NAME

co − check out RCS revisions

SYNOPSIS

co [*options*] *file* . . .

DESCRIPTION

co retrieves a revision from each RCS file and stores it into the corresponding working file.

Pathnames matching an RCS suffix denote RCS files; all others denote working files. Names are paired as explained in **ci**(1).

Revisions of an RCS file can be checked out locked or unlocked. Locking a revision prevents overlapping updates. A revision checked out for reading or processing (e.g., compiling) need not be locked. A revision checked out for editing and later checkin must normally be locked. Checkout with locking fails if the revision to be checked out is currently locked by another user. (A lock can be broken with **rcs**(1).) Checkout with locking also requires the caller to be on the access list of the RCS file, unless he is the owner of the file or the superuser, or the access list is empty. Checkout without locking is not subject to accesslist restrictions, and is not affected by the presence of locks.

A revision is selected by options for revision or branch number, checkin date/time, author, or state. When the selection options are applied in combination, **co** retrieves the latest revision that satisfies all of them. If none of the selection options is specified, **co** retrieves the latest revision on the default branch (normally the trunk, see the **−b** option of **rcs**(1)). A revision or branch number can be attached to any of the options **−f**, **−I**, **−l**, **−M**, **−p**, **−q**, **−r**, or **−u**. The options **−d** (date), **−s** (state), and **−w** (author) retrieve from a single branch, the *selected* branch, which is either specified by one of **−f**, . . ., **−u**, or the default branch.

A **co** command applied to an RCS file with no revisions creates a zero-length working file. **co** always performs keyword substitution (see below).

OPTIONS

−r[*rev*] retrieves the latest revision whose number is less than or equal to *rev*. If *rev* indicates a branch rather than a revision, the latest revision on that branch is retrieved. If *rev* is omitted, the latest revision on the default branch (see the **−b** option of **rcs**(1)) is retrieved. If *rev* is **$**, **co** determines the revision number from keyword values in the working file. Otherwise, a revision is composed of one or more numeric or symbolic fields separated by periods. If *rev* begins with a period, then the default branch (normally the trunk) is prepended to it. If *rev* is a branch number followed by a period, then the latest revision on that branch is used. The numeric equivalent of a symbolic field is specified with the **−n** option of the commands **ci**(1) and **rcs**(1).

−l[*rev*] same as **−r**, except that it also locks the retrieved revision for the caller.

−u[*rev*] same as **−r**, except that it unlocks the retrieved revision if it was locked by the caller. If *rev* is omitted, **−u** retrieves the revision locked by the caller, if there is one; otherwise, it retrieves the latest revision on the default branch.

−f[*rev*] forces the overwriting of the working file; useful in connection with **−q**. See also FILE MODES below.

−kkv Generate keyword strings using the default form, e.g. **$Revision: 5.13 $** for the **Revision** keyword. A locker's name is inserted in the value of the **Header**, **Id**, and **Locker** keyword strings only as a file is being locked, i.e. by **ci −l** and **co −l**. This is the default.

−kkvl Like **−kkv**, except that a locker's name is always inserted if the given revision is currently locked.

−kk Generate only keyword names in keyword strings; omit their values. See KEYWORD SUBSTITUTION below. For example, for the **Revision** keyword, generate the string **$Revision$** instead of **$Revision: 5.13 $**. This option is useful to ignore differences due to keyword substitution when comparing different revisions of a file. Log messages are inserted after **Log** keywords even if **−kk** is specified, since this tends to be more useful when merging changes.

−ko Generate the old keyword string, present in the working file just before it was checked in. For example, for the **Revision** keyword, generate the string **$Revision: 1.1 $** instead of **$Revision:**

5.13 $ if that is how the string appeared when the file was checked in. This can be useful for file formats that cannot tolerate any changes to substrings that happen to take the form of keyword strings.

–kb Generate a binary image of the old keyword string. This acts like **–ko**, except it performs all working file input and output in binary mode. This makes little difference on Posix and Unix hosts, but on DOS-like hosts one should use **rcs –i –kb** to initialize an RCS file intended to be used for binary files. Also, on all hosts, **rcsmerge**(1) normally refuses to merge files when **–kb** is in effect.

–kv Generate only keyword values for keyword strings. For example, for the **Revision** keyword, generate the string **5.13** instead of **$Revision: 5.13 $**. This can help generate files in programming languages where it is hard to strip keyword delimiters like **$Revision: $** from a string. However, further keyword substitution cannot be performed once the keyword names are removed, so this option should be used with care. Because of this danger of losing keywords, this option cannot be combined with **–l**, and the owner write permission of the working file is turned off; to edit the file later, check it out again without **–kv**.

–p[*rev*] prints the retrieved revision on the standard output rather than storing it in the working file. This option is useful when **co** is part of a pipe.

–q[*rev*] quiet mode; diagnostics are not printed.

–I[*rev*] interactive mode; the user is prompted and questioned even if the standard input is not a terminal.

–d*date* retrieves the latest revision on the selected branch whose checkin date/time is less than or equal to *date*. The date and time can be given in free format. The time zone **LT** stands for local time; other common time zone names are understood. For example, the following *date*s are equivalent if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of Coordinated Universal Time (UTC):

8:00 pm lt	
4:00 AM, Jan. 12, 1990	default is UTC
1990-01-12 04:00:00+00	ISO 8601 (UTC)
1990-01-11 20:00:00–08	ISO 8601 (local time)
1990/01/12 04:00:00	traditional RCS format
Thu Jan 11 20:00:00 1990 LT	output of **ctime**(3) + **LT**
Thu Jan 11 20:00:00 PST 1990	output of **date**(1)
Fri Jan 12 04:00:00 GMT 1990	
Thu, 11 Jan 1990 20:00:00 –0800	Internet RFC 822
12-January-1990, 04:00 WET	

Most fields in the date and time can be defaulted. The default time zone is normally UTC, but this can be overridden by the **–z** option. The other defaults are determined in the order year, month, day, hour, minute, and second (most to least significant). At least one of these fields must be provided. For omitted fields that are of higher significance than the highest provided field, the time zone's current values are assumed. For all other omitted fields, the lowest possible values are assumed. For example, without **–z**, the date **20, 10:30** defaults to 10:30:00 UTC of the 20th of the UTC time zone's current month and year. The date/time must be quoted if it contains spaces.

–M[*rev*]

Set the modification time on the new working file to be the date of the retrieved revision. Use this option with care; it can confuse **make**(1).

–s*state* retrieves the latest revision on the selected branch whose state is set to *state*.

–T Preserve the modification time on the RCS file even if the RCS file changes because a lock is added or removed. This option can suppress extensive recompilation caused by a **make**(1) dependency of some other copy of the working file on the RCS file. Use this option with care; it can suppress recompilation even when it is needed, i.e. when the change of lock would mean a change to keyword strings in the other working file.

−w[*login*]
retrieves the latest revision on the selected branch which was checked in by the user with login name *login*. If the argument *login* is omitted, the caller's login is assumed.

−j *joinlist*
generates a new revision which is the join of the revisions on *joinlist*. This option is largely obsoleted by **rcsmerge**(1) but is retained for backwards compatibility.

The *joinlist* is a comma-separated list of pairs of the form *rev2***:***rev3,* where *rev2* and *rev3* are (symbolic or numeric) revision numbers. For the initial such pair, *rev1* denotes the revision selected by the above options **−f**, . . ., **−w**. For all other pairs, *rev1* denotes the revision generated by the previous pair. (Thus, the output of one join becomes the input to the next.)

For each pair, **co** joins revisions *rev1* and *rev3* with respect to *rev2*. This means that all changes that transform *rev2* into *rev1* are applied to a copy of *rev3*. This is particularly useful if *rev1* and *rev3* are the ends of two branches that have *rev2* as a common ancestor. If *rev1*<*rev2*<*rev3* on the same branch, joining generates a new revision which is like *rev3,* but with all changes that lead from *rev1* to *rev2* undone. If changes from *rev2* to *rev1* overlap with changes from *rev2* to *rev3,* **co** reports overlaps as described in **merge**(1).

For the initial pair, *rev2* can be omitted. The default is the common ancestor. If any of the arguments indicate branches, the latest revisions on those branches are assumed. The options **−l** and **−u** lock or unlock *rev1*.

−V Print RCS's version number.

−V*n* Emulate RCS version *n,* where *n* can be **3**, **4**, or **5**. This can be useful when interchanging RCS files with others who are running older versions of RCS. To see which version of RCS your correspondents are running, have them invoke **rcs −V**; this works with newer versions of RCS. If it doesn't work, have them invoke **rlog** on an RCS file; if none of the first few lines of output contain the string **branch:** it is version 3; if the dates' years have just two digits, it is version 4; otherwise, it is version 5. An RCS file generated while emulating version 3 loses its default branch. An RCS revision generated while emulating version 4 or earlier has a time stamp that is off by up to 13 hours. A revision extracted while emulating version 4 or earlier contains abbreviated dates of the form *yy***/***mm***/***dd* and can also contain different white space and line prefixes in the substitution for **Log**.

−x*suffixes*
Use *suffixes* to characterize RCS files. See **ci**(1) for details.

−z*zone* specifies the date output format in keyword substitution, and specifies the default time zone for *date* in the **−d***date* option. The *zone* should be empty, a numeric UTC offset, or the special string **LT** for local time. The default is an empty *zone*, which uses the traditional RCS format of UTC without any time zone indication and with slashes separating the parts of the date; otherwise, times are output in ISO 8601 format with time zone indication. For example, if local time is January 11, 1990, 8pm Pacific Standard Time, eight hours west of UTC, then the time is output as follows:

option	*time output*	
−z	**1990/01/12 04:00:00**	*(default)*
−zLT	**1990-01-11 20:00:00−08**	
−z+05:30	**1990-01-12 09:30:00+05:30**	

The **−z** option does not affect dates stored in RCS files, which are always UTC.

KEYWORD SUBSTITUTION

Strings of the form **$***keyword***$** and **$***keyword***:**. . .**$** embedded in the text are replaced with strings of the form **$***keyword***:***value***$** where *keyword* and *value* are pairs listed below. Keywords can be embedded in literal strings or comments to identify a revision.

Initially, the user enters strings of the form **$***keyword***$**. On checkout, **co** replaces these strings with strings of the form **$***keyword***:***value***$**. If a revision containing strings of the latter form is checked back in, the value fields will be replaced during the next checkout. Thus, the keyword values are automatically updated on checkout. This automatic substitution can be modified by the **−k** options.

Keywords and their corresponding values:

$Author$
The login name of the user who checked in the revision.

$Date$ The date and time the revision was checked in. With **−z***zone* a numeric time zone offset is appended; otherwise, the date is UTC.

$Header$
A standard header containing the full pathname of the RCS file, the revision number, the date and time, the author, the state, and the locker (if locked). With **−z***zone* a numeric time zone offset is appended to the date; otherwise, the date is UTC.

Id Same as **$Header$**, except that the RCS filename is without a path.

$Locker$
The login name of the user who locked the revision (empty if not locked).

Log The log message supplied during checkin, preceded by a header containing the RCS filename, the revision number, the author, and the date and time. With **−z***zone* a numeric time zone offset is appended; otherwise, the date is UTC. Existing log messages are *not* replaced. Instead, the new log message is inserted after **$Log:**. . .**$**. This is useful for accumulating a complete change log in a source file.

Each inserted line is prefixed by the string that prefixes the **Log** line. For example, if the **Log** line is "**// $Log: tan.cc $**", RCS prefixes each line of the log with "**//** ". This is useful for languages with comments that go to the end of the line. The convention for other languages is to use a " ∗ " prefix inside a multiline comment. For example, the initial log comment of a C program conventionally is of the following form:

```
/*
 * $Log$
 */
```

For backwards compatibility with older versions of RCS, if the log prefix is **/∗** or **(∗** surrounded by optional white space, inserted log lines contain a space instead of **/** or **(**; however, this usage is obsolescent and should not be relied on.

$Name$
The symbolic name used to check out the revision, if any. For example, **co −rJoe** generates **$Name: Joe $**. Plain **co** generates just **$Name: $**.

$RCSfile$
The name of the RCS file without a path.

$Revision$
The revision number assigned to the revision.

$Source$
The full pathname of the RCS file.

$State$
The state assigned to the revision with the **−s** option of **rcs**(1) or **ci**(1).

The following characters in keyword values are represented by escape sequences to keep keyword strings well-formed.

char	*escape sequence*
tab	**\t**
newline	**\n**
space	**\040**
$	**\044**
****	****

FILE MODES

The working file inherits the read and execute permissions from the RCS file. In addition, the owner write permission is turned on, unless **−kv** is set or the file is checked out unlocked and locking is set to strict (see **rcs**(1)).

If a file with the name of the working file exists already and has write permission, **co** aborts the checkout, asking beforehand if possible. If the existing working file is not writable or **−f** is given, the working file is deleted without asking.

FILES

co accesses files much as **ci**(1) does, except that it does not need to read the working file unless a revision number of **$** is specified.

ENVIRONMENT

RCSINIT

options prepended to the argument list, separated by spaces. See **ci**(1) for details.

DIAGNOSTICS

The RCS pathname, the working pathname, and the revision number retrieved are written to the diagnostic output. The exit status is zero if and only if all operations were successful.

IDENTIFICATION

Author: Walter F. Tichy.
Manual Page Revision: 5.13; Release Date: 1995/06/01.

SEE ALSO

rcsintro(1), ci(1), ctime(3), date(1), ident(1), make(1), rcs(1), rcsclean(1), rcsdiff(1), rcsmerge(1), rlog(1), rcsfile(5)
Walter F. Tichy, RCS—A System for Version Control, *Software—Practice & Experience* **15**, 7 (July 1985), 637-654.

LIMITS

Links to the RCS and working files are not preserved.

There is no way to selectively suppress the expansion of keywords, except by writing them differently. In nroff and troff, this is done by embedding the null-character **\&** into the keyword.