\LaTeX\,\textsc{X} 2ε for class and package writers

Copyright © 1995–1998 The \LaTeX\,\textsc{X}3 Project
All rights reserved

12 March 1999

Contents

1 Introduction
   1.1 Writing classes and packages for \LaTeX\,\textsc{X} 2ε .................. 2
   1.2 Overview .............................................. 3
   1.3 Further information .................................... 3
   1.4 Policy on standard classes ............................... 4

2 Writing classes and packages ........................................ 4
   2.1 Old versions ........................................... 5
   2.2 Using ‘docstrip’ and ‘doc’ ................................ 5
   2.3 Is it a class or a package? ............................... 5
   2.4 Command names ......................................... 6
   2.5 Box commands and colour .................................. 6
   2.6 Defining text and math characters ....................... 7
   2.7 General style ........................................... 7

3 The structure of a class or package .................................. 10
   3.1 Identification ........................................... 10
   3.2 Using classes and packages ............................. 11
   3.3 Declaring options ....................................... 12
   3.4 A minimal class file ................................... 14
   3.5 Example: a local letter class ............................ 15
   3.6 Example: a newsletter class ............................ 15

4 Commands for class and package writers ................................ 16
   4.1 Identification ........................................... 17
   4.2 Loading files ........................................... 18
   4.3 Option declaration ...................................... 18
   4.4 Commands within option code ........................... 19
   4.5 Moving options around ................................. 20
   4.6 Delaying code .......................................... 21
1 Introduction

This document is an introduction to writing classes and packages for $\LaTeX$, with special attention given to upgrading existing $\LaTeX$ 2.09 packages to $\LaTeX$ 2e. The latter subject is also covered in an article by Johannes Braams published in TUGboat 15.3.

1.1 Writing classes and packages for $\LaTeX$ 2e

$\LaTeX$ is a document preparation system that enables the document writer to concentrate on the contents of their text, without bothering too much about the formatting of it. For example, chapters are indicated by $\backslash$chapter({title}) rather than by selecting 18pt bold.

The file that contains the information about how to turn logical structure (like $\backslash$chapter) into formatting (like ‘18pt bold ragged right’) is a document class. In addition, some features (such as colour or included graphics) are independent of the document class and these are contained in packages.

One of the largest differences between $\LaTeX$ 2.09 and $\LaTeX$ 2e is in the commands used to write packages and classes. In $\LaTeX$ 2.09, there was very little support for writing .sty files, and so writers had to resort to using low-level commands.

$\LaTeX$ 2e provides high-level commands for structuring packages. It is also much easier to build classes and packages on top of each other, for example writing a
local technical report class \texttt{cetechr} (for the Chemical Engineering department) based on \texttt{article}.

1.2 Overview

This document contains an overview of how to write classes and packages for \LaTeX. It does \textit{not} introduce all of the commands necessary to write packages: these can be found in either \textit{\LaTeX: A Document Preparation System} or \textit{The \LaTeX Companion}. But it does describe the new commands for structuring classes and packages.

Section 2.7 contains some general advice about writing classes and packages. It describes the difference between classes and packages, the command naming conventions, the use of \texttt{doc} and \texttt{docstrip}, how \TeX’s primitive file and box commands interact with \LaTeX. It also contains some hints about general \LaTeX style.

Section 3 describes the structure of classes and packages. This includes building classes and packages on top of other classes and packages, declaring options and declaring commands. It also contains example classes.

Section 4 lists the new class and package commands.

Section 6 gives detailed advice on how to upgrade existing \LaTeX 2.09 classes and packages to \LaTeX 2\texttt{e}.

1.3 Further information

For a general introduction to \LaTeX, including the new features of \LaTeX 2\texttt{e}, you should read \textit{\LaTeX: A Document Preparation System} by Leslie Lamport [3].

A more detailed description of the new features of \LaTeX, and an overview of more than 150 packages, is to be found in \textit{The \LaTeX Companion} by Michel Goossens, Frank Mittelbach and Alexander Samarin [1].

The \LaTeX system is based on \TeX, which is described in \textit{The \TeXbook} by Donald E. Knuth [2].

There are a number of documentation files which accompany every copy of \LaTeX. A copy of \texttt{\LaTeX News} will come out with each six-monthly release of \LaTeX, and is found in the files \texttt{ltnews*.tex}. The author’s guide \texttt{\LaTeX 2\texttt{e} for Authors} describes the new \LaTeX document features; it is in \texttt{usrguide.tex}. The guide \texttt{\LaTeX 2\texttt{e} Font Selection} describes the \LaTeX font selection scheme for class- and package-writers; it is in \texttt{fntguide.tex}. Configuring \LaTeX is covered by the guide \texttt{Configuration options for \LaTeX 2\texttt{e} in \texttt{cfgguide.tex}} whilst the philosophy behind our policy on modifying \LaTeX is described in \textit{Modifying \LaTeX} in \texttt{modguide.tex}.  

3
We are gradually turning the source code for \TeX into a \TeX document \textit{\TeX: the program}. This document includes an index of \TeX commands and can be typeset from \texttt{source2e.tex}.

For more information about \TeX and \LaTeX, please contact your local \TeX Users Group, or the international \TeX Users Group. Here are some useful addresses.

\TeX Users Group, P.O. Box 1239, Three Rivers, CA 93271-1239, USA
Fax: +1 209 561 4584 Email: tug@mail.tug.org
UK TUG, 1 Eyemore Close, Selly Oak, Birmingham B29 4LB, UK
Fax: +44 121 476 2159 Email: uk tug-enquiries@tex.ac.uk

1.4 Policy on standard classes

Many of the problem reports we receive concerning the standard classes are not concerned with bugs but are suggesting, more or less politely, that the design decisions embodied in them are ‘not optimal’ and asking us to modify them.

There are several reasons why we should not make such changes to these files.

- However misguided, the current behaviour is clearly what was intended when these classes were designed.
- It is not good practice to change such aspects of ‘standard classes’ because many people will be relying on them.

We have therefore decided not to even consider making such modifications, nor to spend time justifying that decision. This does not mean that we do not agree that there are many deficiencies in the design of these classes, but we have many tasks with higher priority than continually explaining why the standard classes for \TeX cannot be changed.

We would, of course, welcome the production of better classes, or of packages that can be used to enhance these classes. So your first thought when you consider such a deficiency will, we hope, be “what can I do to improve this?”

Similar considerations apply to those parts of the kernel that are implementing design decisions, many of which should be left to the class file but are not in the current system. We realise that in such cases it is much more difficult for you to rectify the problem yourself but it is also the case that making such changes in the kernel would probably be a major project for us; therefore such enhancements will have to wait for \LaTeX3.

2 Writing classes and packages

This section covers some general points concerned with writing \TeX classes and packages.
2.1 Old versions

If you are upgrading an existing \LaTeX{} 2.09 style file then we recommend freezing the 2.09 version and no longer maintaining it. Experience with the various dialects of \LaTeX{} which existed in the early 1990’s suggests that maintaining packages for different versions of \LaTeX{} is almost impossible. It will, of course, be necessary for some organisations to maintain both versions in parallel for some time but this is not essential for those packages and classes supported by individuals: they should support only the new standard \LaTeX{} 2e, not obsolete versions of \LaTeX{}.

2.2 Using ‘docstrip’ and ‘doc’

If you are going to write a large class or package for \LaTeX{} then you should consider using the doc software which comes with \LaTeX{}. \LaTeX{} classes and packages written using this can be processed in two ways: they can be run through \LaTeX{}, to produce documentation; and they can be processed with docstrip, to produce the .cls or .sty file.

The doc software can automatically generate indexes of definitions, indexes of command use, and change-log lists. It is very useful for maintaining and documenting large \TeX{} sources.

The documented sources of the \LaTeX{} kernel itself, and of the standard classes, etc, are doc documents; they are in the .dtx files in the distribution. You can, in fact, typeset the source code of the kernel as one long document, complete with index, by running \LaTeX{} on source2e.tex.

For more information on doc and docstrip, consult the files docstrip.dtx, doc.dtx, and The \LaTeX{} Companion. For examples of its use, look at the .dtx files.

2.3 Is it a class or a package?

The first thing to do when you want to put some new \LaTeX{} commands in a file is to decide whether it should be a document class or a package. The rule of thumb is:

If the commands could be used with any document class, then make them a package; and if not, then make them a class.

There are two major types of class: those like article, report or letter, which are free-standing; and those which are extensions or variations of other classes—for example, the proc document class, which is built on the article document class.

5
Thus, a company might have a local ownlet class for printing letters with their own headed note-paper. Such a class would build on top of the existing letter class but it cannot be used with any other document class, so we have ownlet.cls rather than ownlet.sty.

The graphics package, in contrast, provides commands for including images into a \LaTeX document. Since these commands can be used with any document class, we have graphics.sty rather than graphics.cls.

2.4 Command names

\LaTeX has three types of command.

There are the author commands, such as \texttt{\textbackslash section}, \texttt{\textbackslash emph} and \texttt{\textbackslash times}: most of these have short names, all in lower case.

There are also the class and package writer commands: most of these have long mixed-case names such as the following.

\begin{verbatim}
\InputIfFileExists \RequirePackage \PassOptionsToClass
\end{verbatim}

Finally, there are the internal commands used in the \LaTeX implementation, such as \texttt{\textbackslash tempcnta}, \texttt{\textbackslash gifnextchar} and \texttt{\textbackslash @eha}: most of these commands contain @ in their name, which means they cannot be used in documents, only in class and package files.

Unfortunately, for historical reasons the distinction between these commands is often blurred. For example, \texttt{\textbackslash hbox} is an internal command which should only be used in the \LaTeX kernel, whereas \texttt{\textbackslash one} is the constant -1 and could have been \texttt{\textbackslash MinusOne}.

However, this rule of thumb is still useful: if a command has @ in its name then it is not part of the supported \LaTeX language—and its behaviour may change in future releases! If a command is mixed-case, or is described in \LaTeX: A Document Preparation System, then you can rely on future releases of \LaTeX$\varepsilon$ supporting the command.

2.5 Box commands and colour

Even if you do not intend to use colour in your own documents, by taking note of the points in this section you can ensure that your class or package is compatible with the \texttt{color} package. This may benefit people using your class or package who have access to colour printers.

The simplest way to ensure ‘colour safety’ is to always use \LaTeX box commands rather than \TeX primitives, that is use \texttt{\textbackslash hbox} rather than \texttt{\textbackslash setbox}, \texttt{\textbackslash hbox} rather than \texttt{\textbackslash hbox} and \texttt{\textbackslash parbox} or the \texttt{minipage} environment rather than \texttt{\vbox}. The
\LaTeX{} box commands have new options which mean that they are now as powerful as the \TeX{} primitives.

As an example of what can go wrong, consider that in `{\ttfamily \langle text\rangle}` the font is restored just before the `}`, whereas in the similar looking construction `{\color{green} \langle text\rangle}` the colour is restored just after the final `}`. Normally this distinction does not matter at all; but consider a primitive \TeX{} box assignment such as:

\setbox0=\vbox{\color{green} \langle text\rangle}

Now the colour-restore occurs after the `}` and so is not stored in the box. Exactly what bad effects this can have depends on how colour is implemented: it can range from getting the wrong colours in the rest of the document, to causing errors in the dvi-driver used to print the document.

Also of interest is the command `\normalcolor`. This is normally just `\relax` (i.e., does nothing) but you can use it rather like `\normalfont` to set regions of the page such as captions or section headings to the ‘main document colour’.

### 2.6 Defining text and math characters

Because \LaTeX{} 2ε supports different encodings, definitions of commands for producing symbols, accents, composite glyphs, etc. must be defined using the commands provided for this purpose and described in \LaTeX{} 2ε \textit{Font Selection}. This part of the system is still under development so such tasks should be undertaken with great caution.

Also, `\DeclareRobustCommand` should be used for encoding-independent commands of this type.

Note that it is no longer possible to refer to the math font set-up outside math mode: for example, neither `\textfont 1` nor `\scriptfont 2` are guaranteed to be defined in other modes.

### 2.7 General style

The new system provides many commands designed to help you produce well-structured class and package files that are both robust and portable. This section outlines some ways to make intelligent use of these.

#### 2.7.1 Loading other files

\LaTeX{} provides these commands:
for using classes or packages inside other classes or packages. We recommend strongly that you use them, rather than the primitive \input command, for a number of reasons.

Files loaded with \input \texttt{filename} will not be listed in the \texttt{listfiles} list.

If a package is always loaded with \texttt{RequirePackage}... or \texttt{usepackage} then, even if its loading is requested several times, it will be loaded only once. By contrast, if it is loaded with \input then it can be loaded more than once; such an extra loading may waste time and memory and it may produce strange results.

If a package provides option-processing then, again, strange results are possible if the package is \input rather than loaded by means of \texttt{usepackage} or \texttt{RequirePackage}....

If the package \texttt{foo.sty} loads the package \texttt{baz.sty} by use of \input \texttt{baz.sty} then the user will get a warning:

\texttt{LaTeX Warning: You have requested package \texttt{foo}, but the package provides \texttt{baz}.}

Thus, for several reasons, using \input to load packages is not a good idea.

Unfortunately, if you are upgrading the file \texttt{myclass.sty} to a class file then you have to make sure that any old files which contain \input \texttt{myclass.sty} still work.

This was also true for the standard classes (\texttt{article}, \texttt{book} and \texttt{report}), since a lot of existing \LaTeX\ 2.09 document styles contain \input \texttt{article.sty}. The approach which we use to solve this is to provide minimal files \texttt{article.sty}, \texttt{book.sty} and \texttt{report.sty}, which simply load the appropriate class files.

For example, \texttt{article.sty} contains just the following lines:

\begin{verbatim}
\NeedsTeXFormat{LaTeX2e}
\@obsoletefile{article.cls}{article.sty}
\LoadClass{article}
\end{verbatim}

You may wish to do the same or, if you think that it is safe to do so, you may decide to just remove \texttt{myclass.sty}.
2.7.2 Make it robust

We consider it good practice, when writing packages and classes, to use \LaTeX\ commands as much as possible.

Thus, instead of using \texttt{\def...} we recommend using one of \texttt{\newcommand}, \texttt{\renewcommand} or \texttt{\providecommand}; \texttt{\CheckCommand} is also useful. Doing this makes it less likely that you will inadvertently redefine a command, giving unexpected results.

When you define an environment, use \texttt{\newenvironment} or \texttt{\renewenvironment} instead \texttt{\def\foo{...} and \def\endfoo{...}}.

If you need to set or change the value of a \texttt{\dimen} or \texttt{\skip} register, use \texttt{\setlength}.

To manipulate boxes, use \LaTeX\ commands such as \texttt{\sbox}, \texttt{\mbox} and \texttt{\parbox} rather than \texttt{\setbox}, \texttt{\hbox} and \texttt{\vbox}.

Use \texttt{\PackageError}, \texttt{\PackageWarning} or \texttt{\PackageInfo} (or the equivalent class commands) rather than \texttt{\latexerr}, \texttt{\warning} or \texttt{\wlog}.

It is still possible to declare options by defining \texttt{\do\{option\}} and calling \texttt{\@options}; but we recommend the \texttt{\DeclareOption} and \texttt{\ProcessOptions} commands instead. These are more powerful and use less memory. So rather than using:

\[
\texttt{\def\do\texttt{\{draft\}}{\overfullrule 5pt}}
\]

\[
\texttt{\@options}
\]

you should use:

\[
\texttt{\DeclareOption\{draft\}\{\setlength\{\overfullrule\}\{5pt\}}}
\]

\[
\texttt{\ProcessOptions\relax}
\]

The advantage of this kind of practice is that your code is more readable and, more important, that it is less likely to break when used with future versions of \LaTeX.\texttt{.}

2.7.3 Make it portable

It is also sensible to make your files are as portable as possible. To ensure this; they should contain only visible 7-bit text; and the filenames should contain at most eight characters (plus the three letter extension). Also, of course, it must not have the same name as a file in the standard \LaTeX\ distribution, however similar its contents may be to one of these files.
It is also useful if local classes or packages have a common prefix, for example
the University of Nowhere classes might begin with \unw. This helps to avoid
every University having its own thesis class, all called thesis.cls.
If you rely on some features of the \TeX kernel, or on a package, please specify
the release-date you need. For example, the package error commands were
introduced in the June 1994 release so, if you use them then you should put:

\NeedsTeXFormat{LaTeX2e}[1994/06/01]

2.7.4 Useful hooks

Some packages and document styles had to redefine the command \document
or \enddocument to achieve their goal. This is no longer necessary. You can
now use the 'hooks' \AtBeginDocument and \AtEndDocument (see Section 4.6).
Again, using these hooks makes it less likely that your code breaks with future
versions of \TeX. It also makes it more likely that your package can work
together with someone else's.

However, note that code in the \AtBeginDocument hook is part of the pream-
ble. Thus there are restrictions on what can be put there; in particular, no
typesetting can be done.

3 The structure of a class or package

\TeX \LaTeX classes and packages have more structure than \LaTeX 2.09 style files
did. The outline of a class or package file is:

Identification The file says that it is a \TeX \LaTeX package or class, and gives a
short description of itself.

Preliminary declarations Here the file declares some commands and can also
load other files. Usually these commands will be just those needed for the
code used in the declared options.

Options The file declares and processes its options.

More declarations This is where the file does most of its work: declaring new
variables, commands and fonts; and loading other files.

3.1 Identification

The first thing a class or package file does is identify itself. Package files do this
as follows:
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{package}[\langle date\rangle \langle other information\rangle]

For example:

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{latexsym}[1994/06/01 Standard \LaTeX\ package]

Class files do this as follows:

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{\langle class-name\rangle}[\langle date\rangle \langle other information\rangle]

For example:

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{article}[1994/06/01 Standard \LaTeX\ class]

The \langle date\rangle should be given in the form ‘YYYY/MM/DD’ and must be present if the optional argument is used (this is also true for the \NeedsTeXFormat command). Any derivation from this syntax will result in low-level \TeX\ errors—the commands expect a valid syntax to speed up the daily usage of the package or class and make no provision for the case that the developer made a mistake!

This date is checked whenever a user specifies a date in their \documentclass or \usepackage command. For example, if you wrote:

\documentclass{article}[1995/12/23]

then users at a different location would get a warning that their copy of \texttt{article} was out of date.

The description of a class is displayed when the class is used. The description of a package is put into the log file. These descriptions are also displayed by the \texttt{\listfiles} command. The phrase \texttt{Standard \LaTeX\ must not} be used in the identification banner of any file other than those in the standard \LaTeX\ distribution.

3.2 Using classes and packages

The first major difference between \LaTeX\ 2.09 style files and \LaTeX\2\epsilon\ packages and classes is that \LaTeX\2\epsilon\ supports modularity, in the sense of building files from small building-blocks rather than using large single files.

A \LaTeX\ package or class can load a package as follows:
\requirepackage[(options)]{(package)}{(date)}

For example:
\requirepackage[ifthen][1994/06/01]

This command has the same syntax as the author command \usepackage. It allows packages or classes to use features provided by other packages. For example, by loading the ifthen package, a package writer can use the ‘if...then...else...’ commands provided by that package.

A \LaTeX{} class can load one other class as follows:
\loadclass[(options)]{(class-name)}{(date)}

For example:
\loadclass[twocolumn]{article}

This command has the same syntax as the author command \documentclass. It allows classes to be based on the syntax and appearance of another class. For example, by loading the article class, a class writer only has to change the bits of article they don’t like, rather than writing a new class from scratch.

The following commands can be used in the common case that you want to simply load a class or package file with exactly those options that are being used by the current class. New feature
\loadclasswithoptions{(class-name)}{(date)}\requirepackagewithoptions{(package)}{(date)}

For example:
\loadclasswithoptions{article}\requirepackagewithoptions{graphics}[1995/12/01]

3.3 Declaring options

The other major difference between \LaTeX{} 2.09 styles and \LaTeX{} 2e packages and classes is in option handling. Packages and classes can now declare options and these can be specified by authors; for example, the twocolumn option is declared by the article class. Note that the name of an option should contain only those characters allowed in a \LaTeX{} name; in particular it must not contain any control sequences.

An option is declared as follows: New description
\newoption{twocolumn}{article}{1998/12/01}
\DeclareOption{option}{(code)}

For example, the \texttt{dvips} option (slightly simplified) to the \texttt{graphics} package is implemented as:

\begin{verbatim}
\DeclareOption{dvips}{\input{dvips.def}}
\end{verbatim}

This means that when an author writes \texttt{\usepackage[dvips]{graphics}}, the file \texttt{dvips.def} is loaded. As another example, the \texttt{a4paper} option is declared in the \texttt{article} class to set the \texttt{\paperheight} and \texttt{\paperwidth} lengths:

\begin{verbatim}
\DeclareOption{a4paper}{%
 \setlength{\paperheight}{297mm}%
 \setlength{\paperwidth}{210mm}%
}
\end{verbatim}

Sometimes a user will request an option which the class or package has not explicitly declared. By default this will produce a warning (for classes) or error (for packages); this behaviour can be altered as follows:

\begin{verbatim}
\DeclareOption*{(code)}
\end{verbatim}

For example, to make the package \texttt{fred} produce a warning rather than an error for unknown options, you could specify:

\begin{verbatim}
\DeclareOption*{%
 PackageWarning{fred}{Unknown option ‘\CurrentOption’}%
}
\end{verbatim}

Then, if an author writes \texttt{\usepackage[foo]{fred}}, they will get a warning Package \texttt{fred} Warning: Unknown option ‘\texttt{foo}’. As another example, the \texttt{fontenc} package tries to load a file \texttt{〈ENC〉enc.def} whenever the \texttt{〈ENC〉} option is used. This can be done by writing:

\begin{verbatim}
\DeclareOption*{%
 \input{\CurrentOption enc.def}%
}
\end{verbatim}

It is possible to pass options on to another package or class, using the command \texttt{\PassOptionsToPackage} or \texttt{\PassOptionsToClass} (note that this is a specialised operation that works only for option names). For example, to pass every unknown option on to the \texttt{article} class, you can use:

\begin{verbatim}
\PassOptionsToClass*{option}{article}
\end{verbatim}

\newpage

13
\DeclareOption*{%
  \PassOptionsToClass{\CurrentOption}{article}%
}

If you do this then you should make sure you load the class at some later point, otherwise the options will never be processed!

So far, we have explained only how to declare options, not how to execute them. To process the options with which the file was called, you should use:

\ProcessOptions\relax

This executes the \textit{code} for each option that was both specified and declared (see Section 4.7 for details of how this is done).

For example, if the \texttt{jane} package file contains:

\begin{verbatim}
\DeclareOption{foo}{{\texttt{Saw} foo.}}
\DeclareOption{baz}{{\texttt{Saw} baz.}}
\DeclareOption*{{\texttt{What's} \CurrentOption?}}
\ProcessOptions\relax
\end{verbatim}

and an author writes \texttt{\usepackage[foo,bar]{jane}}, then they will see the messages \texttt{Saw foo.} and \texttt{What's bar}?

### 3.4 A minimal class file

Most of the work of a class or package is in defining new commands, or changing the appearance of documents. This is done in the body of the package, using commands such as \texttt{newcommand} or \texttt{setlength}.

\LaTeX{} provides several new commands to help class and package writers; these are described in detail in Section 4.

There are three things that every class file must contain: these are a definition of \texttt{\normalsize} and values for \texttt{\textwidth} and \texttt{\textheight}. So a minimal document class file\footnote{This class is now in the standard distribution, as \texttt{minimal.cls}.} looks like this:

\begin{verbatim}
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}[1995/10/30 Standard \LaTeX{} minimal class]
\renewcommand{\normalsize}{\fontsize{10pt}{12pt}\selectfont}
\setlength{\textwidth}{6.5in}
\setlength{\textheight}{8in}
\end{verbatim}

However, this class file will not support footnotes, marginals, floats, etc., nor will it provide any of the 2-letter font commands such as \texttt{\rm}; thus most classes will contain more than this minimum!
3.5 Example: a local letter class

A company may have its own letter class, for setting letters in the company style. This section shows a simple implementation of such a class, although a real class would need more structure.

The class begins by announcing itself as neplet.cls.

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{neplet}[1995/04/01 NonExistent Press letter class]

Then this next bit passes any options on to the letter class, which is loaded with the \texttt{a4paper} option.

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{letter}}
\ProcessOptions\relax
\LoadClass{a4paper}{letter}

In order to use the company letter head, it redefines the \texttt{firstpage} page style: this is the page style that is used on the first page of letters.

\renewcommand{\ps@firstpage}{%
  \renewcommand{\@oddhead}{\textit{letterhead goes here}}%
  \renewcommand{\@oddfoot}{\textit{letterfoot goes here}}%
}

And that’s it!

3.6 Example: a newsletter class

A simple newsletter can be typeset with \texttt{LaTeX}, using a variant of the \texttt{article} class. The class begins by announcing itself as smplnews.cls.

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{smplnews}[1995/04/01 The Simple News newsletter class]

\newcommand{\headlinecolor}{\normalcolor}

It passes most specified options on to the \texttt{article} class: apart from the \texttt{onecolumn} option, which is switched off, and the \texttt{green} option, which sets the headline in green.

\DeclareOption{onecolumn}{\OptionNotUsed}
\DeclareOption{green}{\renewcommand{\headlinecolor}{\color{green}}}

\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
It then loads the class \texttt{article} with the option \texttt{twocolumn}.

\LoadClass{twocolumn}{article}

Since the newsletter is to be printed in colour, it now loads the \texttt{color} package. The class does not specify a device driver option since this should be specified by the user of the \texttt{simplenews} class.

\RequirePackage{color}

The class then redefines \texttt{maketitle} to produce the title in 72pt Helvetica bold oblique, in the appropriate colour.

\renewcommand{\maketitle}{%
\twocolumn[
\fontsize{72}{80}\fontfamily{phv}\fontseries{b}\fontshape{sl}\selectfont\headlinerow[
\@title
]
]
}%

It redefines \texttt{section} and switches off section numbering.

\renewcommand{\section}{%
\@startsection
{section}{1}{0pt}{-1.5ex plus -1ex minus -.2ex}{1.5ex plus .2ex}{\large\sfamily\slshape\headlinerow}
}%
\setcounter{secnumdepth}{0}

It also sets the three essential things.

\renewcommand{\normalsize}{\fontsize{9}{10}\selectfont}
\setlength{\textwidth}{17.5cm}
\setlength{\textheight}{25cm}

In practice, a class would need more than this: it would provide commands for issue numbers, authors of articles, page styles and so on; but this skeleton gives a start. The \texttt{simplenews} class file is not much more complex than this one.

\section{Commands for class and package writers}

This section describes briefly each of the new commands for class and package writers. To find out about other aspects of the new system, you should also read \textit{\LaTeX\ A Document Preparation System}, The \LaTeX\ Companion and \LaTeX\ \% for Authors.
4.1 Identification

The first group of commands discussed here are those used to identify your class or package file.

\NeedsTeXFormat {\<format-name\>} {\<release-date\>}

This command tells \TeX\ that this file should be processed using a format with name \textit{\<format-name\>}. You can use the optional argument \textit{\<release-date\>} to further specify the earliest release date of the format that is needed. When the release date of the format is older than the one specified a warning will be generated. The standard \textit{\<format-name\>} is \texttt{LaTeX2e}. The date, if present, must be in the form YYYY/MM/DD.

Example:

\NeedsTeXFormat{LaTeX2e}[1994/06/01]

\ProvidesClass {\<class-name\>} {\<release-info\>}
\ProvidesPackage {\<package-name\>} {\<release-info\>}

This declares that the current file contains the definitions for the document class \textit{\<class-name\>} or package \textit{\<package-name\>}. The optional \textit{\<release-info\>}, if used, must contain:

- the release date of this version of the file, in the form YYYY/MM/DD;
- optionally followed by a space and a short description, possibly including a version number.

The above syntax must be followed exactly so that this information can be used by \texttt{\LoadClass} or \texttt{\documentclass} (for classes) or \texttt{\RequirePackage} or \texttt{\usepackage} (for packages) to test that the release is not too old.

The whole of this \textit{\<release-info\>} information is displayed by \texttt{\ListFiles} and should therefore not be too long.

Example:

\ProvidesClass{article}[1994/06/01 v1.0 Standard \LaTeX\ class]
\ProvidesPackage{ifthen}[1994/06/01 v1.0 Standard \LaTeX\ package]

\ProvidesFile {\<file-name\>} {\<release-info\>}

This is similar to the two previous commands except that here the full filename, including the extension, must be given. It is used for declaring any files other than main class and package files.

Example:
Note that the phrase Standard \LaTeX{} must not be used in the identification banner of any file other than those in the standard \LaTeX{} distribution.

4.2 Loading files

This group of commands can be used to create your own document class or package by building on existing classes or packages.

\index{\texttt{\textbackslash RequirePackage}}
\begin{quote}
\begin{tabular}{ll}
\texttt{\textbackslash RequirePackage} & \{{\texttt{options-list}\}} \{{\texttt{package-name}\}} \{{\texttt{release-info}\}} \\
\texttt{\textbackslash RequirePackageWithOptions} & \{{\texttt{package-name}\}} \{{\texttt{release-info}\}}
\end{tabular}
\end{quote}

Packages and classes should use these commands to load other packages.

The use of \texttt{\textbackslash RequirePackage} is the same as the author command \texttt{\textbackslash usepackage}.

Examples:

\begin{verbatim}
\RequirePackage{ifthen}[1994/06/01]
\RequirePackageWithOptions{graphics}[1995/12/01]
\end{verbatim}

\index{\texttt{\textbackslash LoadClass}}
\begin{quote}
\begin{tabular}{ll}
\texttt{\textbackslash LoadClass} & \{{\texttt{options-list}\}} \{{\texttt{class-name}\}} \{{\texttt{release-info}\}} \\
\texttt{\textbackslash LoadClassWithOptions} & \{{\texttt{class-name}\}} \{{\texttt{release-info}\}}
\end{tabular}
\end{quote}

These commands are for use only in class files, they cannot be used in packages files; they can be used at most once within a class file.

The use of \texttt{\textbackslash LoadClass} is the same as the use of \texttt{\textbackslash documentclass} to load a class file.

Examples:

\begin{verbatim}
\LoadClass{article}[1994/06/01]
\LoadClassWithOptions{article}[1995/12/01]
\end{verbatim}

The two \texttt{WithOptions} versions simply load the class (or package) file with exactly those options that are being used by the current file (class or package).

See below, in 4.5, for further discussion of their use.

4.3 Option declaration

The following commands deal with the declaration and handling of options to document classes and packages. Every option name must be a ‘\LaTeX{} name’.

\index{\texttt{\textbackslash DeclareOption}}
\index{\texttt{\textbackslash DeclareOptionWithCategory}}
\begin{quote}
\begin{tabular}{ll}
\texttt{\textbackslash DeclareOption} & \{{\texttt{option-name}\}} \{{\texttt{category-name}\}} \\
\texttt{\textbackslash DeclareOptionWithCategory} & \{{\texttt{option-name}\}} \{{\texttt{category-name}\}}
\end{tabular}
\end{quote}

\index{\texttt{\textbackslash ProcessOptions}}
\begin{quote}
\texttt{\textbackslash ProcessOptions} \{{\texttt{options\}}\}
\end{quote}

\index{\texttt{\textbackslash ProcessOptionsWithCategory}}
\begin{quote}
\begin{tabular}{ll}
\texttt{\textbackslash ProcessOptionsWithCategory} & \{{\texttt{options\}}\} \{{\texttt{category-name}\}}
\end{tabular}
\end{quote}

\index{\texttt{\textbackslash ProcessOptions}}
\begin{quote}
\texttt{\textbackslash ProcessOptions} \{{\texttt{options\}}\}
\end{quote}

\index{\texttt{\textbackslash ProcessOptionsWithCategory}}
\begin{quote}
\begin{tabular}{ll}
\texttt{\textbackslash ProcessOptionsWithCategory} & \{{\texttt{options\}}\} \{{\texttt{category-name}\}}
\end{tabular}
\end{quote}

\index{\texttt{\textbackslash OptionsDefined}}
\begin{quote}
\texttt{\textbackslash OptionsDefined} \{{\texttt{option-name\}}\}
\end{quote}

\index{\texttt{\textbackslash OptionsDefined}}
\begin{quote}
\begin{tabular}{ll}
\texttt{\textbackslash OptionsDefinedWithCategory} & \{{\texttt{option-name\}}\} \{{\texttt{category-name\}}\}
\end{tabular}
\end{quote}

\index{\texttt{\textbackslash ProcessOptions}}
\begin{quote}
\texttt{\textbackslash ProcessOptions} \{{\texttt{options\}}\}
\end{quote}

\index{\texttt{\textbackslash ProcessOptionsWithCategory}}
\begin{quote}
\begin{tabular}{ll}
\texttt{\textbackslash ProcessOptionsWithCategory} & \{{\texttt{options\}}\} \{{\texttt{category-name\}}\}
\end{tabular}
\end{quote}

\index{\texttt{\textbackslash OptionsDefined}}
\begin{quote}
\texttt{\textbackslash OptionsDefined} \{{\texttt{option-name\}}\}
\end{quote}

\index{\texttt{\textbackslash OptionsDefined}}
\begin{quote}
\begin{tabular}{ll}
\texttt{\textbackslash OptionsDefinedWithCategory} & \{{\texttt{option-name\}}\} \{{\texttt{category-name\}}\}
\end{tabular}
\end{quote}
There are some commands designed especially for use within the \textit{code} argument of these commands (see below).

\begin{quote}
\texttt{\textbackslash DeclareOption \{\textit{option-name}\} \{\textit{code}\}}
\end{quote}

This makes \textit{option-name} a ‘declared option’ of the class or package in which it is put.

The \textit{code} argument contains the code to be executed if that option is specified for the class or package; it can contain any valid \LaTeX\ construct.

Example:

\begin{quote}
\texttt{\textbackslash DeclareOption\{twoside\}\{\texttt{\textbackslash twosidetrue}\}}
\end{quote}

\begin{quote}
\texttt{\textbackslash DeclareOption* \{\textit{code}\}}
\end{quote}

This declares the \textit{code} to be executed for every option which is specified for, but otherwise not explicitly declared by, the class or package; this code is called the ‘default option code’ and it can contain any valid \LaTeX\ construct.

If a class file contains no \texttt{\textbackslash DeclareOption*} then, by default, all specified but undeclared options for that class will be silently passed to all packages (as will the specified and declared options for that class).

If a package file contains no \texttt{\textbackslash DeclareOption*} then, by default, each specified but undeclared option for that package will produce an error.

\section*{4.4 Commands within option code}

These two commands can be used only within the \textit{code} argument of either \texttt{\textbackslash DeclareOption} or \texttt{\textbackslash DeclareOption*}. Other commands commonly used within these arguments can be found in the next few subsections.

\begin{quote}
\texttt{\textbackslash CurrentOption}
\end{quote}

This expands to the name of the current option.

\begin{quote}
\texttt{\textbackslash OptionNotUsed}
\end{quote}

This causes the current option to be added to the list of ‘unused options’.

You can now include hash marks (#) within these \textit{code} arguments without special treatment (formerly, it had been necessary to double them).

\textbf{New feature}

\textit{1995/06/01}
4.5 Moving options around

These two commands are also very useful within the \texttt{(code)} argument of \texttt{\DeclareOption} or \texttt{\DeclareOption*}:

\begin{verbatim}
\PassOptionsToPackage{\{options-list\}}{\{package-name\}}
\PassOptionsToClass{\{options-list\}}{\{class-name\}}
\end{verbatim}

The command \texttt{\PassOptionsToPackage} passes the option names in \texttt{(options-list)} to package \texttt{(package-name)}. This means that it adds the \texttt{(option-list)} to the list of options used by any future \texttt{\RequirePackage} or \texttt{\usepackage} command for package \texttt{(package-name)}.

Example:

\begin{verbatim}
\PassOptionsToPackage{foo,bar}{fred}
\RequirePackage{baz}{fred}
\end{verbatim}

is the same as:

\begin{verbatim}
\RequirePackage[foo,bar,baz]{fred}
\end{verbatim}

Similarly, \texttt{\PassOptionsToClass} may be used in a class file to pass options to another class to be loaded with \texttt{\LoadClass}.

The effects and use of these two commands should be contrasted with those of the following two (documented above, in 4.2):

\begin{verbatim}
\LoadClassWithOptions
\RequirePackageWithOptions
\end{verbatim}

The command \texttt{\RequirePackageWithOptions} is similar to \texttt{\RequirePackage}, but it always loads the required package with exactly the same option list as that being used by the current class or package, rather than with any option explicitly supplied or passed on by \texttt{\PassOptionsToPackage}.

The main purpose of \texttt{\LoadClassWithOptions} is to allow one class to simply build on another, for example:

\begin{verbatim}
\LoadClassWithOptions{article}
\end{verbatim}

This should be compared with the slightly different construction

\begin{verbatim}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
\end{verbatim}
As used above, the effects are more or less the same, but the first is a lot less to type; also the \LoadClassWithOptions method runs slightly quicker.

If, however, the class declares options of its own then the two constructions are different. Compare, for example:

```latex
\DeclareOption{landscape}{\@landscapetrue}
\ProcessOptions\relax
\LoadClassWithOptions{article}
```

with:

```latex
\DeclareOption{landscape}{\@landscapetrue}
\DeclareOption*{\CallToClass{\CurrentOption}{article}}
\ProcessOptions\relax
\LoadClass{article}
```

In the first example, the `article` class will be loaded with option `landscape` precisely when the current class is called with this option. By contrast, in the second example it will never be called with option `landscape` as in that case `article` is passed options only by the default option handler, but this handler is not used for `landscape` because that option is explicitly declared.

### 4.6 Delaying code

These first two commands are also intended primarily for use within the \texttt{\langle code\rangle} argument of \texttt{\DeclareOption} or \texttt{\DeclareOption*}.

```latex
\AtEndOfClass \{\langle code\rangle\}
\AtEndOfPackage \{\langle code\rangle\}
```

These commands declare \texttt{\langle code\rangle} that is saved away internally and then executed after processing the whole of the current class or package file.

Repeated use of these commands is permitted: the code in the arguments is stored (and later executed) in the order of their declarations.

```latex
\AtBeginDocument \{\langle code\rangle\}
\AtEndDocument \{\langle code\rangle\}
```

These commands declare \texttt{\langle code\rangle} to be saved internally and executed while \LaTeX\ is executing \texttt{\begin{document}} or \texttt{\end{document}}.

The \texttt{\langle code\rangle} specified in the argument to \texttt{\AtBeginDocument} is executed near the end of the \texttt{\begin{document}} code, after the font selection tables have been set up. It is therefore a useful place to put code which needs to be executed after
everything has been prepared for typesetting and when the normal font for the
document is the current font.

The \texttt{\textbackslash AtBeginDocument} hook should not be used for code that does any type-
setting since the typeset result would be unpredictable.

The \texttt{\langle code\rangle} specified in the argument to \texttt{\textbackslash AtEndDocument} is executed at the
beginning of the \texttt{\textbackslash end\{document\}} code, before the final page is finished and
before any leftover floating environments are processed. If some of the \texttt{\langle code\rangle} is
to be executed after these two processes, you should include a \texttt{\textbackslash clearpage} at
the appropriate point in \texttt{\langle code\rangle}.

Repeated use of these commands is permitted: the code in the arguments is
stored (and later executed) in the order of their declarations.

\begin{verbatim}
\texttt{\textbackslash AtBeginDvi\{\langle specials\rangle\}}
\end{verbatim}

These commands save in a box register things which are written to the .dvi file
at the beginning of the ‘shipout’ of the first page of the document.

This should not be used for anything that will add any typeset material to the
.dvi file.

Repeated use of this command is permitted.

\subsection{Option processing}

\begin{verbatim}
\texttt{\textbackslash ProcessOptions}
\end{verbatim}

This command executes the \texttt{\langle code\rangle} for each selected option.

We shall first describe how \texttt{\textbackslash ProcessOptions} works in a package file, and then
how this differs in a class file.

To understand in detail what \texttt{\textbackslash ProcessOptions} does in a package file, you have
to know the difference between \texttt{local} and \texttt{global} options.

\begin{itemize}
\item \textbf{Local options} are those which have been explicitly specified for this
particular package in the \texttt{\langle options\rangle} argument of any of these:
\begin{verbatim}
\texttt{\textbackslash PassOptionsToPackage\{\langle options\rangle\} \usepackage[\langle options\rangle]}
\texttt{\textbackslash RequirePackage[\langle options\rangle]}
\end{verbatim}

\item \textbf{Global options} are any other options that are specified by the author in
the \texttt{\langle options\rangle} argument of \texttt{\textbackslash documentclass[\langle options\rangle]}.
\end{itemize}

For example, suppose that a document begins:
\documentclass[german,twocolumn]{article}
\usepackage{gerhardt}

whilst package gerhardt calls package fred with:

\PassOptionsToPackage{german,dvips,a4paper}{fred}
\RequirePackage{errorshow}{fred}

then:

- fred's local options are german, dvips, a4paper and errorshow;
- fred's only global option is twocolumn.

When \ProcessOptions is called, the following happen.

- First, for each option so far declared in fred.sty by \DeclareOption, it
  looks to see if that option is either a global or a local option for fred: if
  it is then the corresponding code is executed.
  This is done in the order in which these options were declared in fred.sty.

- Then, for each remaining local option, the command \dso\-(option) is exec-
  uted if it has been defined somewhere (other than by a \DeclareOption); oth-
  erwise, the 'default option code' is executed. If no default option code
  has been declared then an error message is produced.
  This is done in the order in which these options were specified.

Throughout this process, the system ensures that the code declared for an option
is executed at most once.

Returning to the example, if fred.sty contains:

\DeclareOption{dvips}{\typeout{DVIPS}}
\DeclareOption{german}{\typeout{GERMAN}}
\DeclareOption{french}{\typeout{FRENCH}}
\DeclareOption*{\PackageWarning{fred}{Unknown \CurrentOption}}
\ProcessOptions\relax

then the result of processing this document will be:

```
DVIPS
GERMAN
Package fred Warning: Unknown 'a4paper'.
Package fred Warning: Unknown 'errorshow'.
```

23
Note the following:

- the code for the dvips option is executed before that for the german option, because that is the order in which they are declared in fred.sty;
- the code for the german option is executed only once, when the declared options are being processed;
- the a4paper and errorshow options produce the warning from the code declared by \DeclareOption* (in the order in which they were specified), whilst the twocolumn option does not: this is because twocolumn is a global option.

In a class file, \ProcessOptions works in the same way, except that: all options are local; and the default value for \DeclareOption* is OptionNotUsed rather than an error.

Note that, because \ProcessOptions has a *-form, it is wise to follow the non-star form with \relax, as in the previous examples, since this prevents unnecessary look ahead and possibly misleading error messages being issued.

\ProcessOptions* \@options

This is like \ProcessOptions but it executes the options in the order specified in the calling commands, rather than in the order of declaration in the class or package. For a package this means that the global options are processed first.

The \@options command from \LaTeX{} 2.09 has been made equivalent to this in order to ease the task of updating old document styles to \LaTeX{} 2e class files.

\ExecuteOptions {(options-list)}

For each option in the (options-list), in order, this command simply executes the command \ds@\(option\) (if this command is not defined, then that option is silently ignored).

It can be used to provide a ‘default option list’ just before \ProcessOptions. For example, suppose that in a class file you want to set up the default design to be: two-sided printing; 11pt fonts; in two columns. Then it could specify:

\ExecuteOptions{11pt,twoside,twocolumn}

4.8 Safe file commands

These commands deal with file input; they ensure that the non-existence of a requested file can be handled in a user-friendly way.
If the file exists then the code specified in \textit{true} is executed. 
If the file does not exist then the code specified in \textit{false} is executed. 
This command does not input the file.

This inputs the file \textit{file-name} if it exists and, immediately before the input, the code specified in \textit{true} is executed. 
If the file does not exist then the code specified in \textit{false} is executed. 
It is implemented using \texttt{\textbackslash{}IfFileExists}.

### 4.9 Reporting errors, etc

These commands should be used by third party classes and packages to report errors, or to provide information to authors.

\begin{itemize}
  \item \texttt{\textbackslash{}ClassError}\{\textit{class-name}\}\{\textit{error-text}\}\{\textit{help-text}\}
  \item \texttt{\textbackslash{}PackageError}\{\textit{package-name}\}\{\textit{error-text}\}\{\textit{help-text}\}
\end{itemize}

These produce an error message. The \textit{error-text} is displayed and the ? error prompt is shown. If the user types h, they will be shown the \textit{help-text}.

Within the \textit{error-text} and \textit{help-text}: \texttt{\textbackslash{}protect} can be used to stop a command from expanding; \texttt{\textbackslash{}MessageBreak} causes a line-break; and \texttt{\textbackslash{}space} prints a space.

Note that the \textit{error-text} will have a full stop added to it, so do not put one into the argument.

For example:

\begin{verbatim}
\newcommand\{\foo\}\{FOO\}
\PackageError\{ethel\}\%
  Your hovercraft is full of eels,\MessageBreak
  and \protect\foo\space is \foo
\%
\%
Oh dear! Something’s gone wrong,\MessageBreak
\space \space Try typing \space <return> \space \space to proceed, ignoring \protect\foo.
\}
\end{verbatim}

produces this display: 25
Package ethel Error: Your hovercraft is full of eels, and \texttt{\textbackslash foo} is \texttt{FOO}.

See the ethel package documentation for explanation.

If the user types \texttt{h}, this will be shown:

Oh dear! Something’s gone wrong.
Try typing \texttt{<return>} to proceed, ignoring \texttt{\textbackslash foo}.

\begin{verbatim}
\ClassWarning {\class-name} {\warning-text}
\PackageWarning {\package-name} {\warning-text}
\ClassWarningNoLine {\class-name} {\warning-text}
\PackageWarningNoLine {\package-name} {\warning-text}
\ClassInfo {\class-name} {\info-text}
\PackageInfo {\package-name} {\info-text}
\end{verbatim}

The four \texttt{Warning} commands are similar to the error commands, except that they produce only a warning on the screen, with no error prompt.

The first two, \texttt{Warning} versions, also show the line number where the warning occurred, whilst the second two, \texttt{WarningNoLine} versions, do not.

The two \texttt{Info} commands are similar except that they log the information only in the transcript file, including the line number. There are no \texttt{NoLine} versions of these two.

Within the \texttt{\{warning-text\}} and \texttt{\{info-text\}}: \texttt{\protect} can be used to stop a command from expanding; \texttt{\MessageBreak} causes a line-break; and \texttt{\space} prints a space. Also, these should not end with a full stop as one is automatically added.

\section{4.10 Defining commands}

\LaTeX{} provides some extra methods of (re)defining commands that are intended for use in class and package files.

The \texttt{*}-forms of these commands should be used to define commands that are not, in \LaTeX{} terms, long. This can be useful for error-trapping with commands whose arguments are not intended to contain whole paragraphs of text.

\begin{verbatim}
\DeclareRobustCommand {\cmd} {\num} {\default} {\definition}
\DeclareRobustCommand* {\cmd} {\num} {\default} {\definition}
\end{verbatim}

This command takes the same arguments as \texttt{\newcommand} but it declares a robust command, even if some code within the\texttt{\{definition\}} is fragile. You can use this command to define new robust commands, or to redefine existing commands.
and make them robust. A log is put into the transcript file if a command is redefined.

For example, if \texttt{\seq} is defined as follows:

\begin{verbatim}
\DeclareRobustCommand{\seq}[2][\text{default}]
  \ifmmode
    #1(\ldots#2)\else
    \PackageWarning{fred}{You can't use \protect\seq\space in text}\fi
\end{verbatim}

Then the command \texttt{\seq} can be used in moving arguments, even though \texttt{\ifmmode} cannot, for example:

\begin{verbatim}
\section{Stuff about sequences $\seq{x}$}
\end{verbatim}

Note also that there is no need to put a \texttt{\relax} before the \texttt{\ifmmode} at the beginning of the definition; this is because the protection given by this \texttt{\relax} against expansion at the wrong time will be provided internally.

\begin{verbatim}
\CheckCommand \newcommand \{\cmd\}[\{num\}] [\{default\}] [\{definition\}]
\CheckCommand* \{\cmd\}[\{num\}] [\{default\}] [\{definition\}]
\end{verbatim}

This takes the same arguments as \texttt{\newcommand} but, rather than define \texttt{\cmd}, it just checks that the current definition of \texttt{\cmd} is exactly as given by \texttt{\definition}. An error is raised if these definitions differ.

This command is useful for checking the state of the system before your package starts altering the definitions of commands. It allows you to check, in particular, that no other package has redefined the same command.

### 4.11 Moving arguments

The setting of protect whilst processing (i.e. moving) moving arguments has been reimplemented, as has the method of writing information from the \texttt{.aux} file to other files such as the \texttt{.toc} file. Details can be found in the file \texttt{ldefns.dtx}.

We hope that these changes will not affect many packages.
5 Miscellaneous commands, etc

5.1 Layout parameters

These two parameters are usually set by the class to be the size of the paper being used. This should be actual paper size, unlike \textwidth and \textheight which are the size of the main text body within the margins.

5.2 Case changing

\MakeUppercase{(text)}  \MakeLowercase{(text)}

\TeX provides two primitives \texttt{\upper} and \texttt{\lower} for changing the case of text. These are sometimes used in document classes, for example to set information in running heads in all capitals.

Unfortunately, these \TeX primitives do not change the case of characters accessed by commands like \texttt{\ae} or \texttt{\aa}. To overcome this problem, \LaTeX provides two new commands \texttt{\MakeUppercase} and \texttt{\MakeLowercase} to do this.

For example:

\begin{verbatim}
\upper{abc\ae\aa\ss\oe} \upper{ABC\ae\Ab\oe}
\lower{abc\ae\aa\ss\oe} \lower{abc\ae\Ab\oe}
\MakeUppercase{abc\ae\aa\ss\oe} \MakeUppercase{ABC\ae\Ab\oe}
\MakeLowercase{abc\ae\aa\ss\oe} \MakeLowercase{abc\ae\Ab\oe}
\end{verbatim}

The commands \texttt{\MakeUppercase} and \texttt{\MakeLowercase} themselves are robust, but they have moving arguments.

The commands use the \TeX primitives \texttt{\upper} and \texttt{\lower}, and so have a number of unexpected `features'. In particular, they change the case of everything (except characters in the names of control-sequences) in their text argument: this includes mathematics, environment names, and label names.

For example:

\begin{verbatim}
\MakeUppercase{$x+\$ in \ref{foo}}
\end{verbatim}

produces $X + Y$ and the warning:

LaTeX Warning: Reference `FOO' on page ... undefined on ...
In the long run, we would like to use all-caps fonts rather than any command like \MakeUppercase but this is not possible at the moment because such fonts do not exist.

In order that upper/lower-casing will work reasonably well, and in order to provide any correct hyphenation, \LaTeX \texttt{2e} must use, throughout a document, the same fixed table for changing case. The table used is designed for the font encoding \TeXT1; this works well with the standard \TeX fonts for all Latin alphabets but will cause problems when using other alphabets.

5.3 The ‘openany’ option in the ‘book’ class

The \texttt{openany} option allows chapter and similar openings to occur on left-hand pages. Previously this option affected only \texttt{chapter} and \texttt{backmatter}. It now also affects \texttt{part}, \texttt{frontmatter} and \texttt{mainmatter}.

5.4 Better user-defined math display environments

\begin{Verbatim}
\newenvironment{textheq}
\begin{equation}\begin{minipage}{0.9\linewidth}{}
\end{minipage}\end{equation}
\end{Verbatim}

However, if you have tried this then you will probably have noticed that it does not work perfectly when used in the middle of a paragraph because an inter-word space appears at the beginning of the first line after the environment.

There is now an extra command (with a very long name) available that you can use to avoid this problem; it should be inserted as shown here:

\begin{Verbatim}
\newenvironment{textheq}
\begin{equation}\begin{minipage}{0.9\linewidth}{}
\end{minipage}\end{equation}
\end{Verbatim}
\begin{Verbatim}
\\texttt{\textbackslash ignorespacesafterend}
\end{Verbatim}
5.5 Normalising spacing

This command should be used to restore the normal settings of the parameters that affect spacing between words, sentences, etc.

An important use of this feature is to correct a problem, reported by Donald Arseneau, that punctuation in page headers has always (in all known \TeX{} formats) been potentially wrong whenever a page break happens while a local setting of the space codes is in effect. These space codes are changed by, for example, the command \texttt{\textbackslash frenchspacing} and the \texttt{verbatim} environment.

It is normally given the correct definition automatically in \texttt{\textbackslash begin\{document\}} and so need not be explicitly set; however, if it is explicitly made nonempty in a class file then automatic default setting will be over-ridden.

6 Upgrading \LaTeX{} 2.09 classes and packages

This section describes the changes you may need to make when you upgrade an existing \LaTeX{} style to a package or class but we shall start in optimistic mode.

Many existing style files will run with \LaTeX{} 2\epsilon{} without any modification to the file itself. When everything is running OK, please put a note in the newly created package or class file to record that it runs with the new standard \LaTeX{}; then distribute it to your users.

6.1 Try it first!

The first thing you should do is to test your style in `compatibility mode’. The only change you need to make in order to do this is, possibly, to change the extension of the file to .cls: you should make this change only if your file was used as a main document style. Now, without any other modifications, run \LaTeX{} 2\epsilon{} on a document that uses your file. This assumes that you have a suitable collection of files that tests all the functionality provided by your style file. (If you haven’t, now is the time to make one!)

You now need to change the test document files so that they are \LaTeX{} 2\epsilon{} documents: see \LaTeX{} 2\epsilon{} \textit{for Authors} for details of how to do this and then try them again. You have now tried the test documents in both \LaTeX{} 2\epsilon{} native mode and \LaTeX{} 2.09 compatibility mode.

6.2 Troubleshooting

If your file does not work with \LaTeX{} 2\epsilon{}, there are two likely reasons.
• \LaTeX{} now has a robust, well-defined designer’s interface for selecting fonts, which is very different from the \LaTeX{} 2.09 internals.

• Your style file may have used some \LaTeX{} 2.09 internal commands which have changed, or which have been removed.

When you are debugging your file, you will probably need more information than is normally displayed by \LaTeX{} 2e. This is achieved by resetting the counter errorcontextlines from its default value of −1 to a much higher value, e.g. 999.

6.3 Accommodating compatibility mode

Sometimes an existing collection of \LaTeX{} 2.09 documents makes it inconvenient or impossible to abandon the old commands entirely. If this is the case, then it is possible to accommodate both conventions by making special provision for documents processed in compatibility mode.

\begin{verbatim}
\if@compatibility
\end{verbatim}

This switch is set when a document begins with \verb|\documentstyle| rather than \verb|\documentclass|. Appropriate code can be supplied for either condition, as follows:

\begin{verbatim}
\if@compatibility
    \{code emulating \LaTeX{} 2.09 behavior\}
\else
    \{code suitable for \LaTeX{}2e\}
\fi
\end{verbatim}

6.4 Font commands

Some font and size commands are now defined by the document class rather than by the \LaTeX{} kernel. If you are upgrading a \LaTeX{} 2.09 document style to a class that does not load one of the standard classes, then you will probably need to add definitions for these commands.

\begin{verbatim}
\rm \sf \tt \bf \it \sl \sc
\end{verbatim}

None of these short-form font selection commands are defined in the \LaTeX{} 2e kernel. They are defined by all the standard class files.

If you want to define them in your class file, there are several reasonable ways to do this.

One possible definition is:
\newcommand{\rm}{\rmfamily}
\ ...
\newcommand{\sc}{\scshape}

This would make the font commands orthogonal; for example {\it \textsf text} would produce bold italic, thus: \textit{text}. It will also make them produce an error if used in math mode.

Another possible definition is:
\newcommand{\rm}{\rmfamily}{\mathrm}
\ ...
\newcommand{\sc}{\scshape}{\mathsc}

This will make \textsf{rm} act like \textsf{rmfamily} in text mode (see above) and it will make \textsf{rm} select the \textsf{mathrm} math alphabet in math mode.

Thus $\{\textsf{rm math} = X + 1\}$ will produce \textit{math} = $X + 1$.

If you do not want font selection to be orthogonal then you can follow the standard classes and define:
\newcommand{\rm}{\normalfont \rmfamily}{\mathrm}
\ ...
\newcommand{\sc}{\normalfont \scshape}{\mathsc}

This means, for example, that {\bf \it text} will produce medium weight (rather than bold) italic, thus: \textit{text}.

\textbf{\textnormal{\texttt{\textit normalsize}}}
\textbf{\textnormal{\texttt{\textit @normalsize}}}

The commandawks{\texttt{\textnormal{\texttt{\textit normalsize}}} is retained for compatibility with \textsf{\texttt{\texttt{\textit LATEX}}} 2.09 packages which may have used its value; but redefining it in a class file will have no effect since it is always reset to have the same meaning as \texttt{\textnormal{\texttt{\textit normalsize}}}.

This means that classes \texttt{\textnormal{\texttt{\textit must}}} now redefine \texttt{\textnormal{\texttt{\textit normalsize}}} rather than redefining \texttt{\textnormal{\texttt{\textit @normalsize}}}; for example (a rather incomplete one):
\newcommand{\normalsize}{\fontsize{10}{12}\selectfont}

Note that \texttt{\textnormal{\texttt{\textit normalsize}}} is defined by the \texttt{\textnormal{\texttt{\textit LATEX}}} kernel to be an error message.

\textbf{\texttt{\textnormal{\texttt{\textit tiny \footnotesize \small \large \Large \LARGE \huge \Huge}}}}

None of these other `standard' size-changing commands are defined in the kernel; each needs to be defined in a class file if you need it. They are all defined by the standard classes.
This means you should use \renewcommand for \normalsize and \newcommand for the other size-changing commands.

### 6.5 Obsolete commands

Some packages will not work with \LaTeX\ 2ε, normally because they relied on an internal \LaTeX\ command which was never supported and has now changed, or been removed.

In many cases there will now be a robust, high-level means of achieving what previously required low-level commands. Please consult Section 4 to see if you can now use the \LaTeX\ 2ε class and package writers commands.

Also, of course, if your package or class redefined any of the kernel commands (i.e. those defined in the files \latex.tex, sltex.tex, lfonts.tex, sfonts.tex) then you will need to check it very carefully against the new kernel in case the implementation has changed or the command no longer exists in the \LaTeX\ 2ε kernel.

Too many of the internal commands of \LaTeX\ 2.09 have been re-implemented or removed to be able to list them all here. You must check any that you have used or changed.

We shall, however, list some of the more important commands which are no longer supported.

\begin{verbatim}
\tenrm \elvrm \twlrm ...
\tenbf \elvbff \twlbff ...
\tensf \elvsf \twlsf ...
\end{verbatim}

The (approximately) seventy internal commands of this form are no longer defined. If your class or package uses them then please replace them with new font commands described in \LaTeX\ 2ε \textit{Font Selection}.

For example, the command \texttt{\twlsf} should be replaced by:

```
\fontsize{12}{14}\normalfont\sfamily\selectfont
```

Another possibility is to use the \texttt{rawfonts} package, described in \LaTeX\ 2ε for \textit{Authors}.

Also, remember that many of the fonts preloaded by \LaTeX\ 2.09 are no longer preloaded.
\vpt \\vipt \\vipt ...

These were the internal size-selecting commands in \TeX\ 2.09. (They can still be used in \TeX\ 2.09 compatibility mode.) Please use the command $\texttt{\textbackslash fontsize}$ instead: see \TeX\ \& \LaTeX\ \textit{Font Selection} for details.

For example, \texttt{$\textbackslash fontsize\{6\}\texttt{\textbackslash normalfont\textbackslash selectfont}$}

\pmb, \pbf, \pounds, \pLaTeX ...

\TeX\ 2.09 used several commands beginning with \texttt{\textbackslash p} in order to provide ‘protected’ commands. For example, \texttt{\LaTeX} was defined to be $\texttt{\protect\LaTeX}$, and \texttt{\pLaTeX} was defined to produce the \TeX\ logo. This made \LaTeX\ robust, even though \texttt{\pLaTeX} was not.

These commands have now been reimplemented using \texttt{\DeclareRobustCommand} (described in Section 4.10). If your package redefined one of the \texttt{\p} commands then you must remove the redefinition and use $\texttt{\DeclareRobustCommand}$ to redefine the non-\texttt{\p} command.

\footheight \\
\@maxsep \\
\@dblmaxsep

These parameters are not used by \TeX\ \& \LaTeX\ so they have been removed, except in \TeX\ 2.09 compatibility mode. Classes should no longer set them.

References


\LaTeX\ 2e Summary sheet: updating old styles

Section references below are to \LaTeX\ 2e for Class and Package Writers.

1. Should it become a class or a package? See Section 2.3 for how to answer this question.

2. If it uses another style file, then you will need to obtain an updated version of this other file. Look at Section 2.7.1 for information on how to load other class and package files.

3. Try it: see Section 6.1.

4. It worked? Excellent, but there are probably still some things you should change in order to make your file into a well-structured \LaTeX\ 2e file that is both robust and portable. So you should now read Section 2, especially 2.7. You will also find some useful examples in Section 3.
   If your file sets up new fonts, font-changing commands or symbols, you should also read \LaTeX\ 2e Font Selection.

5. It did not work? There are three possibilities here:
   - error messages are produced whilst reading your file;
   - error messages are produced whilst processing test documents;
   - there are no errors but the output is not as it should be.

Don’t forget to check carefully for this last possibility.
If you have got to this stage then you will need to read Section 6 to find the solutions that will make your file work.