

# Absolute L<sup>A</sup>T<sub>E</sub>X Beginner

Lim Lian Tze

liantze@gmail.com

## Abstract

Most people use the ubiquitous Word<sup>TM</sup> for preparing text documents in the WYSIWYG tradition. L<sup>A</sup>T<sub>E</sub>X is a document typesetting system that produces professional-looking documents while letting the author focus on the overall document structure. With L<sup>A</sup>T<sub>E</sub>X, you concentrate more on *what* you want to write, not *how* you want to format it<sup>1</sup>.

Read this article for your first encounter with the typesetting platform of choice in the scientific publishing community. . .

## 1 Interested in L<sup>A</sup>T<sub>E</sub>X?

So you've heard about L<sup>A</sup>T<sub>E</sub>X. Perhaps you've heard about how easy it is to produce beautiful, professional-looking documents with L<sup>A</sup>T<sub>E</sub>X, or how difficult it is to learn, or a dozen other varying opinions about it. In short, you've read a couple of (or more) articles passionately singing the praises of L<sup>A</sup>T<sub>E</sub>X or cursing it.

Chances are you're wondering: *should I or shouldn't I learn more about this L<sup>A</sup>T<sub>E</sub>X?*

Well wonder no more, because it isn't that difficult to quickly try out L<sup>A</sup>T<sub>E</sub>X to see if it's worth your time.

So let's roll up our sleeves and typeset our first L<sup>A</sup>T<sub>E</sub>X document!

---

<sup>1</sup>Well, yes, *someone* has to worry about it, but it shouldn't be *you*: you, the author, should concentrate on writing and organising the content, not making it look good. The someone who should worry about *that* should provide a L<sup>A</sup>T<sub>E</sub>X formatting style file to you, not continually harassing you about it.

## 1.1 Installation

To start with, you will need a  $\LaTeX$  distribution on your computer.  $\LaTeX$  is free software; this means it can be used, copied, studied, modified and redistributed with minimal restrictions. It is also free-of-charge.

If you are using a GNU/Linux machine, chances are that a  $\LaTeX$  distribution is already included with your GNU/Linux installation. If you are on a Windows platform, I would recommend the Pro $\TeX$ t distribution for its easy installation procedures. You might want to refer to our  $\LaTeX$  installation guide at <http://liantze.googlepages.com/latextypesetting#latexwin> for the instructions.

## 1.2 Editor Tools

Preparing a  $\LaTeX$  document is as simple as creating a plain text file with specialised markups. This means any plain text editor can be used to edit the file: Notepad, *vi*, *emacs*. But just as some people prefer more convenient IDEs to Notepad for editing C++ source code, there are editor environments that streamlines the creation of  $\LaTeX$  documents.

I'll be using screenshots of *TeXnicCenter*, a tool for the Windows platform (included in Pro $\TeX$ t), to illustrate the flow of preparing  $\LaTeX$  documents in this article. For those who prefer a plain text editor and the command-line, I will also show the commands to issue where necessary.

## 2 In the Beginning there was the Document

We'll first create a  $\LaTeX$  article document with a minimal structure. It'll have a title and author name, 2 sections and 1 subsection.

Using *TeXnicCenter* or your text editor of choice, create a new file and type in the contents of Listing 1.

Listing 1: A Minimal  $\LaTeX$  Article

```
1 \documentclass[a4paper,11pt]{article}
2
3 \title{My First Document}
4 \author{John Doe}
5
6 \begin{document}
7
8 \maketitle
```

```

9 \tableofcontents
10
11 \section{Introduction}
12 This is the introduction.
13
14 \section{Main Text}
15 My first \LaTeX{} document! Isn't it exciting?
16
17 \subsection{More Details to Come}
18 This is just the beginning. There are lots of things
19 you can achieve with \LaTeX{}, and I hope to write
20 about them as much as I can. So much to do, so little
21 time\ldots
22
23 Watch this space.
24
25 \end{document}


```

Save the file once you're done typing. Give it a name like `first.tex`. It's a good idea to treat your  $\text{\LaTeX}$  documents as projects: I usually create individual folders (Fig. 1) to hold a  $\text{\LaTeX}$  source file (the `.tex`) and related files, such as bibliography lists and graphics.

If you are using TeXnicCenter, I also recommend that you create a new *project* with `first.tex` as the *main file*.

To do this, first make sure `first.tex` is open and active in TeXnicCenter. Then select from the menu Project → Create with active file as main file (Fig. 2). Select "English" as the language to use, and click OK. The Navigator pane should now be visible on the left of your screen, showing the internal structure of `first.tex`.

## 2.1 Building the PDF Output

We are now ready to build the PDF output from the  $\text{\LaTeX}$  file just created. If you're using TeXnicCenter, select the LaTeX => PDF output profile (Fig. 3 red circle **1**). Then click on the Build and view output button  (Fig. 3 red circle **2**).

If you made no typos in `first.tex`, and if you have a PDF viewer installed, you should see the output `first.pdf` displayed after a short<sup>2</sup> while. You may notice, though, that the table of contents doesn't seem to

---

<sup>2</sup>and agonising? No, surely it's not that bad!

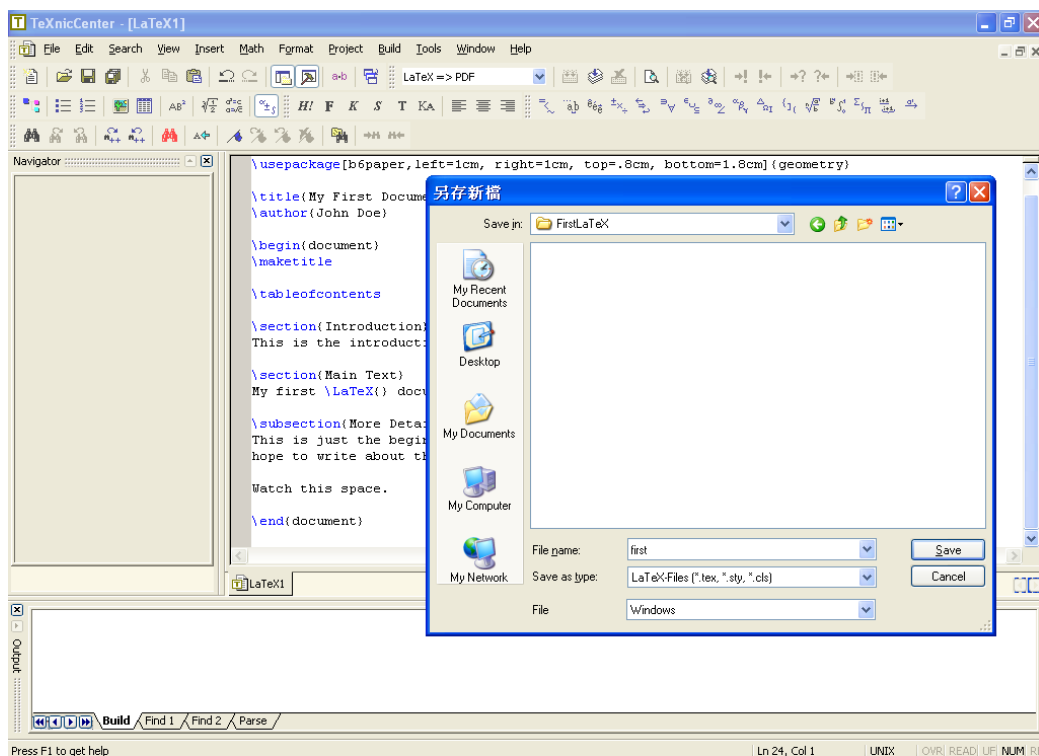



Figure 1: Saving your first L<sup>A</sup>T<sub>E</sub>X document in TeXnicCenter

be complete. Click on  again, and you should see the final PDF output – something similar to Fig. 4.

(If, at this point, TeXnicCenter reports an error about not being able to write to output file and you are using Acrobat Reader as your default PDF viewer, please follow the instructions in section 3.1, step 12 of our installation guide<sup>3</sup> before trying again.)

On the other hand, if you prefer to use the command-line to build your output (especially if you're using GNU/Linux), fire up a console, change to the path where `first.tex` is stored, and issue the following command:

```
pdflatex first.tex
```

You may then view the result, `first.pdf` in a PDF viewer. Again, you may notice that the table of contents is incomplete. Re-run `pdflatex`, and refresh the PDF file in your PDF viewer. You should then see something similar to Fig. 4.

<sup>3</sup><http://liantze.googlepages.com/latextypesetting#latexwin>

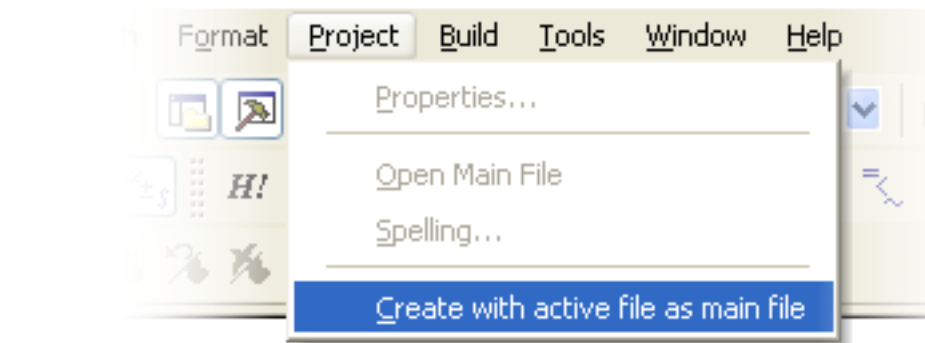


Figure 2: Creating a project

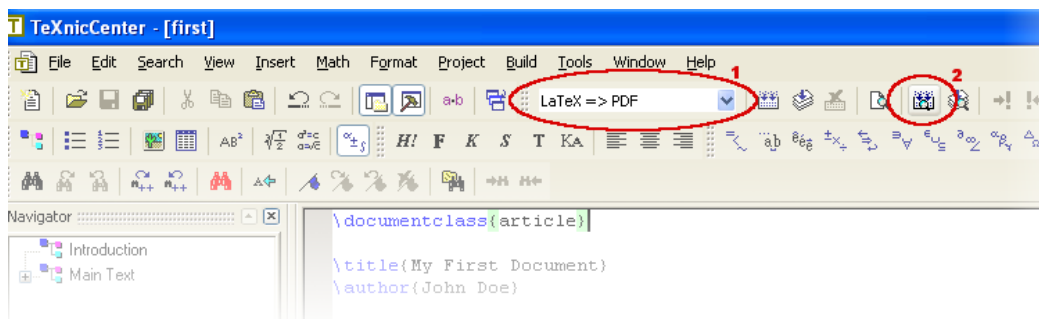


Figure 3: Building  $\LaTeX$  PDF output with TeXnicCenter

There! There's your  $\LaTeX$  PDF output in all its glory. Now that you've achieved your mission of giving  $\LaTeX$  a try, it's mission accomplished; but if you're interested in knowing which bits did what, read on...

## 2.2 Wow. Tell Me How That Worked.

### 2.2.1 We Do It with "Special Mark-ups"...

What you have done was to *define the structure of a document* using special mark-ups. One type of mark-up is the *command*, which looks like this:

```
\cmdname{argument}
```

Some commands take multiple arguments, while others don't need any. For example:

- `\ldots{}` produces the ellipses (...) symbol,
- `\LaTeX{}` produces the  $\LaTeX$  symbol,

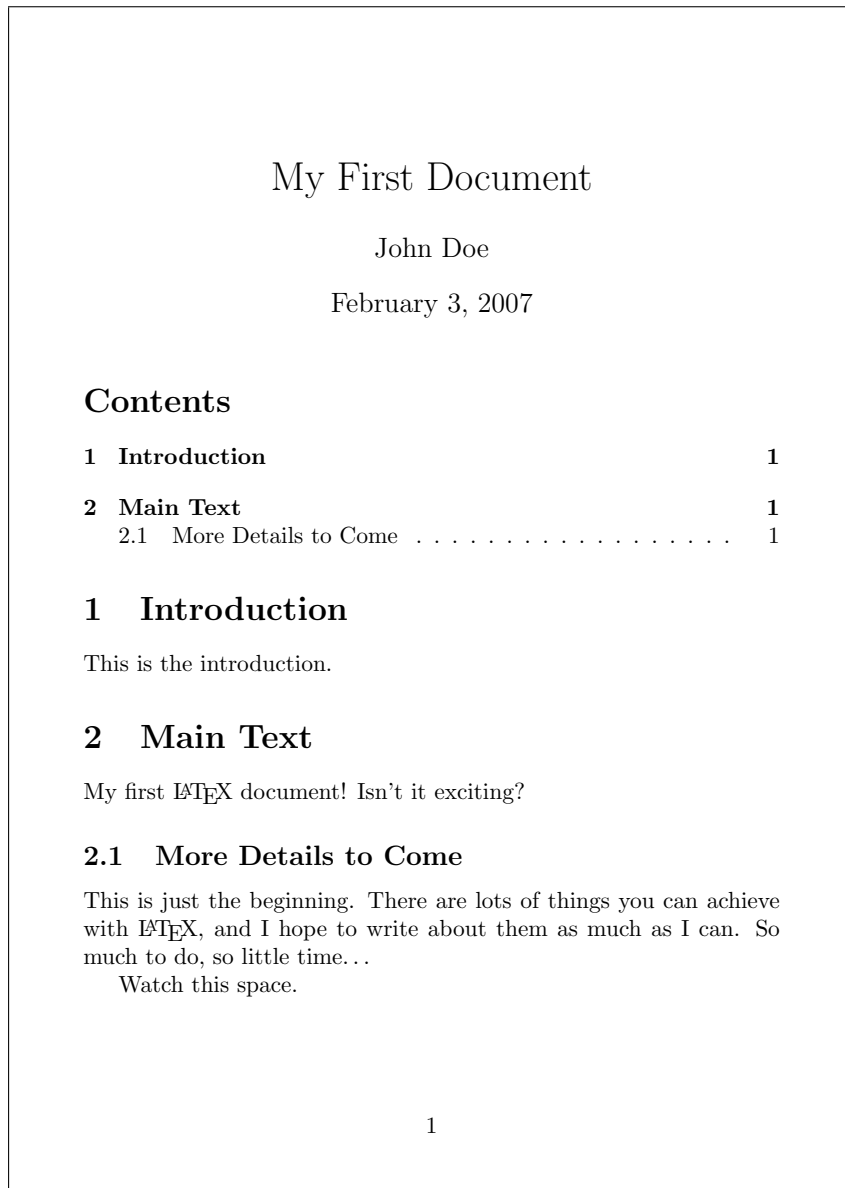


Figure 4: PDF Output from `first.tex` (text and paper size not to scale)

- `\section{Introduction}` signals that you are beginning a new section titled “Introduction”.

You can pass *options* to some commands, too:

```
\cmdname[opt1, opt2]{argument}
```

So if someone tells you to “Use the `\footnote` command with ‘*TODO*’ as the argument, and ‘4’ as the option”, this is what you’d write:

```
\footnote[4]{TODO}
```

There is another type of special mark-up called *environments*. When you use one of these, you have to specify where it *begins* and *ends*. For example, when some text is surrounded by the `center` environment, it will be centre-justified:

```
\begin{center}  
...some contents...  
\end{center}
```

## 2.2.2 ...and Now, Back to Our Document.

A  $\LaTeX$  document, like the one that you just created, is made up of a *document preamble* and the *document body*. The document body is where you write your actual content, and is surrounded by `\begin{document}` and `\end{document}`. (Bingo, that's a  $\LaTeX$  environment.) Everything before that (i.e. lines 1–4 in Listing 1, in blue colour) is the preamble.

## 2.2.3 The Preamble

It's where you define various settings (if any) for your overall document. "What settings," you ask? Well, we started off with the document class declaration, the paper size and default font size, as described a little earlier.

All  $\LaTeX$  documents start with a `\documentclass{class}` declaration. (Yep, that's a  $\LaTeX$  command.) We declared our first document to be an `article`, of which the highest division is the `section`:

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

Other common document classes include `report` and `book`, both of which provide `chapter` as the highest division. We also passed the `a4paper` and `11pt` options to the `article` class, so that the output would be typeset using 11-point size fonts and on A4-sized papers.

We also set the title and author of the document:

```
\title{My First Document}  
\author{John Doe}
```

The preamble is also where *extra packages are loaded* to provide extra or special functionalities. Let's say you want to include JPEG, PNG and/or PDF graphics in your document. Ask a  $\LaTeX$  expert how this can be done,

and the first thing (s)he will probably tell you is to “load the `graphicx` package”.  $\LaTeX$  packages are loaded with the `\usepackage` command, like so:

```
\usepackage{graphicx}
```

## 2.2.4 The Document Body

The document body is where your actual content goes. There are a few nifty commands that will automatically generate some contents for you. For instance, if you’ve specified details such as the title and order of your document, the `\maketitle` command will... well, does what it says: make the title. ;-)

Another useful command is `\tableofcontents`, which generates the table of contents. If you make changes to your document and the page numbers change, simply “compile” it again with `pdflatex`, and the TOC will be updated automatically. Just keep in mind that sometimes you may need to run `pdflatex` *twice* before the changes take effect.

As you may have noticed, you can divide and organise your document contents with sectioning commands like `\section` and `\subsection`.  $\LaTeX$  and friends automatically number the sectioning headings, as well as creating entries for them in the table of contents. In fact, if you load the `hyperref` package and run `pdf $\LaTeX$`  (try it!), you’ll get a PDF document with bookmarks made out of your section headings!

The default sectioning commands are (in order of hierarchy):

- `\part` (only available in report and book document classes)
- `\chapter` (only available in report and book document classes)
- `\section`
- `\subsection`
- `\subsubsection`

There is no `\subsubsubsection`. According to  $\LaTeX$  wisdom, you don’t need it: deep hierarchies are *baaaaaad*. If you think do, you probably need to reorganise your content. If you absolutely insist, you can settle for `\paragraph`, which will *not* be numbered, nor will it make it into the table of contents.



Another thing to notice is the use of blank lines to indicate paragraph breaks. In  $\LaTeX$ , new lines alone won't get you new paragraphs, and multiple blank lines won't get you any wider gaps between your paragraphs. In the same way, multiple space characters and tabs will just render as a single space. In other words, the two code snippets below will both come out the same way in the PDF.

```
1 This is just the beginning. There are lots of things
  you can achieve with \LaTeX{}, and I hope to write
  about them as much as I can. So much to do, so little
  time\ldots
2
3 Watch this space.
```

```
1 This is just the beginning. There are lots
2 of things you can achieve with \LaTeX{}, and I hope to
  write about them as much as I can.
3   So much to do, so little time\ldots
4
5
6 Watch this space.
```

If you think that's weird, remember that the philosophy behind  $\LaTeX$  is that you should care about your contents and organisation (*'I think I'd better break this paragraph here'*) more than how they are laid out exactly (*'I think I want 8pt inter-paragraph spacing after all'*).

In addition,  $\LaTeX$  takes the whole spacing business *very* seriously. It uses a lot of calculations to decide how much space to put between the letters, the words, the paragraphs... so that the end result is aesthetically pleasing to the reader's eye. In other words, it's not going to flinch at the extra space characters and blank lines in your input.

### 3 That's It! Go Forth and $\LaTeX$ !

That was a very cursory introduction to  $\LaTeX$ : I have barely scratched the surface of what  $\LaTeX$  is truly capable of. But I hope you had at least a first "feel" about using  $\LaTeX$ . Don't be afraid to continue to explore further on your own, with the help of a world-wide  $\LaTeX$  community.

To get you started, here are some useful links (amongst myriads oth-

ers):

- Getting to Grips with  $\text{\LaTeX}$ (online tutorial by Andrew Roberts):  
<http://www.andy-roberts.net/misc/latex/index.html>
- LaTeX Wikibook:  
<http://en.wikibooks.org/wiki/LaTeX>
- My own  $\text{\LaTeX}$  bag-of-links:  
<http://liantze.googlepages.com/latextypesetting>
- Online  $\text{\LaTeX}$  Previewer (uses Flash):  
<http://www.sciencesoft.at/flashlatex.jsp?lang=en>

Have Fun with  $\text{\LaTeX}$ !