The `cweb` Class

Joachim Schrod*

November 20, 1995
(Revision 3.6 of `cweb.cls`)

1. `CWEB` is a Literate Programming tool that enables you to combine documentation and actual C or C++ code in one document. The `cweb` class allows you to use LaTeX for the documentation of your `CWEB` programs. Now a `CWEB` document is like a usual LaTeX document—as you have probably written dozens before. With all the features of LaTeX, and with all the quirks of LaTeX.

Originally, `CWEB` was only usable with Plain TeX, as `CWEAVE` outputs TeX code that looks as if it were Plain TeX. Well, it isn't—it's just not the normal LaTeX markup. No change in `CWEAVE` is necessary to use LaTeX.

For the sake of those `CWEB` programmers who have used the original markup interface (i.e., the Plain TeX macros) before, I'll sometimes contrast the approach presented here with the original one. Then I'll call the original interface "Plain `CWEB`". The new interface is called "LaTeX `CWEB`".

If you never used `CWEB` before, you won't find enough information in this manual. Please read the user manual of `CWEB` first, and then come back again. I assume a working knowledge, and will only present the new stuff introduced by this class.

2. This bundle does *not* support inclusion of `CWEB` code in other (general) LaTeX documents. This is not the task of the `cweb` class – its task is the usage of LaTeX for creation of `CWEB` documents.

Note also that you will not be able to LaTeX your `CWEB` documents (i.e., the program files) directly. You must still process them first by `CWEAVE`, then you can use LaTeX to process the resulting TeX file.

3. You have to use `CWEB` version 3 and you need at least LaTeX version ⟨1994/12/01⟩ (or any later one), this class does not work with older versions.

*Email: `jschrod@acm.org`

4. A CWEB document has two concurrent structures, a *document structure* and a *program structure*.

The basic building block of the document structure, the bricks that build up our document, are called *chunks*. We use chunks to present one point of our work to our reader, the presentation has both informal (explanation) and formal parts (code). This is the essence of Literate Programming, that we tell each point twice, once on an abstract and once on a technical level.

A chunk consists of three parts: (1) the *documentation part*, (2) the *definition part*, and (3) the *program part*. Each of these parts can be empty. The documentation part is mostly text with LaTeX tags. In this text material from *restricted program mode* can appear. The definition part consists of a series of either *macro* or *format definitions*. The program part is one piece of a refinement, identified by a name (see below).

A chunk starts with '@␣', and gets assigned a unique identification, a number; we'll use that number to refer to that chunk. In particular, the number is of interest when we want to refer to the code part of the chunk, otherwise it's often regarded as superfluous.

5. And we have higher-level structure elements as well. They combine a sequence of chunks and a sequence of structure elements on a lower level, both sequences may be empty. As usual in LaTeX, we call these elements *section divisions* or short *sections*. We do not give them names like 'part,' 'chapter,' 'subsection,' etc., to denote their level; most often we'll use level-numbers (sometimes called *rank*) instead.

A section division starts with '@*' and is followed by a character, the rank indicator. The highest rank is '@**', you can think of it as an equivalence to LaTeX's \part. (Often we don't use that rank in smaller programs, as we don't use \part in smaller articles.) A section on the next rank is started with '@*␣' (synonym to '@*0'), roughly equivalent to chapters. Then we'll use numbers to mark the next ranks, from @*1 to @*9 (sections to sub^{9}sections). That's outrageous—I'll hope you'll never use 11 hierarchy levels in your documents, they would be unreadable.

A section has a title, that title comes after the section start tag and ends with a full stop. If you need periods in your title, enclose them in braces. (That behavior can be changed if there's enough interest. I can also make it possible to use braces to delimit the section title, as in \section, etc. Send mail with your opinion.)

As a matter of convenience, a section implicitly starts a chunk; i.e., one does not need to use '@␣' after the section title. Of course, it doesn't hurt to use it either.

The explanation might have been a bit too abstract, so let me make one thing plain clear: You should *never ever* use \part, \chapter, \section, or any other LaTeX sectioning command in your CWEB documents. Use the sectioning commands of CWEB, i.e., '@*rank'.

6. The description above does not mean that the output is fixed to the style you'll know from the book or report document classes. Quite in contrary, the default layout is oriented along the article class.

There is a very easy possibility to change that default layout: With the option baseclass (see chunk 16.) you can specify the document class cweb inherits from. If that document class has chapters (like report or book), you'll get the layout as you're used from these classes.

7. In Plain CWEB, the term *section* was used for 'chunk.' We abstain from that usage, as sections are used in LaTeX (and in almost all other documentation systems, for that matter) to designate higher-level structure elements. We prefer to follow the traditional usage of terms here, even if that means that we'll have to confront Plain CWEB users with a slight shift in semantics.

And you'll find the term *starred section* in Plain CWEB and might think these are our sections. Well, almost. There's a subtle, but important difference between these starred sections and our terminology: A starred section is just a chunk with a special attribute (a title). That title will appear in a table of contents, but that does not mean that the starred section is an element to denote a structural *hierarchy*.

You won't find a section number like '1.5.2' in Plain CWEB, do you? — In LaTeX CWEB, you've got the choice. You have the new support of the hierarchical structure, or you may stay with Plain CWEB's (flat) structure.

8. We still have one open point in our explanation of the document's structure. As mentioned at the start, we've got two concurrent structures, let's have a look at the program structure finally.

A CWEB program consists of a tree of *refinements*. A refinement is a sequence of program parts with the same name, ordered in appearance in the document. The root of the tree is the refinement with the special name `@c`.

The complete program text is defined by the depth-first traversal (DFS, i.e. infix-order) of the tree, the tool CTANGLE extracts the program text from a CWEB document.

9. Markup for your document structure.

Like every LaTeX document, a CWEB document starts with a document class specification; then comes the preamble, terminated by the document start. This way we brought some structure in the CWEB limbo, but this shouldn't be a problem for you. You must tag the end of the document, as you do in other LaTeX files. But here you must take a bit more care:

> *Assert that the* `\end{document}` *is in the documentation part of a chunk, neither in the definition nor in the program part.*

If it would be in the program part, CWEAVE would readily process it and LaTeX would never see this tag. Nevertheless, if you make this mistake, you will detect it early: CTANGLE will copy this tag to the C file as well, and your program will be erroneous...

And there's another difference you'll have to take care of: Usually you can put arbitrary text after the end of your document, LaTeX will not see it. Please note that both CTANGLE and CWEAVE will still see it—don't put a program part or even complete chunks there. For technical reasons, there must not be any `\fi` token after the document end, too.

10. The resulting structure of your CWEB document is exemplified in figure 1. Note that `\end{document}` is placed in a new, empty, chunk. This isn't necessary, you could write more documentation there—but it's considered good style to use a sole chunk for document finish.

```
\documentclass{cweb}

\begin{document}

\title{My Program, Doomed for the ACM Software Systems Award}
\author{Joe L. User}

\maketitle

\tableofcontents        % if you want


@* A PSPACE solution for the Traveling Salesman.

< insert your program here >



@
\end{document}
```

Figure 1: Exemplified CWEB document structure

11. If you don't call CWEAVE with the -x option, an identifier index and a list of all refinements is created at the end of the document. You can specify an introductionary text for the index with the tag \cwebIndexIntro, the introduction is the argument of this tag.

12. Configuration.

You may configure the cweb class in several ways, with class options, redefinitions in the preamble, supplying additional packages, and by subclassing it.

At start of the processing, the file `cweb.cfg` is read if it exists. That file may contain configuration code that shall apply to all your CWEB documents. You can set the paper size there, etc. Please don't set the default structure there (see chunk 14.), others would get different results when they process your document.

13. Options are specified with the new "keyword=value"-scheme, first introduced with the graphicx package. I.e., the options have a set of possible values, you may choose one. For instance, the option `structure` has the possible values `flat` and `hierarchic`. You tell that you want a flat document structure by

```
\documentclass[structure=flat]{cweb}
```

As usual, options are separated by commas. From the set of possible option values, two ones are special: The *predefined value* and the *default value*. The predefined value is the one that's used when you don't specify the option at all. The default value is used of you don't specify the value in the option; e.g.,

```
\documentclass[suppress]{cweb}
```

selects the default value of the option `suppress`. (Not that there is one currently. :–)

There is always a predefined value, a default value may not be there. If a default value exists, it is always different from the predefined one. The default value gives you an easy specification of an option value that is expected to be common. If that value would be the predefined one, you won't need to specify the option in the first hand.

The following options are available and explained below:

`structure` Select the structure model used by the cweb class.

`suppress` Suppress output of different document parts.

`baseclass` Select the base document class.

`language` Select a language for inserted texts.

14. We have two possibilities to output chunks and sections, either as a flat or as a hierarchic structure. This configuration is selected by the option `structure`, either through the value `flat` or `hierarchic`.

`flat` The flat structure is the "classic" approach, the way Plain CWEB renders its chunks. Then each chunk starts with its number. The chunk numbers are output in boldface, followed by a dot and a quad.

Sections show chunk numbers as well (remember, they implicitly start a chunk). "Important" sections are added to the table of contents.

`hierarchic` The hierarchic structure is the "modernistic" (aka LaTeX) approach, then we mark the start of a chunk by a start sign. (By default, that's a paragraph sign, '¶'). Chunk numbers are rendered in the margin of the program part. If there's no program part, no chunk number will be rendered—after all, they are only used for identification of program parts.

Section numbers are hierarchic, like in other LaTeX document classes.

The predefined value of `structure` is `hierarchic`. There doesn't exist a default value.

15. The option `suppress` allows to select suppression of different document parts. You have different parts you can suppress. The value of the `suppress` option is a comma-separated list of identifiers, enclosed in braces. If you have only one thing you want to suppress, you don't need to surround that identifier by braces. Possible identifiers are:

`changehints` suppress output of hints that a changefile was used: The change flag (by default a star at the start of a chunk) and the list of changed chunks at the end of the document.

`unchanged` suppress output of chunks that are not changed by a changefile. This value implies `changehints`—by definition all printed chunks are changed, it doesn't make any sense to hint to that fact in addition.

`index` suppress the identifier index.

`reflist` suppress the refinement list.

`format` suppress output of '`@f`' directives.

The predefined value is the empty list, i.e., nothing is suppressed. There is no default value since I don't know which parts are suppressed most often. Send email with your demands, I'll add a default if I see that there is one major application of this option.

16. The option `baseclass` allows to select the base document class used by the cweb class. In particular, if the base class has `\chapter` defined the layout of major section divisions ('`@**`' and '`@*␣`') will be changed to be in a chapter layout.

The predefined value is `article`, the default value is `report`.

17. The option `language` allows to adapt inserted texts to different languages. E.g., with the option '`language=german`' German texts are inserted for cross references, headings, etc. That option will neither switch on language-specific hyphenation, nor will it trigger any other processing – that's the task of Babel or any other LaTeX language style.

There is neither a predefined nor a default value. (Without the `language` option, inserted texts are in English.)

That option actually triggers the usage of a package cwbl-*language*. To see if a language is supported, you have to look for respective files in the LaTeX `CWEB` macro directory.[1] Of course, the easiest way to check the availability is to try it out – you'll get a comprehensible error message if there is no respective `CWEB` language package. In November 1995, `german`, `french`, and `italian` were supported, more might have been added later.

For compatibility with Babel, some other options (e.g., `german` or `french`) are also supported as nicknames for `language=`*option*. If you use packages that make also use of these options (e.g., babel or varioref), you'll probably prefer them over `language` since they are picked up by those packages as well.

[1] That's `texmf/tex/latex/cweb`, on a TDS compliant installation. Please beware that pseudo operating systems like DOS cripple file names and shorten them to eight characters; if you see a file `cwbl-ger.sty` you'll know that the option value `german` is supported.

18. Restrictions.

The following restrictions will not be withdrawn as far as I can see—except if somebody will send me a patch with changes. I tried to sort the restrictions in order of severity.

19. *(Restrictions due to the design of* `CWEAVE`*.)*
Please be aware that the vertical bar ('`|`') is used by `CWEB` to delimit small C code pieces in the documentation parts, and is therefore processed (and mangled) by `CWEAVE`. *You cannot use it for* TEX *any more.*

In particular, you cannot specify rules for the `tabular` or the `array` environment. Since you most certainly want to do so: You have two choices left:

1. Make sure you have the `array` package (by Frank Mittelbach and David Carlisle, from the Tools bundle) installed. Then you may use the package `cwebarray`, it defines '`I`' (that's an uppercase 'i') as a specifier for rules. I.e., instead of `\begin{tabular}{l|l}` you have to write `\begin{tabular}{lIl}`. Not the most elegant solution, but it works...

2. Use '`^^7c`' instead of '`|`'. I.e., instead of `\begin{tabular}{l|l}` you may write `\begin{tabular}{l^^7cl}`.

These two choices are compatible, you may use both in one document. Needless to say, I consider the first alternative the better one.

20. Neither a refinement name nor an index entry made by `@^` may consist of a *single* dot-accented term. I.e., you must not write '`@<\.O@>`', '`@^\.o@>`', or even '`@^\.{foolish}@>`'. Of course you may write '`@^\.o, accent@>`' or '`@< Handle accent \.o @>`'.

21. *(Restriction due to the implementation of* LATEX*.)*
One cannot use restricted C mode in moving arguments. Most notably, this is disturbing in the section titles and in `\caption` tags. Sorry, folks.

(Basically, that's because the definition of `\addcontentsline`, etc., in the LATEX kernel is brain damaged; it wants to expand its argument. And I don't want to maintain redefinitions of such basic macros.)

22. *(Restrictions due to* TEX*.)*
C++ comments (i.e., from `//` to the end of the line) are typeset as C comments. This is especially bad if they are used for a whole block of comment lines, as it is quite common. Please put such comment blocks in the documentation part. (An `\everyline` implementation would be needed to lift that restriction, and that's impossible in general.)

23. Details.

The following tags are reserved and must neither be used nor redefined: `\ATL`, `\B`, `\M`, `\N`, `\PB`, and `\Y`.

`\9` is already explained in the `CWEB` user manual: It's a special control sequence used for the index entries tagged with '`@:`'. Its default definition is setup in such a way that you can cheat `CWEAVE` concerning the sort order of this entry: If you enter '`@:sort}{print@>`' you will get an index entry "print" next to the place where the index entry "sort" would be. But you're allowed to change this default definition.[2]

The names of all other control sequences defined by this class—besides the common LaTeX control sequences—start with `cweb`, `Cweb`, or `cwbb`. Please don't define new control sequences starting with this prefix. (The control sequences starting with `Cweb` may be redefined in a package to change the appearance of the `CWEB` document or the behavior in a controlled manner, check the description of the internal interface for this possibility.)

24. Plain `CWEB` users should note that the macros from `cwebmac.tex` are not available any more. E.g., you cannot use `\.` to typeset typewriter material; use either `\texttt` or `\verb`, as it fits the situation. On the other hand, now you're able to use `\.` for the dot accent, you can define `\3` for the 'sharp s', `\C++` for the C++ logo,[3] etc.

Another detail for ex-plainies: The table of contents is produced by the `\tableofcontents` tag (during the second LaTeX run), not automatically. But this is the standard LaTeX behavior.

[2] Remember: A package is a very good place to place such redefinitions; your document should be concerned with contents, not with appearance.

[3] I will not define any such logos in this class. A package like logos with lots of name definitions is the appropriate place for this.

25. Known Problems.

There is a bunch of known bugs, problems, and ommisions.

Bugs:

- The presentation of `@l` redefinitions is not proper. But it wasn't in Plain `CWEB`, either.

Problems:

- You can't use `\index` and `\makeindex` as-is to create an index. Add your entries instead to `CWEAVE`'s index using '`@^`', '`@.`', or '`@:`'. That problem is caused by a clash in the usage of the filename extension `idx` by `CWEAVE` and LaTeX. The `index` package by David Jones will help you to define additional index categories (that use an other extension) if you don't want to mix identifiers and your other index entries.

Ommisions ("wouldn't it be nice, if"s):

- The current chunk number and most section titles (below rank 1 or 2, depending on the base class) are not available as a mark for inclusion in a running head.
- Better integration of `rcs` package. That package is used to support including information from the revision management system RCS into the document. E.g., the revision log at the end of that document was produced by that package.

26. I would like to thank those who helped me to improve this bundle.

Michael Müller and Zdeněk Wagner did thorough checks that helped me to improve the alpha test version.

Christian Kumpf triggered the addition of language support and pointed out where configuration could be made easier by supplying more macros in the protected interface. Andreas Scherer provided packages for French and Italian `CWEB` documents.

Laurent Desnogues, Felix Gärtner, and John S. Robinson provided error and problem reports. Bronne Louis pointed out the embarrassing error that section numbers were not reset when a higher-level section began.

Revision Log for `cweb-user.tex`

Revision 2.10 (created at November 20, 1995 by Joachim Schrod)
Add language support.
Triggered by Christian Kumpf `<smurf@igd.fhg.de>` and Andreas Scherer `<scherer@physik.rwth-aachen.de>`.
Use my 'official' (ACM) email address.

Revision 2.9 (created at November 20, 1995 by Joachim Schrod)
Explicate that one must not use `\section` etc., that one cannot LaTeX the `CWEB` program directly but must run `CWEAVE` first, and that the cweb class does not provide inclusion of `CWEB` programs in other LaTeX documents.
Clarification triggered by Laurent Desnogues `<laurent.desnogues@aiguemarine.unice.fr>`.

Revision 2.8 (created at September 15, 1995 by Joachim Schrod)
The cweb class needs at least LaTeX version ⟨1994/12/01⟩.
Problem reported by John S. Robinson `jsrobin@umiacs.umd.edu`.

Revision 2.7 (created at September 12, 1995 by Joachim Schrod)
`cwbb` is also a reserved namespace.
Added acknowledgements.

Revision 2.6 (created at August 29, 1995 by Joachim Schrod)
LaTeX `CWEB`, version 1.0.

Revision 2.5 (created at August 29, 1995 by Joachim Schrod)
Discard dependencies on 10 pt fonts.
Support suppression of format directives.

Revision 2.4 (created at August 27, 1995 by Joachim Schrod)
Suppression of index and reference list is supported.
Discard documentation on `\cwebSecNoEject`, that's part of the protected interface.

Revision 2.3 (created at August 27, 1995 by Joachim Schrod)
Make usage of baseclass with chapters work.

Revision 2.2 (created at August 27, 1995 by Joachim Schrod)
Add possibility to suppress change hints. Suppression of unchanged chunks suppress change hints as well, they are meaningless as all printed chunks are changed by definition.
Update problems section, they were partly resolved by the current changes. Added hint to index package that resolves the problem of extensions clashes with *MakeIndex*.

Revision 2.1 (created at August 25, 1995 by Joachim Schrod)
Add keyword-value option style, with new keyvald package.
Hierarchic strucutures are supported now, in addition to the flat structure of the beta-test version. One can choose with an option. For that step, the terminology was cleaned up, too: Chunks are not named sections any more. (That change involved reimplementation of almost all the structure and toc stuff.)
The chunk number supplied by `CWEAVE` is used now, not some computed number. Change flags are printed, too.
One can suppress output of unchanged sections.
One can select the baseclass with an option. That may be used to use report or book to get chapter-style layout. Of course, using an arbitrary baseclass is dangerous, it must conform to the conventions of LaTeX standard classes.

Revision 1.10 (created at August 8, 1995 by Joachim Schrod)
Updated to LaTeX 2ε, the `cweb` style is now a document class. Used my standard templates for that, no changes in functionality.

Revision 1.9 (created at August 10, 1993 by Joachim Schrod)
New page on main section only if group level < `\cwebSecNoEject`. Default for the latter is 3.
Document that logos will not be defined in this style file.
Copy of plain macros for `\CwebNumber` does not work. Repaired the most important one (subscript must be accessed via `\sb`). Incompatibility to NFSS will be addressed later.
(Problems reported by Zdeněk Wagner `<wagner@csearn.bitnet>`.)

Revision 1.8 (created at August 10, 1993 by Joachim Schrod)
Reference to section number does not render a period after the number any more.

Revision 1.7 (created at August 9, 1993 by Joachim Schrod)
Mentioned that '|' cannot be used for LaTeX purposes, in particular, not for ruled tables. Described workarounds, one of them is the new style option `cwebarray`.
(Problem reported by Felix Gärtner `<theedge@rbg.informatik.th-darmstadt.de>`.)

Revision 1.6 (created at June 17, 1993 by Joachim Schrod)
'`cweb`' was still an option in the example.

Revision 1.5 (created at June 17, 1993 by Joachim Schrod)
`CWEB` 3.0 was released officially on June 16, 1993. Mentioned in the documentation that this version is needed for the `cweb` style.

Revision 1.4 (created at June 15, 1993 by Joachim Schrod)
It's a style now, was an option formerly.

Revision 1.3 (created at June 14, 1993 by Joachim Schrod)
Added section about restrictions (no restricted C mode material in moving arguments, no single dot-accented term in an index entry).

Revision 1.2 (created at May 12, 1993 by Joachim Schrod)
Adapted to recent changes of `CWEAVE` (of April 93): Main sections have a group level, represented in the table of contents.
Boxed the example, it was not legible formerly.

Revision 1.1 (created at April 9, 1993 by Joachim Schrod)
Initial revision