Appendix A: Introduction to LaTeX

This tutorial walks you through:

- how to pronounce LaTeX;
- compiling a LaTeX document;
- dealing with compiler errors;
- mathematical formulas; and
- formatting code.

This tutorial assumes that you already have LaTeX installed on your machine. If you don't, you need to install it[56]. Our UNIX lab machines already have LaTeX installed. Several students also report to me that Overleaf[57] is a pleasant alternative to installing LaTeX locally, but I have not tried it myself.

[56] https://www.latex-project.org/get/

[57] https://www.overleaf.com/

Pronouncing LaTeX

LaTeX is pronounced *LAH-tekh*. The "TeX" part consists of the Greek characters, tau, epsilon, and chi, and is a reference to the Greek word, *technē*, meaning "art" or "craft," which is the root of the word *technical*. TeX was written by the computer scientist Donald Knuth. The "La" part comes from Leslie Lamport, who was the person who packaged up TeX with a handy set of macros to make writing easier.[58]

[58] Both Knuth and Lamport have contributed extensively to the field of computer science, and have been subsequently awarded Turing Awards for their work. To be clear, though, those Turing Awards were *not* for TeX/LaTeX.

Compiling a LaTeX document

Suppose you have a LaTeX file called `file.tex`. You can compile `file.tex` to a PDF like so:

```
$ pdflatex file.tex
```

`pdflatex` will print out a lot of stuff. If your `file.tex` has no errors, you should see output that looks something like the following,

```
Output written on file.pdf (1 page, 27256 bytes).
```

and you will have a PDF version of your LaTeX document.

Dealing with compiler errors

If things go wrong, you will need to debug your LaTeX, just as you debug any other computer program. Look for missing brackets (e.g., `[` and `]`) or curly braces (e.g., `{` or `}`), look for typos in commands, and above all, read the output that the compiler is printing. LaTeX error messages are not the most understandable things, but they usually *do* tell you where to find the error.

One important thing to be aware of is that the LaTeX compiler usually drops you into an interactive shell instead of just stopping (like `javac` does). For example,

```
$ pdflatex file.tex
 ... lots of output ...
! Undefined control sequence.
<recently read> \tod

l.19   \item \tod
                {ANSWER}
?
```

Annoyingly, your cursor sits on this line and doesn't give you even a *clue* as to what to do.[59] The correct thing to do is easy: type a capital `X` and press the `ENTER` key.

More importantly, this compiler message actually tells me where to find the problem. On *line 19* (`l.19`), something about the LaTeX commands `\item \tod` causes an `Undefined control sequence`. One might argue that this is not a very good error message, but hey, all the information you need is there.

When I open up my `file.tex` I see the following on line 19:

```
\item \tod{ANSWER}
```

Oops! `\tod` should be `\todo`. After fixing this and running `pdflatex`

[59] By the way, I used LaTeX for years without actually knowing the correct way to escape from this situation. I am not ashamed to admit that I actually used to call the `kill` command to forcibly shutdown LaTeX.

again, `file.tex` again compiles correctly.

Mathematical formulas

One of LaTeX's great strengths is that it can produce beautiful-looking mathematical formulae. This is mostly easy to do, too: just put a mathematical expression in between a pair of dollar signs. For example:

```
$x + 5$
```

will produce a pretty-looking $x + 5$ expression.

I have not yet come across a mathematical figure that I could not reproduce using LaTeX. That's not to say that it is always easy, just that it is *possible*. See the LaTeX/Mathematics[60] reference for more information.

[60] https://en.wikibooks.org/wiki/LaTeX/Mathematics

Formatting code

Code is sometimes frustrating to format in LaTeX. Fortunately, there are two approaches, one easy, and one... less easy. Let's look at the easy one first.

LaTeX has many special "environments" that you can use for specially-formatted blocks of text. A very useful one for code is called `verbatim`. You use it like this:

```
\begin{verbatim}
  code goes here
\end{verbatim}
```

For example,

```
\begin{verbatim}
  public static void main(String[] args) {
    System.out.println("Hello world!");
  }
\end{verbatim}
```

`verbatim` will reproduce the text using a monospaced "typewriter" font that looks a lot like how we normally format code.

But LaTeX actually lets you do some very fancy things with code. Instead of using the `verbatim` environment, you can alternatively use the `lstlisting` environment. You need to do a little extra work:

1. You must add `\usepackage{listings}` to the top of your document.
2. You must supply a `\lstset` command, e.g., `\lstset{language=Pascal}` for the Pascal language.
3. You can then put your code inside a `lstlisting` environment.

The `listings` package can produce genuinely beautiful looking formatted code, with syntax highlighting and everything. If you're interested, see the LaTeX/Source Code Listings[61] page.

[61] https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings