A Beginner's Guide to LaTeX (on the Mac)

Alan Munn

September 18, 2016

Abstract

This document is aimed primarily at completely novice users and is focused on emphasizing just the things that you need to know to actually get stuff done. It's not a complete introduction, and it will be necessary to read other guides as well, but it covers most of the basics, while skipping over much of the details that other intros have which in my opinion are not helpful to beginners. It recommends using XeLaTeX instead of pdfLaTeX.

Contents

1 Some Terms

As with any new domain of knowledge, there are always some new terms to be learned. Learning about LaTeX is no exception. A number of the most important terms revolve around distinctions between the various components needed to use LaTeX. It's important that you understand the basic architecture of the system so that you can get help from others (as these questions are often among the first you may be asked by people trying to help you.)

It's important to realise that there are three main components to using LaTeX on any computer: you need a TeX Distribution, a text editor, and a PDF previewer. Within the TeX distribution are numerous *engines* which are the programs that actually process your LaTeX source and turn it into PDF. Schematically this can be viewed in the following way:

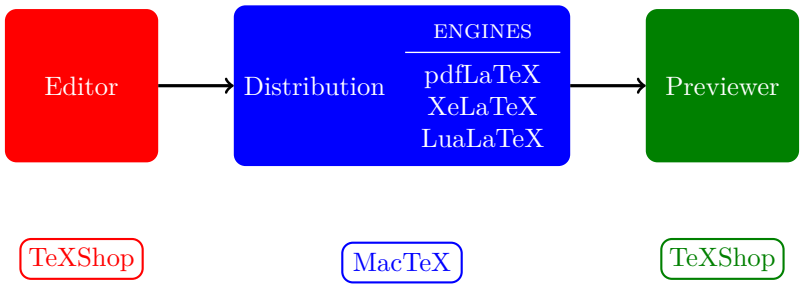


Figure 1: The major components in using LaTeX

distribution: The whole set of Unix programs that are used to process TeX documents. The distribution on the Mac is called MacTeX and is essentially the TeX Live distribution with some extras for use with the Mac.

editor: The application that you use to create the TeX source document. Your editor is TeXShop. It provides an editor, a PDF previewer and a simple user interface between your source document and the TeX distribution.

previewer The application that allows you to view PDF documents. TeXShop incorporates its own previewer, so if you use TeXShop you don't need another one. But Adobe Reader and Preview are also PDF previewers.

engine: The actual TeX program that will be used to process your source file into a PDF file. There are 4 main engines: `latex+DVI`, `pdflaTeX`, `XeLaTeX`, and `LuaLaTeX`. The ones that you need to know about are really `pdfLaTeX` and `XeLaTeX`. Most of the time you will be using `XeLaTeX`. You need to know which one you are using to tell TeXShop how to process the file. In TeXShop, the engine is chosen with the dropdown menu at the top of the document window. It defaults to `pdfLaTeX` (but is displayed as simply `LaTeX`.)

The safest way to choose the engine in TeXShop is to use the `Program` item from the `Macros` menu to add a comment to the file which tells TeXShop which engine to use for that particular document. The comments look like the following:

- When you choose `pdfLaTeX` TeXShop inserts
  `% !TEX TS-program = pdflatex`
- When you choose `XeLaTeX` TeXShop inserts
  `% !TEX TS-program = xelatex`

compiling Processing a source file with a particular engine is often referred to as *compiling* the document.

preamble Every document begins with a `\documentclass` command. The actual text of the document is enclosed with `\begin{document} ... \end{document}`. Everything between the `\documentclass` command and the `\begin{document}` is called the *preamble* of the document. This is the place where packages are loaded, and extra commands are defined or given values.

2 Getting Help

There are a number of ways to get help.

2.1 Online help

- LaTeX workshop (for linguists)
- LaTeX books by Nicola Talbot
- TeX Stackexchange Question/Answer site

2.2 Books

- *The LaTeX Companion, second edition* by Frank Mittelbach and Michel Goosens. Addison Wesley
- *LaTeX and Friends* by Marc van Dongen. Springer.

2.3 Package documentation

The documentation for every package is accessible from within TeXShop. In the `Help` menu there is an item `Show help for package`. If you choose this and enter a package name, the documentation for the package should show up in your default PDF viewer (usually Preview or Acrobat Reader.)

3 Some weird things to understand about LaTeX

For people who are used to using a word processor like Microsoft Word, the idea that your source document is separate from your formatted document is a bit odd at first. The basic thing to understand about a LaTeX document is the following:

Formatting is separated from content!
Formatting should be associated with a semantics!

In reality what does this mean? Well basically, if you want LaTeX to format something a particular way, you should tell it what *kind* of thing it is (give it a semantics) and then associate the formatting with the kind rather than the actual thing. Here's simple example. Suppose you want your section headings to be in 24 point Helvetica and coloured blue. Typically in MSWord, when you wanted a new section you may have just started a new paragraph and manually changed the font and the indention and then changed it back for the text following. Then in the next new section you did the same thing. (Actually Word allows you to associate formatting with styles, so even in Word you shouldn't have done this, but most people do.) This of course gives you the *output* you want, but what if the next day you decide that all the headings should really be 18 point and orange? You'd have to change each one manually. In LaTeX you don't ever do this (although you could, in principle). Instead, you say at the beginning "I want all section titles to be 24 point Helvetica and blue" and then for each section you say "This thing is a section title" by saying `\section{My section title}`. LaTeX associates the formatting you assigned to sections to every bit of text you mark as a section. Now changing all the sections is easy: you just change the definition of the formatting for `\section` and all sections will change automatically.

This idea doesn't apply just to sections, however. You can (and should) apply it to everything that requires repeated formatting. Example sentences are a prime candidate for this kind of treatment: you make things as examples and then LaTeX (or more properly the example formatting package) assigns the correct formatting automatically. This saves you a lot of time, and also ensures that everything in your document is consistent.

4 Some basic packages

Although there are many basic things that LaTeX can do, many useful things are implemented as separate *packages*. Each package is loaded separately in the preamble of the document, and comes with its own documentation.

The following is a list of the basic packages that you are likely to use if you are doing academic linguistic work with LaTeX. Some very basic examples of their use are included. You should read the relevant package documentation for other functionality and a full description of the available commands and their use.

If you find something that you want to do that you can't, you should almost certainly find out if there is a package to do it. The chances are there is, and it will almost certainly be much better than anything you could put together yourself.

4.1 `geometry`: set page margins

The geometry package provides a very simple way to adjust the page margins of your document. The default page margins for TeX are quite large, which produces supposedly very readable text, but most of us are used to reading documents with roughly 1" margins. It's very easy to do this with the `geometry` package:

`\usepackage[margin=1in]{geometry}` sets 1" margins all around

`\usepackage[tmargin=1in,bmargin=1in,lmargin=1.25in,rmargin=1.25in]{geometry}`
sets 1" top and bottom margins and 1.25" left and right margins.

4.2 `fancyhdr`: headers and footers

The `fancyhdr` package allows you to quickly create headers and footers for your document. Headers and footers are implemented with what LaTeX calls a `pagestyle`. There are some default page styles defined:

`\pagestyle{empty}` as its name implies, this has no header or footer information at all.

`\pagestyle{plain}` this has a page number at the bottom centre, but is otherwise empty.

The `fancyhdr` package adds a `\pagestyle{fancy}` which can be modified to suit your needs. (In fact it can do much more than this, but for most applications just having one more page style is sufficient.) The package provides a bunch of independent commands for setting the left, centre and right headers and footers. Here is a simple example:

`\usepackage{fancyhdr}`

`\lfoot{My Name}` sets the left footer to "My Name"

`\cfoot{}` sets the centre footer to nothing

`\rfoot{\thepage}` sets the right footer to the page number

Similar commands can be used to set the header parts (`\lhead`, `\chead`, `\rhead`). The default fancy page style puts a rule in the header. You need to be able to turn this off:

`\renewcommand{\headrulewidth}{0pt}` sets the head rule to 0

To use a pagestyle, you issue the command `\pagestyle` (for all subsequent pages) or `\thispagestyle` for the current page. Here are some examples.

`\pagestyle{fancy}` makes all pages have the fancy page style

`\thispagestyle{empty}` makes the current page empty (useful for the first page of a document)

4.3 `titlesec`: change format of section headings

The standard document classes define formatting for the major divisions of a document: `\chapter`, `\section`, `\subsection`, etc. (The `article` class doesn't define a `\chapter` division.) Although the standard formatting is fine, many people find the section headings overly large, or want to change the fonts. The `titlesec` package makes this easy. It's a very powerful package, so I will only show two simple things that are useful to do with it:

`\usepackage[small]{titlesec}` makes all section headings equal a medium size.

`\usepackage[sf]{titlesec}` makes all section headings a sans serif font.

`\usepackage[center]{titlesec}` centres all headings.

`\usepackage[small,it,compact]` makes all section headings small and italic and reduces the spacing above and below titles.

`\titlelabel{\thetitle.\quad}` (issued after loading the packge) Adds a dot after the section number.

4.4 `enumitem`: change list formatting

LaTeX offers three basic list stuctures: `enumerate` (numbered lists), `itemize` (each item receives the same symbol), and `description` (each item has a specified label.) The `enumitem` package is the simplest way to modify these list properties. As with the `titlesec` package, this package can do a lot, so here we just give some simple examples. List parameters can be specified as an optional argument to the `\begin{<list>}` command. (The corresponding `\end{<list>}` is omitted here.)

`\usepackage{enumitem}`

`\begin{enumerate}[label=\Alph*]` make a list with upper case letters: A, B, C, etc.

`\begin{enumerate}[label=\bfseries(\arabic*)]` bold face numbers in parentheses: **(1), (2), (3)**

`\begin{itemize}[itemsep=0pt]` itemized list single spaced (the default spacing is considerably bigger).

The package also provides ways of defining new lists, and setting all of the various margins (the left margin of the list, plus the indent for each item, etc.).

4.5 `parskip`: blank lines to separate paragraphs

By default, the standard classes indent new paragraphs (except after section headings) and there is no good way to insert a blank line to separate paragraphs. Although it's not usually good practice for regular documents to have blank lines separating paragraphs with no indentation, in linguistics it's usually necessary for handouts. In this case you should use the `parskip` package, which automatically does this in all of the right places.

`\usepackage{parskip}`

4.6 `setspace`: double or 1 and 1/2 spacing

By default, LaTeX also single spaces everything, since this is typographically correct. However, it is sometimes required to submit drafts with double or one and a half spacing. Doing this by hand is very tricky, so the `setspace` package does it for you.

`\usepackage{setspace}`

`\onehalfspacing` set spacing to 1 and a half

`\doublespacing` double spacing

`\singlespacing` (for use in places where single spacing is helpful)

5 Dealing with tables

Tables are sometimes considered difficult in LaTeX because simple text editors aren't designed to deal with tabular data. However, LaTeX can format tables extremely nicely, especially with the `booktabs` package. Before you make any table in LaTeX you should read the documentation of `booktabs` since it contains extremely useful information about what makes a good an bad table. (The simple rule is: a good table has no vertical lines.)

5.1 `booktabs`

The `booktabs` packages adjusts the row spacing and the spacing above and below horizontal lines in a table so that they look much more professional. It also has different thickness of lines for lines at the top and bottom of the table compared to the middle. Here's a side-by-side comparison:

```
\begin{tabular}{ccc}
\hline
ColA & ColB & ColC\\
\hline
43  & 200 & 75\\
280 & 16  & 88\\
102 & 77  & 340\\
\hline
\end{tabular}
```

ColA	ColB	ColC
43	200	75
280	16	88
102	77	340

without `booktabs`

```
\begin{tabular}{ccc}
\toprule
ColA & ColB & ColC\\
\midrule
43  & 200 & 75\\
280 & 16  & 88\\
102 & 77  & 340\\
\bottomrule
\end{tabular}
```

ColA	ColB	ColC
43	200	75
280	16	88
102	77	340

with `booktabs`

5.2 Tables longer than a page

Regular tables in LaTeX cannot span multiple pages. If you have a very large table, you should use the `longtable` package in addition to the `booktabs` package.

5.3 How do I input a table?

For small tables, TeXShop has a the `Matrix Panel` (found in the `Window` menu.) This gives a small spreadsheet like interface which you can use to quickly enter a small table. The formatting of the code itself is not very pleasant, however, and for small tables (such as those above) it's probably easier just to code the table by hand. For larger tables, you can arrange your data in a spreadsheet such as Excel and use the `Paste Spreadsheet Cells` macro in the `Macros` menu. All you do is copy the cells from Excel and then choose `Paste Spreadsheet Cells`. You will be prompted for a table style, and the macro does the rest.

6 Dealing with Fonts

Before the advent of XeLaTeX (and LuaLaTeX), dealing with fonts in LaTeX was an adventure, to say the least, especially if you wanted to stray from the standard fonts that came with the system. For mathematicians, (who for a long time were the primary users of TeX) the need for large inventories of math symbols (which the standard fonts provide) outweighed the need for different font styles or for more esoteric things like non-Latin scripts. For work in Linguistics particularly, fonts are actually much more important: minimally most of us need phonetic fonts, and depending on which languages you work on you may need a variety of accented characters, and possibly non-Latin scripts. XeLaTeX makes all this simple, because

XeLaTeX allows you to use almost any font that is installed on your system. The key to all of this is the `fontspec` package.

6.1 `fontspec`

The `fontspec` package provides a LaTeX interface to the modern OpenType fonts that are now the norm in most operating systems. As with many of the packages described here, `fontspec` can do a lot of things, but for simple use you mainly need to know how to set the main document font and how to use more than one font in the same document. For more details of its capabilities, see the documentation.

There are two main font setting commands; here are examples of their use:

`\usepackage{fontspec}`

`\setmainfont[Ligatures=TeX]{Times}` sets the document font to Times New Roman

`\newfontfamily\phonetic[]{Doulos SIL}` creates a new font command `\phonetic` which changes the font to Doulos SIL.

In the first example, the Times font is loaded. (You can find the names of fonts using the Mac's Fontbook application.) The option `[Ligatures=TeX]` tells XeLaTeX to translate things like the standard TeX quotation marks and dashed (`` `` '' `` and `--`, etc.) into their corresponding Unicode characters. In the second example, the Doulos SIL phonetic font is assigned to a macro `\phonetic`. You can the use `{\phonetic ... }` and enter phonetic characters directly in your source document.

7 Linguistics Packages

The main linguistics package that you will need is something to format examples and glosses. There are three major ones around: `gb4e`, `linguex`, and `ExPex`. The latter package is extremely powerful, but as such is probably not the best for a beginner. Of the remaining two, they are both good; I prefer `gb4e` because its markup is more semantic than `linguex`, and so I will describe it here.

7.1 Examples

`gb4e` implements two environments for examples: the `{exe}` environment is for examples usually regularly numbered with the number in parentheses. The `{xlist}` environment is for sublists of examples under the same number. These are usually numbered *a, b, c,...*. Since examples are often referred to, it's always useful to add a label to every numbered example. Within each of these environments, the actual example itself is introduced with the command `\ex`. This macro takes one optional argument for a grammaticality judgement mark. Here are some examples:

```
\usepackage{gb4e}

\begin{exe}
  \ex[*]{This sentence ungrammatical.}
  \label{NoVerb}
\end{exe}
```

(1) * This sentence ungrammatical.

```
\begin{exe}
  \ex\label{NoNoun}
  \begin{xlist}
    \ex[]{This is a grammatical sentence.}
    \ex[*]{This is a grammatical.}
  \end{xlist}
\end{exe}
```

(2) a. This is a grammatical sentence.
b. * This is a grammatical.

To refer to these examples in the text, you would use the `(\ref{NoVerb})` which would produce (1) and `(\ref{NoNoun})`, which would produce (2).

7.2 Glosses

The `gb4e` package also implements word-by-word glosses. The glossing macro `\gll` introduces a two line example and gloss. Each line of the gloss is ended with `\\`. An optional `\trans` macro can be used for a translation. The glossing macros use spaces to line up the words, so if more than one word in the gloss lines up with a single word in the actual example you need to enclose the multiple words in braces. Conversely, if there are words for which there is multiple source words and only one gloss word, you need to insert `{}` to take up the space. Here is an example:

```
\begin{exe}
\ex[]{\gll O João esta cansado.\\
           the João is tired\\
      \trans ``João is tired.''}
\end{exe}
```

(3) O João esta cansado.
the João is tired
"João is tired."

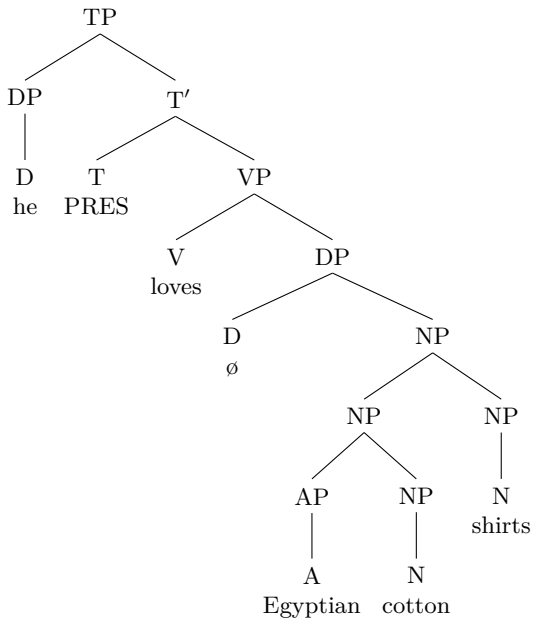
7.3 Trees

LaTeX also has a number of packages to draw trees. The most powerful of these is the `forest` package. The `tikz-qtree` package, which is based on an earlier `qtree` package is also very good. The basic syntax of the packages the similar, and both are very good packages. The `forest` makes annotating trees somewhat simpler, and produces more compact trees but for most trees, either one is a good choice.

`tikz-qtree` The basic syntax is a bracketed structure where each node in the tree is usually introduced by a label. The label is appended to the left bracket of the constituent with a dot. Note that you need to leave a space between words and closing brackets.

Here's an example:

```
\usepackage{tikz-qtree,tikz-qtree-compat}
\tikzset{every tree node/.style=
          {align=center, anchor=north}}
\Tree
[.TP
  [.DP [.D\\he ]]
  [.T\1
    [.T\\PRES ]
    [.VP
      [.V\\loves ]
      [.DP
        [.D\\{ø} ]
        [.NP
          [.NP
            [.AP [.A\\Egyptian ]]
            [.NP [.N\\cotton ]]
          ]
          [.NP [.N\\shirts ]]
        ]
      ]
    ]
  ]
]
```



`forest` The syntax of the bracketed structures in the `forest` package is almost the same, but with two important differences: labels don't have a dot preceding them, but are simply placed right next to their left bracket, and all terminal nodes must be enclosed in brackets. Here's the same example using the `forest` package.

```
\usepackage[linguistics]{forest}

\begin{forest}
[TP
  [DP [{D\\he} ]]
  [T$'$
    [T\\PRES ]
    [VP
      [V\\loves ]
      [DP
        [D\\{ø} ]
        [NP
          [NP
            [AP [A\\Egyptian ]]
            [NP [N\\cotton ]]
          ]
          [NP [N\\shirts ]]
        ]
      ]
    ]
  ]
]
\end{forest}
```