volumes.sty: Support for Printing of only parts of a LaTeX document, with complete indices etc.

Frank Küster

2004/06/02

Abstract

This package tries to help you if you want to produce separate printed volumes from one LaTeX document, as well as one comprehensive "all inclusive" version. It suppresses the parts of the table of contents that are not typeset, while counters, definitions, index entries etc. are kept consistent throughout the input file. The same goal can probably also be achieved by different approaches, e.g. packages to combine separate documents in one typeset version. But for me it was easier this way – I define lots of macros in the document on-the-fly, and have to avoid double definitions in the different parts.

This file also provides nowtoaux.sty which allows writing to the `aux` file, and thus to the `toc`, `lof` and `lot` files, without waiting for a page to be shipped out.

Contents

Part I
User documentation

1 Purpose, alternatives, and credits

If you use \includeonly to typeset only parts of your document, you have only two choices: Either you delete the old aux files, and get no information at all about the parts that are not typeset, or you keep it, but then you get a full table of contents, lists of figures and tables, etc., even for the parts that are not included. Thus, \includeonly is only suitable to keep compilation times short while writing parts of your document, but not for the final typesetting of parts of a document.

If you want to have a more fine-grained control over what is being typeset, volumes.sty may help you. There are, however, some alternatives. If you know in advance that you will want to typeset your package in two or more volumes, it might be better to develop it as separate documents, using xr.sty for references to the other document, or to use tools as shorttoc.

When I developed this package, I had yet written a large part of the document – in fact it was a laboratory diary (using my own labbook class) – and was using lots of automated indexing commands that occurred in all parts, and macros that defined and re-used command sequences on-the-fly. I wanted those commands to be consistent throughout my work in this lab, avoiding dobule definitions, and I wanted the complete index, also for older volumes, in every volume. Therefore it seemed harder to take apart the whole document, than suppressing a part of the table of contents.

In a discussion with Markus Kohm, he came up with the basic idea for this, and I am much in debt to him for his help. Some definitions where also taken and/or adapted from his scrclass.dtx, as well as from basic LaTeX 2ε files, and the contributors of the newsgroup de.comp.text.tex also helped me, as usual.

2 Using the volumes Package

\onlyvolume To use the Package, you must at least load it and call the \onlyvolumes macro:

```
\usepackage{volumes}
\onlyvolume
```

The reason why you need two commands is that it makes it possible to customize it's behavior between the `\usepackage` and `\onlyvolume` (see below).

volumes.sty assumes that you are using `\include` for the parts that you want to typeset conditionally, and standardized, numbered filenames (see below). However, the appearance of your document still won't change if you insert just the lines above. You also need to specify which part should be typeset. There are two ways to accomplish that:

1. You can specify the part you want as an optional argument to `\onlyvolume`, for example:

   ```
   \onlyvolume[2]
   ```

   to get the second part. The disadvantage for this is that you have to change the document when you want to change the part to be typeset.

`\volume`

2. Alternatively, you can define the macro `\volume` outside the file, it should expand to the number of the part you want. This can be done on the command line, e.g. like this:

   ```
   latex "\def\volume{2} \input{filename}"
   ```

volumes.sty will handle each file that is included with `\include` as one volume, and treat the rest as fixed parts. It does not take care of `\input` at all, therefore you're free to use it as you want.

2.1 Customization

`\volumename`

Filenames By default, volumes.sty expects the included files to be named `volume1.tex`, `volume2.tex` etc.[1] The numbers at the end must always be there, but you can change the basename from `volume` to ⟨*anything*⟩ else using
`\volumename{`⟨*anything*⟩`}`

`\allvolumescommand` `\volumecommand`

Commands specific to volumes You might want certain commands only to be executed when the whole document is typeset, and others only for a specific volume. To achieve this, write `\allvolumescommand{`⟨*commands*⟩`}` and/or `\volumecommand{`⟨*number*⟩`}{`⟨*commands*⟩`}`.

The commands are currently[2] executed at the end of the preamble using `\AtBeginDocument`. Therefore you cannot typeset anything, instead you should define commands that are later typeset, e.g. change the `\title` or `\date`.

Excluding `\addtocontents` commands The mechanisms presented so far only exclude things added to the `toc`, `lof`, or `lot` files by sectioning commands or floats, or using `\addcontentsline`. If you add some stuff manually to those lists using `\addtocontents`, it still gets typeset.

[1] In fact this is only true for conditionally included files. If you use `\alwaysinclude` (see below), those files can have arbitrary names.

[2] In future versions, this might be moved to an other place - don't rely on that.

To circumvent this, you can use `\voladdtotoc{`⟨*Text*⟩`}`, `\voladdtolof{`⟨*Text*⟩`}`, and `\voladdtolot{`⟨*Text*⟩`}`, instead of `\addtocontents{`⟨*Text*⟩`}`. These texts will automatically be suppressed if the respective list is suppressed in the part where they are used.

2.2 Options: Determining what to exclude

volumes.sty by default suppresses the unprinted entries into the Table of Contents (`lof`), List of Figures (`lof`) and List of Tables (`lot`). You can change this using the following options:

`tocall` The complete table of contents is printed even when only typesetting one part, i.e. only the lists of tables and of figures is suppressed

`lofall` The complete list of figures is printed even when only typesetting one part, i.e. only the lists of tables and table of contents is suppressed.

`lotall` The complete list of tables is printed even when only typesetting one part, i.e. only the lists of figures and table of contents is suppressed.

`\volumeone`
`\volumetwo`
`\volumethree`
`...`

We redefine the `\include` command to achieve this. If this causes incompatibilities with other packages, use the option

`manual` You must then specify the beginning of a volume by putting the commands `\volumeone`, `\volumetwo`, `\volumethree` and so forth directly before the corresponding `\include`. The filenames still need to follow the conventions described above.

2.3 Future plans

Here are some ideas that I had, and believe can be implemented quite easily – but I didn't have time yet:

- Check whether the filename in `\include` maches the pattern `\volumename`⟨*number*⟩ and include it always if it does not.
- Provide a user interface that allows to suppress entries in other lists, like lists of equations, listings, or user-defined floats.

3 The nowtoaux Package

The `\addtocontents` macro uses the TEX primitive `\write` to insert its text into the `aux` file, from where it will finally end up in the `toc` file etc. `\write` puts its contents into a *whatsit*, and the actual writing is done when the material currently processed fills a page, and the page is shipped out. This way, putting `\thepage` into the arguments of `\addtocontents` will produce the correct page number. However, sometimes one really wants the writing to take place immediately, when the macro is expanded. This can be achieved with the nowtoaux package.

Consider the following example, which tries to typeset the table of contents in the main part of the document in red:

```
\documentclass{report}
```

```
\usepackage{color}
\begin{document}

\tableofcontents

\include{intro}
\addtocontents{toc}{\color{red}}
\include{main}
\end{document}
```

Here, the `\addtocontents` command is at a place where nothing is typeset: The previous `\include` has just caused a page break, and the next include will open its own `aux` file, before also doing a `\clearpage` and starting the typesetting, probably of many pages with sections that go into the toc. It's only after the processing comes back from `main.tex` that TEX notices that a *whatsit* is left, and writes to the `aux` file.

`\immediateaddtocontents`

What you want instead is a command like `\addtocontents` that writes immediately to the `aux` file, without waiting for a page shipout. The nowaux package defines `\immediateaddtocontents{`⟨*table*⟩`}{`⟨*Text*⟩`}` which does exactly that. If you substitute it for the `\addtocontents` in the above example, the entries in the table of contents that come from `main.tex` are colored red (how horrible...).

`\immediatewriteaux`

You can also write arbitrary commands to the aux file, if you find a use for this. `\writeaux{`⟨*commands*⟩`}` writes the commands to the aux file, using a *whatsit* as usual. `\immediatewriteaux{`⟨*commands*⟩`}` does the same, but with `\immediate`. The commands in their argument need to be properly protected. It is best to define one command to perform all the tasks that you want to be done in the aux file. You can then put this command into the aux file with `\macrotoaux{`⟨*one_command*⟩`}`, without further protection, because this is done internally. Here's an example:

4 The test suite

The principles of the test suite will be described elsewhere. Here are just some remarks on how it works in this particular case.

- In the first run, the file switch-onlytwo.tex does not exist. Therefore, `\volume` is undefined, and the complete document is processed, producing aux files also for the files that will not be included in later runs.

Part II
Implementation

5 The nowtoaux package

`\immediateaddtocontents`
`\immediate@protected@write`

The definitions of `\immediateaddtocontents` and `\immediate@protected@write` are taken from the definition of `\addtocontents` in ltsect.dtx and `\protected@write` in ltfiles.dtx, and the only change is the addition of `\immediate` before `\write`.

1 ⟨∗nowtoaux⟩

```
\long\def\immediateaddtocontents#1#2{%
  \immediate@protected@write\@auxout
  {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
  {\string\@writefile{#1}{#2}}
}
\long\def \immediate@protected@write#1#2#3{%
  \begingroup
  \let\thepage\relax
  #2%
  \let\protect\@unexpandable@protect
  \edef\reserved@a{\immediate\write#1{#3}}%
  \reserved@a
  \endgroup
  \if@nobreak\ifvmode\nobreak\fi\fi
}
```

\macrotoaux
\immediatewriteaux
\writeaux

Here come three goodies: `\immediatewriteaux` allows you to write arbitrary commands to the aux file. `\writeaux` does the same, but in the standard way, without `\immediate`. The commands in their argument need to be properly protected. It is best to define one command to perform all the tasks that you want to be done in the aux file. You can then put this command into the aux file with `\macrotoaux`, without further protection, because this is done internally.

```
\long\def\writeaux#1{%
  \protected@write\@auxout
  {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
  {#1}
}
\long\def\immediatewriteaux#1{%
  \immediate@protected@write\@auxout
  {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
  {#1}
}
\long\def\macrotoaux#1{%
  \immediatewriteaux{\string#1}
}
⟨/nowtoaux⟩
```

6 The volumes Package

6.1 Option handling and needed packages

For the all options, we simply create a conditional that is checked later, when the necessary commands have been or will be defined. Furthermore, we need the nowtoaux helper package.

```
⟨*volumes⟩
\newif\if@allincludes\@allincludestrue
\newif\if@tocall\@tocallfalse
\newif\if@lofall\@lofallfalse
\newif\if@lotall\@lotallfalse
\DeclareOption{manual}{\@allincludesfalse}
\DeclareOption{tocall}{\@tocalltrue}
\DeclareOption{lofall}{\@lofalltrue}
```

```
\DeclareOption{lotall}{\@lotalltrue}
\ProcessOptions%
\RequirePackage{nowtoaux}
```

6.2 Suppressing `\addtocontents` commands

```
\newcommand{\vol@do@addto}[1]{#1}
\let\vol@dont@addto\@gobble
\newcommand{\voladdtotoc}[1]{%
  \addtocontents{toc}{\protect\vol@maybe@addto@toc{#1}}%
}%
\let\vol@maybe@addto@toc\vol@do@addto
\newcommand{\voladdtolof}[1]{%
  \addtocontents{lof}{\protect\vol@maybe@addto@lof{#1}}%
}%
\let\vol@maybe@addto@lof\vol@do@addto
\newcommand{\voladdtolot}[1]{%
  \addtocontents{lot}{\protect\vol@maybe@addto@lot{#1}}%
}%
\let\vol@maybe@addto@lot\vol@do@addto
```

6.3 Suppressing entries in the table of contents

In order to get the table of contents, list of figures and list of tables with entries only for the part that we want to print, we redefine `\contentsline` to do nothing. This is done by writing `\let` assignments into the respective files (`*.toc`, `*.lof`, `*.lot`).

`\contentsline`
`\volumes@orig@contentsline`

But we need to be able to suppress parts at the beginning, and reenable later parts. Therefore we save the old definition of `\contentsline` in the macro `\volumes@orig@contentsline`. If the hyperref Package will be loaded later, it will redefine `\contentsline`. Therefore we have to repeat our command after (and only if) hyperref has been loaded:

```
\let\volumes@orig@contentsline\contentsline \RequirePackage{scrlfile}
\AfterPackage{hyperref}{\let\volumes@orig@contentsline\contentsline}
```

The new `\contentsline` macro should do nothing, but just gobble its arguments. Again a hyperref problem: If the package has been loaded, `\contentsline` will have 4 arguments instead of 3. We do the same trick, but this time we also have to check whether hyperref has been loaded yet:

```
\@ifpackageloaded{hyperref}{%
  \let\volumes@new@contentsline\@gobblefour
}{%
  \def\volumes@new@contentsline#1#2#3{}
  \AfterPackage{hyperref}{%
    \let\volumes@new@contentsline\@gobblefour
  }
}
```

We want to `\let` the `\contentsline` macro to the meaning we want in the `*.toc` file etc., but we cannot write directly to those files. Instead, we write to the `*.aux` file using `\immediateaddtocontents`. Commands written to the `aux` and `toc` file this way need to be proctected; in order to make this easier, we define

macros that will do the `\let` assignment and can be proctected with one `\protect` command:

```
\def\volumes@switch@orig@contentsline@toc{%
  \let\contentsline\volumes@orig@contentsline%
  \let\vol@maybe@addto@toc\vol@do@addto
}
\def\volumes@switch@new@contentsline@toc{%
  \let\contentsline\volumes@new@contentsline%
  \let\vol@maybe@addto@toc\vol@dont@addto
}
\def\volumes@switch@orig@contentsline@lof{%
  \let\contentsline\volumes@orig@contentsline%
  \let\vol@maybe@addto@lof\vol@do@addto
}
\def\volumes@switch@new@contentsline@lof{%
  \let\contentsline\volumes@new@contentsline%
  \let\vol@maybe@addto@lof\vol@dont@addto
}
\def\volumes@switch@orig@contentsline@lot{%
  \let\contentsline\volumes@orig@contentsline%
  \let\vol@maybe@addto@lot\vol@do@addto
}
\def\volumes@switch@new@contentsline@lot{%
  \let\contentsline\volumes@new@contentsline%
  \let\vol@maybe@addto@lot\vol@dont@addto
}
% \def\volumes@switch@orig@contentsline{%
%   \let\contentsline\volumes@orig@contentsline}
% \def\volumes@switch@new@contentsline{%
%   \let\contentsline\volumes@new@contentsline}
```

`\volume@switch@off` `\volume@switch@on`

The actual writing to `*.aux` will done by `\volume@switch@on` and `\volume@switch@off`. While `\volume@switch@on` switches on unconditionally, `\volume@switch@off` checks whether any of the `tocall`, `lofall`, or `lotall` options was given; if writes to the respective file only if the option wasn't given.

```
\def\volume@switch@off{%
  \if@tocall\else%
    \immediateaddtocontents{toc}{\protect\volumes@switch@new@contentsline@toc}%
  \fi%
  \if@lofall\else%
    \immediateaddtocontents{lof}{\protect\volumes@switch@new@contentsline@lof}%
  \fi%
  \if@lotall\else%
    \immediateaddtocontents{lot}{\protect\volumes@switch@new@contentsline@lof}%
  \fi%
}
\def\volume@switch@on{%
  \immediateaddtocontents{toc}{\protect\volumes@switch@orig@contentsline@toc}%
  \immediateaddtocontents{lof}{\protect\volumes@switch@orig@contentsline@lof}%
  \immediateaddtocontents{lot}{\protect\volumes@switch@orig@contentsline@lof}%
}
```

6.4 User commands for the appearance of volumes

\allvolumescommand
\volumecommand

In order to allow the user to define commands that should be executed conditionally only if a particular volume is typeset, we provide the command `\volumecommand{`⟨*number*⟩`}{`⟨*LaTeX commands*⟩`}`. Similarly, the macro `\allvolumescommand{`⟨*LaTeX commands*⟩`}` is executed when the whole document is typeset. The internal commands that are generated by both macros need to be set to `\relax`, so that no error occurs if the user does not use (some of) the `\volumecommand`s and `\allvolumescommand`. For the numbered `\volume@command@..`, this is done later.

```
\def\volumecommand#1#2{%
  \expandafter\def\csname volume@command@#1\endcsname{%
    #2%
  }%
}
\def\allvolumescommand#1{%
  \def\all@volumes@command{#1}%
}
\let\all@volumes@command\relax
```

6.5 Determining which part(s) to typeset

Now we define the macro that will finally be used to determine if, and which volume is typeset. The key is the macro `\volume` – if it is undefined, the whole document will be typeset. If it is set to a number, the respective volume will be typeset.

6.5.1 Helper macros: checking for numbers

\ifnumber

In order to test whether `\volume`, if defined, is in fact a number, we use a command taken from scrclass.dtx, and explained there (albeit in german):

```
\providecommand\ifnumber[3]{
 \begingroup\@tempswafalse\let\scr@next\test@number
 \expandafter\scr@next#1\scr@next
 \if@tempswa\endgroup#2\else\endgroup#3\fi
}
\providecommand*{\test@number}[1]{
 \ifx \scr@next#1
  \let\scr@next\relax
 \else
  \@tempcnta=\expandafter\expandafter\expandafter\number
  \expandafter`#1\relax
  \ifnum \@tempcnta>47\relax
   \ifnum \@tempcnta<58\relax
    \@tempswatrue
   \else\@tempswafalse\fi
  \else\@tempswafalse\fi
  \if@tempswa\else\let\scr@next\gobble@till@next\fi\fi
 \scr@next}
\providecommand*{\gobble@till@next}{}
\def\gobble@till@next#1\scr@next{}
```

6.5.2 Specifying the basename of included files

\volumenname The name of the included files corresponding to the individual volumes must end with a number, but it can have any legal filename before this. By default, `Buch1.tex`, `Buch2.tex` etc. are used, but this is stored in `\volume@name` and can be changed with the user command `\volumename`:

```
\newcommand*{\volumename}[1]{%
  \def\volume@name{#1}%
}%
\volumename{volume}%
```

6.5.3 Specifying constant `\include`'s

\alwaysinclude This macro is used to specify parts that should always be included, and can only be used with the `manual` option.

```
\newcommand*{\alwaysinclude}[1]{%
  \if@allincludes
    \PackageError{volumes}{%
      need option "manual" for \string\alwaysinclude.%
    }{%
      You must use the option "manual" when you want to use
      \string\alwaysinclude, and\MessageBreak  specify
       the volumes using \string\volumeone\space etc. - see
       the package documentation.
    }
  \else
    \def\always@include{,#1}
  \fi
}
\def\always@include{}
```

6.5.4 Typesetting the right part

\onlyvolume The command that triggers conditional typesetting is `\onlyvolume`. If it is called with a number as optional argument, it defines `\volume` to expand to that number, meaning the volume to be typeset. If no optional argument is given, `\volume` is not assigned, but the check whether it is assigned is still done. This enables the user to switch between typesetting the whole document or only one volume without changing the file, by assigning `\volume` on the command line:

```
latex "\def\volume{3} \input{filename}"
```

The real work is then done by `\@@onlyvolume`:

```
\newcommand*{\onlyvolume}[1]{}
\def\onlyvolume{%
  \@ifnextchar [{\@onlyvolume}{\@@onlyvolume}
}
\def\@onlyvolume[#1]{%
  \ifnumber{#1}{%
    \def\volume{#1}
  }{%
    \PackageError{volumes}{%
```

```
      Argument to \string\onlyvolume\space must be a number%
    }{%
      The optional argument to \string\onlyvolume\space is used to
      tell volumes.sty which volume it should typeset. You should have
      specified a number there, but instead, you said: \volume%
    }%
  }%
  \@@onlyvolume
}
```

`\volume` `\@@onlyvolume`

To define `\@@onlyvolume`, we first neeed two counters. Then we check whether `\volume` is defined – if not, we `\let` all `\volume@switch@...`es to `\relax`. This is done for `numberofvolumes` volumes – there is no user interface yet to change this value. Then we tell LaTeX to execute `\all@volumes@command` at the begin of the document

```
\newcounter{volume}%
\newcounter{numberofvolumes}\setcounter{numberofvolumes}{10}%
\def\@@onlyvolume{%
  \ifx\volume\undefined%
    \PackageWarningNoLine{volumes}{Typesetting complete document.}
    \setcounter{volume}{0}%
    \@whilenum\c@volume<\c@numberofvolumes\do
    {%
      \expandafter\let%
      \csname volume@switch@\arabic{volume}\endcsname\relax%
      \stepcounter{volume}%
    }%
    \setcounter{volume}{0}%
    \AtBeginDocument{\all@volumes@command}%
```

If `\volume` is defined, we `\let` all `\volume@switch@...`es to `\volume@switch@off`. Then we schedule the execution of the respective `\volume@command@...` to the begin of the document, and make sure only the right part is included.

```
  \else
    \ifnumber{\volume}{%
      \PackageWarningNoLine{volumes}{Typesetting part \volume.}
    }{%
      \PackageError{volumes}{\string\volume\space is defined, but not a
        number}{%
        The macro \string\volume\space is used to tell volumes.sty which
        volume it should typeset. You have defined this macro, but it is
        not a number. Instead, it is: \volume%
      }%
    }%
    \setcounter{volume}{0}%
    \@whilenum\c@volume<\c@numberofvolumes\do {%
      \expandafter\let%
      \csname volume@switch@\arabic{volume}\endcsname%
      \volume@switch@off%
      \stepcounter{volume}%
    }
    \setcounter{volume}{0}%
    \AtBeginDocument{\csname volume@command@\volume\endcsname}
```

```
    \expandafter\let%
    \csname volume@switch@\volume\endcsname\volume@switch@on
    \includeonly{\volume@name\volume\always@include}%
  \fi
```

If the `manual` option was given, we define `\volumeone`, `\volumetwo`,... to be the corresponding `\volume@switch@...`.

```
  \if@allincludes\else%
    \@ifundefined{volumelist}{%
      \def\volumelist{one,two,three,four,five,six,seven,eight,nine,ten}%
    }{}%
    \@for\vol@num:=\volumelist\do
    {%
      \stepcounter{volume}%
      \expandafter\let\csname volume\vol@num\expandafter\endcsname%
        \csname volume@switch@\arabic{volume}\endcsname%
    }%
    \setcounter{volume}{0}%
  \fi
}
```

`\include` If the `manual` option was not given, we redefine `\include` so that it does the necessary switching. Otherwise, we define commands with no `@` signs as aliases for `\volume@switch@`⟨*number*⟩. If the user wants more then ten volumes, or different names, she can define `\volumelist` herself, but before the package is loaded.

TODO: Hier die Nummer als include-Argument einarbeiten!

```
\setcounter{volume}{0}%
\if@allincludes%
  \let\volume@orig@include\include%
  \def\include{%
    \stepcounter{volume}%
    \csname volume@switch@\arabic{volume}\endcsname%
    \volume@orig@include%
  }
\fi
```

⟨/volumes⟩