A Document Class and a Package for handling multi-files projects

Federico Garcia

2002/06/08

Abstract

With the `subfiles` set, the typesetting of a multi-file project consisting of one main file and one or more subsidiary files (subfiles) is more comfortable, since the user can LaTeX either the main file, which will `\input` the subfiles normally, or the subfiles by themselves, which take the preamble of the main file and become self-sufficient LaTeX documents.

1 Introduction

LaTeX commands `\include` and `\input` allow for the creation of different input files to be typeset jointly into a single output. The advantages of this are evident in the creation of large documents with many chapters, but there are also other circumstances in which the author might want to use this feature. I have used it particularly for long-coded examples, tables, figures, etc.[1], which require a considerable amount of trial-and-error.

In this process the rest of the document is of little use, and it can even disturb.[2] Frequently, one ends up wanting to work *only* on the new file, which means following three steps:

- Create a new file, and copy-paste in it the preamble of the main file;
- Work in the example, typeset it *alone* as many times as necessary;
- When the result is satisfactory, delete the preamble from the new file (and the `\end{document}`!), and `\include` or (more frequently) `\input` it from the main file.

It is desirable to reduce these three steps to the only interesting one, the middle one. This would mean that each new, subordinated file (henceforth, 'subfile') should be *both* part of a project and a self-sufficient LaTeX document, depending on whether it is LaTeXed or `\included/\input`. This is what the set of class and package under the name `subfiles` is intended for.

The main idea behind it is the redefinition of `\documentclass` and the 'environment' `document`; while these two features of LaTeX are important to keep

[1]In my case most times it has been musical examples, whose code in MusiXTeX is long, intrincate, and barely readable.

[2]For example, the error messages indicate not only a wrong line number, but even the wrong file.

unchanged, `subfiles` changes them, as far as I know, harmlessly, and I have took care of undoing the changes when finished. This is the first version of `subfiles`, and although I have tried it a few times, it is still susceptible of conflicting with other packages and/or classes.

2 Usage

2.1 Setting up

The files involved have the following basic structures:

MAIN FILE	*SUBFILE*
⟨*some preamble*⟩	`\documentclass[`⟨*main_file_name*⟩`]{subfiles}`
`\usepackage{subfiles}`	`\begin{document}`
⟨*more preamble*⟩	⟨*text, graphics, etc.*⟩
`\begin{document}`	`\end{document}`
⟨*text*⟩	
`\subfile{`⟨*subfile_name*⟩`}`	
⟨*more text*⟩	
`\end{document}`	

The `subfiles` package is to be loaded in the main file of a LaTeX project, and the `subfiles` class is to be loaded by each subordinate file. Note that the `subfiles` class handles only *one* 'option' (whose presence is actually mandatory), the name of the main file. The name should be given following TeX conventions: `.tex` is the default extension; the path has to be indicated (`/`, not `\`) if the main file is in a different directory from the subfile; spaces are gobbled (at least under Windows).

2.2 Results

This done, LaTeXing either the main or the subordinate file produces the following results:

- If the subfile is typeset by itself, it takes as preamble the one of the main file (including its `\documentclass`). The rest is typeset normally.
- If the subordinated file was `\subfile`'d, it ignores everything before and including `\begin{document}`, and the ignores `\end{document}` too. (The body of the file, nothing else, is effectively `\input`.)

The `\subfile` command is more like `\input` than `\include` in the sense that it does not start a new page. It allows nesting, but there is no exclusion mechanism analogous to `\includeonly`.

2.3 Further details and warnings

In all truth, a subfile typeset by itself does not exactly take the preamble of the main file, but *anything outside* `\begin{document}` and `\end{document}`. This has two consequences: *a*) the user can make some commands to be read only when the subfiles are typeset by themselves—which in any case are processed as part of the preamble; 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).

The preamble of the main file can `\input` (not `\include` nor `\subfile`) other files (v.g. files with definitions and shorthand-commands), and the subfiles will too. But it has to be kept in mind that each subfile is `\input` within a group, so definitions made within them might 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 to find them easily.

In principle, nesting files with `\subfile` should work and has worked in my tries, as far as every subfile loads the main file as its option to the `subfiles` class. However, who knows, the behavior can be unpredictable in weird situations. In any case, `subfiles` does *not* disable `\include` nor `\input`, which remain available for free use.

`subfiles` class and package require the `verbatim` package (whose `comment` environment is used to ignore the different parts of different files); this should not be a problem since it makes part of the standard distribution of LaTeX2$_\varepsilon$.

3 The Implementation

3.1 The class

```
⟨*class⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{subfiles}[2002/06/08 Federico Garcia]
\RequirePackage{verbatim}
\DeclareOption*{\typeout{Preamble taken from file `\CurrentOption'}%
    \let\preamble@file\CurrentOption}
\ProcessOptions
```

The first thing to do is to save the regular LaTeX definitions of `\document`, `\enddocument`, and `\documentclass`:

```
\let\old@document@subfiles\document
\let\old@enddocument@subfiles\enddocument
\let\old@documentclass@subfiles\documentclass
```

Now the `document` 'environment' is redefined and equaled to `comment`. As a consequence, the body of the main file is ignored by LaTeX, and only the preamble is read (and anything that comes after `\end{document}`!). For `\documentclass`, having been already loaded one (`subfiles`), it is redefined and equaled to `\LoadClass`. The class and options of the main file are loaded identically.

```
\let\document\comment
\let\enddocument\endcomment
\let\documentclass\LoadClass\relax
```

Now it is possible to `\input` the main file, and then restore the original values of `\document`, `\enddocument` and `\documentclass`. The backup commands are `\undefined` to save memory. That's it.

```
\input{\preamble@file}
```

Here it comes something not so obvious. In the usual situations, the `\preamble@file` contains some `\usepackage` commands, which, at the end, make `@` no longer a letter. That is why the next part needs a `\catcode` command, grouping, and `\global`'s.

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

3.2 The package

Any option will be ignored.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{subfiles}[2002/06/08 Federico Garcia]
\DeclareOption*{\PackageWarning{\CurrentOption ignored}}
\ProcessOptions
\RequirePackage{verbatim}
```

`\skip@preamble` The core of the package. It works by redefining the `document` 'environment,' thus making the `\begin` and `\end{document}` of the subfile 'transparent' to the inclusion. The redefinition of `\documentclass` is analogous, just having a required and an optional arguments which mean nothing to `\subfile`.

```
\newcommand{\skip@preamble}{%
    \let\document\relax\let\enddocument\relax%
    \newenvironment{document}{}{}%
    \renewcommand{\documentclass}[2][subfiles]{}}
```

`\subfile` Note that the new command `\subfile` calls for `\skip@preamble` *within a group*. The changes to `document` and `\documentclass` are undone after the inclusion of the subfile.

```
\newcommand\subfile[1]{\begingroup\skip@preamble\input{#1}\endgroup}
⟨/package⟩
```