What is TEX?

LATEX is a document preparation program for creating documents that are

- beautiful
- easy to correct
- easy to update
- portable to any machine

For simple papers, the essential commands can be learned quickly, and provide easy ways to create sections (and subsections, and subsubsections) that are automatically renumbered, cross-references that have a natural format and are easy to enter, and bibliographies that can easily be formatted in a wide of styles. Beyond these easy-to-use commands is a wide range of capabilities to do just about anything necessary for the typesetting of documents from leaflets to books, including

- creating or importing graphics files
- making complex tables
- writing mathematical equations of all types
- multilingual documents
- nested lists, either enumerated or bulleted

It is important to note that it is a document preparation system, not a word processor. This affects the user in various ways. For instance,

- Rather than simply entering text, you enter text and commands to format the text.
- You can create fully formatted text, if you are writing a book, you can create a file which can simply be copied and bound.

Peter Flom

What Is TEX?

Prelude. The Eiffel Tower is perhaps the world's most beautiful radio tower. It adds beauty to a utilitarian function to such a degree that most of us don't consider its role in radio transmission. And indeed, it was created to be the world's tallest building for Paris' Exposition Universelle in 1889. It is an example of the fusion of modern industrial design, a open steel structure, with the human need to create beautiful objects. At a different scale, the Eighteenth Century Windsor chair combines a light, strong beautiful structure to fulfill an everyday need.

Books share both the public nature of the Eiffel Tower and the personal aspect of a Windsor chair. Whether a book's typography transparently supports the text or burdens readability with excess decoration depends on both the designer and the capability of her typesetting system. TEX offers the designer the capability to create beautiful books that are also highly readable.

Getting Started. I had known about TEX for twenty years before I used it for a project. And for years, I could produce documents that were attractive and useful with a word processor—*PC-Write*, *MS Word*, *OpenOffice*—but did not need the capability to produce books or pamphlets. After my wife and I became docents at the Los Angeles Zoo and Botanical Gardens (`http://www.lazoo.org`), we became involved in various specialities within the docent program. I joined the Botany Committee, a small group of docents whose goal is to present the Zoo's varied plant collection to patrons. The Committee's chairman mentioned that she and another docent had written a self-guided botanical tour complete botanical illustrations. "How do I get *Word* to include my drawings and turn this into a booklet?" she asked. "You don't," I told her. "You need a typesetting system."

Off and Running. I bought Kopka and Daly's *Guide to LATEX*[1], installed TEX and LATEX from the enclosed CD and tried a few of the book's exercises. I also licensed *WinEdt* for editing LATEX documents. This combination was sufficient for several projects and only upgraded when I joined the TEX Users Group (TUG) and received updated versions TEX.

I saved the *Word* files of the botanical tour as plain text files and opened them in *WinEdt*. Next, I scanned the botanical illustrations and then edited them. With text and illustrations available to *WinEdt*, I did a first run and showed the results to the Botany Committee. "This looks so professional," was the initial reaction. Having seen what was possible, expectations rose and we generated several alternatives for the page layout, fonts, table of contents and index.

Looking at design alternatives helped me clarify my own values. The Eiffel Tower and Windsor chair are metaphors of what I find attractive: the structure itself should be beautiful and useful. On a printed page, this means the placement of text, fonts, titles and illustrations without extraneous elements. Viewed as rectangles on the page, the arrangement should be balanced and harmonious. Readability is the primary measure of quality. While layouts are matters of personal taste, there are many helpful references. The documentation for Peter Wilson's memoir class[2] is an excellent overview of book design.

Finale. After weeks of editing and experimenting with layouts, we settled on a final design. TEX is not a monolithic system and can evolve as design goals change. This is done by changing document classes and calling various software packages. There seems to be a package for anything I need, but it's important to know that developers are often revising and writing new packages that add features.

A Botanical Tour of the Los Angeles Zoo and Botanical Gardens was published in November 2004 and is available in the Zoo's gift shops. Many docents are using it as part of the tours for schools and other groups. Earlier this spring, a Botany Committee colleague showed me a botanical tour booklet from another organization. She pointed to the excellent color photographs, but mentioned that the coated paper used for the booklet made it heavier than she liked. She pointed to the page breaks, the index and other features that her now-critical eye immediately spotted. I suspect the source of her comments came from the awareness she developed by seeing what TEX could do when applied as intended. Bravo *Panthera leo*!

Joe Hogg (`Joseph.Hogg@bigfoot.com`)

References

[1] Helmut Kopka and Patrick Daly. Guide to LATEX. Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, MA, USA, 4th edition, 2004. ISBN 0-201-17385-6.

[2] Peter Wilson. The Memoir Class for Configuarable Typesetting User Guide. The Herries Press, Normandy Park, WA, USA, 6th edition, 2004. Available at `http://tug.org/tex-archive/macros/latex/contrib/memoir/memman.pdf`.

What is TEX?

When someone asks you, "What is that word processor you are using?", have you ever wished you had a one page description to hand them? Here is one such brief introduction.

Technically speaking, TEX is a computer application for laying out ("typesetting") pages of text based on the text the user has written and other instructions the user has given. In other words, like the ubiquitous Word, TEX decides where to split lines of text, where to start a new page, and so on.

TEX was created by the noted computer scientist and innovator in computer-based typesetting, Donald Knuth, who developed it over many years and made it freely available to the world. Knuth provided for and invited *users to enhance TEX*, and it has in fact been extended to many types of documents (letters, articles, books, slide shows, concert posters, etc.) and many domains (chemistry, chess, music, poetry, linguistics, critical editions, etc.). The most popular enhancement to TEX is called LATEX, which supports most needs in a straightforward manner.

TEX is always used in conjunction with a text editor that lets you move around in your document adding new text, revising text, and adding instructions for how you want the text formatted. Unlike Word, TEX is available from a variety of commercial, shareware or free sources, configured in ways that different users find suitable (`www.tug.org/begin.html`). But, at their core, all of these have the same TEX "typesetter" from Knuth, and most documents can be moved from one TEX implementation to another without trouble.

Another way that TEX is different than Word and many other word processors is that all typesetting instructions are explicitly typed into and shown in the document file. Here is a short, but perhaps instructive, example LATEX file:

```
\documentclass{article}
\usepackage{a4}
\usepackage{times}
\begin{document}
This is a small example of a
two paragraph document.

This is the \emph{second} paragraph.
\end{document}
```

And here is the formatted output (plus a page number, not shown):

This is a small example of a two paragraph document.
This is the *second* paragraph.

Some points to note: Where Word uses an extra strike of the Enter key to indicate a new paragraph and this information is hidden after the last character of the paragraph (or with the ¶ sign), LATEX uses a visible blank line to indicate a paragraph break (see the example). In Word you can select the style of document, paper size, and font with various menu commands; in LATEX you type these instructions into your file as shown in the first lines of the example (A4 paper, Times fonts). In Word you might type control-I to turn on italics, then type a word, and then type control-I again to turn off italics; in LATEX you indicate *emphasis* explicitly (with `\emph`), as shown in the example's second paragraph.

Our purpose here is to explain what TEX is — not to compare the power of TEX with the power of other types of word processors. Suffice it to say that many people find TEX and its companions useful in a wide variety of applications.

Because TEX from any source has the same extendable basic capability and because the capability for enhancement is very explicit, users are motivated to enhance TEX and there is tremendous sharing of enhancements among TEX users. The Comprehensive TEX Archive Network (CTAN) is a massive collection of TEX enhancements for various application domains, document types, and typesetting flourishes. Discussion groups, for example, `comp.text.tex` and `texhax@tug.org`, provide forums where users can seek help from other (some very expert) users. The TEX Users Group (TUG) and other national user groups provide resources such as user conventions and journals (like this one).

If you aren't already using TEX, you might try proTEXt for Windows (`www.tug.org/protext`), TEX Live for Unix (`www.tug.org/tex-live`), or gwTEX for Mac OS X (`www.rna.nl/tex.html`). When asked how much to install, it's simplest to select *all* packages.

After getting a system installed, a first test is to run `pdflatex sample2e` and view the resulting `sample2e.pdf`. Then start reading documentation, either online in "Not So Short Introduction to LATEX" (`www.tug.org/tex-archive/info/lshort/`) or in print in Kopka & Daly's *Guide to LATEX* (`www.tug.org/books/`). Reading the `sample2e.tex` source file itself can also help in beginning to understand LATEX. If you need help with a specific problem, check the TEX FAQ (`www.tex.ac.uk/faq`). There are *many* other resources available; the sites listed here are a starting point for exploration.

David Walden (`dave@walden-family.com`)

What is TEX?

TEX is a program I use to make my living typesetting books and journals. TEX is software that I'm fortunately not required to install (I have a husband who does that). Me, I'm an end-user. Note that when I say 'TEX', I mean the program which is at the heart of 'plain TEX' and 'LaTeX' and all the other variants out there (e.g., ConTeXt, pdfTeX); it's a bit confusing to have a program name being used this way, and then find out that what I'm really mainly talking about is 'LaTeX', whereas 'TeX' for someone else means they're using the 'ConTeXt' variant. When speaking in generalities, 'TeX' is sufficient.

TEX, just like MS Word, is only part of the ever-expanding context in which text, and documents, are being handled these days. It is not an archival or ancient program confined to math and science typesetting on paper; it can be integrated into the world of PDF, HTML, XML, and so on, because of the efforts of users around the globe, who believe, some quite fiercely, in high-quality output being available to everyone. Yes, it does require a computer, and these days, access to the Internet, so one can argue that it's not really a tool for the entire planet. But once you get the infrastructure, you can make beautiful text appear on paper. You can typeset the textbooks and literary materials that your clients — or your people — need. Beholden to no-one's pockets, or equipment, or fonts. Sounds almost subversive, doesn't it `;-)`

TEX is also the only program I've ever heard of where files coded up in 1983 can still be run. So it's made it possible for me to be rather stunningly lazy — I haven't had to learn much about any other software for almost the past quarter century (you figure it: 1983 till 2005). Well, other than all those TEX add-on packages `;-)` And in almost all cases, they've made things easier, not harder, so the upgrade experience has been very pleasant indeed.

TEX is the entry to a community of awfully kind, awfully clever people who, when they see a need, write add-on packages that expand TEX's capability and make the add-ons publicly available. Sometimes we meet at conferences — often we meet via e-mail — where I try to thank them once I begin to use their offerings. These add-ons sometimes address specific, narrow needs (such as 'soul', which makes spacing out words, letterspacing, underlining, and striking out easier to do) and sometimes major, general enhancements, such the graphics packages ('graphics' and 'graphicx').

TEX is of course frustrating at times — very definitely a program which does exactly what you *tell* it, as opposed to what you'd *like* it to do. But it's also very gratifying in what it does to ASCII characters on a piece of paper (or on-screen). The defaults are pretty damn'd good; with perseverance and good packages, the results are sometimes spectacularly rewarding.

TEX is not only for math or science or the technical fields — it's for all typeset material, be it on paper or on screen (granted, it may be overkill for letters, if that's all you do; but if you want to have letters that seem to stand out because of how the characters sit on paper, then it's a neat thing to do).

TEX is not confined to English (but sometimes it seems to be constrained by it). I periodically require something like a string of Hebrew or Cyrillic or Greek or, most recently, Polish — and it's all there for me. Granted, it's much easier because I have good reference tools.

Books on TEX, once few and far between, are now available in good variety, either general or very focussed — and not just in English `;-)` I find the revised *LATEX Companion* one of my mainstays, along with its first version. Kopka and Daly's *Guide to LATEX* is also necessary — where one explanation doesn't get through to me, the other often does. So resource books and printouts of many free on-line guides are a necessary component to learning and using TEX to do what you want (as opposed to that 'what you tell it to do' thing, mentioned above).

And then there are the internet lists — the usual crew of (c.t.t.) and texhax, ling-tex, typo-l. But trolling via Google also leads you to a massive number of hits that quickly tells you that this is a program with depth — years and years of experience exist out there, and lots of it within all kinds of organisations and associations, institutions beyond the universities, and of course, many well outside the math/science domains which TEX is so often mistakenly relegated to.

TEX isn't word processing, where content and form (two favourite aspects in the linguistics world) are so intermingled and entwined, it's sometimes very difficult to pull them apart and understand the distinction. But that distinction is one of the main aspects to TEX which newcomers are often not (made) aware of. TEX is the second half of the equation — the manipulation of the characters on-screen, the saving of the files, these are all done by a separate program, your text editor.

TEX is something you learn, and appreciate, in layers. If you want to do your thesis in it, start

using it for your first outline! Use it for your reminder notes and letters. Approach complex layouts or equations or tables in layers — get the subsets right first, then build the fraction, or the square root; do the column headings and the first row of the table and make them look right first before adding the 10 or 20 rows of data. Same thing for a book. It's seat-time that counts, so pull your chair over to the keyboard, fire up the editor and process the first bits to see what you get. Don't be a perfectionist and spend hours trying to move some element in the equation over a squidge, just to make it look right — leave the futzing for later. Same for tables. Write and do minimal coding while you're writing, to keep the distractions of TEX to a minimum. If it processes, be happy — and then write some more!

And if it's someone else's splendid work which you have to put into TEX, it's pretty much the same thing: layers. Process a paragraph or equation at a time, to keep on top of mistakes. It's extremely depressing to process a file only after the whole thing's been coded up, untested, and then find that you've made a silly or perfectly honest blunder for 53 equations and 12 tables. Not to mention all those taunting "Undefined control sequence" messages due to typos — often uncannily consistent. These are not cautions which anyone passed on to me — this is direct experience talking.

So, to repeat myself: What is TEX? For me TEX is the software that has allowed me to earn my living typesetting books and journals, since 1983. That's 22 years of TEX — and counting.

Christina Thiele

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)

What is TEX? . . .

. . . by Hans Hagen

Here I reflect on some of the remarks made in the other answers. It's not so much meant as critique, but more as a trigger for further discussion. If you only want to know my answer, you can skip to the last paragraph.

All TEXs are equal . . .

. . . but some are more equal than others.

The answer to this question is not always easy to give. Peter Flom for instance starts his description with "LATEX is . . . " and thereby makes TEX the program equivalent to LATEX the macro-package.

This kind of equivalents are rather common, and many users don't know the difference between PDFTEX (the program) and PDFLATEX (the macro package). This is made even more confusing by the fact that on many systems invoking PDFTEX without explicit macro package mentioned, will load the plain TEX format.

Yet another confusing factor is that TEX is used to describe both a language and its associated interpreter/typesetter. And then there is the TEXbook, which not only describes these two, but also the plain TEX format that ships with the system. So, in the case of TEX we need to distinguish:

language	primitive commands combined with macro definitions
program	interpreter and typesetting engine
package	collection of macros loaded on top of the built in language

If we translate that to commands and files, we end up with:

language	TEX, program specific extensions
program	TEX, PDFTEX, XETEX, ALEPH (OMEGA)
package	plain, AMS-TEX, LATEX, LAMS-TEX, CONTEXT, . . .

LATEX users have several options to invoke TEX:

latex	the (pdf)TEXengine with the LATEX macro package preloaded
pdflatex	idem, but this time the output will be a pdf file
xelatex	the LATEX macro package loaded into the XETEX engine
lambda	the LATEX macro package loaded into the ALEPH or OMEGA

For ConTEXt users life is different. They use a wrapper and thereby use calls like

texexec –pdf somefile.tex the CONTEXT macro package loaded in PDFTEX
texexec –xtx somefile.tex idem but this time loaded in XETEX

For a long time TEX produced DVI output only and one had to postprocess this into a format suitable for a printing engine and for quite a while POSTSCRIPT output was quite popular. Nowadays PDF is the format of choice and PDFTEX can produce this directly. There is no need for a backend like DVIPS (to produce POSTSCRIPT which itself can be converted in PDF) or DVIPDFMX (which converts DVI into PDF).

No matter how you use TEX, you need to keep in mind that when you talk of in terms of what you invoke on the command line, this may not be what others experience. Think of this: by default PDFTEX produces DVI output and unless told explicitly to behave differently, it is just like good old TEX, and in DVI mode still needs backend. Confusing eh?

TEX can produce beautiful documents . . .

. . . but does not give you guarantees.

When people advocate TEX they tend to praise the output of this program as being of high quality and beautiful. In a way this is wishful thinking. There is no doubt that TEX can produce documents that qualify as such, but in practice many documents look just as 'texy' as MSWORD documents look 'wordy' and QUARK output looks 'quarky'. The variations in style (design), font usage and formatting is not that large and a direct result of using the same predefined layout over and over again. For instance, texies make jokes about POWERPOINT presentations (since they can be recognized by the features used) but don't realize that most of their own work stands out in a similar way. They rightfully claim that TEX does a good job on breaking lines into paragraphs but are more tolerant to funny vertical spacing resulting from handcrafted commands that interfere with what the macro package tries to accomplish. Because TEX can do such a good job on justifying text, words sticking into the right margin (overfull boxes) stand out pretty noticeable. I don't want to count the documents posted on the web that demonstrate this 'feature'.

TEX is easy to use . . .

. . . but not everything is easy.

TEX can do clever things with graphics and fonts, but the fact that there are so many questions posted to mailing lists demonstrates that this is less trivial than long time users suggest when they praise TEX to new users. TEX can be an easy system to use, but also a painful experience when one wants to do real clever things. Some things are simply tricky, no matter what system is used.

An important property of TEX usage is that on the average the audience is quite willing to help

newcomers. Nearly always users themselves choose to use TEX. Therefore they are willing to spend time on learning the system.

A strength of TEX and its packages is that one can find resources on usage in bookshop as well as on the web (manuals, faqs, wikis, mailing lists, new groups, etc).

TEX output is always good . . .

. . . it's only you who can mess up things.

In most computer languages, one has to explicitly tell the machinery that some text should be output. Not with TEX. Anything that expands to text will become visible somehow. One can make fun of the fact that those who use word processors may end up with inconsistent spacing, i.e., duplicate spaces in the result. With TEX, you should not be surprised when spacing becomes messed up too due to funny spaces in macros. Be careful of making false claims and dangerous jokes.

In his answer David mentions the visual separation of paragraphs as a characteristic of TEX. He also explains the difference between changing fonts in TEX and for instance MSWORD. In discussions about the the differences between word processors and TEX, one may argue that in a word processor one never knows where exactly a change of fonts takes place: is the space preceding a bold word bold as well or not. But in a way TEX's ways of dealing with font changes or changes in attributes is not less confusing than e.g. MSWORD's.

Say that we want to narrow a paragraph of text.

```
\def\StartNarrow{\bgroup\leftskip1em\rightskip1em\relax}
\def\StopNarrow {\egroup}
\StartNarrow some lines of text \StopNarrow
```

In such cases grouping is used to make sure that we limit the scope of the feature change. However, in this case, you will not get an narrowed paragraph, unless you provide an explicit paragraph end.

```
\StartNarrow some lines of text \par \StopNarrow
```

The solution is to change the definition to:

```
\def\StartNarrow{\bgroup\leftskip1em\rightskip1em\relax}
\def\StopNarrow {\par\egroup}
```

There are many spacing related features that work this way and the effects are not always clear source code. What is true for one document style may be false for another. It all depends on how your TEX is set up and how well macro writers coordinate their work.

TEX is stable and does not change . . .
. . . but do we really want that to be true?

Don Knuth's wishful thinking that TEX the program would be extended for whatever intended purpose has not been fulfilled. In good old TEX there are two examples of extansions: specials and writes. Specials provide a way to control the backend and are used to achieve special effects like color or to insert additional material like graphics. Without specials, we would have been in big trouble and still manually have to cut and paste copies of graphics. Writes are a neccessity for tables of contents, cross references and other features that demand a feedback loop into a successive run. Normally their usage is hidden by macro packages. By providing these examples of extending TEX Don actually made TEX much more future proof.

There are some non-Knuthian extensions, but not that many. For instance, nobody bothered to write a subsystem for typesetting chemistry as companion to the math typesetting subsystem. It has to be done in macros. So far nobody came up with real robust extensions for numbering lines, parallel output streams and other goodies for the humanities and linguistics. Again, one has to revert to macro writing, in this case of a particularly nasty kind. You may call yourself lucky that publishers are not that demanding.

Nevertheless, one may expect extensions and currently the most prominent ones are ε-TEX, which provided some extra programming features, PDFTEX, which kick-started TEX into the 21th century by providing marginal kerning and optical scaling aka *hz* (Hermann Zapf) optimization as well as a full featured PDF backend, and XETEX which boosted TEX towards unicode and opentype fonts. It's only by efforts like this that TEX is still alive and kicking.

Of course, macro packages play their role as well, and as long as we can find people who feel challenged by beating a language and feature set that does not really match today's programming techniques, we're safe. The competition is not doing much better, simply because the problems that we're facing haven't changed much.

Christina mentions that one of the nice things about TEX is that it's virtually bug free. But, Christina, I have to disappoint you: if your 23 year old document depends on `\leaders` behaving like they did at that time, you may have to find yourself an old copy. Among the most recent bug fixes was a fix to this mechanism and it may make a difference, although the changes are small. (Actually, it does make a difference for the TEXbook.) But . . . in general your claim is true, unless of course you forgot to save your old pattern files, along with an old copy of your macro package. And, are you sure that the metrics or appearance of the fonts that you use didn't change? Btw, there are other examples of stable programs: computer language compilers and interpreters, and this is exactly what TEX is.

What is TEX . . .

. . . and why do I like it?

TEX is a system which permits you to create your own typesetting environment. In its 25^+ year existence various environments evolved, for instance LATEX and CONTEXT so users can start right away. Both can be used comfortably in editing programs and previewing your document is no problem. You can extend their functionality or decide to stick to what is offered. You are in control. Okay, some aspects are difficult to deal with, but that is a direct result of the rich functionality.

If you stick to the paradigm of the particular environment you are using, i.e., keep your document code clean, your documents are stable over a long period of time. You edit in a structured way, you define your layout in an abstract manner, and can produce the final result on any platform you want. You can distribute your source code and let others work with it, thanks to the availability of distributions, support mailing lists, a friendly community, user groups, books and manuals. If you want to use their full power, it will take a while to learn such systems, but for most users TEX is a tool that they will use their whole life. Don Knuth gave us the ability to "go out and create beautiful documents" but you need to pay some attention to get it done. He also formulated the important boundary condition that in 100 years from now, the documents should still be valid input for a TEX processor. Such life long validation gives a comfortable feeling.

Beware: in the process, you can get hooked. And: you need to keep an open mind for the shortcomings, myths and somewhat strange solutions that come with TEX.

What is TEX ?

When I was invited to write a brief article entitled "What is TEX ?", I was delighted : the possibilities seemed endless. But when I tried to formulate all the ideas that were bubbling through my head, I found it *considerably* harder to encapsulate them in a single pithy phrase. In the end, I decided to use an analogy, and one with which I hope some readers at least will be familiar : TEX is like a Meccano set, but with one *very* importance difference.

What does this mean ? Well, let me explain to those unfamiliar with the term what a Meccano set is. It is (was) a construction kit for "children of all ages", and was for many of us our first introduction to mechanical engineering. The simplest set was a "No. 1" (possibly known as an "Erector Set #1" in North America, a name far too open to misinterpretation for it ever to be used in the UK !), which consisted of a few metal rods, strips, plates, pulleys, and a hank of green cord. It also contained some nuts and bolts, the former being square- (rather than hex-) headed and therefore requiring a special spanner which was also supplied, as was a screwdriver since the "bolts" had slotted heads. With just these few basic bits and pieces, a child could build absolutely anything that came into his (or her) head : a crane, a lorry, a boat — the possibilities were endless but limited by the few parts (and fewer *types* of part) in a No. 1 set.

With his appetite whetted, a child with a No. 1 set was like an alcoholic at a "sip-and-spit" wine tasting : he just couldn't get enough. His parents would be begged incessantly for a No. 1a ("but then I can build a *'plane*, Mummy !"), and then a No. 2, a No. 3, and so on until finally (and for this he would have to promise to forgo all birthday and Christmas presents for the next five years), he would finally acquire the Holy Grail : a No. 10, in a solid oak case (lesser sets were supplied in cardboard boxes, although I believe that a No. 9 came in something a little special : I don't know for sure, because my parents' pockets became exhausted long before I was even half-way to Nirvana ...).

OK, enough of Meccano : what has this to do with TEX ? Well, to my mind TEX is like a No. 10 Meccano set : there is *nothing* that you can't build with it, given enough time and patience. "Why 'build' ?", I hear many of you ask. Well, TEX is also rather like a newborn child : it knows very little, but it has an almost infinite capacity to learn. In order to get the most out of TEX (in fact, to get anything worthwhile out of TEX at all), it is necessary to invest a not inconsiderable amount of intellectual energy. You can, of course, take the easy way out and allow others to invest that energy for you (by using, say, LATEX or ConTEXt), but to be honest, why would you want to ? The greatest joy in using TEX (to my mind) is the joy of persuading it to do *exactly* what you want. This is hard enough using TEX itself, but it is virtually impossible once you allow format writers (such as Messrs Lamport, Mittelbach, Hagen, *et al.*) to act as intermediaries on your behalf. Only by using TEX as God (oops, Knuth) intended, with an absolute minimum of intervention at the format level, will you ever be able to coerce it into satisfying your every whim.

OK, so TEX is naïve, and needs a fair amount of work in order to coerce it into doing something worthwhile. So are many other systems, yet they don't have the cult following of TEX. What is it that sets TEX apart from the crowd ? And what is it, for that matter, that makes TEX so *very* different from the Meccano set analogy that I have been using so far ? Well, imagine (if you will) a No. 10 Meccano set, taken out of its box and carefully arranged on a very large table with each set of identical pieces separated from every other set. Look at it carefully, and what stands out (apart from the uniform reds and greens in which everything that doesn't rotate is coloured) ? Nothing ! There is no one set of identical pieces that is in some way *fundamentally different* to all of the others. Each has its rôle, none is central (apart, perhaps, from the nuts-and-bolts and maybe the hank of string ...).

Now perform the same experiment on TEX (it will have to be a *Gedankenexperiment*, I am afraid). What do we see ? Well, one pile consists of primitives : commands built into TEX itself which have *a priori* meanings. We have another pile consisting of macro-related bits and pieces (that is, facilities for defining commands in terms of other commands). We have a third pile related to command *execution* (that is, what happens when a command finally reaches TEX's innermost core). And finally we have a black box, on the outside of which is printed "typesetting engine : unauthorized opening will invalidate all warranties, express or implied". And it is this black box that makes TEX unique, and to which everything else is peripheral.

So now we can really answer the question : "What is TEX ?". It is a typesetting system *par excellence*. It is capable of producing printed copy which equals or excels in quality that produced by any of its peers, whether they be public domain, shareware, or incredibly expensive bespoke systems. Surrounding this are a number of peripheral units that can be changed in any way that the user thinks fit. Let me give you just one example. Suppose you don't like TEX's syntax : you find backslashes and braces ugly and inelegant. You are used to writing web pages, and you find angle brackets and CSS notation intuitive and easy to use. *Then implement it !* There is *nothing*

in TEX that says you must use Knuth's original syntax : you are free to implement any other syntax that you choose.

To conclude, TEX is anything that you want it to be. At its heart is a superb typesetting engine, surrounded by a flexible and powerful interface that you, the user, can tailor in any way that you wish. You can use TEX to produce anything, from a one-page letter to your bank manager to a multi-volume work on the world's writing systems. Ask not what TEX can do for you : ask rather whether there is anything that you *cannot* do with TEX !

Philip Taylor
07-Jul-2005