CSC 400

Michael Scott

March 2021

1 TeX and LaTeX

TeX is a programming language for writing documents. It was developed by Donald Knuth (ACM Turing Award, 1974). It was originally written in WEB, a literate programming system built on Pascal. Modern versions are in C. TeX computes with macros, with call-by-name parameter semantics.

LaTeX is a macro package written on top of TeX. It was originally written by Leslie Lamport (ACM Turing Award, 2013). The two level structure (LaTeX on top of plain TeX) means there are often two or more ways to do something:

- `\def v. \[re]newcommand`
- `\vskip v. \vspace`
- `\usepackage v. \RequirePackage`
- `\em or \it v. \emph`

Paremeters to macros are enclosed in curly braces. Multiple parameters require multiple sets of braces. Some macros have one or more optional parameters which, if present, come first and are enclosed in square brackets. *Environments* like `itemize` (used above) are just commands whose last parameter is expected to be long, and is given an explicit end marker.

There is a *huge* amount to learn to be a real LaTeX expert. LaTeX itself has thousands of macros. Standard distributions come with hundreds of additional packages that can be included on demand. (My TeX Live installation has over 3000 of them.) Most people start by copying an existing paper and editing it, so they don't have to create a LaTeX doc from scratch. My goal today is simply to alert you to pitfalls that often trip up novice users.

2 Structure and presentation

As in HTML, good writers distinguish between structure and presentation. TeX is pretty unstructured; LaTeX has lots of structure. Macro packages often change the format. So you say `\section{Introduction}` rather than, say,

```
\vskip 3ex
{\Large\bf Introduction}
\vspace 2ex
```

3 Typography

Kerning is the spacing of adjacent characters. This is usually automatic (and TeX is very good at it), but there are pitfalls. Consider

`{\em if}MMM`	*if*MMM
`\emph{if}MMM`	*if*MMM

Be aware that letters in math mode are unkerned. Never ever ever use math mode to create italics: $eff iciency$ v. *efficiency* v. *efficiency*.

Beware that macros eat following characters. So given

```
\newcommand{\Sysname}{Spiffy}
```

we have

`\Sysname is totally awesome.`	Spiffyis totally awesome.
`\Sysname{} is totally awesome.`	Spiffy is totally awesome.

Most of the time you should let LaTeX space things automatically. If you feel an irresistable urge to tweak things, ~ is a non-breaking space `\,` is a narrow space, `\kern` *len* can be used to get *exactly* the (usually very small) spacing you want. Extra horizontal space can be added with `\hspace`, extra vertical space with `\vspace`. Variants named `\hspace*` and `\vspace*` insert space even when there's nothing on the other side to "push against." Each end of line in your source acts like a space. You can inhibit that by ending the line with a comment (a string starting with `%`).

In traditional English text, the space after a sentence is longer than the space between words. So-called "French spacing" makes all the spaces the same size. Some conference formats use French spacing. To make English spacing work, LaTeX assumes that a period, question mark, or exclamation point (optionally followed by quote marks or right parens, brackets, or braces) ends a sentence, unless the preceding character is a capital. If this assumption is wrong, you need to fix things. So say `Fig.\ 3` (Fig. 3) rather than `Fig. 3` (Fig. 3). Similarly, you need to use `Z\@.` to note that the last letter of the alphabet is Z. Otherwise the next space will be too small. (These fixes are harmless with French spacing, so you should use them all the time.)

As a general rule of thumb, avoid typesetting lines longer than about 60 characters. Ten-point type on letter-size paper with one-inch margins looks terrible:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Be aware that double and single quotes are "aimed" in English: you need to say ``aimed'' there. Also, a hyphen (run-time) is different from an en-dash (March 18–23), a minus sign (−3), or an em-dash—all four have their uses; learn them! An oddity of English is that end-of-sentence punctuation is generally placed *inside* quote marks, rather than outside. In a simlar vein, superscript footnote marks generally come *after* the end-of-sentence punctuation (but square-bracket citations go *inside*).

4 Bibliographies

Use BibTeX to create bibliographies. Start curating your own personal bibliography *now*, and use it for all your papers. You can keep your notes in it, too.

Warning 1: The ACM Digital Library is a great source of bibliographic entries (already in BibTeX format), but *these entries cannot be used as-is*, because ACM uses some of the fields in ways that are incompatible with standard BibTeX macros. For example, ACM uses the "address" field in conferences for the location of the sponsoring organization rather than the location of the conference itself. You can always tell when a paper has snarfed refs straight from the Digital Library, because every ACM conference will appear to have been held in New York City and every IEEE conference will appear to have been held in Washington, DC [1, 2]. As a general rule, you should delete the address field, add a month field, change the Unicode en-dash to a LaTeX -- in the pages field, change the location field to address, probably cut the publisher, and maybe expand or abbreviate certain words or move the series name into the conference name.

Warning 2: You also need to put extra braces around any proper names in the title; otherwise BibTeX will uncapitalize them.

References

[1] C. Watt, C. Pulte, A. Podkopaev, G. Barbier, S. Dolan, S. Flur, J. Pichon-Pharabod, and S.-y. Guo. Repairing and mechanising the javascript relaxed memory model. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2020, page 346–361, New York, NY, USA, 2020. Association for Computing Machinery.

[2] C. Watt, C. Pulte, A. Podkopaev, G. Barbier, S. Dolan, S. Flur, J. Pichon-Pharabod, and S.-y. Guo. Repairing and mechanising the JavaScript relaxed memory model. In *Proc. of the 41st ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI)*, pages 346–361, London, UK, June 2020.