“TeX” Support Proposal

Lorenzo Hess

2024-12-19

Contents

1 Introduction

This document outlines ideas for a comprehensive approach to LaTeX advocacy, covering everything from pro-LaTeX talking points, arguments against word processors (WPs), solutions for group work in LaTeX, and LaTeX code snippets, including full preambles and examples. While any individual could make use of this advocacy material, a group or club would be ideal in terms of providing additional resources and labor. Such a group could be called "TeX Support", so I've used that in the title of this document [1].

2 What's Our Agenda?

We like LaTeX, and we think it's better for preparing documents, in most cases, than Word Processors (WPs). We're also pragmatic, and recognize that many people haven't used LaTeX or may have felt frustrated when they have. Many LaTeX tutorials indeed fail to enumerate and provide what we see as essential foundations for learning, appreciating, and enjoying LaTeX document preparation. For example, many tutorials characterize LaTeX's utility in terms of what it is good at, such as typesetting math or providing robust label and reference management. They fail, however, to provide a comprehensive feature comparison between LaTeX and WPs that would make clear LaTeX's superiority. Tutorials also generally fail to fully compare the LaTeX and WP mindsets, and thus fail to help users realize how WPs hinder writing. Finally, tutorials fail to provide essential tools that make LaTeX editing less tedious, from code templates like preambles to software like Detexify.

We have two general goals: 1) present a comparative, serious, and evidence-based argument for when, and why, individuals and groups should use LaTeX instead of WPs; and 2) to provide code templates, code snippets, and guides for individuals and groups to not only get started with LaTeX, but use it to its full potential.

We're not here to evangelize. WPs do have certain advantages over LaTeX(see the feature comparison in Section X), and it's a free country (and even freer Internet): we don't care what software you prefer or use. We do believe that LaTeX, combined with a proper editor and editing workflow, usually makes writing and formatting faster and better.

3 What is LaTeX?

What is LaTeX? What is a markup language? Do I need to know programming? Is LaTeX programming?

You write instructions (i.e. code) to tell a latex compiler how to prepare your PDF, i.e. how to format, structure, and generally manipulate text, images, tables, etc to make your PDF.

[1]Thanks to Keegan Kuhn for this pun.

Analogies: latex is the movable type of PDFs.

4 Feature Comparison

4.1 The General WP

First cover features, operation (GUI: menus, buttons, and keybinds), and general workflow. Cover the WP mindset.

Advanced Features

Most WPs lack advanced features such as generating a table of contents [2], generating lists of figures, more, and more as well as configuration of all of the above.

A Hidden, Mutable State

WPs are "What You See Is What You Get" [3] editors. What appears in the WP window will appear in your PDF: what you see in the WP is what you're always gonna get. There's no separation between input and output: the writing you put into the WP will be present in the output, whether that's the WP file itself, a generated PDF, or a screenshot of the WP window, perhaps.

Because what you see is what you'll get, any characteristic of your document must be telescoped, projected, or mapped into visual form. Consider boldface styling. Boldface style is the thick version of a font; applying bold to a word makes its font thicker. Crucially, boldface must be applied /to a word/. A word processor, however, has no conception of a word. If you wish to have a word be in boldface, you can select the word by highlighting it and make it bold, often with a keybind or button. Your conception of the bold word, however, has not been mapped one-to-one to the WP: information has been lost. Specifically, your conception of the bold word relies on your conception of the word itself, and this conception in turn relies on the ability to define start and end points of a word. The WP, however, has no conception of a word, and thus cannot define its start and end points. Sure, you may have highlighted the word by starting at one letter and ending at another, but the WP doesn't know this. Your intention, of bolding that word itself (as defined by start and end points), has been lost in the projection of your conception of bolding onto the WPs visual space.

This information loss leads to weird, and often annoying, behaviors. If I bold a word, for example, and then place my cursor in the middle of the word and continue typing, the new letters will remain in bold. Even if I first type a space, which should, in my conception, signify the start of a new word, the letters are bold. In fact, WPs allow you to apply bold style to things that aren't even words, even to spaces. Hopefully my explanation has been clear enough, but consider how ludicrous it is to make a space boldface.

[2]As shown in the next section, many WPs can actually generate a table of contents. Their ToCs, however, are basic and impede workflow, and thus are not considered advanced.

[3]WYSIWYG, pronounced "Wi-see-wig"

The many-to-one mapping from your conception to the visual WP environment doesn't simply cause the lost information to disappear. In fact, WPs require you to keep track of metadata about the document, which can be termed the document's "state".

All this mapping into visual space requires you to keep track of state. You have to remember if you had put a page break somewhere, and if so, where on the page the break was inserted. Or if the boldface ends at the end of a boldface word or if it applies to the space after it as well.

"bold" emphasis style. When you emphasize a word with boldface, you desire that word to be bolded. The WP, however, has little, if any, conception or definition of a "word". In this visual environment

Because of the WP's limitations

4.2 LaTeX

First cover features, operation (TUI: writing code and compiling), and general workflow. Cover the latex mindset.

4.2.1 Don't Switch to LaTeX If...

Don't switch to LaTeX if

1. You're not willing to learn the minimum necessary for your needs (might just be math equations, or might extend to bibliographies).

2. All this being said, LaTeX is just text (another benefit!). If you've started a project but can't finish it in LaTeX, you can always copy/paste it somewhere else, or even export it directly to docx with tools like Pandoc (won't export everything perfectly).

4.3 Table of Features

We should quantify as many features as possible and present them in a table for a comprehensive feature comparison.

4.3.1 LaTeX vs. General WP

4.3.2 Overleaf vs. Google Drive vs. Microsoft Onedrive

5 How Tos

5.1 How to Read Error Messages

5.2 How to Debug LaTeX Code

5.3 How to Use LaTeX Package Documentation

5.4 How to Find Help

tex stackexchanger, overleaf.com/learn, google searches

5.5 How to Use Detexify

Really simple.

6 Building Blocks

6.1 Preambles

6.1.1 Useful Packages

6.1.2 Homework

6.1.3 Reports

IQP

MQP

6.2 Tweaks

We can add the code snippet then link some documentation for people to dive deeper.

1. Header and footer for nice homework.
2. Two figures side by side.
3. Appendices
4. Bibliography
5. acronyms

6. landscape pages
7. block quotes
8. list spacing
9. actually put floats where you want them
10. different fonts and math fonts
11. footnotes
12. glossaries
13. page margins
14. date formatting
15. multi-row table cells
16. code styling
17. include PDF pages
18. charts and other graphs
19. justification
20. line spacing
21. reduce vertical space between word and underline
22. vertical table cells
23. URL colors
24. single space table of contents

6.3 Snippet Plugins

Vim, emacs, vscode.

Naming convetions

Good to have: fraction, etc.

6.4 Naming Convetions

6.4.1 Sections

Good practices for naming sections

6.4.2 Labels

Good practices for naming labels, e.g. `sec.subsec.subsubsec` (dot notation) or `sec:subsec::subsubsec` (colon notation).

6.5 How To Manually Modifying Spacing

How to use positive and negative vspace and hspace. How to use vfill and hfill.

Macros for inserting a preset amount of horizontal space, e.g. to separate answers within a box

6.6 LaTeX Editing Workflow

Your own preamble. File templates. Homework templates. Bibliographies

7 LaTeX Editors

TUI (emacs and vim), VSCode, overleaf. Find other popular ones to cover.

8 Overleaf for Collaborative Editing

Don't give Overleaf tutorial (they have docs for that). We should outline how to manage group projects and collaborative writing in latex.

9 Inkscape and LaTeX For Fancy Diagrams