The `rcs` Package

Joachim Schrod*

August 2, 1995
(Revision 2.10 of `rcs.sty`)

1. An important problem in program development and maintenance is version control, i.e., the task of keeping a software system consisting of many versions and configurations well organized. The *Revision Control System* (RCS) is a software tool that assists with that task. RCS manages revisions of text documents, in particular source programs, documentation, and test data. It automates storing, retrieval, logging and identification of revisions, and it provides selection mechanisms for composing configurations. In addition, it is able to insert management information in the text document, in so-called *RCS fields*.

The *Concurrent Versions System* (CVS) is a front end to RCS which extends the notion of revision control from a collection of files in a single directory to a hierarchical collection of directories consisting of revision-controlled files. These directories and files can be combined to form a software release. CVS provides the functions necessary to manage these software releases and to control the concurrent editing of source files among multiple software developers.

2. If you put LaTeX documents under RCS control, you'll often want to have access to the data of the RCS fields within your document, e.g., to put the date of the last checkin, the revision number, or the author's name in your document. This package provides TeX tags that give you access to this information. In addition, using a configurable RCS, you can typeset your revision logs.

3. There is one restriction: If you checked out with the `-kv` option, you cannot LaTeX your document any more. You'll get an error then. If you're a CVS user, the same holds for the '`cvs export`' command. (Actually, this is equivalent to '`co -kv`'.) This would discard the dollars and all keywords from the RCS fields; the tags explained below need the keywords to work properly.

If you have any idea how a tag might look which circumvents this problem, contact me. (I'm not asking for an implementation, I'm asking for an idea for a user interface!) E.g., are you hit by this problem, and are you willing to enter an additional argument with the keyword for each RCS field?

*Email: `jschrod@acm.org`

This package assumes the file naming possibilities of Unix. It does not work without changes on systems with a restricted file name syntax, like MS-DOS, VMS, etc. In particular, it assumes that RCS uses '`,v`' as the suffix for its files, as it's the default.

4. Let's first consider the access to the values of RCS fields. By writing

`\RCS $`*Keyword*`$`

in your document, a tag

`\RCS`*Keyword*

is created which expands to the value of the field with this respective keyword. E.g., if you write `\RCS $Revision: 1.10 $` and you check in your document, RCS will transform this to something like `\RCS $Revision: 1.10 $` and you can access the revision of your document as `\RCSRevision`. This will expand to '1.1' (note that the trailing space was discarded).

Of course, an RCS field has a value only if you have checked in your file at least once after you inserted it. If the field is brand new, the constructed TEX tag expands to nothing.[1]

5. The Date field is handled in a slightly different manner: Here `\RCSDate` doesn't expand to the original RCS value, it's transformed first. The expansion is the checkin date in a `\today`-like format. The checkin time is available via `\RCSTime` then.

If the Date value is empty, `\RCSDate` will expand to the current date and `\RCSTime` will expand to nothing.

The "original" Date value is stored in `\RCSRawDate`. This whole transformation is done only if the tag `\today` is defined (which is the case for all standard classes).

6. *Tags for your convenience.*

`\RCS` puts information from RCS fields into TEX tags. Some usages of such values are wanted more often than others: placing the checkin date on the title page, and placing the field keyword and value in a footline. This was made available in other `rcs` style options; we use the tag names from there for the sake of compatibility.

7. The tag `\RCSdate` (with a lowercase 'd') is a convenient way to put the last checkin date on the title page. It's used instead of `\date` and expects an RCS Date field afterwards; i.e., to be input as

```
\RCSdate $Date: 2003/02/02 22:30:22 $
```

Afterwards `\maketitle` will use the checkin date.

In fact, the Date value is still placed in `\RCSDate` (with an uppercase 'D') and this tag is used as the argument for `\date`. I.e., this is identical to writing

```
\RCS $Date: 2003/02/02 22:30:22 $
\date{\RCSDate}
```

[1] Well, not quite exactly—this can be changed. Refer to the documentation of the style's internal (programmer's) interface.

8. The tag `\RCSID` places the keyword and the value of the RCS field in a box into the footline. The RCS field is expected after the tag. The value is not transformed—you'll see it exactly as RCS inserted it. Aside from this, `\RCSID` has the same effect as `\RCS`: The field value is stored in `\RCS`*Keyword*.

Note: If you use this tag, you cannot use any other page style or package that sets the footline! The RCS field value must not be a file name with underscores, you get TEX errors otherwise.

9. The tag `\RCSdef` is like `\RCS`, but it additionally outputs the RCS field to the terminal. (As with everything output to the terminal, it will be in the `LOG` file, too.) The RCS field value must not be a file name with underscores, you get TEX errors otherwise.

10. *Typesetting revision logs.*

RCS creates a revision log (sometimes also called change log, version history, etc.) in the lines below a `Log` field. When one's LATEX document is really a program (e.g., if one uses a system of the WEB family like `CWEB` or noweb, or if one uses a documentation system à la MAKEPROG), one might want to incorporate such a log in the typeset document. This is strongly encouraged for every literate program—the version history is an integral part of a program's documentation.

This package provides an environment `rcslog` which supports the typesetting of logs. But you cannot use it with a normal RCS version. You need to be able to configure the format of a revision entry. We have created such a configurable version, you can retrieve it by anonymous ftp from `ftp.tu-darmstadt.de:pub/programming/management/confrcs`. (We welcome comments and bug reports.)

Since version 1.10, CVS has its own internal RCS implementation, therefore you cannot typeset revision logs with CVS, sorry. Please contact me if you find a way nevertheless.

11. You must create an RCS configuration file `.rcsrc` where a backslash is prepended to the revision entries. I.e., it must contain the following configuration line:

```
log_entry "\\Revision %r %d %t %a\n%l\n"
```

This line changes the format of revision entries for *every* file in your directory. If you want to have this special entry format only for specific files, you can put a guard in front of this configuration: the filename and a colon. You can also use wildcards in the filename. For example, to change the entry format for all files with the suffix `doc` use

```
*.doc: log_entry "\\Revision %r %d %t %a\n%l\n"
```

The configurable RCS gives you more possibilities, refer to its documentation.

12. The `rcslog` environment needs the `Log` field as the first line, there must be no material either before or behind it. If you just create a new document, you simply input the three lines

```
\begin{rcslog}
$Log: rcs-user.tex,v $
Revision 1.10  2003/02/02 22:30:22  schrod
Removed dependeny on \LaTeX2.09 font selection commands. (Though
they're still used in the package documentation, as convenience.)

\end{rcslog}
```

(Note that the configurable RCS will now set the comment leader to the empty string, which is exactly the value we need.)

If you have checked in your document already once and if you don't have a revision log by now, you still insert the three lines listed above. Then you have to set the comment leader to the empty string. If you don't have a WIMP interface to RCS, you do this with the command '`rcs -c` *filename*'.

If you have checked in your document already and if you already have a revision log, you insert the environment start (`\begin{rcslog}`) in a new line before the `Log` field and `\end{rcslog}` below the entries. Usually all lines from the revision log will be prefixed with a comment, delete this prefix from each line. Each revision entry starts with '`Revision`', change this to '`\Revision`'. Check that your log texts are valid LaTeX input. Set the comment leader to the empty string (this is described in the previous paragraph). Voilà, you're finished—welcome to the community of Literate Programmers!

13. And since the description of this functionality might have been a bit dry, I want to show you the output you get usually: I insert here the log of this user manual. In fact, it was created by the method described in the previous paragraph—the capability of typesetting revision logs was not available from the very start.

Revision Log for `rcs-user.tex`

Revision 1.9 (created at August 2, 1995 by Joachim Schrod)
Transformed this style option into a LaTeX 2_ε package.

Revision 1.8 (created at August 1, 1995 by Joachim Schrod)
Adapted to LaTeX 2_ε. Spell checked.

Revision 1.7 (created at November 10, 1993 by Joachim Schrod)
barbara proofread the user manual. Besides her minor changes she asked what RCS and CVS are, and if this can be used somewhere else than on Unix. I added appropriate paragraphs to explain the tools. Actually, the package may be used only on Unix, due to the dependency on the '`,v`' filename suffix.

Revision 1.6 (created at November 3, 1993 by Joachim Schrod)
Cleaned up for distribution: Added email address to each document, added copyright info to `rcs.doc`, added acknowledgments. Checked my English and the spacing.
Explained the restriction concerning the `-kv` option of `co`.

Revision 1.5 (created at November 2, 1993 by Joachim Schrod)
Introduced new tag `\RCSdef`. This is done to be upward compatible with Tom Verhoeff's `rcs` style option.

Revision 1.4 (created at November 2, 1993 by Joachim Schrod)
Introduced new tag `\RCSID`. This is done to be upward compatible with Nelson's `rcs` style option.

Revision 1.3 (created at November 1, 1993 by Joachim Schrod)
Added a description of the internal interface, I can refer now to that description. (Previously, I refered to the style's documentation.)

Don't create auxiliary files, we don't need them.

Revision 1.2 (created at November 1, 1993 by Joachim Schrod)
Explained the new feature of typesetting logs. This includes the user manual's log as an example.

Improved the documentation. Told first what happens, then the exceptions. Used "pseudo section headings" to give visual clues for the structure.

Added the style's revision number to the title page.

Used `rcs-doc.sty` for documentation of `rcs` style. It not only loads style options, but it also defines `\RCSStyleRevision` for access of the style's revision.

Prefixed keywords of RCS fields with '`Doc`'. This way we can use the normal keywords in examples and don't have to care about non-expansion.

Revision 1.1 (created at September 3, 1993 by Joachim Schrod)
Re-implemented `rcs` style option. Made it a documented option.

14. Nearly every detail of the formatting of the log is done by macros that can be changed, refer to the documentation of the style's internal interface for more information.

In particular, the `rcslog` environment accepts an optional argument, the configuration. Here LaTeX code is expected, which is inserted after the initial setup, before the heading is set. E.g., this is the place for you to change the type size. (Although this should be done with care—you'll also have to change the heading to get a consistent layout.)

In addition, the tag `\settime` may be used in this optional argument to add the checkin time to all revision entries. (Longtime RCS users will have noticed already that it was omitted.) This tag—as well as `\Revision`—is not defined outside the environment.

15. Longtime RCS users will have noticed another fine point in the log presented above: My full name was presented, not my user id (actually, '`schrod`'). This was done by placing the tag

```
\rcsAuthor{schrod}{Joachim Schrod}
```

somewhere before the log. Of course, if no full name has been presented for a user id, the user id itself is used as the author's name.

16. As explained above, one can configure the layout with an optional argument of the environment. (Or by redefining macros from the style writer's interface in another style option.) What's wanted sometimes are some introductionary words to the log; this can't be done easily this way. (Remember, the heading is not typeset yet.) Therefore, we provide a tag for this obvious need.

The introductionary text can be specified as the parameter of the tag `\rcsLogIntro`. It will be inserted after the heading, before the `Log` list.

17. *Compatibility issues.*

This package is user interface compatible to Piet van Oostrum's `rcs` style option. The internal interface of Piet's style option is not supported; check the documentation of this package's internal interface for its own internal configuration possibilities.

This package is user interface compatible to Nelson Beebe's `rcs` style option. But it does not set the footline unconditionally.

This package is user interface compatible to Tom Verhoeff's `rcs` style option. (Well, the text output by `\RCSdef` is not identical.)

And, of course, it's upward compatible to previous revisions ...