What's a Professor of Neurology doing using LaTeX?

David B. Teplow, Ph.D.

Abstract

In the general biomedical and academic research communities, most people have never heard of LaTeX. Students and professors in the humanities, social sciences, biological sciences, and clinical medicine who have heard of LaTeX often don't appreciate the value of such a sophisticated and elegant typesetting engine. Instead, as with most of the world, they succumb to the forces of the *Dark Side* (traditionally Microsoft Corporation) and use expensive, inflexible, closed source, and relatively primitive programs to compose documents. They also suffer the continuing frustrations of doing so. I discuss here my own journey out of the darkness and into the light of LaTeX.[1]

1 Introduction

My first exposure to document creation, like most in my generation, was in English class in elementary school. Documents all were hand written and one was graded on "penmanship." As one got older, one was expected to use a typewriter to produce professional looking documents. It was not until the 80s, with the introduction of the IBM PC (personal computer) in 1981 and the Apple Macintosh 128K in 1984, that the average consumer could compose documents electronically and print them using line printers or dot matrix devices. I remember my excitement running WordPerfect or WordStar on my IBM PC and MacWrite on my Macintosh Plus (with 1 MB of memory and a 20 MB disk drive large enough to use as a crane counterweight). One actually could "program" how individual letters, sentences, paragraphs, or document sections should look. This was done using keyboard commands embedded in the text. WYSIWYG GUIs came later and were seen as revolutionary. One no longer had to guess how their typescript would look. It was right there in front of you on the screen.

Development of personal computers and software has continued during the almost four decades since a vision for personal computing occurred at IBM. The most important software design principle was "do more and make it easier." Unfortunately,

[1] The reader is cautioned that what follows is my personal perspective on LaTeX. This perspective is not meant to be, nor is it, a definitive review of LaTeX and its uses. I ask the indulgence of *TUGboat* readers if they find any inaccuracies in this article and would be grateful if these inaccuracies were brought to my attention.

and particularly in the case of Microsoft Word, doing more and making it easier actually meant "make it more complicated, inflexible, and buggy." When the Unix-based Mac OS X operating system came to the Macintosh platform with its protected memory and preemptive multitasking architecture, software programming errors only crashed the application in use, not the entire computer and not that often. The exception, of course, was Microsoft Word, which to this day can be counted on to crash at the least opportune moments, often making one's prior work unrecoverable.

As a professor, I had to continually compose documents, be they notes, letters, grant applications, or manuscripts to be published in academic journals. Text processing capability was mandatory and Word was the *de facto* standard for this purpose. This meant that year after year, decade after decade, I, like others, had to suffer the frustrations inherent in trying to get text processing "bloatware" to do what one wanted it to do. These frustrations included, among many, application crashes, the well known "Word has insufficient memory" error messages when one tries to save a file, difficulties embedding images and maintaining their location during subsequent editing, problems formatting tables, bizarre placement of equations constructed using MathType or Equation Editor, and an inability to stop the program from "helping" you by automatically changing formatting, word spelling, and other aspects of document creation.

These computing and composition experiences made clear a desperate need for a better method of document composition. Enter LaTeX.

2 How I met LaTeX and why I fell in love with it

My first exposure to LaTeX occurred in the context of a collaborative effort to understand the mechanistic bases of Alzheimer's disease. The collaboration integrated biochemical and computational studies of protein aggregation. My laboratory carried out the biochemistry work while the computational studies were done by physicists. When our studies were complete, we discussed how and where to publish our results. I had assumed that our manuscript would be composed using Word, which I suggested to my physicist colleagues. To my surprise, the leader of the physics group, a world authority in the field of statistical physics, told me that he did not know what Microsoft Word was! All word processing in his group was done using LaTeX, which I had never heard of. The composition and publishing of scientific manuscripts is an arduous process that requires tremendous attention to detail and many, many iterations during initial manuscript creation and the peer review process. Authors must use the same text processing platform during this process. It thus appeared that either my learned colleague and his group would have to learn Word or I would have to learn LaTeX.

I chose to be the one to learn a new document preparation system. I did so for a number of reasons, some practical and some personal. The practical reason is that professors tend to become ossified as they age, which means that change can be difficult. It would be easier for me, as a new Assistant Professor, to learn a new system than it would for my senior colleague, a distinguished Professor of Physics. The second reason was my high regard for the academic acumen of most physicists, which suggested that the tools they used in their research, including those for document preparation, likely would be powerful and elegant. As I mentioned above, the introduction of the Unix-based Mac OS X operating system made the Mac platform remarkably powerful because now one could take advantage of the huge reservoir of expertise and software associated with Unix, which of course included LaTeX. I quickly learned that the LaTeX source files were simple ASCII text files. I could work on my Mac, either inside a LaTeX GUI or in Terminal, while my colleagues could use their PCs and we could easily exchange files and be certain they would compile,[2] regardless of platform. This eliminated the continuing problem of format alterations in Word files caused by file movement between Mac and PC platforms. I was tremendously impressed with the professional layouts of our manuscript after source file compilation. For the first time in my academic life, I could create manuscripts that, essentially, *were already typeset.* They were beautiful. Finally, from the perspective of a science nerd, I found the prospect of learning what essentially is a programming language to be very exciting.

As I became more adept at using LaTeX, I realized that it provided many capabilities that were superior to those of standard word processing programs. One of the biggest headaches in the composition of scientific manuscripts is the need to change figure and table numbers if such items are added or deleted from manuscript drafts. One can do this manually if one is particularly attentive to detail, or automatically using search and replace functions. However, this is time consuming and often results in multiple figures or tables having the same number, which is confusing — especially to peer reviewers

[2] Assuming no trivial coding errors existed.

and editors who decide if your manuscript will be accepted for publication. Enter the LaTeX `\label{}` command! What an easy and elegant way to ensure that any editorial changes result in automatic, accurate renumbering of figures and tables.

I was equally, if not more, delighted by BibTeX, especially after struggling with EndNote for so many decades. I could now create a single library and format my bibliography using pre-existing bibliography style `.bst` files — no more hassles with constantly having to edit EndNote output styles. I could edit my library in any text editor or use one of the many reference management programs available. I've used JabRef and BibDesk, among others, and find JabRef particularly useful.

The use of pre-existing class and style files illustrates the power, ease of use, and time efficiency of the LaTeX platform. If I am required to use a particular class for a publication, I simply download it from the web or get it from the publisher (as I did for the `ltugboat` class used for this article). I can compose my source file without any concerns about it being properly rendered. Of course, as *TUGboat* readers well know, and as neophyte LaTeX users rapidly learn, TeX and its derivatives are designed to enable writers to focus on the *content* of their work as opposed to its *formatting*. I no longer have to spend time carefully reading document formatting instructions from a publisher or agency to whom I am submitting a grant application and then converting these instructions into an acceptably formatted Word document. The class and style files do it all for me.

Experienced computer users know that it is most time efficient to operate your computer by leaving your fingers on the keyboard, as opposed to constantly having to manipulate a mouse or other input device. This is no more evident than when one is creating mathematical formulas. Although Equation Editor and MathType are useful point-and-click formula creation applications that interface seamlessly with Word, one must invoke either one and then click, click, click . . . to create the formula, which then is inserted, often with bizarre vertical alignment within text lines, into the document. In addition, it is common to find that these formulas are rendered improperly once the file has been typeset by a publisher. When I create formulas in LaTeX, I can do so without lifting my hands off the keyboard and I do not experience any subsequent formatting or rendering errors.

Anyone who has tried to create lists using standard word processors likely has encountered problems with indentation, nesting of list elements, and most vexing, stopping the program from adding new text to the end of an existing list. The fine control of the list environments in LaTeX eliminates these problems. Similar advantages exist with respect to table creation. I am an experienced Word user (unfortunately), but I still can't figure out how to format and align tables in a reasonable amount of time.

I find that figure and caption placement can be problems both for word processor and LaTeX users. One also encounters figures that mysteriously change their positions within a document. In Word, these problems are exacerbated by the fact that figures and captions are entered independently.

3 How I learned LaTeX

As we all know, LaTeX, in essence, is a programming language. Its code may be less complicated than C++, Fortran,[3] Objective-C *et al.*, but it nevertheless requires the user to create source code that instructs a compiler how to produce a properly rendered document. One of the beauties of LaTeX is that a new user need not know anything to begin using LaTeX other than how to open a `.tex` file in a text processor. This was how I began the learning process, by simply editing the text within the source files created by my collaborators. It was easy to learn how to encode underlined, italicized, or bolded text. After all, how hard is it to "escape" the obvious "bf" abbreviation for bold font and type `\bf`?[4] In the process of assimilating this simple syntactical information, one begins to get a sense of how LaTeX programming works and this sense then provides a framework for adding new skills to one's repertoire. "Environments" then are encountered that require learning how they are parameterized and about what can and cannot be done within them. At this point, the neophyte LaTeX user needs to begin studying the language more deeply.

I found myself doing what any self-respecting academic would — I bought books. The first two are well known in the TUG community, namely *Guide to LaTeX* by Kopka and Daly and *The LaTeX Companion* by Mittelbach and Goossens. These two volumes became my "go to" references for questions. I also found *First Steps in LaTeX* by Grätzer, *LaTeX Line by Line* by Diller, and *Learning LaTeX* by Griffiths and Higham to be useful. Scientific publications usually contain tables and figures. In the beginning, as I began creating my own `.tex` files, it was simple to copy and paste a figure environment from a file of my collaborators and just insert the path to my own

[3] Including Fortran 4, which I used a half century ago!

[4] Interestingly and ironically, I just now learned how to escape the backslash so all the following text was not bold!

figure. This got me started. I also cut and paste table environments. However, to gain more expertise in managing these environments, I added *Typesetting Tables with LaTeX* by Voß, and *The LaTeX Graphics Companion* by Goossens, Rahtz, and Mittelbach, to my "go to" references. These days, however, I rarely consult these references, not because they are uninformative, but because so much detailed information is available on the web. I routinely access `tex.stackexchange` if I need help, download package manuals, or access other sources found through web searches. It's remarkable how many preamble lines, environments, minipage formats, and other bits of code one can simply cut from web pages and paste into their source file to achieve a particular typesetting goal without any *pre facto* syntactical knowledge.

I find "playing" with LaTeX to be a lot of fun. It's often a challenge to render and position text, figures, and tables in a specific way. I like trying different things and seeing the output. I might switch between standard `figure` and `wrapfigure` environments, use minipages, or try other methods to achieve a particular document rendering. The process of self-directed investigation provides rich rewards in terms of better understanding how LaTeX works and how to manipulate output, as opposed to memorizing how to perform a single task. One is able to develop an intuition that facilitates learning and accelerates the process of problem solving.

4 How I use LaTeX

"If you got a terminal, you can use LaTeX."

This certainly is true for those who are ⋆nix (Unix, Linux *et al.*) savvy or love the command line. However, the majority of the world's computer users interact with their computers through GUIs. A major advance for the general LaTeX community, one that made the use of these programs much more attractive to the average computer user, was the introduction of GUI front ends to LaTeX compilers. Users were able to run LaTeX by pointing and clicking with their mice. No knowledge of ⋆nix commands and syntax were required. What was required, both for command line users and GUI users, were multiple steps before a finished document could be viewed. For academicians, for whom extensive referencing is required, the process included triple compilation (LaTeX→BibTeX→LaTeX) so that references were numbered correctly and a bibliography was created. Multiple steps also were required to produce an output file that could be easily shared with others who might be relatively computer illiterate or worked using different platforms and operating systems (e.g., Mac and MacOS, PC and Windows, terminals and ⋆nix). This typically involved a tex→dvi→pdf compilation and conversion process. One also could produce output files in other formats, including postscript, html, or rtf, but pdf was the most useful for collaborations and submission of manuscripts to most journals. These processes were not onerous in nature, but they were burdensome and time-consuming. For those used to WYSIWYG document composition, this need to first compile the source code before seeing the finished work product was a bit off-putting. However, with the advent of three-panel application interfaces (file directory, source file, rendered output) and automatic file compilation, users can immediately see the results of their work. This has been an important development because it has streamlined the document preparation process for the average computer user, eliminating the need to understand the source file→compiler→output file process.

My initial LaTeX front end was TeXShop, which provided a simple, useful method for compiling source files and viewing their output. As one who enjoys determining if newly developed or updated applications might offer an easier or more powerful user experience, I also have used Texmaker, TeXworks, TeXnicle, Texpad, TeXstudio, and Latexian, as well as web-based document creation and compilation engines like Overleaf (formerly writeLaTeX) and ShareLaTeX. LyX is unique among these front ends in that its default document view hides the source code from the user and its interface looks more like the icon-based interfaces of non-TeX-based word processors. It also requires the user to port the output file to a different application (e.g., Adobe Acrobat) to view the rendered output. I found this type of GUI to be an unhappy medium between the extremes of document preparation, i.e., using a terminal or using a standard word processor (e.g., Word).

I am composing this document using Texpad, which is my current favorite. A helpful feature of Texpad and other programs is their handling of compilation errors. Error and output logs are instantly available for user review either within the main application window itself or by a simple click on an icon. In addition, the user is provided the means to rapidly edit offending syntax simply by clicking on a particular error message in the message viewer, which then moves the focus of the keyboard to the offending line of code. This saves a lot of time during the error correction process. Other features of these front ends that are particularly useful are code completion, flash bulb-like highlighting of beginning and ending characters (e.g., curly braces), and syntax highlighting.

To further facilitate document creation, and to deal with the progressive memory loss experienced by Professor Emeriti, I also create a variety of preambles, tables, and figure environments and store them in a special directory. I then can simply cut and paste the code into new documents without having to remember any special syntax that I might have used in the past. For example, as a professor, I am asked to compose many different kinds of recommendation letters, including those for undergraduates, graduate students, postdoctoral fellows, different professorial ranks, etc. To do so, I have created a directory in which boilerplate letters for each type of recommendation exist. When I need a template, I can rapidly access these pre-made files. This has made composition of new letters trivial. The same strategy is used for grant application preambles, be they for the National Science Foundation, the National Institutes of Health, or other agencies.

5 Using LaTeX in a non-LaTeX world

Readers of this article already know, and likely much better than I, how powerful, flexible, and efficient LaTeX is. These are some of the reasons we choose to use this document preparation system. Unfortunately, the rest of the world either is not aware of the existence of LaTeX or is precluded from using it due to restrictions on how document preparation is to be done. The latter restriction usually is imposed on employees to ensure company-wide consistency in document preparation, which is a reasonable concern. Establishing a standard application for document preparation allows diverse groups of people to seamlessly exchange documents.[5]

Microsoft Word has become the *de facto* standard document preparation application. There are many reasons for this, all of which can be debated among computer users, businesses, the general public, educators, sociologists *et al.*, but one of the key reasons, vis-à-vis why LaTeX is not a standard, is that Word uses a GUI as opposed to the command line interface of LaTeX. This makes Word easier to learn for the vast majority of computer users, who are not capable of using the command line for document preparation or simply may not want to do so. As a realist, I do not expect this situation to change. I also think it unlikely that proselytizing for LaTeX converts will be particularly effective, especially in a world in which OUIs ("oral user interfaces") appear destined to supplant keyboard, mouse, and other data entry methods, as well as supplant many aspects of application and system control.[6]

Where then do these facts leave the LaTeX community as a whole? I suggest, in the future, that the community will continue to thrive, as it is now. There are myriad reasons for this, many of which have been discussed above. The most important of these is that for many applications, especially in mathematics, physics, and engineering, LaTeX is a superior document preparation system.[7] Given this fact, those who choose to prepare their own documents using LaTeX, but who also work in the larger world of Word and other document preparation applications, must implement strategies for interfacing these two "worlds."

The specific strategies depend on a number of factors, the most important of which are how the master document is to be prepared and what the final output file format must be. The first factor depends primarily on whether document preparation is done by a single person or in the context of a collaboration. The second generally is dictated by the requirements of the end-user of the document, e.g., a publisher. Publishers specify the file types accepted for publication, which increasingly include PDF. The beauty of LaTeX is the facile compilation of the source code as a PDF file, which renders moot the original document preparation system. This PDF output also circumvents problems with providing documents to those working with computer hardware or OSs different from one's own.

If manuscripts can be submitted for publication using one of many file types, e.g., `.tex`, `.doc`, or `.docx`, and the manuscripts present collaborative work, the decision about source file type can be pre-determined among collaborators, usually using a metric based on ease of group composition. Practical considerations also may factor into this decision. For example, if the contributing author, the one who is responsible for the actual compilation of the manuscript and its submission for publication, is not familiar with LaTeX, then Word often becomes the default document preparation application. However, if I am the contributing author and I want to prepare the manuscript using LaTeX, I can do so by adding files from my collaborators into my `.tex` source file. The easiest way to do this is to ask for `.txt` files. Surprisingly, I often encounter collaborators who don't

[5] Of course, *though seamless in theory*, cross-platform (MacOS vs. Windows) document preparation and management using Microsoft Word (or PowerPoint) remains problematic and vexing in practice.

[6] In fact, there is no reason, theoretically, why such OUIs could not be implemented for source file creation in LaTeX.

[7] It should be noted that LaTeX is not restricted to these fields. It has been used effectively in a broad range of fields, including philosophy, economics, theology, the law, and *neurology*.

know how to create `.txt` files and instead provide `.doc` or `.rtf` files. These then must be converted into plain text.

Conversion can be done automatically using a variety of programs or web-based conversion utilities, although I have found that the fidelity of conversion often is lacking. Post-conversion processing of the resulting text file thus is required to remove hidden or special characters that create serious or fatal errors during LaTeX compilation. I have found that utilities that clean up text, e.g., by removing extra spaces, carriage returns, tabs, or forwarding characters, are especially useful in this regard. Once clean, I then execute a second post-conversion process, usually using global find-and-replace functions, to make sure, among other things, that quotation marks will be rendered properly (i.e., converting " and " into `` and ''), percentage symbols are escaped (% to \%), Greek characters are encoded properly, and one-, two-, and three-em dashes will appear correctly. These conversions can be done quite rapidly, after which the plain text can be pasted into the source file. Additional edits, which usually are minor, then are done in the source file after compilation if error messages are displayed or rendering problems exist.

"Cross-world" bibliography creation and management is a bit more cumbersome, if one defines cumbersome as an author needing more than one library. I maintain two comprehensive reference libraries, one in `.bib` format and one in EndNote format (`.enlp`). In collaborative work in LaTeX, the collaborators agree about whose library will be used and simply upload it to a shared directory containing the manuscript source file. This is trivial. If I must incorporate citation markers from Word documents, regardless of their provenance (Word, Bookends, Mendeley), then I do so manually. This requires a significant amount of program switching (LaTeX↔JabRef), but with the powerful search capabilities of library management software, and patience, the process is straightforward. It should be mentioned that neither world is free of typographical problems within rendered bibliographies. Surprisingly, special symbols, capitalizations, and especially Greek characters, are usually not coded properly in references downloaded from the web, especially in the case of EndNote. I automatically examine each downloaded reference to ensure that the bibliography created by BibTeX or Endnote is an exact rendering of the reference as originally published. After having done this for so many decades, this process has become almost autonomic.

The fundamental principle guiding my cross-world collaborations is "LaTeX takes anything and Word takes nothing." This means that if a manuscript is composed in LaTeX, I can take content from essentially any file type provided by a collaborator and incorporate it into my source file. In contrast, if manuscript composition is to be done in Word, then LaTeX is not used at all.

6 Concluding remarks

Among the community of computer users who are free to choose their hardware, software, and style of use, adherence to their choices may have the flavor of religious fanaticism. This long has been true in the Mac community, not even considering Apple's efforts to portray Mac users as "cool," "with it," or "different." I am a long-time Mac fanatic, not because of such superficial characterizations but rather because of my recognition of a superior operating system and how it makes my computational efforts easier and more efficient. My extensive experience with document processing systems has led me to the same recognition with respect to LaTeX. I look forward to a time when this recognition will be universal.

Acknowledgement

The author gratefully acknowledges Dr. Eric Hayden (UCLA) for helpful comments on this manuscript.

◇ David B. Teplow, Ph.D.
Professor Emeritus
Department of Neurology
David Geffen School of Medicine at UCLA
635 Charles E. Young Drive South
Los Angeles, CA 90095
USA
dteplow (at) mednet dot ucla dot edu
http://teplowlab.neurology.ucla.edu/