

L^AT_EX Example

David Dervishi

February 15, 2022

Contents

1	Introduction	1
1.1	Writing L ^A T _E X	2
2	Minimal Example	2
3	Structuring Your Document	3
4	Packages And More On Commands	3
5	Listing Things	4
5.1	Itemize	4
5.2	Enumerate	5
6	Text Formatting	6
7	Math	6
7.1	Useful Commands	7
7.2	Useful Environments	8
8	Labels And References	9
9	Centering Things	10
10	Tables	10
11	Images And Figure Environments	11
12	Bibliography	12

1 Introduction

Hi! This is a small L^AT_EX example that can serve as an introduction to some common features of L^AT_EX. I encourage you to look at the L^AT_EX source of this document, as it is an example that you may find useful, and it contains some comments and features that are not explained in this text (like code snippets or hyperlinks).

1.1 Writing L^AT_EX

If you want to work online, then you can use Overleaf¹, which is frequently used and free software, but requires an account. If you want to work on your machine, you can use a specialized editor like TexStudio², or any other text editor (many of them have a L^AT_EX plugin). In both cases you need to install a L^AT_EX distribution, including some packages. For example, on Debian-based systems, this should get you everything you could ever need

```
apt install texlive-full
```

and this should get you most of what you could ever need on Arch-based systems

```
pacman -S texlive-most
```

2 Minimal Example

A simple L^AT_EX document might look like this:

Listing 1: Simple L^AT_EX document

```
1 \documentclass{article}
2
3 \title{Your Nice Title}
4 \author{Your Name \and Name Of Teammate}
5
6 \begin{document}
7     \maketitle
8
9     % interesting stuff (anything after a '%' is a comment)
10 \end{document}
```

Listing 1 already defines the basic structure of a document. You can see on the first line that we are defining the *class of document* that we will be writing. In this case, we are using the **article** class, which is probably the most common one. It is also the one that I used to write the document you are reading. There are of course other document classes which you may use depending on what you want to do, including **beamer** for slideshows or **book** for documents with chapters.

Anything that starts with a `\` is a *command*. The next commands that we use are `\title` and `\author`, which define the title and authors of your document. If you are not working alone, you can easily specify multiple authors by separating your names with the `\and` command.

Everything up to here is what we call the *preamble* of a document. This is the place where you define document-wide commands, packages and options (take a look at the source of this document for some examples!). Coming next is the interesting part of anything you will write, namely the content. We are creating an *environment* using the `\begin` and `\end` commands. You'll be creating all sorts of interesting environments, but the specific one we are creating here is **document**, which will (quite obviously) contain your document. We start by displaying the title, authors and date of the document using the `\maketitle` command. Note that I am indenting the contents of the environment: this is not mandatory, but quite useful if you want to be able to read your code.

On to the more interesting stuff!

¹<https://www.overleaf.com>

²<https://texstudio.org>. By the way, if you think footnotes are cool, you can create them with `\footnote`

3 Structuring Your Document

Pretty much any document ends up being split into different parts. In the `article` class, you can split your text into sections, subsections, subsubsections and paragraphs.

Listing 2: Basic text structure

```
1 \section{This is a section}
2     You can write text in a section.
3
4     \subsection{This is a subsection}
5         You can also write text in a subsection.
6
7         \subsubsection{This is a subsubsection}
8             You can even write text in a subsubsection.
9
10             Putting a blank line between two blocks of
11             text makes them different paragraphs.
12
13             \paragraph{You can also write paragraphs with a title}
14                 Paragraphs can then have contents.
```

Note that ((sub-)sub-)sections will be automatically numbered by default, like in this document. If you don't want this, you can disable the numbering of a section like in Listing 3. Note that unnumbered sections will not appear in a table of contents by default.

Listing 3: Unnumbered sections

```
1 \section*{Section without number}
2     \subsection*{Subsection without number}
3         \subsubsection*{Subsubsection without number}
```

4 Packages And More On Commands

L^AT_EX has plenty of nice features by default, but some common things you may want to do are easier with or require external packages to work. We'll look at how to do that now, since the following sections will sometimes make use of external packages.

Packages are specified at the start of your file, in the preamble of your document, using the `\usepackage` command.

Listing 4: `\usepackage` example

```
1 \documentclass{article}
2 % ...
3 \usepackage{packagename}
4 %...
5 \begin{document}
6     %...
7 \end{document}
```

Sometimes, packages can be configured directly on importation. To do this, the `\usepackage` command takes this configuration as an *optional parameter*. Any command may take optional or mandatory parameters: optional ones are specified in square brackets, mandatory ones in

curly ones. The details depend on the command, and you can find plenty of documentation online. Note that environments can also use optional parameters.

Listing 5: Command parameter examples

```

1 \command[optional]{mandatory}
2 \usepackage{enumitem} % mandatory package name
3 \usepackage[dvipsnames]{xcolor} % optional parameter
4 \begin{environment}[optional parameters]
5     %...
6 \end{environment}

```

5 Listing Things

Whatever you are writing about, you will very often end up listing things at some point. There are two main ways to make lists in L^AT_EX.

5.1 Itemize

The `itemize` environment lets you create unordered lists of items. You can put pretty much anything in this environment, meaning you can also create nested lists as in Listing 6.

Listing 6: `itemize` example

```

1 \begin{itemize}
2     \item
3         Item 1
4     \item
5         Item 2
6         \begin{itemize}
7             \item
8                 Nested item 1
9             \item
10                Nested item 2
11         \end{itemize}
12     \item
13         Item 3
14 \end{itemize}

```

This will render to the following list:

- Item 1
- Item 2
 - Nested item 1
 - Nested item 2
- Item 3

If you don't like the symbols used to generate the list, you can modify them using the `enumitem` package. Listing 7 is then rendered as the following list:

- / Item 1

/ Item 2

Listing 7: `itemize` labels

```
1 \usepackage{enumitem}
2 %...
3 \begin{itemize}[label = /]
4   \item
5     Item 1
6   \item
7     Item 2
8 \end{itemize}
```

This is usually more useful with ordered lists.

5.2 Enumerate

The `enumerate` environment lets you produce ordered lists of items. Its usage is similar to that of `itemize`.

Listing 8: Nested `enumerate`

```
1 \begin{enumerate}
2   \item
3     Item 1
4   \item
5     Item 2
6     \begin{enumerate}
7       \item
8         Nested item 1
9       \item
10        Nested item 2
11     \end{enumerate}
12 \end{enumerate}
```

Listing 8 renders as follows:

1. Item 1
2. Item 2
 - (a) Nested item 1
 - (b) Nested item 2

You can also change enumeration labels as in `itemize`, but doing so is more subtle, as you cannot simply specify any character: you have to specify a *sequence* of labels, and the `enumitem` package is here to help. The code of the following enumeration is given in Listing 9. Isn't that nice?

- i. Item 1
 - a. Nested item 1.1
 - b. Nested item 1.2

- ii. Item 2
 - 1) Nested item 2.1
 - 2) Nested item 2.2

Listing 9: `enumitem` labels

```

1 \usepackage{enumitem}
2 %...
3 \begin{enumerate}[label = \roman*.]
4   \item
5     Item 1
6     \begin{enumerate}[label = \alph*.]
7       \item
8         Nested item 1.1
9       \item
10        Nested item 1.2
11     \end{enumerate}
12   \item
13     Item 2
14     \begin{enumerate}[label = \arabic*.]
15       \item
16         Nested item 2.1
17       \item
18         Nested item 2.2
19     \end{enumerate}
20 \end{enumerate}

```

6 Text Formatting

There are also commands for text formatting.

- `\textit` renders its argument in *italics*.
- `\textbf` renders its argument in **bold font**.
- `\texttt` renders its argument in **typewriter font**.
- `\textsf` renders its argument in **sans serif font**.
- `\underline` underlines its argument.

7 Math

One of the nice things about L^AT_EX is the math typesetting. You can already do a number of interesting things without packages, but you will pretty much always end up using the `amsfonts`, `amsmath` and `amssymb` packages.

```

\usepackage{amsfonts}
\usepackage{amsmath}
\usepackage{amssymb}

```

To open an inline math environment, just use a pair of dollars and write anything you need in between. For example, `$ax^2 + bx + c = 0$` renders as $ax^2 + bx + c = 0$. You can also write equations using the `equation` environment. Listing 10 renders as the following:

$$ax^2 + bx + c = 0 \tag{1}$$

Listing 10: Simple equation

```
1 \begin{equation}
2     ax^2 + bx + c = 0
3 \end{equation}
```

Note that equations are numbered by default, like sections. If you do not want that to happen, you can simply use the `equation*` environment instead.

There are many, *many* things you can do in a math environment, and I cannot list all of them, but here are a couple of useful commands and environments.

7.1 Useful Commands

- `\mathbb` with some string as argument: $\mathbb{Z}\mathbb{Q}\mathbb{R}\mathbb{C}$. `\mathfrak` does the same with fraktur: $\mathfrak{A}\mathfrak{B}\mathfrak{C}\mathfrak{D}$. `\overline` puts a bar on top of its argument: \overline{abc} . `\hat` puts a hat on top of a letter: \hat{x} . Note that you can also use math versions of `\textbf` and others: `\mathbf` is **abc**.
- For Greek letters, use `\letter` for the normal version, `\Letter` for the capital version: sigma is σ, Σ , delta is δ, Δ .
- Infinity is `\infty`, plus or minus is `\pm`, a left arrow is `\leftarrow`, a long right arrow is `\longrightarrow`, greater or equal is `\geq`, lower or equal is `\leq`, not equal is `\neq`, a subset is `\subset`, inclusion in a set is `\in`.
- Fractions are defined with `\frac{numerator}{denominator}`: $\frac{3}{4}, \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Note that fractions are often more readable in an equation environment:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Any sort of bracket (square, curly, ...) can adapt to the size of its content using `\left` and `\right`.

With simple `()`:

$$\left(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right)$$

With `\left(` and `\right)`:

$$\left(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right)$$

Much better.

- Exponents can be created with a circumflex accent, indices with an underscore. Note that these only take one character into account by default, and taking more characters requires you to put them into curly brackets. Observe the difference between these:

$$\begin{array}{ll} \text{x_i}^2 & x_i^2 \\ \text{x_}\{i^2\} & x_{i^2} \\ \text{x_i}^23 & x_i^23 \\ \text{x_i}^{\{23\}} & x_i^{23} \end{array}$$

- Math functions like sine, cosine and such have their own operators: `\sin(x)` is $\sin(x)$. Note that the `\sin` command (and others) does not take arguments, because some of us like to write $\sin x$ instead. Other commands like `\sqrt` do take an argument, because the square root must know what it should contain.
- Sums (`\sum`), products (`\prod`), limits (`\lim\limits`), integrals (`\int`) and such like to have bounds written over or under them. You can do this with simple exponent and index notation, e.g. `\sum_{i = 0}^n a_i` is

$$\sum_{i=0}^n a_i$$

Note that this cannot be rendered in the same way if you write inline math (with dollars): $\sum_{i=0}^n a_i$.

7.2 Useful Environments

- The `align` environment lets you align multiple equations, i.e. make sure they vertically align at a specified position. Imagine having to compute the derivative of f defined by $f(x) = 2x$. You may want to use multiple `equation` environments for that, like in the following:

$$\begin{aligned} \frac{df}{dx}(a) &= \lim_{x \rightarrow a} \frac{x^2 - a^2}{x - a} \\ &= \lim_{x \rightarrow a} \frac{(x - a)(x + a)}{x - a} \\ &= \lim_{x \rightarrow a} x + a \\ &= 2a, \end{aligned}$$

which is horrible. Luckily, you can use the `align` environment to specify that the lines must be aligned according to the equality symbol they contain. You can use an ampersand (`&`) inside the environment to do this, as in Listing 11.

Listing 11: `align` example

```
1 \begin{align*}
2   \frac{df}{dx}(a) &= \lim\limits_{x \rightarrow a} \dots \\
3   &= \lim\limits_{x \rightarrow a} \frac{(x - a)(x + a)}{x - a} \\
4   &\dots \\
5   &= 2a \\
6   &\% note the \\ at the end of each line (except for the last) \\
7 \end{align*}
```


$$\begin{aligned}
\frac{df}{dx}(a) &= \lim_{x \rightarrow a} \frac{x^2 - a^2}{x - a} \\
&= \lim_{x \rightarrow a} \frac{(x - a)(x + a)}{x - a} \\
&= \lim_{x \rightarrow a} x + a \\
&= 2a
\end{aligned}$$

Much better. You can also use multiple & per line if you want to align more than one thing.

- The `cases` environment is similar to `align`, but is used when you want to distinguish multiple cases in an equation. The following is shown in Listing 12. Note that the `cases` environment must itself be contained in a math environment, unlike `align`.

$$\max(a, b) = \begin{cases} a, & a > b \\ b, & a \leq b \end{cases}$$

Listing 12: `cases` example

```

1 \begin{equation*}
2   \max(a, b) =
3   \begin{cases}
4     a, & a > b \\
5     b, & a \leq b
6   \end{cases}
7 \end{equation*}

```

8 Labels And References

Noticed how this document is full of references to listings, and how the numbers always match? This is automatically managed by labels and references. You can define a label using the `\label` command, which takes a name as argument. A common practice is to have a label name of the form `kind-of-thing:name-of-thing`, so a label inside an equation would look like `eq:cauchy-riemann`, or `equation:amdahl` (just make sure you stay consistent), while a listing might look like `lst:align`. This assigns a number to the label you defined, and using the command `\ref` with a label name gives you back the number that was assigned. For example, labeling a section and referencing it would give you the section number. If you do the same with a listing, it would give you the listing number, and the same with an equation.

As an example, I am going to label and refer to this section as follows:

Listing 13: `label` example

```

1 \label{sec:labels-references}
2 This is Section \ref{sec:labels-references}

```

This is Section 8.

If you don't want to type the kind of thing you are referring to every time you use a reference, you can use the `cleveref` package instead:

```

1 \usepackage{cleveref}
2 %...
3 This is \Cref{sec:labels-references}

```

This is Section 8.

Note that some environments take a label name as optional argument, so you may not always have to use `\label` directly. Take a look at this document's source for examples.

9 Centering Things

As we will shortly discuss tables and figures, you may want to know how to horizontally center objects within a page, be it text, images or anything else. \LaTeX provides the `\centering` command and the `center` environment, which essentially do the same thing, except that `center` adds some vertical space that you may find convenient.

Listing 14: `\centering` and `center`

```

1 % Notice the brackets! They tell \centering where to stop.
2 % Without them you may end up centering too many things.
3 % Also note that the text is terminated by \\ (or an empty line)
4 {\centering
5   Here is a centering example. \\
6 }
7
8 \begin{center}
9   And here is a center example.
10 \end{center}

```

Here is a centering example.

And here is a center example.

10 Tables

To create a table, you first have to determine how many columns it will contain and how text will be aligned within them.

Listing 15: Basic table structure

```

1 \begin{table}[options]
2   \centering
3   \begin{tabular}{column specification}
4     %...
5   \end{tabular}
6 \end{table}

```

Note that the columns specification is mandatory. Every column is specified by a letter: `l` is for left-alignment, `c` for centering, `r` for right-alignment. `crr` would thus declare three columns, the first one being horizontally centered and the two other ones being aligned to the right. Columns can also be separated by vertical lines, which are specified with a vertical bar (`'—'`). The specification `|c|rr|` declares three columns with the same alignments as before, but now the table has vertical bars to the right and left, as well as a vertical separation between the first and second columns.

The optional `table` parameters define the position of the table with respect to the text. To make it (mostly) attached to the previous paragraph, you can use `ht`. Otherwise, \LaTeX is allowed to place your tables at whatever location it determines is best. You'll observe that \LaTeX has a lot of freedom to when it comes to placing objects.

The content of the table is specified in a way similar to the `align` environment: you can use ampersands to tell \LaTeX where the cells of your table end. Lines are also terminated by `\\`. An example is given in Listing 16. Note the addition of `\hline`: this command adds a horizontal line above the current line, which is quite useful depending on the table format you want to have.

Listing 16: Table content example

```
1 \hline first cell , first line & second cell , first line \\
2 \hline first cell , second line & second cell , second line \\
3 %...
```

A complete example is given in Table 1. You can note that lines 3, 4 and 5 do not declare a `hline`, and that you do not have to put content in cells, as in the last line.

Table 1: `table` example

Item	Cost	Quantity	Total
Apple	4	3	12
Broccoli	8	2	16
Coriander	5	3	15
Dill	2	4	8
			51

11 Images And Figure Environments

To add images to your document, you should use the `graphicx` package:

```
\usepackage{graphicx}
```

The easiest way to include an image is to directly use the `\includegraphics` package as follows:

```
\includegraphics[options]{path/to/image}
```

The optional arguments control for example whether the image should be resized, cropped or rotated. The path may be absolute or relative. It is common practice to put all of your images inside one or multiple predefined directories like `images/` or `res/images/`. In that case, typing the base directory of every image in your document is annoying, and you can tell `graphicx` where it should look for images using `\graphicspath`:

```
\graphicspath{{path/to/images1/} {path/to/images2/}}
```

Note that your directories must be separately enclosed by curly braces, and they must end with a `/`. Check out the source for an example!

Usually, simply adding an image is not sufficient for your needs: you want captions, labels and some control over the location of your image. Worry not, for the `figure` environment is here to help. It can be used as in Listing 17. The position of the `\caption` command determines whether the caption is positioned above or below the image.

Listing 17: `figure` example

```

1 \begin{figure}[ht]
2   \label{img:example}
3   \centering
4   \includegraphics[options]{path/to/image}
5   \caption{Caption example}
6 \end{figure}

```

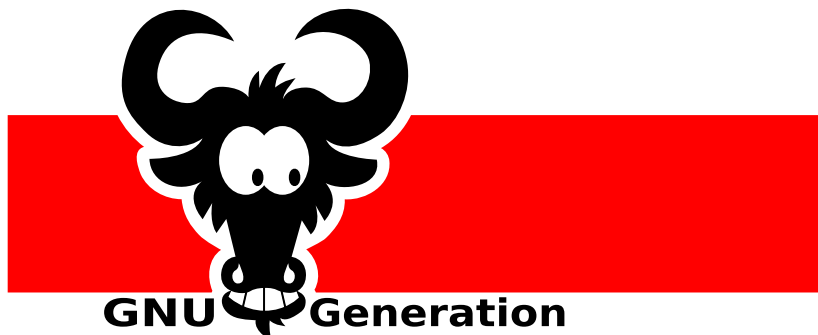


Figure 1: Image example

Sometimes, you want your text to wrap around your image instead of being separated. Without surprise, there is a package for that:

```
\usepackage{wrapfig}
```

The `wrapfigure` environment works mostly like `figure`, but you have to give it two mandatory arguments instead of optional ones: the image position with respect to the text, and the width of the figure. \LaTeX offers many ways to specify lengths, but I think the most useful one is the `\textwidth` command. If you want something to use half of the space available for text, you can give it a width of `0.5\textwidth`. The position argument is specified with one letter, the most commonly used ones being `r` and `l` for right and left, respectively. You can also use capital letters if you want to let \LaTeX decide where to put the figure.

Listing 18: `wrapfigure` environment

```

1 \begin{wrapfigure}{position}{image width}
2   %...
3 \end{wrapfigure}

```

12 Bibliography

Last but not least, you may want to list your sources in a pretty bibliography using `biblatex`:

```

\usepackage[
  % not necessary, but recommended (you need to install biber)
  backend = biber
]{biblatex}

```

You can then specify your sources in a separate file, which you must declare using the `\addbibresource` command:

```
\addbibresource{path/to/sources.bib}
```

The general format to specify a source is as follows:

Listing 19: **biblatex** source format

```
1 @<source-type>{label ,
2   % example fields
3   title = "Nice Title",
4   author = "LastName1, FirstName1 and Last2, First2 and Last3, First3",
5   url = "www.example.com"
6   % other "field = value" pairs
7 }
```

There are many types of sources and fields, many of which you can easily find online, for example in [this neat cheat sheet](#).

Once you have your sources, you can cite them in your document using the `\cite` command: Something something as showed in `\cite{label}`.

Where `label` is the name you gave to your source in the `.bib` file. You can then display your bibliography at the end of your document using the `\printbibliography` command. **biblatex** also lets you change the style of your citations or bibliography.

License

Copyright © 2022 David Dervishi. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found under <https://www.gnu.org/licenses/fdl-1.3.html>.