif.....	20
4.10 sadd.....	21
4.11 sdelete.....	22
4.12 sfreeze.....	23
4.13 saddtree.....	24
5. CVS Documentation.....	25
6. Emacs Editor.....	26
7. Problem Reporting System.....	26
8. Other Formats of this Document.....	26
9. Copyright and License.....	28

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

AI Dev (Alavoor Vasudevan) alavoor@yahoo.com

v11.0, 10 June 2000

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

1. [Introduction](#)

2. [Which one is for me? CVS or RCS](#)

3. [Setting up CVS](#)

- [3.1 Environment variables](#)
- [3.2 Migrate RCS to CVS](#)

4. [Shell Scripts](#)

- [4.1 sget](#)
- [4.2 sedit](#)
- [4.3 scommit](#)
- [4.4 supdate](#)
- [4.5 sunlock](#)
- [4.6 slist](#)
- [4.7 sinfo](#)
- [4.8 slog](#)
- [4.9 sdif](#)
- [4.10 sadd](#)
- [4.11 sdelete](#)
- [4.12 sfreeze](#)
- [4.13 saddtree](#)

5. [CVS Documentation](#)

6. [Emacs Editor](#)

7. [Problem Reporting System](#)

8. [Other Formats of this Document](#)

9. [Copyright and License](#)

1. [Introduction](#)

Source code control system is a MUST to manage the changes occurring 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 a user checks out files/directories from the repository 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 or enable concurrent editing of files using shell scripts or PERL.

Disadvantages of CVS

- Needs a little more administration than RCS
- Very highly sophisticated and complex system. It is the "State of the Art" technology.
- 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

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 CVSRROOT=/home/cvsroot
export CVSREAD=yes
```

Create a directory to store the source code repository and give read, write access to unix group/user.

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

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'
#         fi
        else
            echo >&2 "`basename "$0"`: can't change to RCS subdirectory '$RCSROOT/$RCS'."
        fi)
    done
done

```

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

4. [Shell Scripts](#)

The following are wrappers around the basic 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 scripts 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.

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 a 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](#)
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](#)
13. **saddtree** <revision name> <directory name> To add a directory tree to CVS. Click [saddtree](#)

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

4.1 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

cmdname=`basename $0`
```


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

```
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 getopt's 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

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

# Check if file already exists....
if [ -f $1 ]; then
    user_perms=" "
    group_perms=" "
    other_perms=" "
    user_perms=`ls -l $1 | awk '{print $1 }' | cut -b3-3`
    group_perms=`ls -l $1 | awk '{print $1 }' | cut -b6-6`
    other_perms=`ls -l $1 | awk '{print $1 }' | cut -b9-9`
    if [ "$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

cur_dir=`pwd`
#echo $cur_dir

len=${#hme}
len=$((len + 2))
#echo $len
```

```

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

if [ "$subdir" = "" ]; then
    fdname=$1
else
    fdname=$subdir/"$1
fi

# Move the file
touch $1 2>/dev/null
\mv -f $1 $1.$bkextn

# Create subshell
(
cd $hme
#echo $fdname

# Use -A option to clear all sticky flags
if [ "$FLAG1" = "" ]; then
    cvs -r checkout -A $fdname
else
    cvs -r checkout -A -$FLAG1 $OARG1 $fdname
fi
)
#pwd

if [ -f $1 ]; then
    print "\nREAD-ONLY copy of the file $fdname obtained."
    print "Done $cmdname"
    #print "\nTip (Usage): $cmdname <file/directory name> \n"
fi

```

4.2 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

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 getopt instead.
#while getopt r:F ii
while getopt 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

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

bkextn=sedit_bak

cur_dir=`pwd`
#echo $cur_dir

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

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

if [ "$subdir" = "" ]; then
    fdname=$1
else
    fdname=$subdir/"$1"
fi

# 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
cldir=$CVSROOT/$subdir/Locks
mkdir $cldir 2>/dev/null
rcsfile=$CVSROOT/$subdir/$1,v
#echo $rcsfile

if [ ! -e $rcsfile ]; then

```

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

```
print "\nError: File $1 does not exist in CVS repository!!\n"
exit

fi

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

lockfile=$cldir/$1-$OARG1
#if [ -e $lockfile -a "$FLAG2" = "" ]; then
if [ -e $lockfile ]; then
    print "\nError: File $1 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 $1 ]; then
(
    cd $hme
    cvs -r checkout $fdname 1>/dev/null
)
fi

# Check if file already exists....
if [ -f $1 ]; then
    user_perms=" "
    group_perms=" "
    other_perms=" "
    user_perms=`ls -l $1 | awk '{print $1 }' | cut -b3-3`
    group_perms=`ls -l $1 | awk '{print $1 }' | cut -b6-6`
    other_perms=`ls -l $1 | awk '{print $1 }' | cut -b9-9`
    if [ "$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 $1 is read-only."
    #print "Hence I am moving it to $1.$bkextn ....\n"
    \mv -f $1 $1.$bkextn
    chmod 444 $1.$bkextn
```

```

elif [ -d $1 ]; then
    print "\nError: $1 is a directory and NOT a file. Aborting $cmdname ....\n"
    exit
fi

# Create subshell
print "\nNow getting the file $1 from CVS repository ... \n"
(
    cd $hme
    #echo $fdname
    # Use -A option to clear the sticky tag and to get
    # the HEAD revision version
    if [ "$FLAG1" = "" ]; then
        cvs -w checkout -A $fdname
    else
        cvs -w checkout -A -$FLAG1 $OARG1 $fdname
    fi
)

if [ -e $1 ]; then
    touch $lockfile
fi

#pwd

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

```

4.3 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

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

```

```

        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

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

# Find sub-directory
cur_dir=`pwd`
#echo $cur_dir
len=${#hme}
len=$((len + 2))
#echo $len
subdir=`echo $cur_dir | cut -b $len-2000`
#echo $subdir
if [ "$subdir" = "" ]; then
    fdname=$1
else
    fdname=$subdir/"$1
fi

# 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
cldir=$CVSROOT/$subdir/Locks
mkdir $cldir 2>/dev/null

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

if [ "$OARG1" = "" ]; then
    print "The file $fdname is NEW, it is not in the CVS repository"
else
    lockfile=$cldir/$1-$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 $fdname 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
    if [ -f $CVSROOT/$subdir/$1,v ]; then
        print "You did not lock the file $fdname with sedit!!"
        print "Aborting the $cmdname ...."
        exit 2
    else
        print "\nThe file $fdname does not exist in CVS repository yet!!"
        print "You should have done sadd on $fdname ...."
    fi
fi

fi

if [ -d $1 ]; then
    Usage
    exit 2
    # Do not allow directory commits for now ...
    #cvs commit
else
    cvs commit $1
    exit_status=$?
fi

if [ $exit_status -eq 0 ]; then
    print "\nDone $cmdname. $cmdname successful"
    #print "\nTip (Usage): $cmdname <filename/directory name>\n"
fi

```

4.4 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

cmdname=`basename $0`

if [ $# -lt 1 ]; then

```

```

        print "\nUsage: $cmdname <filename>"
        exit
    fi

    # Check if file already exists....
    if [ $# -gt 0 -a -f $1 ]; then
        user_perms=" "
        group_perms=" "
        other_perms=" "
        user_perms=`ls -l $1 | awk '{print $1 }' | cut -b3-3 `
        group_perms=`ls -l $1 | awk '{print $1 }' | cut -b6-6 `
        other_perms=`ls -l $1 | awk '{print $1 }' | cut -b9-9 `
        if [ "$user_perms" = "w" -o "$group_perms" = "w" \
            -o "$other_perms" = "w" ]; then
            while :
            do
                print "\n$cmdname will backup your working file "
                print "$1 to $1.supdate_bak before doing any merges."
                print "Are you sure you want the merge the changes from"
                print -n "CVS repository to your working file ? <y/n> [n]: "
                read ans
                if [ "$ans" = "y" -o "$ans" = "Y" ]; then
                    if [ -f $1.supdate_bak ]; then
                        print "\nWarning : File $1.supdate_bak already exists!!"
                        print "Please examine the file $1.supdate_bak and delete"
                        print "and than re-try this $cmdname "
                        print "Aborting $cmdname ...."
                        exit
                    else
                        cp $1 $1.supdate_bak
                        break
                    fi
                elif [ "$ans" = "n" -o "$ans" = "N" -o "$ans" = " " -o "$ans" = " " ]; then
                    exit
                fi
            done
        fi
    fi

    if [ -d $1 ]; then
        print "\nDirectory update is disabled as cvs update"
        print "merges the changes from repository to your working directory"
        print "So give the filename to update - as shown below: "
        print " Usage: $cmdname <filename>"
        exit
    #
    # cvs update
    else
        cvs update $1
    fi

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

```

4.5 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

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

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

cur_dir=`pwd`
#echo $cur_dir

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

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

if [ "$subdir" = "" ]; then
    fdname=$1
else
    fdname=$subdir/"$1"
fi

# 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
```

```

fi
cldir=$CVSROOT/$subdir/Locks
rcsfile=$CVSROOT/$subdir/$1,v
#echo $rcsfile

if [ ! -e $rcsfile ]; then
    print "\nError: File $1 does not exist in CVS repository!!\n"
    exit
fi

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

lockfile=$cldir/$1-$OARG1
#echo lockfile is : $lockfile
if [ ! -e $lockfile ]; then
    print "\nFile $1 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 "$ans" = "Y" ]; then
        print "\n\n\n\n "
        print "CAUTION: You may lose all the changes made to file!!"
        print -n "Do you really want to unlock the file <y/n>? [n]: "
        read ans
        if [ "$ans" = "y" -o "$ans" = "Y" ]; then
            break
        else
            exit
        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 $1 is NOT locked by anyone"
        print " "
fi

```

4.6 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.

```

#!/bin/ksh

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

#cmdname=`basename $0`

#echo "no of params : " $#
#echo "all args : " $@

recurse_flag=""

if [ "$1" = "" ]; then
    dir=.
    recurse_flag=""
else
    dir=$@
    recurse_flag=" -prune "
fi

FOUT=slist_temporary_file.out

\rm -f $FOUT

find $dir $recurse_flag -type f -exec ls -ltr {} \; \
| grep -v "/CVS/" \
| grep ^\-rw \
| grep -v /\.o \
| grep -v /\.log \
| grep -v /\.out \
| grep -v /\.pid \
| awk '{ if ($NF != "tags") print $0 }' \
| awk '{ if ($NF != "a.out") print $0 }' \
| awk '{ if ($NF != "core") print $0 }' \
| awk '{ print $NF }' > $FOUT

aa=`cat $FOUT`
\rm -f $FOUT

for ii in $aa ; do
    ftype=" "
    ftype=`file $ii | awk '{print $2 }'`

    # find . -type f -exec file {} \;

```

```

# 1)ELF 2)commands 3)[nt]roff, 4)c 5)English 6)executable
# 7)ascii 8)current 9)empty
# Binaries are ELF, lib.a are current
#
if [ "$ftype" = "ascii" -o "$ftype" = "commands" \
    -o "$ftype" = "[nt]roff," -o "$ftype" = "c" -o "$ftype" = "data" \
    -o "$ftype" = "English" -o "$ftype" = "executable" ]; then
pcfile=` echo $ii | cut -d'.' -f1`
pcfile=${pcfile}.pc
if [ ! -f $pcfile ]; then
    ls -l $ii
else
    if [ "$ii" = "$pcfile" ]; then
        ls -l $ii
    fi
fi
done;

#| grep -v ^\-rwx \

#ls -l | grep ^\-rw | grep -v \\.o
#ls -l | grep ^\-rw | grep -v \\.o | awk '{ if ($NF != "tags") print $0 }'
#ls -l | grep ^\-rw | grep -v ^\-rwx | grep -v \\.o | awk '{ if ($NF != "tags") print $0 }' | aw

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

```

4.7 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

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

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

tmpfile=$hme/cvs_sinfo.tmp

```

```

rm -f $tmpfile

cur_dir=`pwd`
#echo $cur_dir

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

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

if [ "$subdir" = "" ]; then
    fdname=$1
else
    fdname=$subdir/"$1
fi

# Create subshell
if [ -f $1 ]; then
    (
        cd $hme
        clear
        cvs status $fdname
    )
elif [ -d $1 ]; then
    (
        cd $hme
        clear
        echo " " >> $tmpfile
        echo " *****" >> $tmpfile
        echo " Overall Status of Directory" >> $tmpfile
        echo " *****" >> $tmpfile
        cvs release $fdname 1>>$tmpfile 2>>$tmpfile << EOF
Y
EOF
        echo "\n ----- \n" >> $tmpfile

        aa=`cat $tmpfile | grep ^"M " | awk '{print $2}' `
        for ii in $aa
        do
            jj="(cd $hme; 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

```

4.8 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

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

cvs log $1 | /usr/local/bin/less

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

4.9 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

cmdname=`basename $0`

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

FLAG1=""
FLAG2=""
OARG1=""
OARG2=""
# Command getopt will not supported in next major release.
# Use getopt 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

if [ "$FLAG1" = "" ]; then
    cvs diff -r HEAD $1 | less
else
    cvs diff -s$FLAG1 $OARG1 -s$FLAG2 $OARG2 $1 | less
fi

```

4.10 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

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

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

# Check if file exists ....
if [ -f $1 ]; then
    cvs add $1
    exit
fi

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

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

```

```

        exit
    fi

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

    if [ "$subdir" = "" ]; then
        if [ -d $CVSROOT/$1 ]; then
            print "\nDirectory $1 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 $1;
                    cvs import $1 $2 $3
                )
            fi
        fi
    else
        # If current directory exists in CVS...
        if [ -d $CVSROOT/$subdir ]; then
            if [ -d $CVSROOT/$subdir/$1 ]; then
                print "\nDirectory $1 already in CVS repository!"
            else
                cvs add $1
            fi
        else
            print "\nSub-directory $subdir does not exist in CVS"
            print "You need to first add $subdir to CVS"
            exit
        fi
    fi
fi

```

4.11 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

cmdname=`basename $0`

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

```



```

fi

# Check if file does not exist...
if [ ! -f $1 ]; then
    # Try to get the file from CVS
    sget $1
    if [ ! -f $1 ]; then
        print "\nError: $1 does NOT exist in CVS repository. Aborting $cmdname ....."
        exit
    fi
fi

bkextn=cvs_sdelete_safety_backup
\mv -f $1 $1.$bkextn

cvs remove $1

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 "\nTip (Usage): $cmdname <filename>\n"
\mv -f $1.$bkextn $1

```

4.12 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()
{
    print "\nUsage: $cmdname symbolic_tag <directory name> "

    print "\nFor example :- "
    print "    cd \${HOME}"
    print "    $cmdname REVISION_1 mesa"
    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 mesa"
    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

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

print "\nDone $cmdname. $cmdname successful"

```

4.13 saddtree

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

#####
# Sample Program to checkin a directory tree (let's say SAMP) into CVS
# Note that if SAMP directory is not in CVS than you would use sadd
# command and -
#      cd SAMP; cvs import SAMP V1_0 R1_0
# After running this program do -
#      cd $HOME/foo/SAMP
#      cvs import foo/SAMP V1_0 Rev1_0
#####

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

```

```

        exit
    fi
    sampdir=$hme/foo/SAMP

check_out_files()
{
    # Now check out the files
    tmp2f=$hme/tmp2.baksamp.sh

    cd $hme
    \rm -rf foo/SAMP
    cvs -w checkout foo/SAMP
    cd $hme/foo
    find SAMP -type f -print > $tmp2f

    cd $hme
    for ii in `cat $tmp2f`
    do
        iidir=`dirname $ii`
        iifile=`basename $ii`
        if [ "$iifile" = "Root" -o "$iifile" = "Repository" -o "$iifile" = "Entries" ]; then
            continue
        fi
        jjdir=` echo $iidir | cut -d'/' -f2-1000 `

        cp $hme/foo/SAMP.tobe/$jjdir/$iifile $hme/foo/$iidir/$iifile
        echo "cp $hme/foo/SAMP.tobe/$jjdir/$iifile $hme/foo/$iidir/$iifile "

        cvs add foo/$iidir/$iifile
    done

    print
    print "===== "
    print " Now run cvs commit foo/SAMP"
    print " After commit. Do - "
    print "          cd foo; rm -rf SAMP and "
    print " get fresh copy, sget SAMP"
    print " Verify with slog filename.samp to see new revision"
    print "===== "
    print
}

check_out_files

```

5. 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'. Then 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> WinCVS can be used along with Samba – <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.
-

6. [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>.

7. [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.

8. [Other Formats of this Document](#)

This document is published in 11 different formats namely – DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU–info, HTML, 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/>
- Plain text format is in: <ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO>

- Translations to other languages like French, German, Spanish, Chinese, Japanese are in <ftp://sunsite.unc.edu/pub/Linux/docs/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.

9. [Copyright and License](#)

Copyright Al Dev (Alavoor Vasudevan) 1998–2000.

License is GNU GPL, but it is requested that you retain the author's name and email on all copies.
