More essential LaTeX 2ε

Richard Kaye*

12th August 1998

Contents

1 Introduction

This document is a supplement to *Essential LaTeX 2ε* and to *Essential Mathematical LaTeX*. The main points covered are:

- an overview of how the components of the TeX system work together;
- the rationale behind LaTeX 2ε, and in what way it differs from the previous version, LaTeX 2.09;
- more information on the use of fonts in LaTeX 2ε;
- some ideas and pointers more advanced use of LaTeX 2ε, such as the use of user-defined commands.

As always, you are encouraged to modify this file to experiment with the ideas here or as your own reference to suit your own use later.

*School of Mathematics and Statistics, The University of Birmingham, Birmingham B15 2TT, U.K.

2 The components of TEX

Much of this is just background information relating to how TEX and its various components fit together, and the need for a new version of LATEX.

TEX is a typesetting program which takes your document (which is a standard text file) and typesets it according to certain built-in rules and other rules which are automatically read in from other files. Basically, TEX is a glorified typewriter. When it sees the letter 'a', it just goes and typesets a letter a in the current font.

To do this it reads information about all the fonts you will use from special files (called 'TEX font metric' files, having suffix `.tfm`) which tell TEX how large each character is and how much space is normally left between characters. Using this information, TEX decides what characters should be put where on the pages, and stores this information in the output (`.dvi`) file.

To further control how the output looks, TEX has various other primitive commands which can be used. You will almost never have to work with these (although the details can be found in Knuth's *The TEXbook*). Fortunately TEX also has a mechanism to define other commands from old ones (using a device called *macro expansion*), and you normally use TEX by first loading a package of new commands (or macros) and working with those commands instead.

Packages for TEX are available for typesetting almost any language, and there are many other packages too, even one to typeset music, called MusicTEX.

The popularity of TEX is due in part to the popularity of these packages. The original macro package for TEX is (oddly) called *plain TEX*, even though it is much more powerful than the primitive set of commands that are used. Then, the American Mathematical Society developed another one, AMS-TEX, and Lesie Lamport developed LATEX. Arising from experience of these two, AMS-LATEX was developed. No two of these systems are compatible with each other, although it is almost true that plain TEX is a subset of the other three. In fact, it is even worse than this, and many dialects of LATEX are in use, some with and some without the 'new font selection scheme', and even ones with this scheme are set up differently. LATEX itself has the ability to load new packages (so-called style or `.sty` files), and hundreds are available for all sorts of applications. Unfortunately standardization in these packages seems to be rare, and the situation is rather a mess. The latest version of LATEX, LATEX 2_ε, was developed to sort out the muddle with the different dialects of LATEX and its various packages, and is discussed more below.

There are two further components of the TEX system that are worth mentioning. The first (and most important to the user) is the family of so-called *drivers* which convert the `.dvi` file to a printed output, or at least to

something you can read. By necessity, these drivers are machine dependent, and you will need a different driver for each type of printer or screen. On the sun network you will use `dvips`, which converts the `.dvi` file to postscript and prints it normally on a laser printer, and `xdvi` which views a `.dvi` file on an X-terminal. Drivers are also available for most other computers and output devices.

Roughly what a driver does is read the `.dvi` file and also information about the shape of characters in a font (usually from `.pk` files—this part is done automatically) and produces a graphic image which can be printed or viewed.

The other component of the TEX system is called *Metafont*, and is used to create the font files from specially written programs. So in principle, you can design your own characters in your mathematical papers, but of course in practice this is time-consuming and difficult to get to look good. But sometimes, if `dvips` can't find a font you will find it calling Metafont automatically to generate it.

TEX was designed for maximum portability between computers and different printers. (That means it is an ideal format to send academic papers to a colleague by email. But it is often wise to check that your colleague has similar fonts and macro packages.) A disadvantage is that you cannot always directly use special features of your computer/printer when working with TEX and its normal fonts, for example drawing long diagonal lines and incorporating images is difficult to do with TEX's normal `.tfm`, `.pk` fonts. There are several ways to work round the problem if you can be sure you (and your colleagues) only work with one driver (`dvips` for example) or can print out postscript files directly.

Finally, the TEX software is extensive, covering typesetting in almost every language and in almost any application, and just about all of this software is public domain and/or free. If you need some extra package it can usually be obtained by ftp from `ftp.tex.ac.uk` for example.

3 What's new in LaTeX 2_ε

LaTeX is not just one package, but also a rather general way of choosing packages, loading new commands and further packages, and also (if you should want or need to) designing document styles. Packages and styles are read in by LaTeX as `.sty` files. You can write these yourself, or you can use a `.sty` file written by someone else (there are literally hundreds available). Unfortunately this proliferation of styles and packages caused more confusion, since many were given the same name (or new versions of a `.sty` file were written with the same name and different functionality). LaTeX 2_ε was devised to provide a standard base to all users. It was released in the summer of 1994 and is widely available, but unfortunately is not

loaded onto every system yet!

The *new font selection scheme* (nfss) built into LaTeX 2$_\varepsilon$ makes using special symbol fonts for mathematics, and using postscript fonts (rather that Knuth's 'Computer Modern' fonts) much easier—provided of course you use a driver like `dvips`. If you want to use the additional features developed for typesetting mathematics by the AMS, or if you want to use postscript fonts, you are strongly advised to use LaTeX 2$_\varepsilon$.

LaTeX 2$_\varepsilon$ has a 'compatibility mode', and if it sees a line like

```
\documentstyle[11pt,a4,amssymb]{article}
```

it recognises the document as a LaTeX 2.09 document and tries to emulate LaTeX 2.09. (This is usually but not always successful—some documents will be typeset differently, especially if you try to use an incompatible package.) On the other hand a true LaTeX 2$_\varepsilon$ document will start with

```
\documentclass[11pt,a4paper]{article}
\usepackage{amssymb}
```

say, and the document can then use the new features of LaTeX 2$_\varepsilon$ (and typesetting should be quicker as well).

Key to LaTeX 2$_\varepsilon$ is the idea of a 'document-class'—what sort of document it is (an article, a book, or whatever)—and the distinction between this and a 'package'. In LaTeX 2.09 the distinction was blurred. There are many other enhancements made 'behind the scenes' that make writing classes, packages and styles easier and enable them to work more reliably and predictably. This probably is the most important improvement, and is part of the longer-term design of the next release of LaTeX, LaTeX 3, which will not have any obvious new features to the user, but will have a host of new features to the writer of LaTeX packages.

4 About using fonts

The new font selection scheme (nfss) works in a significantly different way to the old font mechanism in LaTeX 2.09 and plain TeX, although in most cases the difference should be transparent to the user. Occasionally, a more detailed knowledge than given in 'Essential LaTeX 2$_\varepsilon$' is required.

Every font has: (a) a family name; (b) a series; and (c) a shape. Each of these three attributes can be changed independently. The three main families are (1) roman, as here, (2) sans-serif, like this, and (3) typewriter, `like this`. The commands for selecting these families are respectively `\rmfamily`, `\sffamily` and `\ttfamily`. Each font in each family has *series*, indicating whether it is bold or medium-weight. The commands for selecting these series are `\bfseries`, `\mdseries`. Finally, the *shape* of the font is whether the letters are upright, italic, slanted, or caps-and-small-caps.

Here's a summary table.

1. Roman family

	\upshape	\itshape	\slshape	\scshape
\mdseries	Upright	*Italic*	*Slanted*	Small Caps
\bfseries	**Upright**	***Italic***	***Slanted***	**Small Caps**

2. Sans serif family

	\upshape	\itshape	\slshape	\scshape
\mdseries	Upright	*Italic*	*Slanted*	Small Caps
\bfseries	**Upright**	**Italic**	**Slanted**	**Small Caps**

3. Typewriter family

	\upshape	\itshape	\slshape	\scshape
\mdseries	Upright	*Italic*	*Slanted*	Small Caps
\bfseries	Upright	*Italic*	Slanted	Small Caps

Note that is the current shape is italic, `\bfseries` would select bold italic, and if text is currently upright, `\bfseries` would select upright bold. Note also that some family/series/shape combinations are not always available. This shouldn't create problems as suitable substitutions are made (and a warning message issued), but if you really want these fonts, it is reasonably straightforward to get Metafont to make them for you.

These commands are typically used in contexts delimited with braces in the usual way. If you only want a *small* amount of text to be changed, you can use the commands:

- `\textrm{Roman family}` Roman family;
- `\textsf{Sans serif family}` Sans serif family;
- `\texttt{Typewriter family}` Typewriter family;
- `\textmd{Medium series}` Medium series;
- `\textbf{Bold series}` **Bold series**;
- `\textup{Upright shape}` Upright shape;
- `\textit{Italic shape}` *Italic shape*;
- `\textsl{Slanted shape}` *Slanted shape*;
- `\textsc{Small Caps shape}` Small Caps shape;
- `\emph{Emphasized}` *Emphasized*;

the last of these chooses its shape ('up' or 'it') depending on context. All of these commands will insert the appropriate extra space if a slanted shape butts into an upright one.

To change fonts in maths mode you use the analogous commands

- `\mathrm{Roman}`,
- `\mathnormal{Normal}`,
- `\mathcal{CALIGRAPHIC}`,
- `\mathbf{Bold}`,
- `\mathsf{Sans serif}`,
- `\mathtt{Typewriter}`,
- `\mathit{Italic}`.

`\mathbb` gives 'blackboard bold' when the appropriate AMS symbols package loaded. The AMS packages also provide cyrillic, gothic and script letters.

Here's a test:

Roman	$Normal$	$\mathcal{CALIGRAPHIC}$
$\mathbf{Bold}$	$\mathsf{Sans\ serif}$	$\mathtt{Typewriter}$
Italic	$\mathbb{N}, \mathbb{Z}, \mathbb{R}, \mathbb{C}$	$\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}$

`\mathnormal` is the usual font for algebraic equations, etc., and the spacing between letters is designed for this. If you want a word typed in italic, as in T_{max} use `\mathit`. (Compare this with `T_{max}`, T_{max} which looks pretty horrible.

There's a subtle but important point about changing fonts, often missed by beginners (and also people who are not beginners) at LaTeX, which is the so called *italic correction*. Typesetters should add a little extra space after a

slanted letter

if the next letter is upright. If this is not done the text looks too cramped, as in

slanted letter.

The commands `\textit`, `\textsl`, and `\emph` do this automatically, but you have to add the correction by hand with the command `\/` when you use any of the other font-changing commands, including `\it` and `\sl`.

One final word on fonts: the commands `\textrm`, `\rm` and so on are defined by the *document class* in terms of the more basic commands discussed here, so they might not work in the same way if you change from the

`article` to a publisher's special document class. So you should probably use them rather than the basic commands, since that way major stylistic changes can be made with a single change.

5 User-defined commands in LaTeX

LaTeX always included the factility to add new commands. For example, you could say

```
\newcommand{\wake}{baba\-badalgharagh\-takam\-min\-%
arron\-nkonn\-bronn\-tonn\-er\-ronn\-tuonn\-thunn\-%
tro\-varr\-houn\-awnskawn\-too\-hoo\-hoor\-denen\-thur\-nuk}
```

in a document that makes a lot of use of the hundred-letter word on the first page of Joyce's *Finnegans Wake*, and then you can save yourself a lot of typing (as well as remembering where all the suitable places to hyphenate it are) by just saying `\wake` every time you need this word as in

```
The fall (\wake!) of a once wallstrait oldparr\ldots
```

'The fall (bababadalgharaghtakamminarronnkonnbronntonnerronntuonnthunntrovarrhounawnskawntoohoohoordenenthurnuk!) of a once wallstrait oldparr…'

Note the use of the comment character (%) to ensure that LaTeX doesn't see the end-of-line symbol and therefore doesn't break up this beautiful word into three. (Unfortunately, LaTeX does have to hyphenate it somewhere though.) Note also that new commands defined this way still suffer from the problem that any spaces following them will be ignored.

LaTeX commands may take arguments too. For example, in

```
\newcommand{\lis}[2]{#2_{1},\ldots,#2_{#1}}
```

we define a command `\lis` of two arguments (which will replace the `#1` and `#2` in the definition. Thus `consider \( \mathbf{x} = (\lis{n}{x}) \)` becomes 'consider $\mathbf{x} = (x_1, \ldots, x_n)$.'

LaTeX 2$_\varepsilon$ improves the `\newcommand` command of LaTeX 2.09 by allowing you to define commands with an optional argument as well.

For example, in

```
\newcommand{\seq}[2][n]{#2_{1},\ldots,#2_{#1}}
```

we define `\seq` with two arguments, the first being optional (default value 'n'). This means

```
For some \(i \in \{ \seq{t} \} \) we have
\( f(i)=\seq[m]{\alpha} \)
```

gives 'For some $i \in \{t_1, \ldots, t_n\}$ we have $f(i) = \alpha_1, \ldots, \alpha_m$'.

LaTeX also allows you to define environments, usually variations of existing ones. So the definition

```
\newenvironment{qsi}[1]%
{#1 wrote,\begin{quote}\begin{sloppypar}\it}%
{\end{sloppypar}\end{quote}}
```

Makes the following LaTeX source

```
\begin{qsi}{Joyce}
The fall (\wake!) of a once wallstrait oldparr\ldots
\end{qsi}
```

yield: Joyce wrote,

> *The fall (babababadalgharaghtakamminarronnkonnbronntonnerronntuonnthunntrovarrhounawnskawntoohoohoordenenthurnuk!) of a once wallstrait oldparr...*

6 Packages

The best feature of LaTeX 2$_\varepsilon$ is its ability to load and use packages from various sources in a smooth, uniform way. Several are bundled with the new LaTeX, including ones to print graphics, and one to vary 'theorem' environments. The AMS packages now work well with LaTeX and are highly recommended for tricky mathematical typesetting. For example, if you require the AMS fonts, you are strongly advised to use LaTeX 2$_\varepsilon$ with the appropriate package.

For further information on LaTeX 2$_\varepsilon$ and its packages, see 'LaTeX 2$_\varepsilon$ for authors' and 'AMS-LaTeX Version 1.2 User's guide' which are on fourier and copies can be made available.