

L^AT_EX for Engineers: T_EX to PDF

Run to the hills, run for your lives

Lamprinos Chatziioannou

[2026-09-08 Tue]

Contents

1	Roadmap	1
2	My L ^A T _E X experience	2
3	Why would I encourage you to give L ^A T _E X a shot?	2
4	Understanding L ^A T _E X	3
4.1	Understanding L ^A T _E X	3
4.2	Working Environment	4
5	Testing your environment	7
6	Dissecting the sample document	8
7	Minimizing work: Your preamble	9
8	Bibliography	12
9	Going forward	12
9.1	Some generic tips:	13
9.2	On templates	14
10	Further reading	15

1 Roadmap

In this guide

1. **The Case for L^AT_EX:** Why plain-text typesetting outperforms word processors for engineering coursework.
2. **Environment Setup:** Getting started with a L^AT_EX distribution and configuring your editor.
3. **Core Foundations:** Compiling a minimal document and understanding the separation of preamble and content.
4. **Workflow Optimization:** Automating continuous compilation with `latexmk` and citation management with `biblatex` + Zotero.
5. **Project Architecture:** Structuring scalable, multi-file documents with a reusable preamble.
6. **Templates** and my `Thesis.cls`. A ready-to-use template for your thesis, with a document to get you started.

2 My $\text{\LaTeX}$ experience

It's already been 2 years since I started writing my thesis. Since then, I've successfully submitted and defended it and gotten my degree. I first got into $\text{\LaTeX}$ by reading the wonderfully written and, from my point of view, immaculate articles of [Gilles Castel](#). I had been using `vim` for the better part of 5 years back then, so his workflow did not require much tinkering on my part. It was something I could easily replicate on my own system. The school of engineering quickly helped me realize that, while taking notes on my computer was a piece of cake back in those COVID days, it was not optimal for me: in class I was not able to quickly capture sketches (without focusing too much on those and sacrificing understanding). I turned to a more conservative solution of handwritten notes which I could easily then integrate with my much simpler [org-mode](#). Still, $\text{\LaTeX}$ knowledge and experimentation is what allowed me to have consistently high-quality reports, without ever (except for one, very particular course) having to force myself through the tedious process of creating a presentable office doc.

3 Why would I encourage you to give $\text{\LaTeX}$ a shot?

First, two easy examples for you:

- This blog post was **automatically** turned to [this](#) using my $\text{\LaTeX}$ preamble
- This [sample thesis PDF](#) was generated with **less than 40 lines** of $\text{\LaTeX}$

At first, let's try to only focus on the facts:

1. **Reusability:** Since you're writing in plain text you can easily share snippets across your works without having to manually adjust preferences such as fonts, colors and generic structure. I'll build on this in [the preamble section](#).
2. **Version control:** You can use `git` and have it all nicely tracked. No more files named `report.doc`, `report1.doc`, `report1.final.doc`, `report1.final.final.doc`¹ (and you can also upload them on GitHub and include them on your [vanity page](#))
3. **Predictability:** This one is counterintuitive but $\text{\LaTeX}$ is ridiculously simple for 99% of the time (statistics source: my imagination). It is difficult to properly set up, but after that, it is deceptively easy.
4. **Maths and Equations (and Bibliography):** If you need maths in a report, don't even try to use Word. Just use $\text{\LaTeX}$.
5. **Beauty and Consistency:** I do not need to build on this, but all my reports look consistently beautiful. There is something so nice about having a *signature* in your work (without even trying).

Seriously, I do not see a good reason why not. But, once again, you might not be convinced. I know I would need something more tangible. I suggest that you take a look at the following [image 1](#). These are the first pages of different projects I undertook while an undergraduate. Notice the similarities among all of those and be aware that I **never** worked manually on creating the table of contents (toc) nor did I mess around with the placement of the title... $\text{\LaTeX}$ handled most of this stuff for me.

¹Unless you ... <https://xkcd.com/1597/>

Eternal War In Memory		HW1: Term Project		HW2 Course Project	
Augustus Vasilakos		Report on the implementation of a Verilog Model for the RISC-V Processor		Lamprinos Chatziioannou	
Karyofyllis Ekdipos 2023-2024		Lamprinos Chatziioannou, 10474		June 16, 2024	
January 21, 2025					
Contents		Contents		Contents	
1	1	1	1	1	1
2	2	2	2	2	2
3	3	3	3	3	3
4	4	4	4	4	4
5	5	5	5	5	5
6	6	6	6	6	6
7	7	7	7	7	7
8	8	8	8	8	8
9	9	9	9	9	9
10	10	10	10	10	10
11	11	11	11	11	11
12	12	12	12	12	12
13	13	13	13	13	13
14	14	14	14	14	14
15	15	15	15	15	15
16	16	16	16	16	16
17	17	17	17	17	17
18	18	18	18	18	18
19	19	19	19	19	19
20	20	20	20	20	20
21	21	21	21	21	21
22	22	22	22	22	22
23	23	23	23	23	23
24	24	24	24	24	24
25	25	25	25	25	25
26	26	26	26	26	26
27	27	27	27	27	27
28	28	28	28	28	28
29	29	29	29	29	29
30	30	30	30	30	30
31	31	31	31	31	31
32	32	32	32	32	32
33	33	33	33	33	33
34	34	34	34	34	34
35	35	35	35	35	35
36	36	36	36	36	36
37	37	37	37	37	37
38	38	38	38	38	38
39	39	39	39	39	39
40	40	40	40	40	40
41	41	41	41	41	41
42	42	42	42	42	42
43	43	43	43	43	43
44	44	44	44	44	44
45	45	45	45	45	45
46	46	46	46	46	46
47	47	47	47	47	47
48	48	48	48	48	48
49	49	49	49	49	49
50	50	50	50	50	50
51	51	51	51	51	51
52	52	52	52	52	52
53	53	53	53	53	53
54	54	54	54	54	54
55	55	55	55	55	55
56	56	56	56	56	56
57	57	57	57	57	57
58	58	58	58	58	58
59	59	59	59	59	59
60	60	60	60	60	60
61	61	61	61	61	61
62	62	62	62	62	62
63	63	63	63	63	63
64	64	64	64	64	64
65	65	65	65	65	65
66	66	66	66	66	66
67	67	67	67	67	67
68	68	68	68	68	68
69	69	69	69	69	69
70	70	70	70	70	70
71	71	71	71	71	71
72	72	72	72	72	72
73	73	73	73	73	73
74	74	74	74	74	74
75	75	75	75	75	75
76	76	76	76	76	76
77	77	77	77	77	77
78	78	78	78	78	78
79	79	79	79	79	79
80	80	80	80	80	80
81	81	81	81	81	81
82	82	82	82	82	82
83	83	83	83	83	83
84	84	84	84	84	84
85	85	85	85	85	85
86	86	86	86	86	86
87	87	87	87	87	87
88	88	88	88	88	88
89	89	89	89	89	89
90	90	90	90	90	90
91	91	91	91	91	91
92	92	92	92	92	92
93	93	93	93	93	93
94	94	94	94	94	94
95	95	95	95	95	95
96	96	96	96	96	96
97	97	97	97	97	97
98	98	98	98	98	98
99	99	99	99	99	99
100	100	100	100	100	100

Figure 1: Reusing the same preamble (with slight modifications)

If you're still not convinced, well, take a look at the following repos I have published on GitHub. I am not particularly proud of the work behind the scenes (I don't even remember most of it), but you can see $\text{\LaTeX}$ work in action: the reports look consistently high-quality.

- [MicroProcessors notes](#) (useful for the exams)
- [Optimization Techniques repo](#)
- [Algorithm Design and Analysis repo](#)

4 Understanding $\text{\LaTeX}$

Disclaimer 4.1

This is not intended to be a full introduction to $\text{\LaTeX}$. You don't need to know anything beforehand, but you won't become a master either. Hopefully, you'll have enough information to let you focus on your thesis, without getting in greater depth than necessary.

An interesting bit from Wikipedia: [on pronunciation](#)

Quote 4.1

The characters 'T', 'E', and 'X' in the name come from the Greek capital letters tau, epsilon, and chi, as the name of $\text{\TeX}$ derives from the Ancient Greek: τέχνη ('skill', 'art', 'technique'); for this reason, $\text{\TeX}$'s creator Donald Knuth promotes its pronunciation as *tex* (tek).

4.1 Understanding $\text{\LaTeX}$

$\text{\LaTeX}$ is an extension of $\text{\TeX}$, essentially its successor (though there are others claiming that place as well). It is a much friendlier format to work with, since it includes a satisfactory collection of macros and is, at the same time, further extended by a wide range of packages, that are incompatible with plain $\text{\TeX}$.

One creates files following these formats, but these are plain text files, usually with the `.tex` extension. Much like `.c` or `.cpp` files, the author usually wants to compile them to unlock their functionality. Compilers, in the context of $\text{\TeX}$ are called engines, and there are multiple different of those: Xe $\text{\TeX}$, Lua $\text{\TeX}$ and Pdf $\text{\TeX}$, to name a few. Usually these engines have their $\text{\LaTeX}$ equivalent as well, as shown in the graph below.

Engines receive our plain-text formatted files, and produce output, usually in the form of PDF documents. Getting this terminology right might seem tedious but helps better understand the process later on.

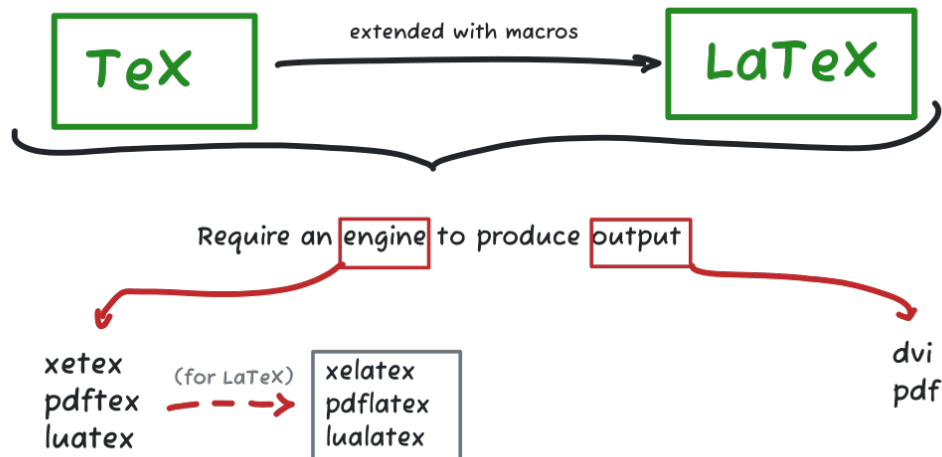


Figure 2: T_EX and L^AT_EX: Formats, Engines, Outputs

4.2 Working Environment

One can work with L^AT_EX in many different environments. Just like most *applications* nowadays, there is the option of working in the cloud, a route which will almost surely lead you to [Overleaf](#), but I prefer working locally.

To work with L^AT_EX locally one needs two things:

- An editor
- A T_EX Distribution

4.2.1 The editor

I am not a zealot. I suggest you find an editor that works for you and use just that. If you're comfortable with vi keybindings, then nvim is a powerhouse for LaTeX purposes, but VSCode must also be pretty good, according to some friends of mine. Do not hassle over it, you could edit the files with Notepad and still have no problem.

Nvim This bit might be helpful if you use nvim:

```

1  -- Vimtex configuration, responsible for:
2  -- 1) Setting server name - so that inverse search works
3  -- 2) Setting up vimtex
4  --
5  -- Sources:
6  -- 1) https://jdhao.github.io/2021/02/20/inverse_search_setup_neovim_vimtex/
7  -- 2) https://jdhao.github.io/2019/03/26/nvim_latex_write_preview/
8  local function set_server_name()
9      -- This only executes on windows.
10     local servername = vim.v.servername

```

```

11     local filename = "nvimsocket"
12     local path
13
14     path = vim.env.TEMP .. "\\ " .. filename
15     vim.g.vimtex_view_general_viewer = "SumatraPDF"
16     vim.g.vimtex_view_general_options =
17         ↪ "-reuse-instance -forward-search @tex @line @pdf"
18
19     local cmd = string.format('echo %s > "%s"', servername, path)
20     vim.fn.system(cmd)
21 end
22
23 return {
24     {
25         "lervag/vimtex",
26         ft = "tex", -- Load only when editing .tex files
27         init = function()
28             vim.g.vimtex_syntax_enabled = 0
29             vim.g.vimtex_compiler_latexmk = {
30                 aux_dir = "./aux",
31                 options = {
32                     "-interaction=nonstopmode",
33                     "-synctex=1",
34                 },
35             }
36             vim.g.vimtex_compiler_latexmk_engines = {
37                 _ = "-xelatex",
38             }
39             if vim.fn.has("win32") == 1 then
40                 set_server_name()
41             else
42                 vim.g.vimtex_view_method = "zathura"
43             end
44         end,
45     },
46 }

```

VSCode A friend shared those

- [Git in VSCode](#)
- [L^AT_EX workshop in VSCode](#)

4.2.2 The T_EX Distribution

Everything I have written about so far is platform-agnostic. It's true for L^AT_EX, and L^AT_EX does not care about Windows, Linux or MacOS. However, when it comes to the *implementation*, things are slightly different.

There are different distributions for the different operating systems, and the most dominant ones are:

- T_EX Live (cross-platform / Linux default)
- MacTeX (macOS bundle of T_EX Live)

- MiKTeX (Windows/cross-platform)

By installing those, you effectively install **multiple** engines, core $\text{\LaTeX}$ files, a package manager and some CLI tools that help you interact with the different parts of the $\text{\LaTeX}$ ecosystem.

Note 4.1

Though (latex) package managers are usually installed within the distribution, on Linux it is advisable to use your (Linux) distribution's package manager (apt, pacman, whatever) instead of tlmgr. Mainly to ensure that everything is automatically updated, with all components remaining compatible.^a

^aDistro repositories usually lag CTAN by 1-2 years. Due to that, vanilla Tex Live installations might be useful under certain circumstances.

My only tip for the Windows and MacOS users is this one: **when in doubt, install more packages**. If I remember correctly, both installers ask whether you want a full or a minimal installation. For beginners, the fuller installation will undoubtedly help you avoid some problems with missing packages.

Now, for Linux users, feel free to check this one out, it's the list of texlive packages I have currently installed.

```
1 # A gift: ;)
2 # pacman -Qs texlive | awk '/local/ {print $1}' | cut -d/ -f2
3 sudo pacman -S texlive-basic texlive-bibtexextra texlive-bin texlive-binextra
  ↳ texlive-fontsextra texlive-fontsrecommended texlive-latex
  ↳ texlive-latexextra texlive-latexrecommended texlive-pictures
  ↳ texlive-plaingeneric texlive-xetex
```

4.2.3 The CLI tools

Once you have your .tex files ready, I'm sure you'll want to have them compiled and admire your newly created documents. To do that, though GUI solutions exist, you might want to interact with the engine via CLI.

Disclaimer 4.2

After spending some time researching, I settled on using xelatex and configured my workflow around it. That's why you'll only see xelatex moving on^a.

^aI would argue that 99% of this article is applicable to all engines, but xelatex was chosen to make my life writing documents with both English and Greek as easy as possible.

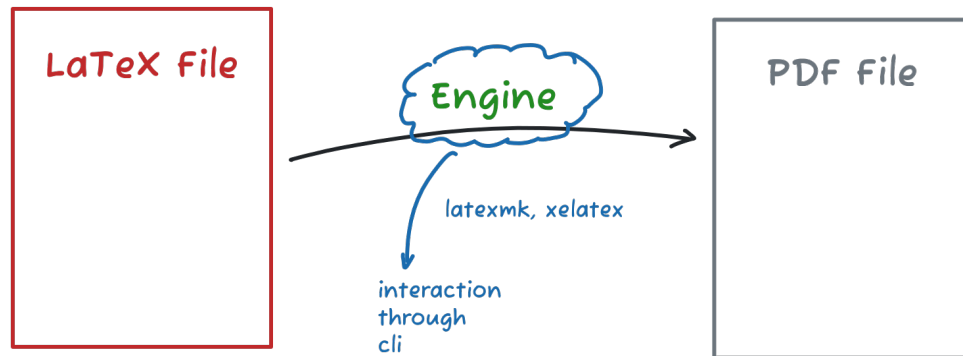
You can use the engine as is, but I prefer using latexmk. I like to think of it as a naive little helper: it helps me keep my directories clean, have continuous preview, have my bibliography without much hassle, and much more.

```
1 # Assuming that your tex file is master.tex
2 xelatex master.tex
3 latexmk -xelatex master.tex
4 # Using xelatex, open the preview and keep updating on changes to tex
5 latexmk -xelatex -pvc master.tex
6 # Force is usually helpful here, to ignore unimportant warnings, however make
7 # sure not to overdo it. I prefer working without it most of the time (since
  ↳ it
```

```

8 # might hide other problems until it's too late)
9 latexmk -xelatex -pvc -f master.tex
10 latexmk -xelatex master.tex -bibtex # if you use bibtex
11 latexmk -C # use this to delete files used in the compilation process

```

Figure 3: L^AT_EX: The practical overview

5 Testing your environment

If you're following this article like a tutorial, this would be a good time to take a step back and test your environment. Copy the following into a `latex` file, read it and then use the commands found above to produce a PDF.

- Before getting overloaded, open [this cheatsheet](#).
- All lines starting with `%` are comments, make sure to use them wisely in your own work as well. They are not shown in the end result.
- Make sure to fool around: add some lists, add bold, italics, images. Everything you'd normally do in a report. The cheatsheet I linked is more than enough.
- Try to write an **equation** or two!

```

1 % !TeX program = xelatex
2 % Try with book, report, article
3 % notice the differences
4 \documentclass[12pt]{report}
5
6 % You can play around with this one too
7 \date{\today}
8
9 % Add your name here
10 \author{Chatziiola}
11
12 \title{Hello World}
13
14 \usepackage{graphicx} % includegraphics, for images
15
16 % If you do not begin your document then nothing will be produced.

```

```

17 \begin{document}
18
19 \maketitle % comment this line out, see the output
20
21 % It is very easy to add sections
22 % \chapter{Chapter title}
23 % \section{This is a section}
24 Hello world
25
26 \begin{equation}
27   E = mc^2
28 \end{equation}
29
30 \end{document}

```

6 Dissecting the sample document

If you take a quick look at the piece of code you'll see that we can split it in three:

- The documentclass line
- Before the document environment
- The document environment

```

1 \documentclass[12pt]{report}

```

The documentclass line is something you will usually ignore. Going in depth here, would be way too time-consuming, so let's consider it out of scope. You only need to know that:

- [these,are,optional,arguments]{not-optional}.
- there are different classes such as book, report, article
- some optional arguments, such as a4paper, 12pt², are widely universal.

```

1 % A minimalistic preamble
2 \date{\today}
3 \author{Chatziola}
4 \title{Hello World}
5
6 % you can use \usepackage{package}
7 \usepackage{graphicx} % allows you to use \includegraphics{} for images

```

That second part³ is often called the preamble. I've kept it simple in the example but it does not need to be like that: my thesis preamble was more than 100 lines long. In it you:

- \usepackage{package}
- configure fonts, margins, colors and every other property you can think of.

²Font size in general

³It can be integrated within a class, but having it standalone offers greater flexibility.

Note 6.1

It is important to note that most of these tasks **can only be done in the preamble**. Try to `\usepackage` inside the document environment and you'll get compilation errors.

```

1 \begin{document}
2
3 \maketitle % comment this line out, see the output
4
5 % It is very easy to add sections
6 % \chapter{Chapter title}
7 % \section{This is a section}
8 Hello world
9
10 \begin{equation}
11     E = mc^2
12 \end{equation}
13
14 \end{document}

```

The last part is the document itself. You can use environments and commands (if you have properly included their packages) and practically everything you write there will be visible to the resulting document.

7 Minimizing work: Your preamble

I'm sure some of you have already guessed why I broke the document down like that. The goal is the **separation of structure and content**. In your preamble you want to carefully define everything that should be consistent across your different projects: fonts, margins, macros, colors, packages available. That will allow you to reuse it, without needing to reinvent the wheel every time.

That's my final *tip* to connect the dots. In $\text{\LaTeX}$ you can reuse whole files by using `\input{filename}` and `\include{filename}` statements.

Note 7.1

There is a subtle difference between the two:

- `\input{input_file}` means that the contents of `input_file` will be inserted as is to your existing document (behind the scenes). Thus, even if `input_file` did not change at all, it will be fully recompiled.
- `\include{include_file}` means that `include_file` will be independently compiled and then *inserted* to the output file.

Your preamble **must** be inserted using `\input` (trust me) and for smaller projects performance will be the same. **But** `\include` inserts the file at a new page.

A sample preamble you can use is this one:

```

1 % !TEX program = xelatex
2 % TODO add this to a file called preamble.tex
3
4 % Your name probably does not change across documents :P

```

```

5 \author{YOURNAME}
6
7 % this is a minimal configuration. feel free to expand
8 \usepackage{graphicx}
9 \usepackage{listings}
10 \usepackage{longtable}
11 \usepackage{wrapfig}
12 \usepackage{rotating}
13 \usepackage{amsmath}
14 \usepackage{amssymb}
15 \usepackage{capt-of}
16 \usepackage{geometry}
17 \geometry{
18     margin=2.5cm,
19     includeheadfoot
20 }
21
22 \usepackage{fontspec}
23
24 % Add your own fonts
25 % Tinos: available from google fonts
26 \setmainfont{Tinos}
27 \newfontfamily\greekfont{Tinos}
28
29 \usepackage{xcolor}
30 % define your own colors
31 \definecolor{YOURCOLOR}{RGB}{69,69,69}
32
33 \usepackage{hyperref}
34 \hypersetup{
35     colorlinks=true,
36     citecolor=YOURCOLOR,
37     linkcolor=YOURCOLOR,
38     filecolor=YOURCOLOR,
39     urlcolor=YOURCOLOR,
40 }
41
42 \usepackage{fancyhdr}
43 \pagestyle{fancy}
44 \fancyhf{}
45 \fancyhead[L]{\textcolor{YOURCOLOR}{\rightmark}}
46 \fancyhead[R]{\textcolor{YOURCOLOR}{YOUR NAME}}
47 \fancyfoot[C]{\textcolor{YOURCOLOR}{\thepage}}
48
49 \usepackage{titlesec}
50 \titleformat{\section}
51     {\normalfont\Large\bfseries\color{YOURCOLOR}}{\thesection}{1em}{}
52 \titleformat{\subsection}
53     {\normalfont\large\bfseries\color{YOURCOLOR}}{\thesubsection}{1em}{}
54 \titleformat{\subsubsection}

```

```

55  {\normal-
    ↳ font\normalsize\bfseries\color{YOURCOLOR}}{\thesubsubsection}{1em}{}
56  \titleformat{\chapter}[display]
57  {\normalfont\huge\bfseries\color{YOURCOLOR}}{\chaptertitlename\
    ↳ \thechapter}{2em}{\Huge}
58
59  % This is for justification purposes
60  \usepackage[document]{ragged2e}
61  \justifying
62  \sloppy
63  \setlength{\parindent}{2em}
64  \setlength{\RaggedRightParindent}{\parindent}

```

Disclaimer 7.1

Αν γράφετε σε ελληνικά, δώστε ιδιαίτερη προσοχή στις γραμμές του fontspec. Δεν χρειάζεται να μπλέξετε με περίεργα packages, ούτε να χρησιμοποιείτε εντολές για αλλαγή γλώσσας. Χρειάζεται απλά (όταν έχετε xelatex) να χρησιμοποιείτε γραμματοσειρά που να υποστηρίζει τα ελληνικά.

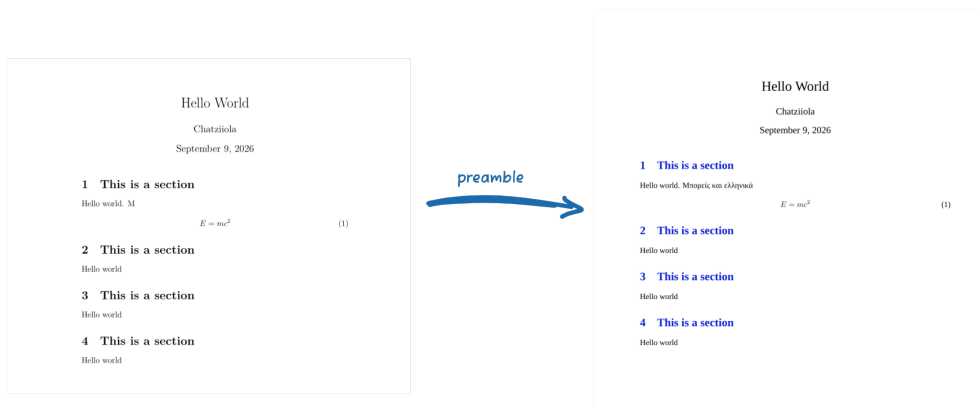


Figure 4: Changes using just this preamble - notice that you can now use Greek ;)

Having a preamble, your documents can start looking more like:

```

1  % !TeX program = xelatex
2  \documentclass[12pt]{article}
3
4  % Configuration is hidden in here
5  \input{preamble.tex}
6  \title{YOUR TITLE}
7
8  \begin{document}
9  % Your content
10 \end{document}

```

8 Bibliography

For some reason I get a feeling of satisfaction when reading papers and being able to see the sources, or coming across blogs like gubern.net that document their inspiration. When I took up latex I saw that I could do that very easily. All I had to use was a package like `biblatex` (please do not go for `bibtex`, you'll regret it down the line), have a `.bib` file and have my references...

It can be that simple, but not if you decide to manually write your entries. `@bookclauswitzWar1984, title = On War, author = family=Clausewitz, given=Carl, prefix=von, useprefix=true and Howard, Michael Eliot and Paret, Peter, date = 1984, publisher = Princeton University Press, location = Princeton, N.J, isbn = 978-0-691-05657-9, langid = english`

This bib entry does not have many fields nor is it particularly complex. Most of the entries I used on my thesis have 2 or 3 times the size. I did not add them manually. I used [Zotero](#), per [this guide](#). In essence, you install Zotero locally, then the browser plugin, and whenever you find a good article or book, you use your plugin to create an entry on your Zotero library. Then inside Zotero you can export your selected library to `biblatex` (and keep it updated!), thus automating the creation of your `.bib` file.

If you follow this workflow, then you do not need to change anything⁴, other than

- `\usepackage[ ... ]{biblatex}` to actually use `biblatex`.
- `\addbibresource{your-bib-file.bib}`, to select your bibliography file.
- use `\cite` to make your citations
- use `\printbibliography` to have a beautiful bibliography (with backlinks) automatically created for you

Take a look at this snippet (it will not compile as is):

```

1 % ...
2
3 % use this in the preamble
4 \usepackage[backend=biber, style=numeric, sorting=ynt]{biblatex}
5 \addbibresource{your-bib-file.bib}
6
7 % ...
8
9 % main document
10 \begin{document}
11
12 blah blah blah\cite[p.4]{LamprinosChatziioannou2022}. Yes I truly said that.
13
14 \printbibliography[heading=bibintoc]
15
16 \end{document}

```

9 Going forward

At this point you should get the picture. Your directories should look like that:

```
preamble.tex % Common across projects, no need to copy
thesis-or-big-project/
```

⁴You might need to install `biber` separately with your package manager.

```

├─ master.tex
├─ thesis.bib
├─ figures/
├─ chapters/
│   └─ 01_introduction.tex
│   └─ 02_methodology.tex
small-project/
├─ report.tex
├─ figures/
└─ references.bib

```

9.1 Some generic tips:

1. Your `\included` files do not need to have a preamble nor a document environment. You can write in those as if you're already in your document.
2. Tables are tricky to get right, I settled for the `xltabular` package, with slight modifications using `booktabs`, `array`, `rotating`, `threeparttablex`, and commands like `\newcolumnstype{Z}{>\raggedright\arraybackslash}X`.
3. Creating graphs with pure $\text{\LaTeX}$ isn't practical. While you might get something out of that, I prefer creating the graphs with `python`, `matlab` or anything else. $\text{\LaTeX}$ is my last resort for that.
4. You can use `label` and `cref` (from the `cleveref` package) to make references to figures and sections in your documents easily
5. If you want justified text, make sure to use the `ragged2e` package. I used it like this:

```

1 \usepackage[document]{ragged2e}
2 \justifying
3 \sloppy
4 \setlength{\parindent}{2em}
5 \setlength{\RaggedRightParindent}{\parindent}

```

6. You can download `.cls` files and then replace your `\documentclass` with the "new" class. You'll probably find yourself doing that for big projects, such as your thesis.
7. You can download fonts locally and configure `fontspec` to load the fonts from files and not to rely on your system's fonts (that is particularly useful if you're using a niche font and you want to collaborate with others). Also, check out [latex-fonts](#).

```

1 % Example of local fonts
2 \setmainfont{AvenirNext}[
3   Extension = .otf,
4   Path = ./fonts/AvenirNext/,
5   UprightFont = *-Regular,
6   BoldFont = *-Bold,
7   ItalicFont = *-It
8 ]

```

8. Use `git` vigorously - it's better to have the option of looking back.
9. If you want to use multiple images, `graphicx`, `subcaption` are all you need:

```

1  \begin{document}
2
3  \listoffigures
4  \vfill
5
6  \begin{figure}[ht]
7    \centering
8    \begin{subfigure}[b]{0.4\textwidth}
9      \centering
10     \framebox[3cm]{\strut Subfigure 1}
11     \caption{First subfigure}
12   \end{subfigure}
13   \hfill
14   \begin{subfigure}[b]{0.4\textwidth}
15     \centering
16     \framebox[3cm]{\strut Subfigure 2}
17     \caption{Second subfigure}
18   \end{subfigure}
19   \caption{Main figure container}
20 \end{figure}
21
22 \end{document}

```

10. If you use multiple languages, such as Greek, xelatex is the way to go. [Look at this](#). **You only need to use a font that supports the necessary characters.**
11. If you ever want to use glossaries,

9.2 On templates

- [IEEE Template Selector](#): Gives you the ability to download L^AT_EX templates.
- Feel free to modify the preamble, while also using the sample master.tex
- I have created this [gist](#), which I encourage you to copy and use. If you download it and have the necessary fonts available then the following master.tex is a good place to start.

```

1  % !TeX program = xelatex
2  \documentclass{Thesis}
3
4  % Details
5  \date{\today}
6  \title{Test}
7  \titlegreek{Τεστάκι}
8  \authors{Lamprinos Chatziioannou}
9  \university{\texorpdfstring{\href{https://www.auth.gr}{Aristotle University}
10 ↪ of Thessaloniki}}{University name}}
11 \department{\texorpdfstring{\href{http://www.ece.auth.gr/el/}{School of
12 ↪ Electrical and Computer Engineering}}{Department or School Name}}
13 \faculty{\texorpdfstring{\href{https://www.auth.gr/en/faculty/eng-en/}{
14 ↪ Faculty of Engineering}}{Faculty Name}}
15 \degree{Integrated Master in Electrical and Computer Engineering}

```

```

13 \supervisor{\texorpdfstring{Supervising Professor's name} {Supervising
    ↪ Professor's name}}
14 \universitylogo{figures/logo-auth.png} % find yours
15 \universitygreek{\texorpdfstring{\href{https://www.auth.gr}{Αριστοτέλειο
    ↪ Πανεπιστήμιο Θεσσαλονίκης}}{Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης}}
16 \facultygreek{\texorpdfstring{\href{https://www.auth.gr/en/faculty/eng-en/}{Πολυτεχνική
    ↪ Σχολή}}{Πολυτεχνική Σχολή}}
17 \departmentgreek{\texorpdfstring{\href{http://www.ece.auth.gr/el/}{Τμήμα
    ↪ Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών}}{Τμήμα Ηλεκτρολόγων
    ↪ Μηχανικών και Μηχανικών Υπολογιστών}}
18 \authorsgreek{Λαμπρινός Χατζηιωάννου}
19
20 \begin{document}
21   \maketitle
22
23   \begin{abstract}{Abstract}
24     Hey hey my my
25   \end{abstract}
26
27   \begin{abstractgreek}{Περίληψη}
28     Ελληνικό
29   \end{abstractgreek}
30
31   \dedicatory{Για το φώς στα σκοτάδια}
32   \tableofcontents
33   \chapter{First chapter}
34     This is a test
35   \chapter{Αρχίζει καλά}
36     Λαμπρινός Χατζηιωάννου
37
38 \end{document}

```

10 Further reading

- Knuth's Interview: $\text{\TeX}$ origins
- Castel's preamble
- Greek LUG archive: has some material on $\text{\LaTeX}$