NAME

rcsput − RCS check-in utility

USAGE

rcsput *[options] [file-specifications]*

SYNOPSIS

Rcsput is a simple, easy to use interface to *rcs* (**r**evision **c**ontrol **s**ystem). For each file specified as input, it checks differences against the previously archived version and prompts you for change history comments.

DESCRIPTION

Rcsput uses the *rcs* utility **ci** to maintain versions of a given source file in a dependent directory named "RCS":

- It checks to ensure that each file is indeed a text file (so that you do not accidentally archive ".o" files, for example).
- If you give **rcsput** a directory name, it will recur, checking-in files in the directory.
- For each file which has a corresponding ",v" file, **rcs put** compares the two (using **diff**) and pipes the result through the pager.
- An option is provided so that you may direct **rcsput** to perform the differencing without checking the file into rcs.
- The ",v" file is post-processed by **rcsput** so that the check-in date matches the file's modification date.

The last point is the fundamental advantage offered by **rcsput**. The ordinary *rcs* methodology uses the current date as the check-in date. This works well only for large projects in which a central project administrator is responsible for controlling the versions of source files. It does not work well for small projects, for which *rcs*'s primary advantage is its compact storage of multiple versions of a file.

By using the file's modification date as a reference, you can more easily back up to a meaningful version − by date, rather than version number.

Rcsput integrates all of the functions used in the *rcs* check-in process into one utility program.

OPTIONS

Some of the options which you may specify to **rcsput** are passed through to the underlying **ci** utility. Others represent extensions:

−b is passed to **diff**, and directs it to ignore trailing blanks on a line, and to treat repeated blanks as a single blank.

−c directs **rcsput** to use **cat** rather than the **PAGER** (usually **more**) to display differences. This is most useful in an Apollo pad, since the **more** program would otherwise switch to VT100 emulator mode.

−d instructs **rcsput** to test for differences, but not to check the files into *rcs*.

−h is passed to **diff**, and permits it to handle huge differences.

−L *file*
causes **rcsput** to generate a log-file of the files which are processed, and all differences which are encountered. The log-file is inherited in recursion to lower directory levels (i.e., it is written to the same place). If no argument is specified, rcsput assumes "logfile".

−T *path*
specifies an alternate tool to invoke, overrides the default "**checkin**".

OPERATIONS

The **rcsput** utility is designed for use in small development projects. The methodology for this tool follows:

- Develop source files "normally". Each file should contain rcs keywords (see *ci (1)*) so that you will be able to distinguish checked-out files. The rcs keywords should appear at the top of your source file, for consistency. In C language programs, the convention is to make a string which will permit the **ident** utility to show the versions of the modules which make up a program:

```
#ifndef lint
static char ident[] = "$Id: rcsput.man,v 10.1 92/02/06 10:01:45 dickey Exp $";
#endif
```

- Periodically archive (with **rcsput**) those versions of files which you wish to keep (you should never have programs which have new features which you wish to keep, while there are defects in other parts of the program. That would be an unsound approach to development!).
- When you reach the point of releasing the program, ensure that all source files have been checked-in. The directory editor (**ded**) is useful for reviewing the check-in dates.
- Copy the directory containing your program to the release directory. Purge all files, except those which are stored in the *rcs* subdirectories. Use **rcsget** to extract the files. The unadorned **co** utility will work, of course, but it retains the file modification dates. You may also use **checkout** to retain file dates.
- Ensure that all files have been checked-in and released. You may use **diff** to compare the directories − the only differences should be the substituted *rcs* keywords.
- Build the released version of your program. All files should be present. No embedded path names should refer to your development copy. To ensure good isolation, you may change the permissions on your development directory temporarily.

When checking files into *rcs*, it is a good idea to make a test run (using the "**−d**" option) so that you can inspect the differences. For example, you may have forgotten to remove (or bypass) debugging stubs. Or, you may have been editing a checked-out file (with the *rcs* keywords substituted). **Rcsput** would archive this anyway.

ENVIRONMENT

Rcsput is written in C, and runs on POSIX systems.

Environment variables imported by **rcsput** include:

PAGER
identifies the program to use in displaying differences between the file which is being checked in, and the previously archived version. There may be a lot of differences − more than can be shown on one screen.

FILES

Rcsput uses the following files

checkin
A utility which invokes **ci**, and modifies the *rcs* ",v" file after check-in so that the check-in date matches the file's modification date.

ANTICIPATED CHANGES

None.

SEE ALSO

checkin, rcsget, checkout, ded, ci (1), co (1), diff (1), ident (1)

AUTHOR:

Thomas E. Dickey <dickey@invisible-island.net>