What is TEX?

Computer programming languages, such as C, Java or Fortran, all require writing source code using a text editor, and then converting this code into a binary executable. TEX is also a type of computer language, however it is a *typesetting* language, so the source code is converted into a typeset document, rather than an executable.

As with other languages, TEX has its own syntax and set of predefined commands, however there are many different formats of TEX, which can extend the language and provide a more comprehensive set of commands. Just as some programmers prefer to use low-level languages, some TEX users prefer to use plain TEX, and just as some programmers prefer to use high-level languages, some TEX users prefer to use one of the TEX formats. One such format is LATEX, and it is this format that I prefer to use.

I teach LATEX as part of the IT courses run by the University of East Anglia. The participants are staff and postgraduate students from across the university, with diverse backgrounds ranging from mathematics to linguistics. Their motives in learning LATEX tend to reflect their backgrounds. Those from a mathematics background want a system that will typeset complicated mathematical equations, others may want a system that can produce professional looking typeset documents.

Generating a basic document in LATEX is quite straight-forward. You first need to specify what type of document you want to create (is it an article, a book etc?) This is called the document class, and it is the document class that defines the document layout (what the section headings look like and so on). For example, you might use the command `\documentclass{report}` if you wanted to write a technical report. The text that makes up the document is placed inside the document environment, delimited by `\begin{document}` and `\end{document}`. Within this document environment, commands are provided to start chapters (e.g. `\chapter{Introduction}`) and sections (e.g. `\section{Results}`), so you don't need to worry about consistent formatting or keeping track of the section numbers, as it's all dealt with automatically, as is generating a table of contents (`\tableofcontents`) or list of figures (`\listoffigures`). If at a later date you decide to change the style of your document, it can be done simply by changing the document class.

Unlike word processors, TEX follows certain typographical rules which give a professional look to your document with very little effort from the user. You rarely need to worry about minor things such as remembering to put two spaces between sentences and only one space between words, as LATEX will do this automatically, and floating objects, such as figures and tables, will be positioned according to certain rules, so you do not have to keep repositioning them every time you add in an extra paragraph. TEX also encourages users to consider the structure of the document—am I referring to some mathematics $yo - yo$ (`$yo-yo$` note the minus sign and the correct spacing) or am I merely emphasizing the word *yo-yo* (`\emph{yo-yo}` note the hyphen and no spacing)? Some of these points may perhaps seem minor to someone outside the publishing industry, but they all contribute towards the impact of the entire document. When writing technical documents, the presentation as well as the content is important. All too often examiners or reviewers are put off reading a document because it is badly formatted. This provokes an immediate negative reaction and provides little desire to look favourably upon your work.

Suppose you have written a numerical program in C or Matlab. You can save the results in a table, e.g. values and variables separated by commas or spaces, but you can also modify your program to produce a file of TEX commands interleaved with your numeric results so that the TEX program will format your results in a beautiful typeset table. In fact, some applications allow you to save your work in TEX format, for example, the symbolic maths toolbox in Matlab has the function `latex` which will convert a piece of symbolic maths into the appropriate LATEX code, which can then be written to a file using the Matlab `fprintf` command.

For me, as both a mathematician and computer programmer, TEX is an invaluable tool not only for typesetting complicated equations, but also for assisting with the complex task of generating a technical manual with cross-references, bibliography, index and glossaries. Also, TEX can easily be instructed to produce output as Postscript (which some publishers prefer), PDF (which looks nice on a printer or the web), or HTML (which is good for browsing on the web).

Nicola Talbot (nlct@cmp.uea.ac.uk)