GNU RCS

for version 5.10.1, 27 January 2022

Thien-Thi Nguyen

This manual is for GNU RCS (version 5.10.1, 27 January 2022).

Table of Contents

1 Overview

GNU RCS (Revision Control System) manages multiple revisions of files. RCS can store, retrieve, log, identify, and merge revisions. It is useful for files that are revised frequently, e.g. programs, documentation, graphics, and papers. It can handle text as well as binary files, although functionality is reduced for the latter.

A normal installation includes the commands: **ci**, **co**, **ident**, **merge**, **rcs**, **rcsclean**, **rcsdiff**, **rcsmerge** and **rlog** (see Chapter 2 [Usage], page 7). These are small and fast programs (amenable to scripting) and indeed the distribution also includes the script **rcsfreeze** showing some of the possibilities.

RCS works with versions stored on a single filesystem or machine, edited by one person at a time. Other version control systems, such as Bazaar (`http:///www.gnu.org/software/bazaar`), CVS, Subversion, and Git, support distributed access in various ways. Which is more appropriate depends on the task at hand.

1.1 Credits

RCS was designed and built by Walter F. Tichy of Purdue University. RCS version 3 was released in 1983.

Adam Hammer, Thomas Narten, and Daniel Trinkle of Purdue supported RCS through version 4.3, released in 1990. Guy Harris of Sun contributed many porting fixes. Paul Eggert of System Development Corporation contributed bug fixes and tuneups. Jay Lepreau contributed 4.3BSD support.

Paul Eggert of Twin Sun wrote the changes for RCS versions 5.5 and 5.6 (1991). Rich Braun of Kronos and Andy Glew of Intel contributed ideas for new options. Bill Hahn of Stratus contributed ideas for setuid support. Ideas for piece tables came from Joe Berkovitz of Stratus and Walter F. Tichy. Matt Cross of Stratus contributed test case ideas. Adam Hammer of Purdue QAed.

Paul Eggert wrote most of the changes for RCS 5.7. K. Richard Pixley of Cygnus Support contributed several bug fixes. Robert Lupton of Princeton and Daniel Trinkle contributed ideas for '`$Name`' expansion. Brendan Kehoe of Cygnus Support suggested rlog's `-N` option. Paul D. Smith of Data General suggested improvements in option and error processing. Adam Hammer of Purdue QAed.

Thien-Thi Nguyen is responsible for RCS 5.8. He modernized the code base, build system, and manual pages, fixing some bugs on the way. He added standard `--help`, `--version` processing, and wrote the documentation you are reading (gladly taking inspiration from the paper[1] and manpages originally written by Walter F. Tichy).

1.2 Concepts

1.2.1 Interaction model

The interaction model is straightforward. For each working file, you initialize its RCS file once, then enter a cycle of checkout, modification, and checkin operations. Along the way,

[1] Source (troff) and several output formats are available from the RCS homepage (`http://www.gnu.org/software/rcs/`).

you can tweak some of the RCS file's metadata, as well. All of this is done through RCS commands; you need not modify the RCS file directly (and in fact you should probably avoid doing so lest RCS become confused). This model is somewhat analogous to using a library (of books). With a library, you sign up for a library card (initialize), then enter a cycle of taking a book home (checkout), enjoying it (NB: **without** modification, one hopes), and returning it to the library (checkin).

Furthermore, you can compare revisions in the RCS file against each other, examine the user- (hopefully high) quality descriptions of the changes each revision embodies, merge selected revisions, and so forth.

1.2.2 Working file

RCS commands operate on one pair of files at a time. The *working file* is what you normally view and edit (e.g., a file of C programming language source code named `a.c`). Because the working file's contents can be extracted from the RCS file (called *instantiating a working file*), it can be safely deleted to regain some disk space.

1.2.3 RCS file

The *RCS file* is a separate file, conventionally placed in the subdirectory `RCS`, wherein RCS commands organize the initial and subsequent *revisions* of the working file, associating with each revision a unique revision number along with the remembered particulars of the checkin that produced it. It also contains a *description* of the working file and various other metadata, described below.

The RCS file is also known (colloquially) as the "comma-v file", due to its name often ending in `,v` (e.g., `a.c,v`).

A *revision number* is a branch number followed by a dot followed by an integer, and a *branch number* is an odd number of integers separated by dot. A revision number with one dot (implying a branch number without any dots) is said to be *on the trunk*. All integers are positive. For example:

```
1.1         -- revision number for initial checkin (typically);
               branch number: 1

9.4.1.42    -- more complicated (perhaps after much gnarly hacking);
               branch number: 9.4.1

333.333.333 -- not a valid revision number;
               however, a perfectly valid branch number
```

The *branch point* of a non-trunk branch is the revision number formed by removing the branch's trailing integer. To compute the *next higher* branch or revision number, add one to the trailing integer. The highest-numbered revision on a branch is called the *tip* of the branch (or *branch tip*). Continuing the example:

```
1.1         -- on trunk; no branch point;
               next higher branch number:   2
               next higher revision number: 1.2
```

```
9.4.1.42    -- not on trunk; branch point:  9.4
               next higher branch number:   9.4.2
               next higher revision number: 9.4.1.43

333.333.333 -- not on trunk; branch point:  333.333
               next higher branch number:   333.333.334
               next higher revision number: 333.333.333.1
```

In addition to this "tree" of thus-linked revisions, the RCS file keeps track of the *default branch*, i.e., the branch whose tip corresponds to the most recent checkin; as well as the *symbolic names*, a list of associations between a user-supplied (and presumably meaningful) symbol and an underlying branch or revision number.

The RCS file contains two pieces of information used to implement its *access control policy*. The first is a list of usernames. If non-empty, only those users listed can modify the RCS file (via RCS commands). The second is a list of *locks*, i.e., association between a username and a revision number. If a lock *`username:revno`* exists, that means only *username* may modify *revno* (that is, do a checkin operation to deposit the next higher revision, or a higher revision number on the same branch as *revno*).

1.2.4 Fundamental operations

The *checkin* operation records the contents of the working file in the RCS file, assigning it a new (normally the next higher) revision number and recording the username, timestamp, *state* (a short symbol), and user-supplied *log message* (a textual description of the changes leading to that revision). It uses **diff** to find the differences between the tip of the default branch and the working file, thereby writing the minimal amount of information needed to be able to recreate the contents of the previous tip.

The *checkout* operation identifies a specific revision from the RCS file and either displays the content to standard output or instantiates a working file, overwriting any current instantiation with the selected revision. In either case, the content may undergo *keyword expansion*, which replaces text of the form '`$Keyword$`' with (possibly) different text comprising the keyword and its *value*, depending on the current keyword expansion mode (see Section 2.1.4 [Substitution mode option], page 9).

1.2.5 Keywords

The keywords and their values are:

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

`Date` The date and time the revision was checked in. May include an appended timezone offset.

`Header` A standard header containing the absolute RCS filename, the revision number, the date and time, the author, the state, and the locker (if locked). May include an appended timezone offset.

`Id` Same as '`Header`', except that only the basename appears (no directory components).

`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. May include an appended timezone offset.

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 $
```

then RCS prefixes each line of the log with '`// `' (slash, slash, space). This is useful for languages with comments that go to the end of the line.

The convention for other languages is to use a '` * `' (space, asterisk, space) 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 basename of the RCS file.

`Revision` The revision number assigned to the revision.

`Source` The absolute RCS filename.

`State` The state assigned to the revision with the `-s` option of **rcs** or **ci**.

1.3 Quick tour

This section complements the preceding section (see Section 1.2 [Concepts], page 1), presenting a handful of RCS commands in quick succession. For details on individual RCS commands, See Chapter 2 [Usage], page 7.

Suppose you have a file `f.c` that you wish to put under control of RCS. If you have not already done so, make an `RCS` directory with the command:

```
mkdir RCS
```

Then invoke the checkin command:

```
ci f.c
```

This command creates an RCS file in directory `RCS`, stores `f.c` into it as revision 1.1, and deletes `f.c`. It also asks you for a description. The description should be a synopsis of the contents of the file. All later checkin commands will ask you for a log entry, which should summarize the changes that you made.

To get back the working file `f.c` in the previous example, use the checkout command:

```
co f.c
```

This command extracts the latest revision from the RCS file and writes it into `f.c`. If you want to edit `f.c`, you must lock it as you check it out, with the command:

```
co -l f.c
```

You can now edit `f.c`. Suppose after some editing you want to know what changes that you have made. The command:

```
rcsdiff f.c
```

tells you the difference between the most recently checked-in version and the working file. You can check the file back in by invoking:

```
ci f.c
```

This increments the revision number properly. If **ci** complains with the message:

```
ci error: no lock set by your name
```

then you have tried to check in a file even though you did not lock it when you checked it out. Of course, it is too late now to do the checkout with locking, because another checkout would overwrite your modifications. Instead, invoke:

```
rcs -l f.c
```

This command will lock the latest revision for you, unless somebody else got ahead of you already. In this case, you'll have to negotiate with that person.

Locking assures that you, and only you, can check in the next update, and avoids nasty problems if several people work on the same file. Even if a revision is locked, it can still be checked out for reading, compiling, etc. All that locking prevents is a checkin by anybody but the locker.

If your RCS file is private, i.e., if you are the only person who is going to deposit revisions into it, strict locking is not needed and you can turn it off. If strict locking is turned off, the owner of the RCS file need not have a lock for checkin; all others still do. Turning strict locking off and on is done with the commands:

```
rcs -U f.c     # disable strict locking
rcs -L f.c     # enable strict locking
```

If you don't want to clutter your working directory with RCS files, create a subdirectory called `RCS` in your working directory, and move all your RCS files there. RCS commands will look first into that directory to find needed files. All the commands discussed above will still work, without any modification. See Section 2.1 [Common elements], page 7.

To avoid the deletion of the working file during checkin (in case you want to continue editing or compiling), invoke one of:

```
ci -l f.c      # checkin + locked checkout
ci -u f.c      # checkin + unlocked checkout
```

These commands check in `f.c` as usual, then perform an implicit checkout. The first form also locks the checked in revision, the second one doesn't. Thus, these options save you one checkout operation. The first form is useful if you want to continue editing, the second one if you just want to read the file. Both update the keyword substitutions in your working file see Section 1.2 [Concepts], page 1.

You can give **ci** the number you want assigned to a checked-in revision. Assume all your revisions were numbered 1.1, 1.2, 1.3, etc., and you would like to start release 2. Either of the commands:

```
ci -r2 f.c
ci -r2.1 f.c
```

assigns the number 2.1 to the new revision. From then on, **ci** will number the subsequent revisions with 2.2, 2.3, etc. The corresponding **co** commands:

```
co -r2 f.c
co -r2.1 f.c
```

retrieve the latest revision numbered 2.x and the revision 2.1, respectively. **co** without a revision number selects the latest revision on the trunk, i.e. the highest revision with a number consisting of two fields. Numbers with more than two fields are needed for branches. For example, to start a branch at revision 1.3, invoke:

```
ci -r1.3.1 f.c
```

This command starts a branch numbered 1 at revision 1.3, and assigns the number 1.3.1.1 to the new revision. Here is a diagram showing the new revision in relation to its branch and the trunk.

```
1.1  --  1.2  --  1.3  --  1.4  --  1.5
                   |
                [1.3.1]  --  1.3.1.1
```

For more information about branches, See Section 1.2 [Concepts], page 1.

2 Usage

This chapter describes how to invoke RCS commands, including common command-line elements, as well options specific to each command.

2.1 Common elements

All RCS commands accept `--help` and `--version`. See Section "Command-Line Interfaces" in *The GNU Coding Standards*.

Aside from `--help` and `--version`, RCS commands take the form '`-letter[arg]`', i.e., a hyphen followed by a single letter, optionally followed by extra information. The square braces mean that the extra information is optional. (No square braces means that the extra information is required.) In any case, when specified, the extra information **must** abut the letter; there can be no intervening whitespace.

```
co -u 1.4 foo   # wrong, space between -u and 1.4
co -u1.4  foo   # ok
```

Furthermore, options must appear before file names (if any) on the command line.

```
ident foo -q    # wrong, option after file name
ident -q foo    # ok
```

Lastly, pairs of RCS and working files can be specified in three ways: (a) both are given, (b) only the working file is given, (c) only the RCS file is given. For (a), both RCS and working files may have arbitrary directory components; RCS commands pair them up intelligently. For (b), RCS commands will look first into the directory `./RCS`, if it exists, to find the associated RCS file.

2.1.1 Revision options

As to be expected in a revision control system, many options are of the form '`-flag[rev]`', where *flag* is a single letter (e.g., '`r`'). If ommitted, *rev* defaults to the latest revision on the default branch. A revision can be specified in many ways:

`br.n` Straightforward dot-notation, where *br* specifies the branch.

`.n` Like *br.n*, using the default branch.

`br` Like *br.n*, using the a command-specific computation of *n*, given the current tip *i*. For **ci** (see Section 2.2 [ci], page 13), *n* would be *`i + 1`*, while for other commands *n* would be simply *i*.

`name` This is the symbolic name of a revision, as assigned previously by a `ci -n` or `ci -N` command.

`$` The command computes the effective revision by examining the values of keyword expansions in the working file.

For commands that accept a range of revisions, the syntax is generally *`rev1:rev2`*, i.e., two revisions (specified as described above) separated by a colon.

2.1.2 Date option

Some commands accept an option of the form '`-ddate`' to specify a *date*, an absolute point in time (to second resolution), expressed in a *date format*. These also accept '`-zzone`' to specify the timezone. The special value '`LT`' stands for the *local time zone*. RCS recognizes many date formats and time zones. For example, the following dates 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. Note that for the shell, the date/time must be quoted if it contains spaces.

RCS also accepts some other formats which specify only the date portion (omitting the time portion). In the following table, *year* is the four-digit year `YYYY`, and all examples specify 20 April 2018.

format	**example**	**description**
year`-`*doy*	`2018-110`	*doy* is the day of year `DDD`, 1-366.
year`-w`*week*`-`*dow*	`2018-w16-5`	*week* is the ISO week number `WW`, 0-53 (actually, ISO week numbers are 1-53; week 0 is a GNU RCS extension); and *dow* is the ISO day number `D`, 1-7 (Monday through Sunday). Note the literal `w` that precedes *week*.

2.1.3 Description option

Some commands accept an option of the form '`-t-`*text*' or '`-t`*file-name*'. This option is to set or update the RCS file description text. In the first form, *text* is used directly, excluding the leading hyphen ('`-`') that distinguishes the two forms. In the second form, the description text is taken from the contents of *file-name*.

2.1.4 Substitution mode option

Some commands accept an option of the form `-ksubst`, used to control how keywords (see Section 1.2 [Concepts], page 1) are expanded in the working file. In the following table of *subst* values, the example keyword is '`Revision`' and its value is '`5.13`'.

`kv` Generate '`$Revision: 5.13 $`' (dollar-sign, keyword, colon, space, value, space, dollar-sign). 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 substitution mode.

`kvl` Like `-kkv`, except that a locker's name is always inserted if the given revision is currently locked.

`k` Generate '`$Revision$`' (dollar-sign, keyword, dollar-sign). This is useful 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.

`o` Like `-kkv`, but use the old value present in the working file just before it 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.

`b` Like `-ko`, but do all file i/o 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** normally refuses to merge files when `-kb` is in effect.

`v` Generate '`5.13`' (value only). Further keyword substitution cannot be performed once the keyword names are removed, so this should be used with care. Because of this danger of losing keywords, `-kv` 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`.

2.1.5 Log message option

Both `ci` and `rcs` allow a log message to be specified with the `-m` option. If the *msg* argument to this option is empty, RCS uses the default '`*** empty log message ***`'. This particular message is handled specially (i.e., filtered out) by `rlog`.

2.1.6 State option

Some commands accept an option of the form `-sstate` to specify a state (see Section 2.2 [ci], page 13, see Section 2.3 [co], page 14). For **frob** (see Section 2.6 [rcs], page 16), you can also specify a revision. For **log** (see Section 2.10 [rlog], page 20), you can specify more than one state (see Section 2.1.9 [Delim-separated list], page 11).

RCS uses `Exp` (for "experimental") as the default state, but does not attach any meaning to it or any state you choose. Other common states are `Rel` (for "release"), `Prod` (for "production"), `Stable`, and so on. A state should be an `id` ("identifier", see Section 3.1.1 [comma-v grammar], page 22).

2.1.7 Working file mtime option

Two commands that mutate the working file (see Section 2.2 [ci], page 13, see Section 2.3 [co], page 14) accept an option of the form `-M` or `-Mrev`.

This option works with `-u` or `-l` to specify that the working file's modification time should be set to the date associated with revision *rev* (defaults to the branch tip if unspecified).

Like `-T`, this can be useful when the working file is named in a Makefile target's list of prerequisites.

2.1.8 Misc common options

Other common options are `-I`, `-q`, `-T`, `-V`, `-w`, `-x`.

`-I` This option enables *interactive mode*. More precisely, it **forces** interactive mode, whereby RCS commands believe that their standard input is a terminal, normally a precondition for displaying a prompt to receive input (such as a log message on checkin). The intention of `-I` is for scripting situations where standard input is actually not a terminal but you know beforehand (without prompting) that input is needed and you are ready to provide it on standard input anyway.

`-q` This option enables *quiet mode*. Commands work silently (unless there is an error condition), and suppress warnings and prompts.

`-T` This option controls how some commands (see Section 2.2 [ci], page 13, see Section 2.3 [co], page 14, see Section 2.6 [rcs], page 16, see Section 2.7 [rcsclean], page 18) timestamp the RCS file. Normally, RCS commands set the RCS file's timestamp when modifying it in the "natural" way (without taking any particular care). With `-T`, on the other hand, the commands either preserve the timestamp (for standalone lock/unlock operations), or use the timestamp of the working file (for **ci**).

This can be useful if the RCS file is found in a makefile target's list of prerequisites (see Section "Rule Syntax" in *The GNU Make Manual*), that is, if some target should be rebuilt if the RCS file is newer than it. In that case, you can do '`rcs -u -T`', for example, to unlock a revision in the RCS file without triggering a recompilation.

See Section 3.2 [Stamp resolution], page 25, for details on support for subsecond resolution.

`-V` Behave like `--version`, i.e., display command version information and exit successfully. **NB**: This option is obsolete and its **support will be removed** in some future release.

`-Vn` *n* specifies the RCS (major) version to emulate. Valid values for *n* are: 3, 4, 5. Version 5 is the current version, so `-V5` does nothing special.

In versions prior to 5, RCS outputs '`\t`' (tab, `U+09`) between the '`:`' (colon) and the value (for keyword substitution) instead of space, uses the RCS file `comment` string to prefix each line in the `Log` expansion instead of computing it on the fly from the input text, writes/reads localtime instead of UTC, and displays slightly different output for **rlog**.

For version 4, the `Header` expansion unconditionally includes `Locker: locker`, as if the `kvl` substitution mode were specified (see Section 2.1.4 [Substitution mode option], page 9).

For version 3, the `Header` expansion omits the directories from the filename and says only `Locked` instead of the state.

`-wlogin` Some commands accept an option of the form '`-wlogin`' to specify the login name of the author of a revision, i.e., "who" is responsible.

`-xsuff` Specify *suff* as the slash-separated list of file name suffixes used to recognize an RCS file. The default value is '`,v/`', that is, first try with '`,v`' then try with an empty suffix.

This *basename search* occurs within (i.e., starting from the beginning) the larger *directory search* loop, which comprises two candidates: `d/RCS` and `d`, where *d* is the directory component of the working file name. For example, given the working file `a.c` in the current directory, RCS tries, in order, these candidates:

```
./RCS/a.c,v
./RCS/a.c
./a.c,v
./a.c
```

Note that the last candidate is impossible (and is in fact discarded), because the working and RCS files cannot have the same name.

2.1.9 Delim-separated list

Some options that require (or allow) additional information can take multiple items of that information in the form of a *delim-separated list*, a concatenation of items with one or more delimiter characters between adjacent items. Multiple adjacent delim characters count as a single delimiter.

Most often you will use comma (`,` or `U+2C`), but RCS also permits others, depending on context. In the following table, "SPC" is the space character (`U+20`), "LF" is the linefeed character (`U+0A`), "TAB" is the tab character (`U+09`), and "semi" is `;` (`U+3B`).

context	**permitted delimiter characters, notes**
`co -jjoins`	SPC TAB comma *joins* is a delim-separated list, each item of which is a *join pair* of the form *rev*`:`*rev*. See Section 2.3 [co], page 14.
`rcs -aaccessors` `rcs -eaccessors`	SPC LF TAB comma *accessors* is a delim-separated list of logins. See Section 2.6 [rcs], page 16.

`rcs -o`*revspecs* semi comma
`rlog -r`*revspecs* *revspecs* is a delim-separated list, each item of which has one of the forms:

`REV`	
`REV1:`	`REV1-`
`REV1:REV2`	`REV1-REV2`

The variants in the second column use hyphen (`-`, `U+2D`). They are obsolete and should be avoided. They are ambiguous in the presence of symbolic branch and revision names that include a hyphen in the name (see Section 2.1.1 [Revision options], page 7). See Section 2.6 [rcs], page 16, See Section 2.10 [rlog], page 20. Note that although the "join pair" for **co -j** above is identical to a `REV1:REV2` revspec, the set of delim characters is different.

`rlog -l`*lockers* SPC TAB LF semi comma
`rlog -w`[*authors*] Both *lockers* and *authors* are a delim-separated list of logins. Note that *authors* is completely optional. See Section 2.10 [rlog], page 20.

`rlog -s`*states* SPC TAB LF semi comma
states is a delim-separated list of states (see Section 2.1.6 [State option], page 9). See Section 2.10 [rlog], page 20.

`rlog -d`*dates* SPC TAB LF semi comma
dates is a delim-separated list of date/time specifications (see Section 2.1.2 [Date option], page 8). See Section 2.10 [rlog], page 20.

Maintainer's Note: This embarrassment of choice for delim characters will probably be reduced to simply one character in RCS 6: comma.

2.1.10 Environment

Various environment variables influence how RCS works.

`RCSINIT` [Environment Variable]

Another way to set common options is with the '`RCSINIT`' environment variable. This is a space-separated list of options. Use '\' (backslash) to escape significant space. For example:

```
# Set the value; make it available to subsequent commands.
RCSINIT="-q -x/,v -zLT"
export RCSINIT

# Use it (implicitly).
rlog -L foo
```

This example, in Bourne shell syntax, arranges for RCS commands to operate as if each command-line had prepended '`-q -x/,v -zLT`' to the rest of the command-line. The effective command-line that **rlog** sees is thus '`-q -x/,v -zLT -L foo`'.

`RCS_MEM_LIMIT` [Environment Variable]

Normally, for speed, commands either memory map or copy into memory the RCS file if its size is less than the *memory limit*, currently defaulting to "unlimited". Otherwise (or if the initially-tried speedy ways fail), the commands fall back to using standard i/o routines.

You can adjust the memory limit by setting the '`RCS_MEM_LIMIT`' environment variable to a numeric value (measured in kilobytes). An empty value is silently ignored.

As a side effect, specifying the memory limit inhibits fall-back to slower routines. (This env var is mostly intended for testing RCS; normally, you can leave it unset. Probably it will be removed in a future release.)

`TMPDIR` [Environment Variable]
`TMP` [Environment Variable]
`TEMP` [Environment Variable]

Commands sometimes create temporary files, normally in a system-dependent directory, such as `/tmp`. You can override this directory by specifying another one as the value of one of the environment variables `TMPDIR`, `TMP`, or `TEMP` (checked in that order).

`LOGNAME` [Environment Variable]
`USER` [Environment Variable]

Absent `-w`*login*, or when *login* is omitted (see Section 2.1.8 [Misc common options], page 10), commands check environment variables `LOGNAME` and `USER` in that order[1]. If neither of these are set, RCS queries the host for, and uses, your login.

2.2 Invoking ci

rcs ci `[options] file ...`
(or "**ci**" instead of "**rcs ci**")

The **ci** command adds a revision to the RCS file reflecting the current state of the working file. This operation is also known as "checkin".

`-f[`*rev*`]` Force new entry, even if no content changed.

`-I[`*rev*`]`
`-q[`*rev*`]` See Section 2.1.8 [Misc common options], page 10.

`-i[`*rev*`]` Initial checkin; error if the RCS file already exists.

`-j[`*rev*`]` Just checkin, don't initialize; error if the RCS file does not exist.

`-k[`*rev*`]` Compute revision from working file keywords.

Do not confuse this with `-k`*subst* (see Section 2.1.4 [Substitution mode option], page 9).

`-r` Release lock and delete working file.

`-r`*rev* Do normal checkin.

`-l[`*rev*`]` Like `-r`, but immediately checkout locked (`co -l`) afterwards.

[1] However, on systems where env var `LOGNAME` is readonly at configure time, RCS checks `USER` first.

`-u[rev]` Like `-l`, but checkout unlocked (`co -u`).

`-M[rev]` See Section 2.1.7 [minus-M], page 10.

Multiple flags in `-{fiIjklMqru}` may be given, except for `-r`, `-l`, `-u`, which are mutually exclusive. For a fully specified revision of the form *`br.n`*, *n* must be greater than any existing on *br*, or *br* must be new. If *rev* is omitted, compute it from the last lock (`co -l`), perhaps starting a new branch. If there is no lock, use *`defbr.(L+1)`*. See Section 2.1.1 [Revision options], page 7.

`-d[date]`
`-z`*`zone`* See Section 2.1.2 [Date option], page 8. If no *date* specified, use the working file modification time.

`-m[msg]` Use *msg* as the log message. See Section 2.1.5 [Log message option], page 9.

`-n`*`name`*
`-N`*`name`* Assign symbolic *name* to the entry. For `-n`, *name* must be new (no previous assignment). For `-N`, overwrite any previous assignment.

`-s`*`state`* Set the state (see Section 2.1.6 [State option], page 9).

`-t-`*`text`*
`-t`*`file-name`*
See Section 2.1.3 [Description option], page 8.

`-T` Set the RCS file's modification time to the new revision's time if the former precedes the latter and there is a new revision; preserve the RCS file's modification time otherwise. See Section 2.1.8 [Misc common options], page 10.

`-w`*`who`* Use *who* as the author. See Section 2.1.8 [Misc common options], page 10.

`-V`
`-V`*`n`*
`-x`*`suff`* See Section 2.1.8 [Misc common options], page 10.

2.3 Invoking co

rcs co `[options] file ...`
(or "**co**" instead of "**rcs co**")

The **co** command retrieves a revision from the RCS file, writing a new working file. This operation is also known as "checkout".

`-f[rev]` Force overwrite of working file.

`-I[rev]`
`-q[rev]` See Section 2.1.8 [Misc common options], page 10.

`-p[rev]` Write to standard output instead of the working file.

`-r[rev]` Normal checkout.

`-l[rev]` Like `-r`, but also lock.

`-u[rev]` Like `-l`, but unlock.

`-M[rev]` See Section 2.1.7 [minus-M], page 10.

Multiple flags in `-{fIlMpqru}` may be given, except for `-r`, `-l`, `-u`, which are mutually exclusive. See Section 2.1.1 [Revision options], page 7.

`-ksubst` See Section 2.1.4 [Substitution mode option], page 9.

`-ddate`
`-zzone` See Section 2.1.2 [Date option], page 8. Select latest before or on *date*.

`-jjoins` Merge using *joins*, a list of *rev*:*rev* pairs. **NB**: This option is obsolete (see Section 2.9 [rcsmerge], page 19).

`-sstate` Select matching state (see Section 2.1.6 [State option], page 9).

`-S` Enable "self-same" mode. In this mode, the owner of a lock is unimportant, just that it exists. Effectively, this prevents you from checking out the same revision twice.

```
$ whoami
ttn

$ co -l -f z
RCS/z,v  -->  z
revision 1.1 (locked)
done

$ co -S -l -f z
RCS/z,v  -->  z
co: RCS/z,v: Revision 1.1 is already locked by ttn.
```

`-T` Preserve the modification time on the RCS file even if it changes because a lock is added or removed. See Section 2.1.8 [Misc common options], page 10.

`-wwho` Select matching login *who*. See Section 2.1.8 [Misc common options], page 10.

`-V`
`-Vn`
`-xsuff` See Section 2.1.8 [Misc common options], page 10.

2.4 Invoking ident

```
ident [options]   [file ...]
```

If no *file* is specified, scan standard input. The **ident** command scans its input for keywords (see Section 1.2 [Concepts], page 1), displaying to standard output what it finds.

`-q` Normally, if no patterns are found for a file, **ident** emits a warning. This option suppresses the warning.

`-V` Note that `-Vn` is *not* a valid option for **ident**, in contrast to most other RCS commands (see Section 2.1.8 [Misc common options], page 10).

In addition to the normal keyword pattern, for Subversion 1.2 (and later) compatibility[2], **ident** also recognizes patterns having one of the forms:

`$`*keyword*`:: `*text*` $`

[2] The *fixed-length keyword syntax* is described in detail in Version Control with Subversion, chapter "Advanced Topics", section "Keyword Substitution".

```
;; two colons and space after keyword
;; space before ending $

$keyword:: text#$
;; two colons and space after keyword
;; hash before ending $
```

2.5 Invoking merge

```
merge [options]   receiving-sibling parent other-sibling
```

The **merge** command combines the differences between a the parent and the other sibling, and the differences between the parent and the receiving sibling. It writes the result to the receiving sibling.

`-A`
`-E`
`-e` Use `diff3 -A`, `-E` (default), or `-e`, respectively.

`-p` Write to standard output instead of overwriting *receiving-sibling*.

`-q` See Section 2.1.8 [Misc common options], page 10. Suppress conflict warnings.

`-L label` (up to three times) Specify the conflict labels for *receiving-sibling*, *parent* and *other-sibling*, respectively.

`-V` Note that `-Vn` is *not* a valid option for **merge**, in contrast to most other RCS commands (see Section 2.1.8 [Misc common options], page 10).

2.6 Invoking rcs

The **rcs** command is unique in the set of RCS programs in that it has two usages, the modern (for RCS 5.9.0 and later) and the legacy.

2.6.1 modern

```
rcs [options]   command [command-arg ...]
```

This **rcs** usage dispatches to *command*, passing along *command-arg*... without interpretation.

`--commands`
Display a list of available commands, including a one-line description, and exit successfully.

`--aliases`
Display a list of command aliases and exit successfully.

`--help command`
Display help for a particular *command* and exit successfully. For example, to display help for the legacy interface, use:

```
--help frob
```

2.6.2 legacy

rcs frob [options] file ...
(or " **rcs**" instead of "**rcs frob**")

This **rcs** usage performs various administrative operations on the RCS file, depending on the options given.

`-i` Create and initialize a new RCS file.

`-L` Set strict locking.

`-U` Set non-strict locking.

`-M` Don't send mail when breaking someone else's lock.
Do not confuse this with `-Mrev` (see Section 2.1.7 [minus-M], page 10).

`-T` Preserve the modification time on the RCS file unless a revision is removed.

`-I`
`-q` See Section 2.1.8 [Misc common options], page 10.

`-a`*logins* Append *logins* (see Section 2.1.9 [Delim-separated list], page 11) to access-list.

`-e[`*logins*`]`
Erase *logins* (see Section 2.1.9 [Delim-separated list], page 11) from access-list. If *logins* is omitted, clear the access-list.

`-A`*file-name*
Append access-list of *file-name* to current access-list.

`-b[`*rev*`]` Set default branch to that of *rev* or highest branch on trunk if *rev* is omitted.

`-l[`*rev*`]` Lock a revision.

`-u[`*rev*`]` Unlock a revision.

`-c`*string* Set comment leader to *string*. **NB**: Don't use; obsolete.

`-k`*subst* See Section 2.1.4 [Substitution mode option], page 9.

`-m`*rev*`:[`*msg*`]`
Replace log message with *msg*. See Section 2.1.5 [Log message option], page 9.

`-n`*name*`[:[`*rev*`]]`
If *:rev* is omitted, delete symbolic *name*. Otherwise, associate *name* with *rev*; *name* must be new.

`-N`*name*`[:[`*rev*`]]`
Like `-n`, but overwrite any previous assignment.

`-o`*range* Delete (also known as "outdate") revisions in *range*:

rev	single revision
br	latest revision on branch *br*
rev1`:`*rev2*	*rev1* to *rev2* on same branch, inclusive
`:`*rev*	beginning of branch to *rev*

rev: *rev* to end of branch

`-sstate[:rev]`
: Set state (see Section 2.1.6 [State option], page 9).

`-t-text`
`-tfile-name`
: See Section 2.1.3 [Description option], page 8. Replace description.

`-V`
`-Vn`
`-xsuff` See Section 2.1.8 [Misc common options], page 10.

These options have no effect, and are included solely for consistency with other commands (see Section 2.1.10 [Environment], page 12): `-zzone`.

2.7 Invoking rcsclean

rcs clean `[options] [file ...]`
(or "**rcsclean**" instead of "**rcs clean**")

The **rcsclean** command removes working files that are not being worked on. If given `-u`, it also unlocks and removes working files that are being worked on but have not changed. If no *file* is specified, operate on all the working files in the current directory.

`-r[rev]` Specify revision.

`-u[rev]` Unlock if is locked and no differences found.

`-n[rev]` Dry run (no act, don't operate).

`-q[rev]` See Section 2.1.8 [Misc common options], page 10.

`-ksubst` See Section 2.1.4 [Substitution mode option], page 9.

`-T` Preserve the modification time on the RCS file even it changes because a lock is removed.

`-V`
`-Vn`
`-xsuff` See Section 2.1.8 [Misc common options], page 10.

`-zzone` See Section 2.1.2 [Date option], page 8.

2.8 Invoking rcsdiff

rcs diff `[options] file ...`
(or "**rcsdiff**" instead of "**rcs diff**")

The **rcsdiff** command runs **diff** to compare two revisions in an RCS file. See Section "Invoking diff" in *The GNU Diffutils Manaual*.

`-rrev` (zero, one, or two times) Name a revision. If given two revisions ('`-rrev1 -rrev2`'), compare those revisions. If given only one revision ('`-rrev`'), compare the working file with it. If given no revisions, compare the working file with the latest revision on the default branch.

`-ksubst` See Section 2.1.4 [Substitution mode option], page 9.

`-q` See Section 2.1.8 [Misc common options], page 10.

`-V`
`-Vn`
`-xsuff` See Section 2.1.8 [Misc common options], page 10.

`-zzone` See Section 2.1.2 [Date option], page 8.

These options have no effect, and are included solely for consistency with other commands (see Section 2.1.10 [Environment], page 12): `-T`.

Additionally, the following options (and their argument, if any) are passed to the underlying **diff** command:

```
-0, -1, -2, -3, -4, -5, -6, -7, -8, -9, -B, -C, -D, -F, -H, -I,
-L, -U, -W, -a, -b, -c, -d, -e, -f, -h, -i, -n, -p, -t, -u, -w, -y,
long options (that start with "--")
```

(Not all of these options are meaningful.)

2.9 Invoking rcsmerge

rcs merge `[options] file`
(or "**rcsmerge**" instead of "**rcs merge**")

The **rcsmerge** command incorporates the changes between two revisions of an RCS file into the corresponding working file.

`-A`
`-E`
`-e` Passed to the **diff3** command. The default if none are specified is `-E`. With `-e`, suppress warnings on conflict. The `-A` style generates the most verbose output. See Section "Invoking diff3" in *The GNU Diffutils Manual*.

`-p[rev]` Write to standard output instead of overwriting the working file.

`-q[rev]` See Section 2.1.8 [Misc common options], page 10.

`-rrev` (one or two times) specify a revision.

One or two revisions must be specified (using `-p`, `-q`, `-r`). If only one is specified, the second revision defaults to the latest revision on the default branch.

`-ksubst` See Section 2.1.4 [Substitution mode option], page 9.

`-V`
`-Vn`
`-xsuff` See Section 2.1.8 [Misc common options], page 10.

`-zzone` See Section 2.1.2 [Date option], page 8.

These options have no effect, and are included solely for consistency with other commands (see Section 2.1.10 [Environment], page 12): `-T`.

2.10 Invoking rlog

rcs log `[options] file ...`
(or "**rlog**" instead of "**rcs log**")

The **rlog** command displays information about RCS files.

`-L` Ignore RCS files with no locks set.

`-R` Print only the name of the RCS file.

`-h` Print only the "header" information.

`-t` Like `-h`, but also include the description.

`-N` Omit symbolic names.

`-b` Select the default branch.

`-d`*dates* See Section 2.1.2 [Date option], page 8. Select revisions based on timestamp, in the range *dates*, with spec:

d single revision *d* or earlier

d1`<`*d2*
d2`>`*d1* between *d1* and *d2*, exclusive

`<`*d*
d`>` before *d*

`>`*d*
d`<` after *d*

Instead of '`<`' or '`>`', you can use '`<=`' or '`>=`', respectively, to specify inclusive ranges. *dates* may also be a list of semicolon-separated specs.

`-l[`*who*`]` Select revisions locked by *who* (see Section 2.1.9 [Delim-separated list], page 11) only, or by anyone if *who* is omitted.

`-r[`*revs*`]` Select revisions in *revs* (see Section 2.1.9 [Delim-separated list], page 11), one of: *rev*, *rev*`:`, `:`*rev*, *rev1*`:`*rev2*.

`-s`*state*`[,`*state*`...]`
Select revisions with specified state(s) (see Section 2.1.6 [State option], page 9).

`-w[`*who*`]` Select revisions checked in by *who* (see Section 2.1.9 [Delim-separated list], page 11), or by the user if *who* is omitted.

`-V`
`-V`*n*
`-x`*suff* See Section 2.1.8 [Misc common options], page 10.

`-z`*zone* See Section 2.1.2 [Date option], page 8. This option also changes the output format of the date to use hyphens instead of slashes. For example:

```
$ rlog t,v  # without -z
...
date: 2010/10/02 04:35:26;  [...]
...
```

```
$ rlog -z+0200 t,v
...
date: 2010-10-02 06:35:26+02;  [...]
...
```

These options have no effect, and are included solely for consistency with other commands (see Section 2.1.10 [Environment], page 12): ‘`-q`’, ‘`-T`’.

3 Hacking

This chapter, in contrast to the previous (see Chapter 2 [Usage], page 7), is introspective. It describes important aspects of RCS interop with other programs, and development ideas and methods.

3.1 File format

An RCS file's contents are described by the grammar below[1]. Overall, the format is free–format text. In most environments RCS uses the ISO 8859/1 encoding: visible graphic characters are (octal) codes 041–176 and 240–377, and whitespace characters are codes 010–015 and 040.
TODO: Discuss or point to encoding compatibility issues.

3.1.1 File format grammar

The meta syntax in this section uses the following conventions: '`|`' (`U+7C`) separates alternatives; '`{`' (`U+7B`) and '`}`' (`U+7D`) enclose optional phrases; '`{`' and '`}*`' (trailing `U+2A`) enclose phrases that can be repeated zero or more times; '`{`' and '`}+`' (trailing `U+2B`) enclose phrases that must appear at least once and can be repeated; terminal symbols are in '`""`' (two `U+22`).

```
rcstext   ::=  admin {delta}* desc {deltatext}*

admin     ::=  "head"         {num} ";"
               { "branch"     {num} ";" }
               "access"       {id}* ";"
               "symbols"      { sym ":" num }* ";"
               "locks"        { id ":" num }* ";"
               { "strict" ";" }
               { "integrity " {intstring} ";" }
               { "comment"    {string} ";" }
               { "expand"     {string} ";" }

delta     ::=  num
               "date"       num ";"
               "author"     id ";"
               "state"      {id} ";"
               "branches"   {num}* ";"
               "next"       {num} ";"
               { "commitid" sym ";" }

desc      ::=  "desc"  string

deltatext ::=  num
               "log"   string
               "text"  string
```

[1] This section is adapted from the '`rcsfile(5)`' manpage, written by Walter F. Tichy.

```
num       ::=  { digit | "." }+

digit     ::=  "0" through "9"

id        ::=  { idchar | "." }+

sym       ::=  {idchar}+

idchar    ::=  any visible graphic character except special

special   ::=  "$" | "," | "." | ":" | ";" | "@"

string    ::=  "@" { any character, with @ doubled }* "@"

word      ::=  id | num | string | ":"

intchar   ::=  any character, except @

thirdp    ::=  "^L" {intchar}*

intstring ::= "@" {intchar}* {thirdp}* "@"
```

3.1.2 Additional particulars of the file format

- In releases prior to 5.8 (2011-08-30), the grammar included the production:

  ```
  newphrase ::= id word* ";"
  ```

  and used it in the `admin`, `delta` and `deltatext` productions. This allowed third-party programs to interoperate with RCS by storing opaque (to RCS) data in the file.

  As of 5.8, in the name of progress (towards more systematic file integrity support), the only area reserved for third-party interop is in the `string` value of the `integrity` field, specifically after the first formfeed (`U+0C`). A further restriction (for all programs) is that the `integrity` value must not contain '`@`'.

  Warning: This change means you cannot use `rlog` (or `rcs log`) as a workalike for `cvs log` for versions of CVS that write other kinds of metadata into the file. If you use CVS and have access to the `*,v` files it writes, you can determine if they require `cvs log` by the following command:

  ```
  if grep -E -q '^(deltatype|permissions|kopt)' *,v
    then echo 'must use "cvs log"'
    else echo 'probably safe to use "rcs log" (for now)'
  fi
  ```

  The "(for now)" bit is a nod to the most probable trajectory for both RCS and CVS: away from interop.

- Whitespace has no significance except in `string` values. However, whitespace cannot appear within an `id`, `num`, or `sym`, and an RCS file must end with newline (`U+0A`). A

`string` value is enclosed by '`@`' (`U+40`) with internal '`@`' characters doubled. All other bytes (arbitrary binary data) represent themselves. For example:

conceptual string	**persistent representation**
a string of five words	`@a string of five words@`
another, with one '@' char	`@another, with one '@@' char@`
with newline and unquoted @	`@with newline` `and unquoted @@@`

- Identifiers are case sensitive. Keywords are in lower case only. The sets of keywords and identifiers can overlap.

- Dates, which appear after the `date` keyword, are of the form `Y.m.d.H.M.S`, where `Y` is the year, `m` the month (01–12), `d` the day (01–31), `H` the hour (00–23), `M` the minute (00–59), and `S` the second (00–60). (These correspond to `strftime` format strings, with the exception of `Y`, which depends on the particular year.)

  `Y` contains just the last two digits of the year for years from 1900 through 1999, and all the digits of years thereafter. Dates use the Gregorian calendar; times use UTC.

- The `delta` nodes form a tree. All nodes whose numbers consist of a single pair, e.g.:

  ```
  2.3
  2.1
  1.3
  ```

  are on the trunk, and are linked through the `next` field in order of decreasing numbers. The `head` field in the `admin` node points to the head of that sequence (i.e., contains the highest pair). The `branch` node in the `admin` node indicates the default branch (or revision) for most RCS operations. If empty, the default branch is the highest branch on the trunk.

  All `delta` nodes whose numbers consist of $2n$ fields ($n \geq 2$), e.g.:

  ```
  3.1.1.1
  2.1.2.2
  ```

  are linked as follows. All nodes whose first $2n-1$ number fields are identical are linked through the `next` field in order of increasing numbers. For each such sequence, the `delta` node whose number is identical to the first $2n-2$ number fields of the `delta` nodes on that sequence is called the *branchpoint*.

  The `branches` field of a node contains a list of the numbers of the first nodes of all sequences for which it is a branchpoint. This list is ordered in increasing numbers. See Figure 3.1.

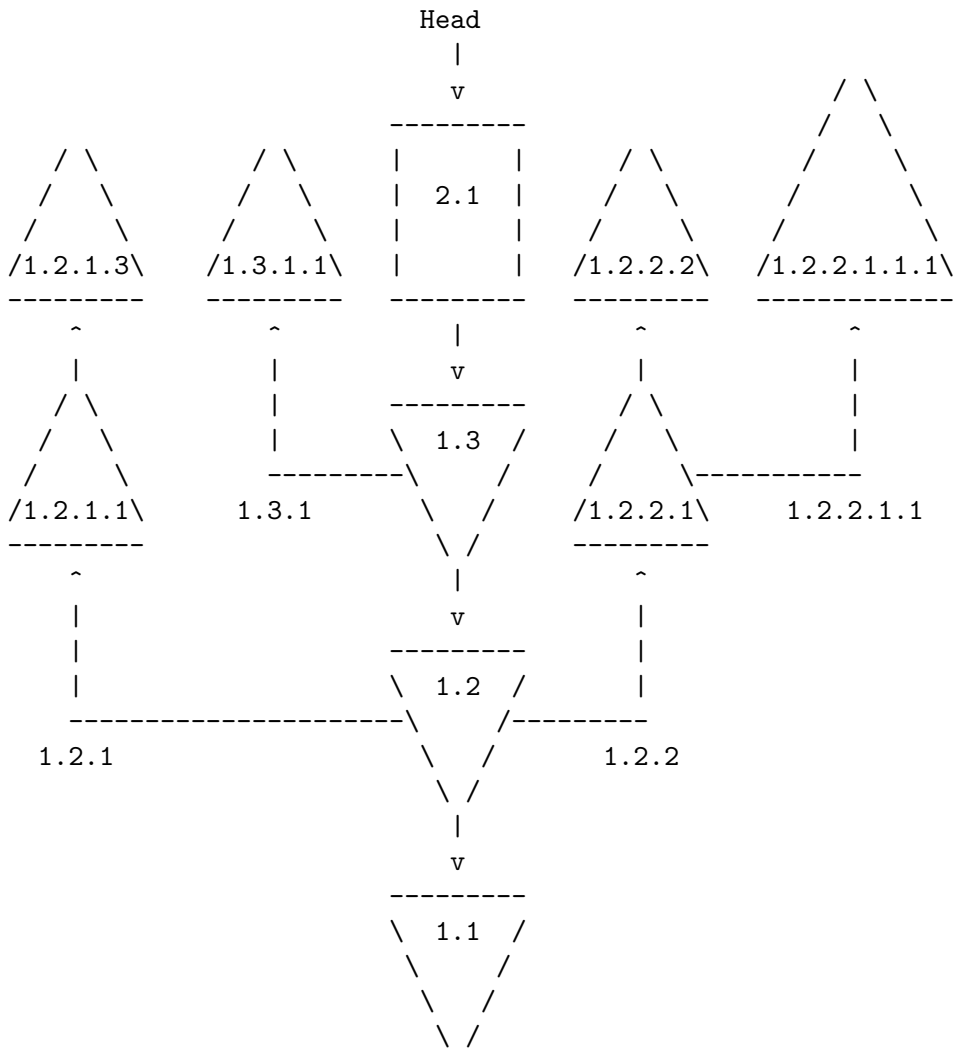

Figure 3.1: The organization of an example RCS file.

3.2 Stamp resolution

Regarding RCS, recorded timestamps come into play in two places:

- The `delta` production of the file format grammar includes component `date` (see Section 3.1.1 [comma-v grammar], page 22). The recorded information has second (whole number) resolution.
- The metadata of a file on the filesystem usually includes its *modification time*. The resolution of this information depends on the capabilities of the filesystem; modern ones — e.g., ext4 (`https://en.wikipedia.org/wiki/Ext4`) — tend to support subsecond (fractional) resolution.

Historically, up through version 5.9.4, RCS behaved "agnostically" with respect to the subsecond component of the file modification time, relying on the operating system and filesystem to take care of things at whatever resolution was available at the time, with the

single exception of the '`-T`' option (see Section 2.1.8 [Misc common options], page 10). In the presence of this option, RCS would:

- (reading) Ignore the subsecond component.
- (writing) Specify 0 as the subsecond component.

For versions after 5.9.4, if the filesystem supports it, RCS reads and writes file modification time with subsecond resolution, given the '`-T`' option.

It's important to keep in mind that by design, the delta `date` component is limited to second resolution, so subsecond resolution is only guaranteed for operations where the file modification time originates from a file actually existing on the filesystem (i.e., via the `stat(2)` system call).

3.3 Still missing

RCS is still missing some features. The following is an unordered list of "to-do musings" kept by the RCS maintainers. If you would like to hack on an item, See Section 3.4 [Reporting bugs], page 27.

- Add an option to **rcsmerge** so that it can use an arbitrary program to do the 3-way merge, instead of the default **merge**. Likewise for **rcsdiff** and **diff**. It should be possible to pass arbitrary options to these programs, and to the subsidiary **co** invocations.
- Add format options for finer control over the output of **ident** and **rlog**. E.g. there should be an easy way for rlog to output lines like '`src/main.c 2.4 wft`', one for each locked revision. **rlog** options should have three orthogonal types: selecting files, selecting revisions, and selecting rlog format.
- Add format options for finer control over the output of keyword strings. E.g. there should be some way to prepend '`@(#)`', and there should be some way to change '`$`' to some other character to disable further substitution. These options should make the resulting files uneditable, like '`-kv`'.
- Add long options, e.g. `--keyword-substitution`. Unfortunately RCS's option syntax is incompatible with getopt. Perhaps the best way is to overload **rcs**, e.g., '`rcs diff --keyword-substitution=old file`' instead of '`rcsdiff -ko file`'.
- `rlog -rM:N` should work even if M and N have different numbers of fields, so long as M is an ancestor of N or vice versa.
- **rcs** should evaluate options in order; this allows `rcs -oS -nS`.
- Be able to redo your most recent checkin with minor changes.
- `co -u` shouldn't complain about a '`+w`' working file if contents don't change.
- Add a '`-`' option to take the list of file names from standard input. Perhaps the file names should be null-terminated, not newline-terminated, so that those that contain newlines are handled properly.
- Permit multiple option–filename pairs, e.g., `co -r1.4 a -r1.5 b`.
- Add an option to break a symbolic link to an RCS file, instead of breaking the hard link that it points to.
- Add ways to specify the earliest revision, the most recent revision, the earliest or latest revision on a particular branch, and the parent or child of some other revision.

- If a user has multiple locks, perhaps **ci** should fall back on the method of `ci -k` to figure out which revision to use.
- Add an option to **rcsclean** to clean directories recursively.
- Write an **rcsck** program that repairs corrupted RCS files, much as fsck repairs corrupted file systems. For example, it should remove stale lock files.
- Update the date parser to use the more modern `getdate.y` by Bellovin, Salz, and Berets, or the even more modern `getdate` by Moraes. None of these getdate implementations are as robust as RCS's old warhorse in avoiding problems like arithmetic overflow, so they'll have to be fixed first. (Perhaps we can use gnulib module `getdate`.)
- Break up the code into a library so that it's easier to write new programs that manipulate RCS files, and so that useless code is removed from the existing programs. For example, the **rcs** command contains unnecessary keyword substitution baggage, and the **merge** command can be greatly pruned.
- Make it easier to use your favorite text editor to edit log messages, etc., instead of having to type them in irretrievably at the terminal.
- Let the user specify a search path for default branches, e.g., to use L as the default branch if it works, and M otherwise. Let the user require that at least one entry in the search path works. Let the user say that later entries in the search path are read only, i.e. one cannot check in changes to them. This should be an option settable by '`RCSINIT`'.
- Add a way for a user to see which revisions affected which lines.
- Have `rlog -nN F` print just the revision number that N translates to. E.g., `rlog -nB. F` would print the highest revision on the branch B. Use this to add an option `-bB` to rcsbranch, to freeze the named branch. This should interact well with default branches.
- Add a **co** option that prints the revision number before each line, as SCCS's `get -m` does. [I implemented this for Emacs 22 as a subroutine of `vc-annotate`, q.v. —ttn]

3.4 Reporting bugs

To report bugs or suggest enhancements for GNU RCS, please visit its homepage (`http://www.gnu.org/software/rcs/`) to find directions on how to "file a bug report" online, or send electronic mail to `help-rcs@gnu.org`. (If you use the web interface, you don't need to also send email, since that is done automatically.)

For bug reports, please include enough information for the maintainers to reproduce the problem. Generally speaking, that means:

- The RCS version, command(s) and manual section(s) involved.
- Hardware and operating system names and versions.
- The contents of any input files necessary to reproduce the bug.
- The expected behavior and/or output.
- A description of the problem and samples of any erroneous output.
- Options you gave to `configure` other than specifying installation directories.
- Anything else that you think would be helpful.

When in doubt whether something is needed or not, include it. It's better to include too much than to leave out something important.

Patches are welcome; if possible, please make them with '`git format-patch`' and include `ChangeLog` entries (see Section "Change Log" in *The GNU Emacs Manual*). Please see file `HACKING` in the repo, for coding standards.

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

I

K

L

M

N

O

P

Q

R

S

T

U

V

W