A Document Class and a Package for Handling Multi-File Projects

Federico Garcia, Gernot Salzer

2019/09/28

Abstract

The `subfiles` package allows authors to split a document into one main file and one and more subsidiary files (subfiles) akin to the `\input` command, with the added benefit of making the subfiles compilable by themselves. This is achieved by reusing the preamble of the main file also for the subfiles.

1 Introduction

The LaTeX commands `\include` and `\input` allow the user to split the TeX source of a document into several input files. This is useful when creating documents with many chapters, but also for handling large tables, figures, and code samples, which require a considerable amount of trial-and-errors. [1]

In this process the rest of the document is of little use, and can even interfere. For example, error messages may indicate not only the wrong line number, but may point to the wrong file. Frequently, one ends up wanting to work only on the new file:

- Create a new file, and copy-paste the preamble of the main file into it.
- Work on this file, typeset it *alone* as many times as necessary.
- Finally, when the result is satisfactory, delete the preamble from the file (alongside with `\end{document}`!), and `\include` or `\input` it from the main file.

It is desirable to reduce these three steps to the interesting, middle one. Each new, subordinate file (henceforth 'subfile') should behave both as a self-sufficient LaTeX document and as part of the whole project, depending on whether it is LaTeXed individually or `\included`/`\input` from the main document. This is what the class `subfiles.cls` and the package `subfiles.sty` are intended for.

[1] Frederico had to typset numerous musical examples, whose code in MusiXTeX is long, intrincate, and barely readable.

2 Basic usage

The main file, i.e., the file with the preamble to be shared with the subfiles, has to load the package `subfiles` *at the very end of the preamble*:

```
\usepackage{subfiles}
\begin{document}
```

Subordinate files (subfiles) are loaded from the main file or from other subfiles with the command

```
\subfile{⟨subfile_name⟩}
```

The subfiles have to start with the line

```
\documentclass[⟨main_file_name⟩]{subfiles}
```

which loads the class `subfiles`. Its only 'option', which is actually mandatory, gives the name of the main file. This name follows TeX conventions: `.tex` is the default extension, the path has to be provided if the main file is in a different directory, and directories in the path have to be separated by `/` (not `\`). Thus, we have the following structure:

main file	subfile
`\documentclass[...]{...}`	`\documentclass[⟨main_file_name⟩]{subfiles}`
⟨shared preamble⟩	`\begin{document}`
`\usepackage{subfiles}`	...
`\begin{document}`	`\end{document}`
...	
`\subfile{⟨subfile_name⟩}`	
...	
`\end{document}`	

Now there are two possibilities.

- If LaTeX is run on the subfile, the line `\documentclass[..]{subfiles}` is replaced by the preamble of the main file (including its `\documentclass` command). The rest of the subfile is processed normally.
- If LaTeX is run on the main file, the subfile is loaded like with an `\input` command, except that the three lines `\documentclass[..]{subfiles}`, `\begin{document}`, and `\end{document}` are ignored.

3 Advanced usage

3.1 Hierarchy of directories

Sometimes it is desirable to put a subfile together with its images and further files into its own directory. The difficulty now is that these additional files have to be addressed by different pathes depending on whether the main files or the subfile is typeset. As of version 1.3, the `subfiles` package handles this problem by using the `import` package.

As an example, consider the following hierarchy of files:

```
main.tex
mypreamble.tex
dir1/subfile1.tex
dir1/image1.jpg
dir1/text1.tex
dir1/dir2/subfile2.tex
dir1/dir2/image2.jpg
dir1/dir2/text2.tex
```

where `main`, `subfile1`, and `subfile2` have the following contents:

main.tex

```
\documentclass{article}
\input{mypreamble}
\usepackage{graphicx}
\usepackage{subfiles}
\begin{document}
\subfile{dir1/subfile1}
\end{document}
```

subfile1.tex

```
\documentclass[../main]{subfiles}
\begin{document}
\input{text1}
\includegraphics{image1.jpg}
\subfile{dir2/subfile2}
\end{document}
```

subfile2.tex

```
\documentclass[../../main]{subfiles}
\begin{document}
\input{text2}
\includegraphics{image2.jpg}
\end{document}
```

Then each of the three files can be typeset individually in its respective directory, where LaTeX is able to locate all included text files and images.

3.2 Additional definitions for subfiles

Usually all definitions and packages required by the subfiles should go into the preamble of the main file. There are some places, though, where one might consider adding definitions for the subfiles.

Code after the end of the main document is added to the preamble of the subfiles, but is ignored when typesetting the main file. This happens because a subfile typeset by itself does not really take the preamble of the main file, but *everything outside* of `\begin{document}` and `\end{document}`. This has two consequences: *a)* the user can add some commands to be processed as part of the preamble only when the subfiles are typeset by themselves; but also *b)* the user has to be careful even *after* `\end{document}` in the main file, for any syntax error there will ruin the LaTeXing of the subfile(s).

Code in the preamble of a subfile is processed as part of the text when typesetting the main file, but as part of the preamble when typesetting the subfile. This means that the preamble of a subfile can only contain stuff that is acceptable for both, the preamble and the text area. One should also keep in mind that each subfile is `\input` within a group, so definitions made within may not work outside. A good practice when using `subfiles` (and also when not using it) is to make any definitions in the preamble of the main file, avoiding confusion and allowing the reader to find them easily.

3.3 Avoiding extra spaces

Sometimes you may want to load the contents of a subfile without white space separating it from the contents of the main file. In this respect `\subfile` behaves like `\input`. Any space or newline before and after the `\subfile` command will appear in the typeset document, as will any white space between the last character of the subfile and `\end{document}`. Therefore, to load the contents of a subfile without intervening spaces, you have either to add comment signs:

`main.tex`	`sub.tex`
`...`	`\documentclass[main.tex]{subfiles}`
`text before%`	`\begin{document}`
`\subfile{sub.tex}%`	`contents of subfile%`
`text after`	`\end{document}`

or to put everything on the same line:

```
text before\subfile{sub.tex}text after
  contents of subfile\end{document}
```

4 Dependencies

The `subfiles` package requires the `verbatim` package, whose `comment` environment is used to ignore the text area of the main file when typesetting subfiles separately. Moreover, the `import` package is needed to load subfiles and their auxiliary files from different directories. Both packages are part of the standard TeX distributions.

5 Version history

v1.1 (FG): Start of version history.

v1.2 (GS): The `subfiles` package becomes compatible with classes and packages that modify the `\document` command, like the class `revtex4`.

v1.3 (GS): Use of `import` package to handle directory hierarchies. `\ignorespaces` added to avoid spurious spaces. Incompatibility with commands removed that expect `\document` to be equal to `\@onlypreamble` after the preamble (thanks to Eric Domenjoud for analysing the problem).

6 The Implementation

6.1 The class

```
⟨*class⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{subfiles}[2019/09/28 v1.3 Federico Garcia, Gernot Salzer]
\DeclareOption*{\typeout{Preamble taken from file `\CurrentOption'}%
    \let\preamble@file\CurrentOption}
\ProcessOptions
```

We start by saving the regular LaTeX definition of `\documentclass`:

```
\let\subfiles@documentclass\documentclass
```

Now `\documentclass` is set equal to `\LoadClass` such that the class and the options of the main file will be loaded as usual.

```
\let\documentclass\LoadClass\relax
```

To handle subfiles in separate directories, we load the `import` package.

```
\RequirePackage{import}
```

The `\subimport` command requires the path and the basename of the file to be loaded in separate arguments. Therefore we have to decompose file names into these two components.

```
\def\subfiles@split#1{%
  \edef\subfiles@filename{#1}%
  \def\subfiles@dir{}%
  \def\subfiles@base{}%
  \def\subfiles@sep{}%
  \expandafter\subfiles@split@\subfiles@filename/\@nil/%
}
\def\subfiles@split@#1/{%
  \def\tmp{#1}%
  \ifx\tmp\@nnil
    \let\subfiles@next\relax
  \else
    \edef\subfiles@dir{\subfiles@dir\subfiles@base\subfiles@sep}%
    \def\subfiles@base{#1}%
    \def\subfiles@sep{/}%
    \let\subfiles@next\subfiles@split@
  \fi
  \subfiles@next
}
```

After executing e.g. `\subfiles@split{../dir1/dir2/file.tex}`, the commands `\subfiles@dir` and `\subfiles@base` expand to `../dir1/dir2/` and `file.tex`, respectively.

Now we split the name of the main file that has been provided as optional argument of the document class, and `\subimport` the main file.

```
\subfiles@split{\preamble@file}
\subimport{\subfiles@dir}{\subfiles@base}
```

The main file loads the package `subfiles` as part of the preamble, which saves the contents of `\document` and `\enddocument` as `\subfiles@document` and `\subfiles@enddocument`, respectively. Then we restore the original values of `\document`, `\enddocument`, and `\documentclass`. The backup commands are `\undefined` to save memory. That's it.

```
{\catcode`\@=11
\global\let\document\subfiles@document
\global\let\enddocument\subfiles@enddocument
\global\let\documentclass\subfiles@documentclass
\global\let\subfiles@document\undefined
\global\let\subfiles@enddocument\undefined
\global\let\subfiles@documentclass\undefined
}
⟨/class⟩
```

It may not be obvious why `@` has to be catcoded to a letter, since we are in a style file anyway. However, the `\preamble@file` occasionally contains `\usepackage` commands that make `@` a non-letter. This is why the part after loading the main preamble needs a `\catcode` command, grouping, and `\global`'s.

6.2 The package

Any option will be ignored.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{subfiles}[2019/09/28 v1.3 Federico Garcia, Gernot Salzer]
\DeclareOption*{\PackageWarning{\CurrentOption ignored}}
\ProcessOptions
\RequirePackage{verbatim}
```

To handle subfiles in separate directories, we load the `import` package.

```
\RequirePackage{import}
```

The `\import` command requires the path and the basename of files in separate arguments. Therefore we have to decompose file names into these two components.

```
\def\subfiles@split#1{%
  \edef\subfiles@filename{#1}%
  \def\subfiles@dir{}%
  \def\subfiles@base{}%
  \def\subfiles@sep{}%
  \expandafter\subfiles@split@\subfiles@filename/\@nil/%
}
\def\subfiles@split@#1/{%
  \def\tmp{#1}%
  \ifx\tmp\@nnil
    \let\subfiles@next\relax
  \else
    \edef\subfiles@dir{\subfiles@dir\subfiles@base\subfiles@sep}%
    \def\subfiles@base{#1}%
    \def\subfiles@sep{/}%
    \let\subfiles@next\subfiles@split@
  \fi
  \subfiles@next
}
```

After executing e.g. `\subfiles@split{../dir1/dir2/file.tex}`, the commands `\subfiles@dir` and `\subfiles@base` expand to `../dir1/dir2/` and `file.tex`, respectively.

`\subfile` The command `\subfile` first redefines `\documentclass` and the `document` environment to do nothing. To avoid spurious spaces we `\ignorespaces`. Moreover, we have to set `\document` to the value it usually has after the end of the preamble, since some commands check this value and raise an error saying that the command cannot be used in the preamble.

```
\newcommand\subfile[1]{%
  \begingroup
  \renewcommand\documentclass[2][subfiles]{\ignorespaces}%
  \renewenvironment{document}{\let\document\@onlypreamble\ignorespaces}{}%
```

Now we split the file name into path and base name and `\subimport` it.

```
  \subfiles@split{#1}%
  \subimport{\subfiles@dir}{\subfiles@base}%
  \unskip
  \endgroup
}
```

Note that the changes to `\documentclass` and the `document` environment happen *within a group*, so they are undone after inclusion of the subfile.

If the package is being processed as part of the main file, then we are done. However, if the initial document class was `subfiles`, then the main file is loaded as part of a subfile. In this case anything between `\begin{document}` and `\end{document}` has to be skipped, while the contents of the commands `\document` and `\enddocument` has to be retained for later use in the subfile. Therefore we save the contents of the two commands as `\subfiles@document` and `\subfiles@enddocument`, respectively. Now the `document` environment is redefined to become the `comment` environment from the `verbatim` package. Consequently, the body of the main file is ignored by LaTeX, and only the preamble is read (as well as anything that comes after `\end{document}`!).

```
\@ifclassloaded{subfiles}{%
  \let\subfiles@document\document
  \let\subfiles@enddocument\enddocument
  \let\document\comment
  \let\enddocument\endcomment
}{}
⟨/package⟩
```

By loading the `subfiles` package immediately before `\begin{document}` we ensure that `\subfiles@document` and `\subfiles@enddocument` contain all modifications that the class and the preamble of the main file may have applied to the `document` environment. This happens e.g. with the class `revtex4` and the package `pythontex`.