LaTeX – the very idea

Peter Smith

What is LaTeX? How does it differ from a word processing programme? Isn't it ante-diluvian? Isn't it really only for mathematicians and scientists? Isn't it quite horrendously difficult to learn?

These are the questions briefly answered here. I'll say almost nothing about the nuts-and-bolts of LaTeX for there is a mass of other documentation that tells you all you could possibly want to know about that. What I try to do here is stand back from the details, and give a sense of what you might call 'the LaTeX philosophy', i.e. its guiding conception. For the crucial thing is to grasp what *kind* of programme LaTeX is. Once you've got your head round that, you'll have a much better idea if LaTeX is for you. If you then decide to try it out, you can fill in the details you need to master on a need-to-know basis.

1 Not just another word-processor

LaTeX is described by its creator Leslie Lamport as 'a document preparation system'. And it is very important indeed to understand straight away that this *isn't* a fancy way of saying that LaTeX is just another word-processor. For the LaTeX approach is very different from that of a standard word-processing program. But different in what ways?

This is most easily explained by example. So dive in and read through the following sample LaTeX *source document* (of course, you won't get the significance of every detail, but you should easily get a rough grasp of what's going on):

```
\documentclass{article}
\author{Peter Smith}
\title{A Simple Document using \LaTeX}

\begin{document}
\maketitle
\tableofcontents
```

```
\section{\LaTeX: source and output}
\subsection{The source document}
  A \LaTeX\ source document is a pure ascii text document
  like this --  which can be prepared using any ‘text
  editor’ (though there are special editors which are
  particularly adapted to writing \LaTeX\ source
  documents).
\subsection{The output}
  Once you’ve written a source document, you run the
  \LaTeX\ program with the document as input.

  And (in a typical modern set-up) doing this generates
  a nicely typeset PDF document which can be viewed
  onscreen and sent to a printer.

\section{Using commands}
\subsection{‘Mark-up’ commands}
  The source document crucially uses \emph{‘mark-up’
  commands} to show e.g. what is a section-heading or
  what needs to be emphasized by being printed in
  \emph{italics}.
\subsection{Other commands}
  Other key commands give instructions to make a
  document title or make a table of contents. Further
  commands produce e.g. Greek letters, such as
  $\alpha, \beta, \gamma$, or mathematics symbolism
  as in $(x+y)^2= x^2 +2xy+y^2$.
\end{document}
```

So that’s a simple example of the kind of input document that we can then use LaTeX to process in order to generate nicely typeset output.

And if you *do* invoke LaTeX to process this source document as input, then you’ll get the output as printed on the next page (and before reading on, it is probably worth briefly pausing to compare the input and output).

The first, and most essential, thing revealed by our example is that producing (say) an article using LaTeX is essentially a two-step process. You first write a source document – this is a plain text file, composed using your favourite *text editor* (not a standard word-processor like *MS Word*,

A Simple Document using LaTeX

Peter Smith

June 7, 2005

Contents

1 LaTeX: source and output

1.1 The source document

A LaTeX source document is a pure ascii text document like this – which can be prepared using any 'text editor' (though there are special editors which are particularly adapted to writing LaTeX source documents).

1.2 The output

Once you've written a source document, you run the LaTeX program with the document as input.

And (in a typical modern set-up) doing this generates a nicely typeset PDF document which can be viewed onscreen and sent to a printer.

2 Using commands

2.1 'Mark-up' commands

The source document crucially uses *'mark-up' commands* to show e.g. what is a section-heading or what needs to be emphasized by being printed in *italics*.

2.2 Other commands

Other key commands give instructions to make a document title or make a table of contents. Further commands produce e.g. Greek letters, such as α, β, γ, or mathematics symbolism as in $(x + y)^2 = x^2 + 2xy + y^2$.

which produces files with a lot of invisible gunk). The source document not only contains the raw *content* to be typeset, but also contains *explicit 'mark-up' instructions* indicating the structure of the article (e.g. indicating what counts as a section heading, what needs to be set for emphasis, etc.: these commands start with the backslash character '\'). Then, second, you invoke the LaTeX program to interpret the instructions in your source document – probably just by pressing a button in the toolbar of your favourite editor – and you thereby typeset the article's contents. This second stage happens, so to speak, behind the scenes: if all goes well, and the source document is properly constructed, the typeset document appears on the screen (nowadays, typically in PDF form) without further ado.

If you have ever written web pages then you'll probably be familiar with a very similar two-stage process. For in that case, you first construct an HTML document which again is a plain text document including both the textual content of your web-page but also containing mark-up code which specifies what is to be displayed as a heading, what is to be indented, and so on. Then second you render this page using a web-browser, which interprets all your mark-up and displays the resulting page.

Here are four more assorted comments:

(1) The table of contents for our sample mini-article is of course entirely trivial, because all the sections and subsections appear on the same single page! But still, the example illustrates how LaTeX can make use of the structural mark-up of a document (the `\section`, `\subsection` commands) to generate automatically a table of contents. Other kinds of mark-up can similarly be used to generate indices, bibliographies, etc. And since all the required mark-up is done quite explicitly – using plain text commands (and not hidden coding like a typical word-processor) – correcting and editing it is a breeze.

In fact, LaTeX's resources for producing complex documents which include tables of contents, indices, bibligoraphies, etc. are second to none. These resources already make a highly compelling reason for adopting the program for structured documents from articles to whole books.

(2) In our example, the typographical decisions – the choice and size of font for regular text, the style of headings and subheadings, the use of italics for emphasis – have all been taken by LaTeX, using its default settings for an 'article'. Even with these rather utilitarian defaults, how-

ever, the printed output already is much more professional-looking than the output from any standard word-processor: that's because the underlying type-setting engine used by LaTeX is vastly better (more about this in a moment).

We can of course change all the typographical default settings for LaTeX. We do this by adding new commands to the 'preamble' of our source document – i.e. the part *before* the command `\begin{document}` – to tell LaTeX how we want it to set section-headings, etc. If you have an eye for good typography, LaTeX enables you produce really beautiful output (to get results even approximately as good as those routinely generated by LaTeX, the only alternative is to use a very expensive commercial typesetting program). That's another compelling reason for adopting LaTeX.

(3) Note though that, if we do want to change the defaults, LaTeX pretty much insists that we still keep quite separate the business of (i) marking up some text to be set as a section-heading (for example), and (ii) choosing to set section-headings e.g. as centered, in italic 12pt type with section numbers in bold, etc. The advantages of keeping the task of writing structural mark-up entirely distinct from the task of typographical design should be pretty obvious. For a start, they are typically the responsibility of different people. The author, when composing her text, should concentrate on developing the document's content, and on imposing a clear structure on that content (e.g. by breaking the document into sections and subsections as appropriate). She can let LaTeX use its own default styles when she prints out drafts of the article or book as she is working on it (or she can use some other off-the-shelf styles, e.g. by downloading a journal's or book-series's standard typographical preamble).

Then she can pass over the document to a publisher's designer who impose a house-style or customized typographical layout. So long as every section heading has been properly marked-up using `\section{...}`, every new theorem marked `\theorem{..}` and so on, the various typographical features which the designer wants to use to distinguish section-headings, theorems etc. will be correctly fixed by the preamble definitions, without the designer having to delve significantly into the body of the document. Even when the author is acting as her own designer from beginning to end, keeping the writing tasks and design tasks strictly separate makes both *vastly* easier to manage – which is another very strong reason for taking the LaTeX route.

(4) By contrast, using a standard word-processor encourages you to mark section-headings – to stick with that example – by typography rather than by a structural marker. For example, your fingers reach for the key combination for bold type; you type the heading; revert to ordinary type and hit an extra return key to separate the section heading from the ensuing text. And on you go. Given you have probably also used bold-type for other purposes, there is no easy way of later systematically revising the design of section-headings other than searching through the whole document and changing each one by hand.

Now, true, you *can* make use of a modern word-processor's facility for defining paragraph and character styles, and you *can* define a special style for section-headings (for example): and then you can change the look of every heading by redefining the style *if you've been entirely systematic in applying your defined styles*. The only-too-familiar snag is that, while a word-processor *allows* you to do this, it certainly doesn't *require* any systematicity: in particular, it also allows you to change the style for particular paragraphs on the fly and fiddle with settings in an ad hoc way. You have to be *very* self-disciplined not to cheat and to keep a document consistently styled. LaTeX, on the other hand, more or less enforces systematicity. When you are writing a long structured document, this imposed discipline is simply invaluable.

2 But isn't LaTeX really only for mathematicians?

LaTeX is *spectacularly* good at setting complicated mathematical material. We saw above that the input `$(x+y)^2= x^2 +2xy+y^2$`, with its quite inconsistent spacing, effortlessly produces the prettified output $(x+y)^2 = x^2 + 2xy + y^2$. And that's only the very simplest of examples.

It is no wonder that LaTeX is now the program of choice for type-setting articles and books in the mathematical sciences. Understandably, then, there is a great deal of emphasis in the LaTeX literature on the handling of symbolic equations, tabular material, and so on. Which all helps to encourage the widely held view that LaTeX is really only for mathematicians and scientists.

But absolutely not so.

True, you perhaps won't want to use LaTeX for a quick thank-you letter to your aunt or for a shopping list. But, as I've already stressed, if

you need to produce *any* kind of moderately complex, structured document (especially one which might grow up to deserve a table of contents and/or index and/or bibliography), then LaTeX is simply unbeatable.

So non-mathematicians too have every reason to use the best way of processing structured documents available on planet Earth. Moreover LaTeX runs on any platform. And – not a small point – it's free (and most LaTeX add-on tools like editors are also free or shareware, and it's all supported by a very large, enthusiastic, community who are generous with on-line advice if you run into esoteric problems).

3 But isn't LaTeX difficult to learn?

No, not really. You need to get used to the way that LaTeX does things by using explicit commands written into the source document: but once you've grasped the basic concepts, producing articles documents using what I'll call *core* LaTeX is actually plain sailing.

Now, this claim might seem decidedly implausible if you have ever dipped into the two standard contemporary books on LaTeX, wondering whether to take the plunge. The latest edition of Helmut Kopka and Patrick W. Daly's *Guide to LaTeX* is a weighty 597 pages, and Frank Mittelbach and Michel Goossens' *The LaTeX Companion* has a mere 1090 further pages of detailed additional information. Both are quite terrific in their way. But the sheer size of these volumes is only too likely indeed to put off potential users (especially non-scientists): and the style and organization of those volumes is in very real danger of hiding the basic simplicity of core LaTeX.

So why are the *Guide* and *Companion* so substantial even though core LaTeX is (I claim) pretty simple? Some history will help explain.

In the late 1970s, the great computer scientist Donald Knuth started developing a typesetting program he called TeX (that's '$\tau\epsilon\chi$' in capitals, so you pronounce it 'tech', as in 'technical'). Like every word-processor or typesetting program, this deploys algorithms for arranging characters into lines, breaking the lines to make evenly set paragraphs, breaking the paragraphs across pages, etc., etc. Knuth's algorithms are vastly better than nearly all the competitors (and certainly better than any standard word-processor's), so TeX-produced pages always look a lot more elegant and professional, other things being equal. However, using 'raw' TeX requires the author to engage with the programming language for

giving instructions to place type on the page in a very hands-on way, requiring considerable skills.

So since the 1980s various 'higher level' document preparation programs have been written which use TeX as the typesetting engine under the bonnet, while insulating the user to a greater or lesser extent from having to get their hands dirty messing with the engine's mechanics. Essentially these are suites of TeX macros. And Leslie Lamport's LaTeX is by far the most widely used of these 'higher-level' programs. Like TeX, it is free to users, and runs on any platform. The standard version, released in 1994, is now LaTeX 2_ε; it is described in Lamport's own (relatively slim) book *LaTeX: A Document Preparation System*.

But standard LaTeX is very readily extensible. If you want to add 'packages' defining new commands to help you to do something using TeX that unaugmented LaTeX 2_ε can't do, then that is relatively easy. And if you don't like the way LaTeX 2_ε handles some task, then your add-on 'package' can even cheerfully change the effect of LaTeX 2_ε commands. By now, there are many hundreds of these packages freely available; and a good number are included in a standard LaTeX installation. It is the availability of this enormous range of extensions to the original LaTeX which gives the whole system its enormous power and flexibility (want to set chess diagrams, chemical diagrams, even typeset music? – then there will be a package to solve the problem). But the plethora of add-ons is also the reason why the *Guide* and *Companion* – which even taken together don't pretend to be comprehensive – are dauntingly weighty volumes.

Suppose, however, we get back to basics, and ignore most of the add-on packages. Suppose we also largely ignore the wonderful mathematical capabilities (which after all aren't of much concern to non-scientific users). Suppose too we ignore at the outset most of the controls for typographical fine-tuning that are built into LaTeX 2_ε. Then the residual *core* LaTeX mark-up language is revealed again as, at its heart, conceptually very simple and intuitive, and hence perfectly accessible to the humanities user.

4 But isn't LaTeX showing its age?

In computer terms, LaTeX is ancient and TeX is simply antediluvian. So you might wonder 'Isn't it mad to use an antique when you could have

all the benefits of *WhizzWord 2010* – or whatever the latest, greatest, commercial word-processor might be? Retro-chic is fine in its place: but isn't using a program which indeed dates from before the era of PCs – in fact, a program initially designed to run on mainframe computers without a graphical user interface – just making life quite ludicrously difficult for yourself?'

If that strikes you as a good question, that's because you haven't yet quite appreciated the kind of program LaTeX is. Fair enough – but let me try to explain the key point again about the two-step process of working in LaTeX, saying just a bit more on the user-experience.

As I said before, to use LaTeX as a document processor you first need to produce a source-document which includes the content you want to typeset together with all mark-up commands to structure it. To write this source-document, you use some text-editor (and this editor *isn't* part of LaTeX: you can use any suitable tool). You then invoke the LaTeX program which interprets your mark-up instructions and then does very clever things involving TeX behind the scenes in order to generate some typeset output.

So the overall user-experience will depend crucially on (a) the user-friendliness of the text-editor you initially use, and (b) the speed with which your computer can interpret the LaTeX commands and run TeX to turn source-code into output you can read on the screen and print out. And *both* these crucial things have improved radically as personal computers have improved, even while LaTeX and TeX have remained stable behind the scenes – we now have very user-friendly editors, and benefit from the development of the PDF standard for output.

So, for example, in writing this present document, I am (a) using the latest version of an elegant, free, Mac OSX editor called 'TeXShop'. I have two large windows open: on the left, I am writing the source-document (occasionally using some of the neat tools that TeXShop provides to expedite the process). By hitting the 'Typeset' button at the top of that TeXShop window, I can from time to time (b) get LaTeX to process the document almost instantaneously. The result (c) appears as a PDF graphic in the right-hand window (assuming I've not made a formatting mistake like forgetting to close the curly brackets that must surround many a LaTeX command, in which case I get an error message). The overall (a)-(b)-(c) user-experience is perfectly smooth and a lot nicer than battling with the messy interface of the *MS Word* bloatware. But if you don't

like my set-up, there are – even on the Mac – a variety of other editing tools you could use instead. Further, most of the LaTeX editing tools on the Mac and other platforms are heavily user-customizable, so you can work with LaTeX in the way that suits you best, in an up-to-the-moment user-environment with all the bells and whistles you could want.

Indeed, because of the ease and lightning speed of using LaTeX on modern computers, the program seems to be have been recently taken up by more and more non-scientists. In joining them, you certainly *aren't* taking a step back into the computational dark ages!

And it is worth mentioning here the not unimportant point that, in using LaTeX, you are also future-proofing your work. For source documents are plain ascii text files and will remain readable, if not to the end of time, at least for as long as any of us will be around to want to use them. Commercial word-processors, by contrast, use proprietary file formats: recovering *all* the content from files produced by superseded work-processors can become a frustrating business a lot sooner than you might think (as I've found to my cost since Adobe decided not to port the once-wonderful FrameMaker to the new Mac OS).

OK, to be frank, there *are* aspects of using LaTeX which are tiresome in a way that *is* due to the age of the underlying program. The obvious case is its font handling, if you want to use something other than the usual repertoire of Times, Palatino, etc. The program predates the ready availability of a vast range of postscript fonts, and getting standard LaTeX to use new fonts is slightly effortful (though you can use a variant called XeTeX which copes with fonts wonderfully). However, this isn't a serious issue for the ordinary author, unless and until you reach the point where you are engaged in designing sophisticated typography for a book.

5 Begin with core LaTeX

To expand my earlier suggestion, in order to impose some order on its ramifications, it is quite helpful (no, make that '*very* helpful') to think of LaTeX as comprising various 'modules' built onto a basic core program. Not that this division into modules can be made very sharp: but imposing a rather rough-and-ready map is a great deal better than imposing none at all.

So let's think in terms of the being

Core LaTeX A restricted set of mark-up commands needed to typeset a basic document (which might be divided into chapters and sections, have footnotes, indented material, numbered lists, have headers and footers, etc., and might use some special symbols),

and then a collection of modules

For more advanced text manipulation Commands for using more than one column, using tables, inserting figures, adding marginal notes, etc.

For mathematicians Commands for displaying more or less complex mathematics.

For importing graphics Commands for placing various sorts of graphics, on the page, running text round graphics etc.

For book-making Tools for producing indices and bibliographies.

For designers Commands for typographical fine-tuning – e.g. for changing the LaTeX default fonts and the leading between lines; for changing the layouts for section headings or the indentation of lists; and so on and so forth.

For added user control Commands that enable a user to define her own commands to abbreviate (clusters of) frequently used verbose commands; also commands that allow a long document to be built up by 'including' other LaTeX files, etc.

LaTeX 2$_\varepsilon$ already contains all these modules (and a modern LaTeX installation enhances each of those add-on modules in various ways, as well as adding more esoteric modules).

Now, while I think it is very natural to think of LaTeX as coming chunked up into modules like that, this isn't how the standard documentation approaches things. In particular – and arguably flatly against the spirit of the program – the *Guide to LaTeX* often runs together in the same section basic structural mark-up information with much more complex information related to typographical fine-tuning. That makes the Guide quite unnecessarily daunting.

For a mini-example, suppose that you want to frame some text in a simple box like this. Then the structural mark-up is simplicity itself:

you just use the command `\fbox{...}` to enclosed the text you want to frame. End of story.

However, all sorts of typographical fine-tunings are possible – changing the thickness of the surrounding lines, the distance of the enclosing lines from the text, allowing the text to be centred in a rather longer box with white space before and after the text, etc., etc. And, after giving the basic structural command, the *Guide* immediately explains these typographical commands. I suppose that, looked at in one way, it makes good sense to put all the stuff relevant to framed boxes in one place. But looked at another way, this is running together a bit of *author*-relevant core LaTeX with, quite differently, *designer*-relevant fine-tunings (it's the author, let's suppose, who makes the basic decision to set off some bit of text in a box, the designer who worries about the thickness of the lines). And even if you are being your own designer, these are quite separate and independent matters.

It is worth keeping the separation between core author-relevant commands and typographical niceties clearly in mind when you approach LaTeX. So, at least at the beginning, just set aside issues of typographical detail (and other bells and whistles) when you read the *Guide* or other documentation. And if you are not a mathematician, ignore for the moment the early chapters on setting formulae and equations. Then what's left – core LaTeX – can be pretty briskly mastered. Honestly! For this core is relatively clean and simple, yet contains what the non-science user needs for most documents. And if you start using core LaTeX and find it useful and congenial, then learning how to use commands from other modules (including the typographical one) on a need-to-know basis is relatively easy.

6 So how do I get started?

I'll here be very brief indeed. First, you need a working LaTeX installation. There are two likely situations.

1. You are using a networked computer, probably in an academic environment. Then you may very well find that LaTeX is already available on your server (particularly if you are running a species of unix, such as Mac OSX or Linux). If it isn't – perhaps because you are on a small humanities network and no-one has previously asked for LaTeX – then your local friendly computer support ser-

vices should be happy to install it together with a suitable editor. (After all, it is in the university's interest that as many people as possible use free software.)

2. You want to acquire and install LaTeX your own computer. Just download everything you need over the internet. Start by visiting `http://www.tug.org/interest.html` for links to free (La)TeX implementations. You may need separately to get a suitable editor: use e.g. TeXShop for the Mac, WinEdt for Windows, Emacs for Linux, all available for download over the internet.

And now you need some more documentation to get you under way. Lamport's original *LaTeX: A Document Preparation System* is still a very good place to start. Or delve into on-line resources.[1]

Almost certainly, however, in an academic environment there will be colleagues or fellow students who are already LaTeX converts, and like other converts will be happy to spread the gospel and help get you started. Take their advice too.

And enjoy!

[1] One short word of explanation, though. You'll often see reference in older books or documentation to 'dvi' files: think of these as a superseded, LaTeX specific, precursor to PDF files. With a modern LaTeX installation you produce PDF files instead.