Getting to Grips with LaTeX

Andrew Roberts

Chapter 1

Absolute beginners

This chapter is aimed at getting familar with the bare bones on LaTeX. We will begin with creating the actual source LaTeX file, and then take you through how to feed this through the LaTeX system to produce quality output, such as postscript or PDF.

1.1 The LaTeX source

The first thing you need to be aware of is that LaTeX uses a markup language in order to describe document structure and presentation. What LaTeX does is to convert your source text, combined with the markup, into a high quality document. For the purpose of analogy, web pages work in a similar way: the HTML is used to describe the document, but it is your browser that presents it in its full glory — with different colours, fonts, sizes, etc.

1.1.1 Hello World!

Ok, so let us begin by deciding what we will actually get LaTeX to produce. As I said, we will produce the absolute bare minimum that is needed in order to get some output, and so I think the well known 'Hello World! approach will be suitable here.

1. Open your favorite text-editor. If you use vim or emacs, they also have syntax highlighting that will help to write your files.
2. Reproduce the following text in your editor. This is the LaTeX source.

```
% hello.tex - Our first LaTeX example!

\documentclass{article}

\begin{document}

Hello World!

\end{document}
```

3. Save your file as '`hello.tex`'. (Without the quotes!)

1.1.2 What does it all mean?

`% hello.tex - Our first LaTeX example!` The first line is a *comment*. This is because it begins with the percent symbol (%), which when LaTeX sees it simply ignores the rest of the line. Comments are useful for humans to annotate parts of the source file. For example, you could put information about the author and the date, or whatever you wish.

`\documentclass{article}` This line tells LaTeX to use the article document class. A document class file defines the formatting, which in this case is a generic article format. The handy thing is that if you want change the appearance of your document, substitute article for another class file that exists.

`\begin{document}` An educated guess would tell you that this command alerts LaTeX that content of the document is about to commence. Anything above this command are known generally to belong in the preamble.

`Hello World!` This was the only actual line containing real content — the text that we wanted displayed on the page.

`\end{document}` Once again, this is not too difficult to understand. It tells LaTeX that the document source is complete.

You should also notice that each of the LaTeX commands begin with a backslash (\). This is LaTeX's way of knowing that whenever it sees a backslash, to expect some commands. Comments are not classed as a command, since all they tell LaTeX is to ignore the line. Comments never affect the output of the document.

Note, if you want to use the backslash or percent symbols within your text, you need to actually issue a command to tell LaTeX to draw the desired symbols, otherwise it will expect a command or a comment! The commands are:

Symbol	Command
%	`\%`
\	`\textbackslash`

1.2 Generating the document

It is clearly not going to be the most exciting document you have ever seen, but we want to see it nontheless. I am assuming that you are at a command prompt, already in the directory where `hello.tex` is stored.

1. Type the command: `latex hello` (the `.tex` extension is not required, although you can include it if you wish.)

2. Various bits of info about latex and its progress will be displayed. If all went well, the last two lines displayed in the console will be:

```
Output written on hello.dvi (1 page, 232 bytes).
Transcript written on hello.log.
```

This means that your source file has been processed and the resulting document is called hello.dvi, which takes up 1 page and 232 bytes of space.

Note, in this instance, due to the simplicity of the file, you only need to run the LaTeX command once. However, if you begin to create complex documents, including bibliographies and cross-references, etc., LaTeX needs to be executed multiple times to resolve the references. But this will be discussed in the future when it comes up.

1.3 Viewing the document

LaTeX has now done its job, so we can view the output. The default format is DVI (device independent), of which viewers exist freely for most platforms. However, the chances are that you would prefer to have a postscript file or PDF. Fortunately, there exist tools that can convert DVI to PS (and PDF) easily.

1.3.1 Converting to Postscript

Type the command: `dvips hello.dvi -o hello.ps`

`dvips` is the utility that actually performs the conversion. The first argument is the DVI file to be converted. The `-o` argument says that you want the output to be saved as a file. And the argument immediately after is the name you wish to call it. You could give it any name, but it makes sense to stick with hello, as well as giving it an informative `.ps` extension.

1.3.2 Converting to PDF

There are two easy routes to get a PDF:

1. Type the command: `dvipdf hello.dvi hello.pdf` (Note that there is no `-o` with this command, because although the utilities look almost identical, they have slighly differing syntax)
2. If you already have a postscript version, then type: `ps2pdf hello.ps hello.pdf`

Now it is simply a matter of using your preferred PS or PDF viewer to see the output. What you should see at the top left of the page are the words Hello World! and at the bottom is the current page number. All in a standard times font.

1.4 Summary

Ok, we've created possibly the simpliest possible document that LaTeX will produce (except for a blank page of course!) which is why it is not much to look at. However, now we have seen the basics, and how to actually use the LaTeX software, we can progress towards the more typical documents that you are likely to produce.

Chapter 2

Document structure

This chapter progresses significantly from the previous — very simplistic — chapter. The goal is to produce a fairly basic article, of similar style to what a research paper would resemble. To achieve this efficiently, we will focuse largely on document structure.

LaTeX practically forces you to declare structure within your documents. This is a good thing though. Because once LaTeX understands how you want your document organised, it will take care of all the tedious business of the layout and presentation for you. The separation of content and layout allows you to concentrate on the job at hand, i.e., communicating your ideas.

2.1 Preamble

If you recall from the previous tutorial, the preamble the everything from the start of the LaTeX source file until the `\begin{document}` command. It normally contains commands that affect the entire document.

```
% simple.tex - A simple article to illustrate document structure.

\documentclass{article}
\usepackage{times}

\begin{document}
```

The first line is a comment (as denoted by the `%` sign). The `\documentclass` command takes an argument, which in this case is `article`, because that's the type of document we want to produce. Other default classes that exist are `book`, `report`, `letter`, etc. It is also possible to create your own, as is often done by journal publishers, who simply provide you with their own *class* file, which tells LaTeX how to format your content. But we'll be happy with the standard article class for now!

`\usepackage` is an important command that tells Latex to utilise some external macros. In this instance, I specified `times` which means LaTeX will use the Postscript Times type 1 fonts, which look nicer :) And finally, the `\begin{document}`. This strictly isn't part of the preamble, but I'll put it here

anyway, as it implies the end of the preamble by nature of stating that the document is now starting.

2.2 Top Matter

At the beginning of most documents will be information about the document itself, such as the title and date, and also information about the authors, such as name, address, email etc. All of this type of information within Latex is collectively referred to as top matter. Although never explicitly specified, that is, there is no such `\topmatter` command, you are likely to encouter the term within Latex documentation.

An example:

```
\title{How to Structure a \LaTeX{} Document}
\author{Andrew Roberts\\
  School of Computing,\\
  University of Leeds,\\
  Leeds,\\
  United Kingdom,\\
  LS2 9JT\\
  \texttt{andyr@comp.leeds.ac.uk}}
\date{\today}
\maketitle
```

The `\title` command is fairly obivous. Simply put the title you want between the curly braces. `\author` would also seem easy, until you notice that I've crammed in all sorts of other information along with the name. This is merely a common, albeit, ungraceful hack, due to the default article class being a tad basic. If you are provided with a class file from a publisher, or if you use the AMS article class (amsart), then you have a more logical approach to entering author information. In the meantime, you can see how the new line command (`\\`) has been used so that I could produce my address. My email address is at the end, and the `\texttt` command formats the email address using a monospaced font. The `\date` command takes an argument to signify the date the document was written. I've used a built-in command called `\today` which, when processed by LaTeX, will be replaced with the current date. But you are free to put whatever you want as a date, in no set order. If braces are left empty, then the date is then omitted. Without `\maketitle`, the top matter would not appear in the document. So it is needed to commit your article attributes to paper.

2.3 Abstract

As most research papers have an abstract, then there is a predefined command for telling LaTeX which part of the content makes up the abstract. This should appear in its logical order, therefore, after the top matter, but before the main sections of the body.

```
\begin{abstract}
Your abstract goes here...
```

Command	Level
\part{*part*}	-1
\chapter{*chapter*}	0
\section{*section*}	1
\subsection{*subsection*}	2
\subsubsection{*subsubsection*}	3
\paragraph{*paragraph*}	4
\subparagraph{*subparagraph*}	5

Table 2.1: Possible section commands.**??**

```
...
\end{abstract}
```

2.4 Sectioning commands

The commands for inserting sections are fairly intuitive. Of course, certain commands are appropriate to different document classes. For example, a book has chapters but a article doesn't. Here is an edited version of some of the structure commonds in use.

```
\section{Introduction}
This section's content...

\section{Structure}
This section's content...

\subsection{Top Matter}
This subsection's content...

\subsubsection{Article Information}
This subsubsection's content...
```

As you can see, the commands are fairly intuitive. Notice that you do not need to specify section numbers. Latex will sort that out for you! Also, for sections, you do not need to markup which content belongs to a given block, using `\begin` and `\end` commands, for example.

Numbering of the sections is performed automatically by Latex, so don't bother adding them explicitly, just insert the heading you want between the curly braces. If you don't want sections number, then add an asterisk (`*`) after the section command, but before the first curly brace, e.g., `\section*{A Title Without Numbers}`.

2.5 The bibliography

Any good research paper will have a whole list of references. In this example document, I have included one. If you look at the PDF version, then after the first instance of 'Latex' in the introduction, you should notice a numbered reference. And at the end of the document, you can see the full reference.

Fortunately, Latex has a slightly more intelligent approach to managing your references than the average word processor, like MS Word for example, where everything has to be inputted manually (unless you purchase a 3rd party add-on). There are two ways to insert your references into Latex: the first is to store them in an external file and then link them via a command to your current document, or secondly, embed them within the document itself. In this chapter, I shall quickly cover the latter. Although, the former will be covered in depth in a future tutorial, as it is by far the most efficient and flexible.

There are two stages to setting up your biblography/references in a document. The first is to set up a bibliography environment, which is where you provide Latex with the details of the references. The second is the actual citation of your references within your document.

The following code was used in creating the bibliography environment for the example document in this tutorial. It is located immediately after the last line of the document content, but before the `\end{document}` command.

```
\begin{thebibliography}{9}
\bibitem{lamport94} Leslie Lamport, \emph{\LaTeX: A Document
Preparation System}.  Addison Wesley, Massachusetts, 2nd Edition,
1994.
\end{thebibliography}
```

Ok, so what is going on here? The first thing to notice is the establishment of the environment. `thebibliography` is a keyword that Latex recognises as everything between the `begin` and `end` tags as being data for the bibliography. The optional argument which I supplied after the `begin` statement is telling Latex how wide the item label will be when printed[1]. Note however, that it is not a literal parameter, i.e the number 9 in this case, but a text width. Therefore, I am effectively telling Latex that I will only need reference labels of one character in width, which means no more than nine references in total. If you want more than ten, then input a two-digit number, such as '99' which permits less than 100 references.

Next is the actual reference entry itself. This is prefixed with the \bibitem{*cite_key*} command. The *cite_key* is should be a unique identifier for that particular reference, and is often some sort of mnemonic consisting of any sequence of letters, numbers and punctuation symbols (although not a comma). I often use the surname of the first author, followed by the last two digits of the year (hence '`lamport94`'). If that author has produced more than one reference for a given year, then I add letters after, 'a', 'b', etc. But, you should whatever works for you. Everything after the key is the reference itself. You need to type it as you want it to be presented. I have put the different parts of the reference, such as author, title, etc., on different lines for readability. These line breaks are ignored by Latex. I wanted the title to be in italics, so I used the `\emph` command to achieve this.

To actually cite a given reference within your document is very easy. Goto the point where you want the citation to appear, and use the following: `\cite{cite_key}`, where the *cite_key* is that of the bibitem you wish to cite. When Latex processes the document, the citation will be cross-referenced with the bibitems

[1] An item label is simply the number that appears before the actual reference that allows you to cross reference it with the cited number within the document

and replaced with the appropriate number citation. The advantage here, once again, is that Latex looks after the numbering for you. If it was totally manual, then adding or removing a reference can be a real chore, as you would have to re-number all the citations by hand.

Of course, it may be your preference to use a different referencing system, such as Harvard, instead of the default numerical. This will be covered in the future, in the mean time, why not try to experiment with the `\package{Natbib}` package.

Appendix A

History of LaTeX

Some interesting stuff about LaTeX...