Upgrading old styles

Johannes Braams

PTT Dr Neher Laboratorium
Leidschendam
The Netherlands
`j.l.braams@research.ptt.nl`

Abstract

This article deals with the things you have to look for when upgrading and old `.sty` file for LaTeX2ε, turning it into either a Document Class or a Package.

1 Introduction

This article is meant for those LaTeX users that have developped their own extensions to the LaTeX system and want to retain that functionality when they to start to use LaTeX2ε.

2 A .sty file is not a .sty file

With the old LaTeX 2.09, *all* extensions to the LaTeX system were stored in files with the extension `.sty`. Therefore it was hard to tell without looking at the contents of a file whether is was a documentstyle, an option or a package implementing entirely new functionality. As you can see in table 1 in the article on Document Classes and Packages, we have changed that and introduced a number of new extensions.

This means that you will have to ask yourself when you upgrade an existing `.sty` file what kind of functionality it implements.

Here are a few points which might help you in deciding what to do with your `.sty` file.

- Was the original `.sty` file a documentstyle? Then turn it into a Document Class.
- Was the original `.sty` file meant to be used for a certain type of document? In that case you should consider turning it into a Document Class, possibly by building on top of an existing class. An example of this is `proc.sty` which is now `proc.cls`.
- Was it just changing some aspects of the way LaTeX does things? In that case you would probably want to turn your `.sty` file into a Package.
- Was it adding completely new functionality to LaTeX? Examples of this kind of `.sty` file are files such as `seminar.sty` and `XYpic.sty`. This you most certainly will want to turn into a Package for LaTeX2ε.

3 Getting it to work again

3.1 General

When you start to work on your `.sty` file the first thing to do is see if it runs with LaTeX2ε unmodified. This assumes that you have a suitable test set that tests all functionality provided by the `.sty` file. (If you haven't, now is the time to make one!) We expect that quite a large number of the available `.sty` files will run with LaTeX2ε without any modification. Yet if it does run, please enter a note into the file that you have checked that it runs and resubmit it to the archives if it was a distributed file.

When your file .sty file uses commands that used to be part of the way LaTeXused to deal wih fonts than your file will almost certainly *not* work. You will have to look in the LaTeX companion and the file `features.tex` which is part of the LaTeX2ε distribution to see what needs to be done. Macros such as `\tenrm` or `\xipt` do not exist any longer.

Also when you use internal commands from NFSS version 1 you will have to be very carefull to check if everything still works as it was once intended.

When you need to debug your TeX code you have to know that LaTeX2ε sets `\errorcontextlines` to -1. When you need more information from TeX on where what kind of error occurs you most probably will want to change the value of `\errorcontextlines`.

3.2 Document styles

When upgrading your own document style you should start by asking yourself if it isn't more easy to reimplement it by building on an existing Document Class. This has now become very easy, you can pass any user supplied options to the underlying class with:

```
\DeclareOption*{%
  \PassOptionsToClass{%
    \CurrentOption}{class}}
```

You can make sure that certain options are used by default with:

```
\PassOptionsToClass{%
  option,option,...}{class}
```

You can disallow certain options with:

```
\DeclareOption{option}%
  {\@latexerr{%
    Option ‘option’ not supported}{}}
```

Then don't forget to call `\ProcessOptions` and `\LoadClass{class}`.

When you are upgrading a documentstyle, please note that some parameters that were in the LaTeX 2.09 documentstyles have been removed. The parameter `\footheight`, which was given a value in some documentstyles, was never even used, but it took one of the precious dimension registers, so we decided to remove it. The change in the way floats are handled rendered `\@maxsep` and `\@dblmaxsep` useless. Removing these freed another two dimension registers.

Another subject to address when you are upgrading a documentstyle, is that of options. The way options are handled in LaTeX2$_\varepsilon$ is considerably different from LaTeX 2.09. You now have to declare each option that a Document Class is to recognise. But you can now also give a Package a number of options, something that was impossible in LaTeX 2.09.

When your Document Class doesn't support floating objects it is no longer necessary to set the parameters for the handling of floats. For instance, the Document Class `letter`, which doesn't support figurs or tables no longer sets parameters such as `\floatsep`, `\textfloasep`, etc.

When your documentsytle used to perform major surgery on the inner workings of LaTeX, be very carefull when upgraing. A lot of code might have changed, so please check if your redefinitions are still valid. When you needed to redefin either `\begin{document}` or `\end{document}` to accompish your wishes, please not that this might not be necessary any longer. We have introduced 'hooks' into these commands. These hooks can be accessed with `\AtBeginDocument` or `\AtEndDocument`.

4 Using LaTeX2$_\varepsilon$ features

When you upgrade your file (or write a new one) you may wish to use one or more of the new features of LaTeX2$_\varepsilon$. If you do, *please* include a line such as the following at the top of your file:

```
\NeedsTeXFormat{LaTeX2e}[1994/02/14]
```

This would give generate a warning if someone used your file with the first test release of LaTeX2$_\varepsilon$. When you want this mechanism to work for your file as well, be sure to use one of `\ProvidesClass`, `\ProvidesPackage` or `\ProvidesFile`.

The first argument to these commands is the name of your Document Class or Package[3] and should match the name of the file it is stored in. The second argument is optional and should provide version information. Here is an example form the `a4` Package:

```
\ProvidesPackage{a4}%
  [\filedate\space
   \fileversion\space
   A4 based page layout]
```

Note that the macros `\filedate` and `\fileversion` were defined earlier in the Package file.

The session 'Advanced class(es) and packages' will discuss this subject in greater detail.

5 Compatibility with LaTeX 2.09

The subject of remaining compatible with LaTeX 2.09 is very complex. To start with you may want to remain fully compatible with the old version of LaTeX because you work together with people that can't upgrade for some reason. This means that you will either be forced not to use *any* of the new features of LaTeX2$_\varepsilon$, or you will have to find a different solution[4]. Such a solution could be the use of DOCSTRIP modules to produce two different files from your code, one for the old LaTeX installation and one to be used with LaTeX2$_\varepsilon$.

Another kind of compatibility is that which is provided in the compatibility mode of LaTeX2$_\varepsilon$. This mode was introduced to be able to run old LaTeX 2.09 documents through LaTeX2$_\varepsilon$, yeilding (almost) the same result. If this is what you need to acheive than you may be please to know that the `\if@compatibility` switch can be used to test for compatibility mode. Using this switch, you can develop a full blown LaTeX2$_\varepsilon$ Package or Document Class out of a LaTeX 2.09 style file and yet still be able to print your old documents without changing them.

6 Possible Pitfalls

Some mistakes that might be easiliy made and that can lead to unexpected results:

- You declare options in your Package using `\DeclareOption` but forget to call `\ProcessOptions`. LaTeX2$_\varepsilon$ will die horribly this way.
- The usage either of `\footheight`, `\@maxsep` or `\@dblmaxsep` outside of compatibility mode will lead to a complaint from TeX about an unknown command sequence.
- With LaTeX 2.09 the order in which options to a documentstyle were specified was *very* significant. A document might fail if the options are given in the wrong order. LaTeX2$_\varepsilon$does not necessarily process the options in the order that they were specified. When the order is relvenat to your code you have to use `\ProcessOptions*` to let LaTeX2$_\varepsilon$ evaluate them in the order specified. For the babel package for instance, the order is significant, because the language specified last will be the one the document starts off with.

[3]The first argument of `\ProvidesFile` should be a full filename, including the extension.

[4]A dirty trick for this would be to enclose new features in a test that checks whether `\documentclass` is defined

7 General advice

This section contains some general points of advice for people who are upgrading their `.sty` files.

- Carefully read the `features.tex` which is part of the LaTeX2$_\varepsilon$ disribution.
- When you distribute your work *please* distribute it in `.dtx` fomat and supply a `.ins` file to produce the stripped version(s) of your code.
- Use LaTeX commands as much as possible; this will be of benefit to you when LaTeX 3 appears.

7.1 Option naming conventions

No option names are built into the LaTeX core system, but we hope that package and class writers will follow these conventions for names of options that have general applicability. Of course, packages may also have options with names particular to the package.

language options If your package produces 'fixed strings' that produce text in the output then please consider supporting language options as defined by the babel package.

paper size options The standard classes support the options letterpaper, legalpaper, executivepaper, a4paper, a5paper, b5paper; any paper size can be modified by the option landscape. These set the `\paperheight` and `\paperwidth` parameters. When writing a class that supports different paper sizes you should make similar options, with names ending in `paper`, for those sizes that you support.

driver options Many graphics related packages need to know what dvi-driver is being used (so that the correct syntax may be used in the `\special` command). It is hoped that conventional option-names will be agreed denoting the major drivers. Currently the suggested list includes: emtex, dvips, oztex etc.

debugging options errorshow, warningshow, infoshow, debugshow, pausing.