NAME

checkin – RCS check-in utility

SYNOPSIS

checkin *[options] [file-specifications]*

DESCRIPTION

Checkin is an extension of the RCS utility **ci**. It uses the file's modification date rather than the current date as the RCS delta-date.

Checkin uses the *rcs* utility **ci**. It is normally invoked from the **rcsput** script, but may be invoked in a standalone manner. **Checkin** differs from **ci** primarily in its treatment of the delta date: after invoking **ci**, **checkin** modifies the delta-date in the archive to reflect the file's modification date.

This is the fundamental advantage offered by **checkin**. 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.

Archive Directory

If the archive directory (e.g., "./RCS") does not exist, **checkin** creates it before invoking the **ci** program.

Set-UID Operation

The rcs **ci** and **co** utilities work to a degree in set-uid mode (i.e., the "u+s" protection is set on the programs). However, the code assumes that the effective uid is **root**, and does not concern itself with maintaining file ownership.

The **checkin** package is able to run as a set-uid process for any particular user (e.g., the administrator of a project). For example, suppose that **/proj** is the location of project-specific tools, and is owned by **admin**. Then (running as the **admin** user):

```
cp checkin /proj            # admin now owns this copy
chmod 4755 /proj/checkin    # sets u+s mode
```

Thereafter, users who invoke **/proj/checkin** will have the rights of **admin** – for this application. They may check into *rcs* any files which they own, into archives which **admin** owns. **Checkin** will maintain **admin**'s ownership of the archive files, and the user's ownership of his working files.

If **checkin** does not need the set-uid rights (e.g., if the user already owns the archive), **checkin** resets its effective uid to the user's. This permits a single copy of **checkin** to be used for both configuration management as well as individual developers.

Sharing RCS Archives

Checkin provides support for shared files by using *rcs*'s access lists, and providing special handling for set-uid operation:

- When you first archive a file using **checkin**, it invokes the **rcs** administrative utility to initialize the access list of the file. It puts the *effective* user into the list.
- If **checkin** is running in set-uid mode, it puts the *real* user on the access list as well.

With the access list is initialized, only those users who appear on an access list may place locks on files, even when running in set-uid mode.

Directory-Level Permissions

Before attempting to create or lock an archive file, **checkin** looks first for the directory-level permissions which may be set with the **permit** utility. If they exist, **checkin** limits further access rights to those permitted.

OPTIONS

Checkin recognizes all of the "**ci**" options.

If the "**−k**" option is used, **checkin** supplies a default log-message

```
FROM_KEYS
```

Options specific to **checkin** are:

−B directs **checkin** to ignore the **baseline** version. Normally, **checkin** supplies a default version number which augments that of **ci**, by looking at the **baseline** version.

−D causes it to display the actions it would perform, but not to do them (e.g., invocation of **rcs** and **ci**).

−M*filename*
provide the check-in message in the given file. Normally **ci** prompts you for a multiline message.

If the input is not a terminal, **checkin** assumes that is a pipe, and passes the text (escaped) to **ci**. That is done best in a script. For random use, to supply the same check-in message for more than one file, the **−M** option lets you provide the message via a file.

OPERATIONS

Checkin is used exactly as one would use **ci**. Place a lock on the file using the "**−l**" option with **ci** (or with **co**) when you wish to edit a file. Check the file in using the "**−u**" option to retain a working copy after modification.

ENVIRONMENT

Checkin is a C-language program. It invokes **ci** (with an explicit path, to protect against mishaps in set-uid mode), and performs pre- and postprocessing of the archive and working file to determine the version to which the file's modification date applies.

Checkin uses the following environment variables:

RCS_BASE
is used to specify a default value for initial revision numbers. If the user does not specify the initial version number of a file, **ci** assigns the value "1.1". This is used to support the use of module-level version numbers, while preserving the relationship between changes and revisions: a new version is made only if the file is changed.

The directory-level revision set by the **permit** utility may override this environment variable. See *baseline* and *permit* for more details.

RCS_COMMENT
is set to a string controlling the initial setting of the rcs "**−c**" option. For example, the strings

```
setenv RCS_COMMENT '/.c/ *> /'
```

and

```
setenv RCS_COMMENT '/.d/# /,/.bas/REM /'
```

define comment-prefixes for ".c", ".d" and ".bas" suffixes. (The suffix is delimited with the first "." in the leaf-name).

RCS_DIR
if defined, specifies the directory in which *rcs* archive files are found. Normally files are found in "./RCS".

TZ is the POSIX time zone, which is overridden internally so that file modification dates are independent of the local time zone.

FILES

Checkin uses the following files

ci the RCS check-in program

rcs the RCS administrative program

ANTICIPATED CHANGES

None.

AUTHORS

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

SEE ALSO

baseline, rcsput, permit, ded, ci (1), co (1), rcs (1)