

The `chiddoc` Package

Niklas Beisert

Institut für Theoretische Physik
Eidgenössische Technische Hochschule Zürich
Wolfgang-Pauli-Strasse 27, 8093 Zürich, Switzerland

nbeisert@itp.phys.ethz.ch

17 January 2018, v1.6

Abstract

`chiddoc` is a $\text{\LaTeX} 2_{\epsilon}$ package that enables the direct compilation of document sections included by `\include` to individual files.

Contents

1	Introduction	1
2	Usage	2
2.1	Considerations	3
2.2	Conditional Processing	3
2.3	Flags and Forwarding	4
2.4	Manual Code	5
2.5	Command Line Processing	6
3	Information	6
3.1	Copyright	6
3.2	Files and Installation	6
3.3	Related CTAN Packages	7
3.4	Revision History	7
A	Sample	8
B	Implementation	9

1 Introduction

\LaTeX provides a mechanism to structure a large document (such as a book) into a main file and several child files (containing the chapters) using the `\include` command. This mechanism is beneficial for documents which span hundreds of pages in order to make the source file(s) more manageable. Moreover, compilation can be restricted to selected child files by means of the `\includeonly` command. The latter feature can be used to reduce the compilation time while editing (this was significantly more useful in the earlier days of \LaTeX) or to generate a smaller document which is easier to navigate. Another application of `\includeonly` is to generate documents consisting of selected parts of the complete document.

However, there are a few drawbacks of the plain `\include` mechanism:

- The child files cannot be compiled on their own, they can only be compiled via the main file. A naive editing environment (such as a text editor with an option to have the current file processed by L^AT_EX) may require one to switch to the main file before compiling; attempting to compile the child file produces errors.
- The main file must be modified (each time) to adjust the `\includeonly` command to the present needs. This easily leaves the main file in a messy state.
- The generated document will always carry the filename of the main document. This is inconvenient if several child files are to be compiled and to be kept for distribution.

The present package provides a simple interface to make child files individually compilable by L^AT_EX. Compiling a child file then has the same effect as compiling the main file with an `\includeonly` command to select the appropriate child. Moreover the generated document will carry the name of the child rather than the main file. This resolves all three above issues.

This feature is meant to make the editing of books, thesis documents and lecture notes somewhat more convenient. However, the package can also be used efficiently for composing a series of documents (such as exercise sheets) which are typically distributed individually. It then assists the author in generating the individual documents (potentially in different versions) as well as a document containing the collected series. Another application is in developing style files or other kinds of included material where compilation of the style file could redirect to a sample or test file.

2 Usage

The package `childdoc` is *not* a standard L^AT_EX 2_ε `.sty` style file! Therefore it needs to be invoked in a non-standard way.

`\childdoc` To use the package add the commands

```
\input{childdoc.def}
\childdocmain{main}
```

at the very top of your main L^AT_EX file, in particular *before* the `\documentclass` statement! Furthermore, you must add the command

```
\input{childdoc.def}
\childdocof{main}
```

at the top of every child file which is included by `\include` from within the main file (or at least for those files which you would like to compile individually). In each case, the argument *main* must be the filename of the main file. Alternatively, it suffices to start a child file with just `\input{main}`; this has the same effect, but the nesting of included files is slightly different. Note that the closed loop generated by this combination of `\input` and `\include` commands is broken by `\childdocmain`.

Please note the following restrictions:

- The argument *main* of `\childdocmain` must match the filename of the file in which it is specified. This is necessary because T_EX does not store the name of files included via `\input`.
- The filename *main* must be specified without the `.tex` extension.
- The filename *main* is case sensitive (even in case-insensitive file systems) due to internal string comparison.
- The argument *main* should be fully expanded, it cannot be a macro.

- Subdirectories and special characters should be avoided in filenames.
- The command `\childdocmain{main}` must be followed by a whitespace. It should not be followed immediately by another command or by a comment mark `'%`. This is because the \TeX parser reads the token immediately following the argument of `\childdocmain` and puts it at the beginning of every child section; however, a white-space is ignored.

2.1 Considerations

Let us mention a couple of considerations in setting up the main and child documents:

Content of Main File. It is advisable to place all output in the child files included by `\include`. Any output contained in the main file will appear in all child documents; it cannot be suppressed by the `\includeonly` directive and thus should normally be avoided. Below we shall describe a method to include some output in the main file by means of conditional processing.

Page Numbering. When only a part of the document is compiled, the appropriate numbering of pages (as well as other status parameters) is determined from the `.aux` files. The latter contain information from previous passes. However this information needs to propagate through all intermediate child documents. Therefore the page numbering in child documents may well be inconsistent until the complete document is compiled at least once.

A useful (if unconventional) way to always ensure a consistent page numbering is to restart the numbering in each child document and denote the pages by `'child.page'` where *child* represents the chapter/section number of the child file. This can be achieved by the command `\numberwithin{page}{child}` of the `amsmath` package where *child* can be `chapter` or `section` depending on the chosen structuring. Alternatively, one can modify the macro `\thepage` appropriately and reset the counter `page` at the start of each child file.

Include Files. The `childdoc` mechanism can also be used for the development of include files such as \LaTeX styles or classes, but also other material included by `\input`. This case differs from the above setup with multiple parts included by `\include` in that no `\includeonly` should be invoked. This can be achieved by starting the include file by:

```
\input{childdoc.def}
\childdocforward{main}
```

The main file is prepared as described above.

2.2 Conditional Processing

The package provides a mechanism to compile different versions of a document. To customise the versions further some conditional processing can come in handy to distinguish which version is being compiled. The package provides two macros to describe the compilation context:

`\ifchilddoc` The conditional `\ifchilddoc` distinguishes between the compilation of child documents and the main document:

```
\ifchilddoc child-code [\else main-code] \fi
```

`\childdocname` The macro `\childdocname` contains the filename (without extension) of the main or child file being processed. Note that `\jobname` will always contain the name of the main file.

Title Page. For example, conditional processing can be used to include a title or banner page in the main document when proper precautions are taken. Importantly, the code in the main file should ensure that the page counter (as well as other status parameters which are stored in the `.aux` files) takes the same value after the conditional processing. Otherwise the page numbers may take divergent values depending on which part is compiled.

For example, a title page could be declared by:

```
\ifchilddoc\else
\addtocounter{page}{-1}
code for title page
\newpage
\fi
```

A banner page for the child documents can be generated by:

```
\ifchilddoc
\addtocounter{page}{-1}
code for banner page
\newpage
\fi
```

Here one could write a message such as:

```
This is the part \childdocname{ } of \jobname.
```

2.3 Flags and Forwarding

The package allows to easily generate different versions of the main or child documents and to (permanently) store these in different files. To this end compilation flags can be defined and assigned different default values.

Defining Flags. Suppose we want to define a flag `\version` which can be set to `draft` or `final`. The document source will contain some conditional code depending on the value of `\version`. Suppose further, the flag should default to `final` for the main file and to `draft` for child files which is a natural assignment for editing the document. This is achieved by placing the following code in the preamble of the main document (below the `\childdocmain` directive):

```
\ifchilddoc
\providecommand{\version}{draft}
\else
\providecommand{\version}{final}
\fi
```

By using `\providecommand` we make sure that previous definitions are not overwritten. We can thus add further statements `\providecommand{\version}{...}` before the above code to override it.

For the main file, one might add a line (between `\childdocmain` and the above block)

```
%\ifchilddoc\else\providecommand{\version}{draft}\fi
```

which can be uncommented to produce a draft version. Likewise one can add a line to the very top of a child file (above the `\childdocof{main}` directive)

```
%\providecommand{\version}{final}
```

which can be uncommented to produce the final version of this child document.

Forwarding. Once compilation flags are defined, we can permanently set up files to produce certain versions of the documents. To this end, the package defines a command to pass on compilation to a different file:

```
\childdocforward
childdocforwardprefix

The commands \childdocforward[prefix] redirect processing to the another source file:

\childdocforward{main}{dest}
\childdocforwardprefix{main}{prefix}{dest}
```

The argument *dest* is the destination file (without extension). It should be the main file or one of the child files. In the second form, the destination file is determined by a pattern depending on the current file: To make this work, the current file must be called '*prefix suffix*' and processing is passed on to the file '*dest suffix*'. Surely, the same effect is achieved by directly specifying the argument '*dest suffix*' in the first form. However, that requires to set up a different file for each child. With the alternative form of the command all these files can have exactly the same content which simplifies setting them up and maintaining them. Finally, the optional argument *main* allows to pass on directly to the main file *main* while pretending to compile *dest*.

For example, the following file `draft.tex` compiles the main document as a draft:

```
\def\version{draft}
\input{childdoc.def}
\childdocforward{main}
```

Likewise, the following files `finalnn.tex` compile the final version of the child document `childnn.tex`:

```
\def\version{final}
\input{childdoc.def}
\childdocforwardprefix{final}{child}
```

Note that when several versions of a main file and/or of each child file are to be generated, it will be convenient to set up a Makefile or shell script to automatise the process.

2.4 Manual Code

In case one cannot be certain whether the definitions file `childdoc.def` is installed on the target T_EX distribution and one prefers not to ship it, it is conceivable to paste a few relevant commands into the sources.

To that end, drop all statements `\input{childdoc.def}` and perform the replacements as outlined below. Instead of `\childdocmain{main}` add the following code to the top of the main file:

```
\ifdefined\childdocname\endinput\fi\newif\ifchilddoc
\edef\childdocname{\scantokens\expandafter{\jobname\noexpand}}
\def\childdocmain{main}\ifx\childdocmain\childdocname\else
\childdoctrue\includeonly{\childdocname}\let\jobname\childdocmain\fi
```

Instead of `\childdocof{main}` just include the main file at the top of each child file:

```
\input{main}
```

A simple redirection `\childdocforward{dest}` is achieved by:

```
\def\jobname{dest}\input{\jobname}
```

The redirection with prefix `\childdocforwardprefix[prefix]{dest}` is accomplished by:

```
{\edef\jobname{\scantokens\expandafter{\jobname\noexpand}}
\def\redirectjob prefix#1~~~{\gdef\jobname{dest#1}}
\expandafter\redirectjob\jobname~~~\input{\jobname}
```

2.5 Command Line Processing

The effect of redirection files can also be achieved by invoking the L^AT_EX compiler with a more elaborate command line. Most conveniently this should be done as part of a shell script or a Makefile.

When using `childdoc` in the main file, the following command line effectively performs a redirection (note that depending on the shell being used, backslashes may have to be doubled: ‘\’ → ‘\\’)

```
... -jobname "target" "[flags]\def\jobname{dest}\input{main}"
```

Here *target* is the name of the output file, *main* is the name of the main file and *dest* is the name of the main or child file to be processed (all filenames without extensions). Optionally, compilation *flags* can be defined via `\def` commands.

This command line makes the T_EX engine believe it is compiling the file *target* whose content is specified as the latter parameter. The provided code in turn tweaks the definition of `\jobname` to *dest* which is later passed on to `\includeonly` by `\childdocmain` and then hands over to the main file *main*.

In fact, a similar effect can be achieved without the `childdoc` mechanism by using the command line:

```
... -jobname "target" "[flags]\includeonly{dest}\input{main}"
```

However, some of the functionality of `childdoc` is lost, e.g. child documents cannot be processed individually and the conditional `\ifchilddoc` is not defined.

3 Information

3.1 Copyright

Copyright © 2017–2018 Niklas Beisert

This work may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in <http://www.latex-project.org/lppl.txt> and version 1.3 or later is part of all distributions of L^AT_EX version 2005/12/01 or later.

This work has the LPPL maintenance status ‘maintained’.

The Current Maintainer of this work is Niklas Beisert.

This work consists of the files `README.txt`, `childdoc.ins` and `childdoc.dtx` as well as the derived files `childdoc.def`, `cdocsamp.tex` with `cdocsch1.tex`, `cdocsch2.tex`, `cdocsdrf.tex`, `cdocsfn1.tex`, `cdocsfn2.tex` and `childdoc.pdf`.

3.2 Files and Installation

The package consists of the files:

<code>README.txt</code>	readme file
<code>childdoc.ins</code>	installation file
<code>childdoc.dtx</code>	source file
<code>childdoc.def</code>	definition file
<code>cdocsamp.tex</code>	sample main file
<code>cdocsch1.tex</code>	sample include file
<code>cdocsch2.tex</code>	sample include file
<code>cdocsdrf.tex</code>	sample redirection file
<code>cdocsfn1.tex</code>	sample redirection file
<code>cdocsfn2.tex</code>	sample redirection file
<code>childdoc.pdf</code>	manual

The distribution consists of the files `README.txt`, `childdoc.ins` and `childdoc.dtx`.

- Run (pdf)L^AT_EX on `childdoc.dtx` to compile the manual `childdoc.pdf` (this file).
- Run L^AT_EX on `childdoc.ins` to create the definitions file `childdoc.def` and the sample `cdocsamp.tex` with include files `cdocsch1.tex`, `cdocsch2.tex`, `cdocsdrf.tex`, `cdocsfn1.tex`, `cdocsfn2.tex`. Copy the file `childdoc.def` to an appropriate directory of your L^AT_EX distribution, e.g. `texmf-root/tex/latex/childdoc`.

3.3 Related CTAN Packages

There are several other packages which offer a similar functionality:

- The packages `docmute`, `includex` and `standalone` provide commands to include only the document body of a child file thus allowing both files to be compiled individually.
- The packages `subdocs` and `subfiles` provide structures in which the main and child documents can be encapsulated and allowing them to be compiled individually. The inclusion mechanism is different from the conventional `\include`.
- The package `combine` is an elaborate solution to combine several documents into one.

See also the CTAN topic `subdocs` for further related packages. The present package differs from the above solutions in that a document structure constructed with the conventional `\include` mechanism just needs two extra commands at the top of every file such that all constituent files can be compiled individually.

3.4 Revision History

v1.6: 2018/01/17

- application for development of include files
- corrections to manual

v1.5: 2017/05/21

- more complete structuring introduced
- `\childdocof` introduced
- `\childdoc` renamed to `\childdocmain`
- `\childredirect` renamed to `\childdocforward` and `\childdocforwardprefix` and functionality expanded

v1.0: 2017/04/27

- manual and install package
- first version published on CTAN

v0.6: 2017/04/26

- redirection mechanism added

v0.5: 2016/04/26

- functionality in definition file

A Sample

Here we present a sample document with two chapters, a title page, a compile flag as well as three forwarding files to set the flag. It consists of six `.tex` files:

<code>cdocsamp.tex</code>	main file
<code>cdocsch1.tex</code>	include file for chapter 1
<code>cdocsch2.tex</code>	include file for chapter 2
<code>cdocsdrf.tex</code>	forwarding file for main file in draft mode
<code>cdocsfi1.tex</code>	forwarding file for final version of chapter 1
<code>cdocsfi2.tex</code>	forwarding file for final version of chapter 2

Each of the six files can be compiled directly by the \LaTeX compiler.

Main File. The main file is called `cdocsamp.tex`.

Load the `childdoc` definitions and declare the filename for the main document:

```
1 \input{childdoc.def}
2 \childdocmain{cdocsamp}
```

Optional override for `\version` flag:

```
3 %%\ifchilddoc\else\providecommand{\version}{draft}\fi
```

Define the default values for the `\version` flag (`final` for the main file and `draft` for childs):

```
4 \ifchilddoc
5 \providecommand{\version}{draft}
6 \else
7 \providecommand{\version}{final}
8 \fi
```

Load the standard document class:

```
9 \documentclass[12pt]{article}
```

Start the document body:

```
10 \begin{document}
```

Declare a title page. Print title, part of document being processed and version flag:

```
11 \addtocounter{page}{-1}
12 \begin{center}
13 {\LARGE\bfseries}childdoc example\par
14 \vspace{1cm}
15 part:
16 \ifchilddoc
17 '\childdocname' of '\jobname'\par
18 \else
19 main\par
20 \fi
21 version: \version\par
22 \end{center}
23 \newpage
```

Include the two chapters:

```
24 \include{cdocsch1}
25 \include{cdocsch2}
```


End of document body:

```
26 \end{document}
```

Chapter Include Files. The include files are called `cdocsch1.tex` and `cdocsch2.tex`.

Optional override for `\version` flag:

```
27 %%\providecommand{\version}{final}
```

Include the main document:

```
28 \input{childdoc.def}
29 \childdocof{cdocsamp}
```

Some text for chapter 1:

```
30 \section{one}
31 some text in chapter one
```

Some text for chapter 2:

```
32 \section{two}
33 more text in chapter two
```

Forwarding for a Complete Draft. The following forwarding file `cdocsdrf.tex` compiles the main document in draft mode:

```
34 \def\version{draft}
35 \input{childdoc.def}
36 \childdocforward{cdocsamp}
```

Forwarding for Final Version of the Chapters. The following forwarding files `cdocsfn1.tex` and `cdocsfn2.tex` (with identical content) compile the final versions of the child documents `cdocsch1.tex` and `cdocsch2.tex`, respectively:

```
37 \def\version{final}
38 \input{childdoc.def}
39 \childdocforwardprefix[cdocsamp]{cdocsfn}{cdocsch}
```

B Implementation

In this section we describe the definitions file `childdoc.def`.

The definitions cannot be loaded using `\usepackage` or `\RequirePackage` which has a mechanism to prevent loading a style file more than once. When loading the definitions by means of `\input` we have to prevent multiple instances manually:

```
40 %\ifdefined\childdocmain\endinput\fi
```

`\ifchilddoc` The conditional `\ifchilddoc` tells whether a child (true) or main (false) document is being compiled. The definition initialises to false:

```
41 \newif\ifchilddoc
```

`\childdocname` The macro `\childdocname` stores the name of the document to be compiled. The content of `\jobname` cannot be compared to filenames specified in the source due to different catcodes. The following code rescans `\jobname` and stores the result in `\childdocname`:

```
42 \edef\childdocname{\scantokens\expandafter{\jobname\noexpand}}
```

`\childdocmain` The macro `\childdocmain` is to be called at the top of the main file with the main filename (without extension) as argument. First, it overwrites its own definition to end processing of the present file (`\endinput`) on subsequent calls. It also overwrites the definition of `\childdocof` and `\childdocforward` to prevent further inclusions of the main document. This prevents the main file from being processed more than once. Then the current filename is compared to the main filename and in case of mismatch `\ifchilddoc` is set to true. In that case `\includeonly` is applied to the child file and `\jobname` is set to the main file (for proper handling of .aux files):

```

43 \newcommand{\childdocmain}[1]
44 {
45   \renewcommand{\childdocmain}[1]{\endinput}
46   \renewcommand{\childdocof}[1]{}
47   \renewcommand{\childdocforward}[2] [] {}
48   \begingroup
49     \def\childdoctmp{#1}
50     \ifx\childdocname\childdoctmp
51       \def\childdoctmp{\childdocfalse}
52     \else
53       \def\childdoctmp{\childdoctrue}
54     \fi
55     \expandafter
56   \endgroup
57   \childdoctmp
58   \ifchilddoc
59     \includeonly{\childdocname}
60     \def\jobname{#1}
61   \fi
62 }
```

`\childdoc` The deprecated macro `\childdoc` is a legacy version of `\childdocmain`:

```

63 \newcommand{\childdoc}{\childdocmain}
```

`\childdocof` The command `\childdocof` redirects compilation to the main file #1.

```

64 \newcommand{\childdocof}[1]
65 {
66   \input{#1}
67 }
```

`\childdocforward` The command `\childdocforward` redirects compilation to the main or a child file. `\jobname` and `\childdocname` are set to the new filename and compilation is handed over to the new file:

```

68 \newcommand{\childdocforward}[2] []
69 {
70   \def\jobname{#2}
71   \def\childdocname{#2}
72   \begingroup
73     \def\childdoctmp{#1}
74     \def\childdocempty{}
75     \ifx\childdoctmp\childdocempty
76       \def\childdoctmp{\input{#2}}
77     \else
78       \def\childdoctmp{\input{#1}}
79     \fi
80     \expandafter
```

```

81 \endgroup
82 \childdoctmp
83 \endinput
84 }

```

childdocforwardprefix The command `\childdocforwardprefix` redirects compilation to the main or a child file by means of a pattern. The prefix `#1` in the current filename is replaced by `#2` and the suffix of the current filename is kept (it is assumed that the filename does not contain the substring `'~~~'` which is used as a delimiter). Compilation is handed over to the new file by `\childdocforward`:

```

85 \newcommand{\childdocforwardprefix}[3] []
86 {
87   \begingroup
88   \def\childdocextract #2##1~~~{\def\childdoctmp{\childdocforward[#1]{#3##1}}}
89   \expandafter\childdocextract\childdocname~~~
90   \expandafter
91   \endgroup
92   \childdoctmp
93 }

```

\childdocredirect The deprecated macro `\childdocredirect` is a legacy version of `\childdocforward` and `\childdocforwardprefix`:

```

94 \newcommand{\childdocredirect}[2] []
95 {
96   \begingroup
97   \def\childdoctmp{#1}
98   \def\childdocempty{}
99   \ifx\childdoctmp\childdocempty
100     \def\childdoctmp{\childdocforward{#2}}
101   \else
102     \def\childdoctmp{\childdocforwardprefix{#1}{#2}}
103   \fi
104   \expandafter
105   \endgroup
106   \childdoctmp
107 }

```