1. CVS -- Concurrent Versions System

1.1 SYNOPSIS

cvs [cvs_options]

cvs_command [command_options] [command_args]

1.2 Contents:

- Introduction
- Commands
- Files
- Environment variables
- Management
- RCS

1.3 Authors

Dick Grune -- Original author of the cvs shell script version posted to comp.sources.unix in the volume6 release of December, 1986. Credited with much of the cvs conflict resolution algorithms.

Brian Berliner -- Coder and designer of the cvs program itself in April,1989, based on the original work done by Dick.

Jeff Polk -- Helped Brian with the design of the cvs module and vendor branch support and author of the checkin(1) shellscript (the ancestor of 'cvs import').

1.4 SEE ALSO

ci(1), co(1), cvs(5), diff(1), grep(1), mkmodules(1),patch(1), rcs(1), rcsdiff(1), rcsmerge(1), rlog(1), rm(1),sort(1).

1.5 COPYING

1.6 Introduction to CVS

cvs is a front end to the rcs (1) revision control system which extends the notion of revision control from a collection of files in a single directory to a hierarchical collection of directories consisting of revision controlled files. These directories and files can be combined together to form a software release. cvs provides the functions necessary to manage these software releases and to control the concurrent editing of source files among multiple software developers.
cvs keeps a single copy of the master sources. This copy is called the source "repository"; it contains all the information to permit extracting previous software releases at any time based on either a symbolic revision tag, or a date in the past.

1.6.1 ESSENTIAL 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.

1.6.2 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.

1.6.3 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.

1.6.4 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.

1.6.5 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'.

1.6.6 cvs commit file...

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

1.6.7 OPTIONS

The cvs command line can include cvs_options, which apply to the overall cvs program; a cvs_command, which specifies aparticular action on the source repository; and command_options and command_arguments to fully specify what the cvs_command will do.
Warning: you must be careful of precisely where you place options relative to the cvs_command. The same option can mean different things depending on whether it is in the cvs_options position (to the left of a cvs command) or in the command_options position (to the right of a cvs command).
There are only two situations where you may omit cvs_command: 'cvs -H' elicits a list of available commands and 'cvs -v ' displays version information on cvs itself.

1.6.8 CVS OPTIONS

Use these options for overall control of the cvs program:

H Display usage information about the specified cvs_command (but do not actually execute the command). If you don't specify a command name, 'cvs -H' displays a summary of all the commands available.

Q Causes the command to be really quiet; the command will generate output only for serious problems.

q Causes the command to be somewhat quiet; informational messages, such as reports of recursion through subdirectories, are suppressed.

b bindir Use bindir as the directory where RCS programs are located. Overrides the setting of the RCSBIN environment variable. This value should be specified as an absolute pathname.

d CVS_root_directory Use CVS_root_directory as the root directory pathname of the master RCS source repository. Overrides the setting of the CVSROOT environment variable. This value should be specified as an absolute pathname.

e editor Use editor to enter revision log information. Overrides the setting of the EDITOR environment variable.

l Do not log the cvs_command in the command history (but execute it anyway). See the description of the history command for information on command history.

n Do not change any files. Attempt to execute the cvs_command, but only to issue reports; do not remove, update, or merge any existing files, or create any new files.

t Trace program execution; display messages showing the steps of cvs activity. Particularly useful with -n to explore the potential impact of an unfamiliar command.

r Makes new working files files read -only. Same effect as if the CVSREAD environment variable is set.

v Displays version and copyright information for cvs.

w Makes new working files read -write (default). Overrides the setting of the CVSREAD environment variable.

1.6.9 USAGE

Except when requesting general help with 'cvs -H', you mustspecify a cvs_command to cvs to performa specific release control function. Each cvs command accepts its own collection of options and arguments. However, many options are available across several commands. You can display a usage summary for each command by specifying the -H option with the command.

1.6.10 CVS COMMAND SUMMARY

Here are brief descriptions of all the cvs commands :

add Add a new file or directory to the repository, pending a 'cvs commit' on the same file. Can only be done from within sources created by a previous 'cvs checkout' invocation. Use 'cvs import' to place whole new hierarchies of sources under cvs control. (Does not directly affect repository; changes working directory.)

admin Execute RCS control functions on the source repository. (Changes repository directly; uses working directory without changing it.)

checkout Make a working directory of source files for editing. (Creates or changes working directory.)

commit Apply to the source repository changes, additions, and deletions from your working directory. (Changes repository.)

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

export Prepare copies of a set of source files for shipment off-site. Differs from 'cvs checkout' in that no cvs administrative directories are created (and therefore 'cvs commit' cannot be executed from a directory prepared with 'cvsexport'), and a symbolic tag must be specified. (Does not change repository; creates directory similar to working directories).

history Show reports on cvs commands that you or others have executed on a particular file or directory in the source repository. (Does not change repository or working directory.) History logs are kept only if enabled by creation of the '$CVSROOT/CVSROOT/history' file; see cvs(5).

import Incorporate a set of updates from off -site into the source repository, as a "vendor branch". (Changes repository.)

log Display RCS log information. (Does not change repository or working directory.)

rdiff Prepare a collection of diffs as a patch file between two releases in the repository. (Does not change repository or working directory.)

release Cancel a 'cvscheckout' abandoning any changes. (Can delete working directory; no effect on repository.)

remove Remove files from the source repository, pending a 'cvs-commit' on the same files. (Does not directly affect repository; changes working directory.)

rtag Explicitly specify a symbolic tag for particular revisions of files in the source repository. See also 'cvstag'. (Changes repository directly; does not require or affect working directory.)

status Show current status of files: latest version, versionin working directory, whether working version has been edited and, optionally, symbolic tags in the RCS file. (Does not change repository or working directory.)

tag Specify a symbolic tag for files in the repository. Tags the revisions that were last synchronized with your working directory. (Changes repository directly; uses working directory without changing it.)

update Bring your working directory up to date with changesfrom the repository. Merges are performed automatically when possible; a warning is issued if manualresolution is required for conflicting changes.(Changes working directory; does not change repository.)

1.6.11 COMMON COMMAND OPTIONS

This section describes the command_options that are available across several cvs commands. Not all commands support all of these options; each option is only supported for commands where it makes sense. However, when a command has one of these options you can count on the same meaning for the option as in other commands. (Other command options, which are listed with the individual commands, may have different meanings from one cvs command to another.) Warning: the history command is an exception; it supports many options that conflict even with these standard options.

D date_spec Use the most recent revision no later than date_spec (a single argument, date description specifying a date in the past). A wide variety of date formats are supported by the underlying RCS facilities, similar to those described in co(1), but not exactly the same. The date_spec is interpreted as being in the local time zone, unless a specific time zone is specified. The specification is "sticky" when you use it to make a private copy of a source file; that is, when you get a working file using -D, cvs records the date you specified, so that further updates in the same directory will use the same date (unless you explicitly override it; see the description of the update command). -D is available with the checkout, diff, history, rdiff, rtag, and update commands. Examples of valid date specifications include:

```
1 month ago
2 hours ago
400000 seconds ago
last year
last Monday
yesterday
a fortnight ago
3/31/92 10:00:07 PST
January 23, 1987 10:05pm
22:00 GMT
```

f When you specify a particular date or tag to cvs commands, they normally ignore files that do not contain the tag (or did not exist on the date) that you specified. Use the -f option if you want files retrieved even when there is no match for the tag or date. (The most recent version is used in this situation.) -f is available with these commands: checkout, export, rdiff, rtag, and update.

-H Help; describe the options available for this command. This is the only option supported for all cvs commands.

-k kflag Alter the default RCS processing of keywords; all the -k options described in rcs(1) are available. The -k option is available with the add, checkout, diff, rdiff, and update commands. Your kflag specification is "sticky" when you use it to create a private copy of a source file; that is, when you use this option with the checkout or update commands, cvs associatesyour selected kflag with the file, and continues to use it with future update commands on the same file until you specify otherwise.

-l Local; run only in current working directory, rather than recurring through subdirectories. Available with the following commands: checkout, commit, diff, export,remove, rdiff, status, tag, and update. Warning: this is not the same as the overall 'cvs -l' option, which you can specify to the left of a cvs command!

-n Do not run any checkout/commit/tag/ program. (A program can be specified to run on each of these activities, in the modules database; this option bypasses it.) Available with the checkout, commit, export, and rtag commands. Warning: this is not the same as theoverall 'cvs -n' option, which you can specify to the left of a cvs command!

-P Prune (remove) directories that are empty after being updated, on checkout, or update. Normally, an empty directory (one that is void of revision -controlled files) is left alone. Specifying -P will cause these directories to be silently removed from your checked -out sources. This does not remove the directory from the repository, only from your checked out copy. Note that this option is implied by the -r or -D options of checkout and export.