StarTeX — a TeX for beginners

Dag Langmyhr
Department of Informatics
University of Oslo
Norway
dag@ifi.uio.no

Abstract

This article describes StarTeX, a new TeX format for students writing their first report and other novice users. Its aim is to provide a simpler and more robust tool for users with no previous knowledge of TeX and LaTeX.

The problem

Students taking courses at our department are required to write short project reports, and LaTeX (Lamport, 1994) has been the preferred tool. Several years experience has, however, shown us that LaTeX is not ideal for this.

This project report is the first encounter most students have with LaTeX, and they face many problems:

- The major problem is the error messages. They are very terse at best, and since they are sometimes produced by LaTeX and at other times by TeX, understanding the messages requires reasonably good knowledge of both systems. Most students tend to look only at the line numbers when examining their error logs.
- LaTeX is not very robust; trivial syntax errors can cause a serious burst of confusing error messages, like when you forget a `\\` prior to `\hline` in an `array` environment.

  You can also experience undesired effects if you use the commands incorrectly, for instance if you write

  `\abstract{`*text*`}`

  rather than the correct

  `\begin{abstract}`
  text
  `\end{abstract}`

  This error produces no error message, but will cause the whole article to be set in a smaller font.
- LaTeX does not hide the primitive commands of TeX, making it possible for the users to access them accidentally. For example, one of our users defined a macro for her name:

  ```
  \def \else {Else Hansen}
  ```

  This error alone produced more then 100 error messages.
- LaTeX uses ten special characters: `#`, `$`, `%`, `&`, `~`, `^`, `_`, `{`, `}` and `\`. Users need to remember that these characters are special, and they must learn which commands are necessary to produce them if they are required in the text. Fewer special characters would be an advantage.
- The command notation `\xxx` used in LaTeX often causes problems with the space following it.
- LaTeX has borrowed its error recovery philosophy from plain TeX: the user is expected to manually correct each detected error to allow LaTeX to proceed. The problem with this approach is that you will get many confusing error messages if you do not correct the error properly.

  None of our students use this interactive recovery facility; they either restart after having discovered the first error, or they let the processing run to completion without any interaction. An automatic error recovery scheme like that employed by compilers would be a great benefit for these users.
- LaTeX provides a mixture of structural mark-up commands as well as visual mark-up. The advantage is that experienced users can achieve the visual appearance they desire; the disadvantage is that less experienced users — particularly those who have used other document processing tools — spend too much of their time trying to coerce LaTeX into producing exactly the layout they think is proper.
- LaTeX is a large system and running it is not as fast as for instance plain TeX. For instance, a 2½ page sample document takes from 2.8 seconds on a Sun SparcStation20 to 8.7 seconds

on a Silicon Graphics Indy. Since novice users tend to process their documents very frequently to remove errors or test the effect of a feature, execution times do matter — even in this range.

The requirements

All these problems indicate that LaTeX in its present form is not the tool we want for our students, at least not for their first report. We want a document processing program with the following properties:

- It must be based on TeX to achieve the desired quality in mathematical formulae.
- It should use a different notation for its mark-up commands; one which caused less confusion concerning spaces and has fewer special characters.
- It must hide all the internal TeX commands; this is the only safe way to avoid students using them accidentally.
- It must be small and easy to understand, so that it may easily be adapted to the particular need of each installations.
- It should contain structural mark-up commands only, and no visual mark-up.
- It should be robust.
- It should produce better error messages. If possible, no messages from TeX should ever appear. If this is impossible, error messages from TeX should be preceded by a message produced by the new tool.
- Since most students tend to just disregard all messages about under- and over-full boxes, it should try to reduce the number of such messages.
- It should run in nonstop mode and use automatic error recovery to detect as many genuine errors as possible.
- It should be as fast as plain TeX.
- The command handling should be insensitive to uppercase and lowercase. This is not an important issue, but case confusion has caused problems for some.

The solution

Attempting to achieve the goals mentioned above, StarTeX was designed. The name was chosen to indicate that it was a *Starters' TeX*.

StarTeX is a new format, and is thus a simple cousin of AMS-TeX (Spivak, 1986) and LaTeX. It is built on top of the plain TeX (Knuth, 1991) commands.

The notation

At the EuroTeX conference in Arhem in September last year, Philip Taylor (Taylor, 1995) proposed a different notation for (LA)TeX commands:

`<xxx>` rather than `\xxx`.

I decided to use this notation in StarTeX as it solves many of our problems:

- Spaces following the command are no longer a problem. There is no need for special rules like "When a space comes after a control word, it is ignored by TeX." (Knuth, 1991, p. 8).
- Only one special character is needed: `<`. The characters `#`, `$`, `%`, `&`, `~`, `^`, `_`, `{`, `}` and `\` can be defined to be just ordinary characters.
- The command name may contain almost any character, not just letters.
- The scheme is easy to implement: all that is required is to make `<` an active character, and let the corresponding command regard everything up to the following `>` as a parameter.
- Since all commands are called through this interface, it is easy to make all internal TeX commands invisible.
- It is easy to check whether the user command is defined, and provide suitable error recovery if it is not.
- It is easy to `\lowercase` the user command, thus making the command handling insensitive to case.
- This command notation is the same as in HTML (Raggett, 1995) with which many students are familiar.

I could have used any bracketing symbol pair, like `[xxx]` or `{xxx}` or `/xxx\`, but I chose `<xxx>` because it resembles HTML and because `<` and `>` are not used very frequently.

Command parameters A few commands need a parameter to specify non-printing matter like a file name or a label. I chose to use square brackets for this, as in

`<ref>[`*label*`]`

Using a special notation indicates more clearly that the parameter is not to be typeset.

The command set

The set of available StarTeX commands was chosen with the following aims in mind:

- There should be sufficient commands for writing a student report, but otherwise there should be as few commands as possible.

- There should be no commands for visual mark-up, only structural specifications.
- The commands should have a form that makes them easy to check for errors, and to automatically recover from the errors.

In table 1 are listed most of the StarTeX commands with their LaTeX counterpart.

Paragraph separation It was decided to use `<p>` to separate paragraphs, as in HTML. Even though the blank line used by (LA)TeX is easier to type, it does cause problems with indentation of the paragraph following an environment like a list. Using `<p>` alleviates this problem.

Another advantage of using the `<p>` notation is that it can be employed as line separator (like `\\` in LaTeX) in environments where the concept of paragraph makes little sense, as in the `<title>` or `<author>` environments. This provides a double benefit: a special command for line breaking is no longer necessary, and using `<p>` in a `<title>` environment is now legal.

Font selection A few commands for font selection are necessary, but my belief is that `<b>` (for **bold text**), `<i>` (for *italic*) and `<tt>` (for `typewriter text`) form a sufficient set of commands. The commands may of course be nested to provide for instance *`italic typewriter text`*.

Some might argue that these commands are visual rather than structural, and that the HTML approach of providing a wider selection of structural commands like `<dfn>` for definitions, `<em>` for emphasis, `<kbd>` for keyboard input and `<samp>` for literal characters, is more logical. My own experience is that there are seldom enough definitions to suit my needs, so I will for instance use a specification like `<strong>` when I really want to indicate a reserved word in a programming language. Providing a few simple type changing commands is simpler.

PostScript figures Since nearly all figures used in LaTeX documents at our department are PostScript files, it seems reasonable to specialize the interface for this. The notation

`<psfig>[`*file name*`]` *caption text*`</psfig>`

was chosen as only two keywords were necessary. All figures are automatically scaled and they float to the top of the current or following page.

Tables The notation for tables was also chosen to be as simple as possible, and to ease error detection and recovery. Only very regular tables are catered for, but this is the price one has to pay for a simple notation.

Table 2: A small table sample

Index	**Data**
12	199
17	0

A table is a complex structure, with entries in columns within rows inside the table, but a notation was found which will seldom give grouping errors:

```
<table>caption text
<row>text<col>text<col>...
<row>text<col>text<col>...
    :
</table>
```

Every `<row>` starts a new row, and each `<col>` starts another column. The text prior to the first row is regarded as the table caption.

The number of columns is determined automatically. All columns are centered, and a grid of horizontal and vertical rules is always added. For example, the code

```
<table>
  A small table sample
  <row> <b>Index</b> <col> <b>Data</b>
  <row> 12 <col> 199
  <row> 17 <col> 0
</table>
```

will generate the table shown as table 2.

Document styles All documents need some adaption to conform to a particular style. I propose to let the user decide this by stating

`<style>[`*style file*`]`

The style file is written in plain TeX and contains the necessary definitions and modifications. Since the user has no visual mark-up commands at his or her disposal, all design decisions are made by the style designer. This makes it easier to have all reports conform to the approved standard.

My hope is that each site using StarTeX will develop styles of their own. These styles should be comprehensive, so the user should only have to specify that one style. For instance, our style `ifi-report` defines

- the page size (A4 paper),
- Norwegian format of `<today>` and `<now>`,
- Norwegian translations of fixed texts like “Figure” and “Table”,
- the page headers and footers, and
- various minor typographic details.

	StarTEX	**LaTEX**
Document bounds	<body>*text*</body>	\begin{document}*text*\end{document}
Document style	\style[*style file*]	\documentclass{*style file*}
Document head	<title>*text*</title> <author>*text*</author> <info>*text*</info>	\title{*text*} \author{*text*} \date{*text*}
Font change	*text* <i>*text*</i> <tt>*text*</tt>	\textbf{*text*} \textit{*text*} \texttt{*text*}
Paragraph break	<p>	⟨blank line⟩
Mathematical formula	$*formula*$ <displaymath>*formula*</displaymath>	\(*formula*\) \[*formula*\]
Sectioning	*text* *text* *text* *text*	\section{*text*} \subsection{*text*} \subsubsection{*text*} \paragraph{*text*}

Itemized list	<list>
	  <item> ...
	    :
	</list>
		\begin{itemize}
		  \item ...
		    :
		\end{itemize}

Enumerated list	<list>
	  <numitem> ...
	    :
	</list>
		\begin{enumerate}
		  \item ...
		    :
		\end{enumerate}

Description list	<list>
	  <textitem>*text*</textitem> ...
	    :
	</list>
		\begin{description}
		  \item[*text*] ...
		    :
		\end{description}

PostScript figure	<psfig>[*file name*] *caption text*
	</psfig>
		\begin{figure}
		  \caption{*caption text*}
		  \begin{center}
		    \epsfig{file=*file name*,...}
		  \end{center}
		\end{figure}

Table	<table>*caption text*
	  <row>*text*<col>*text*<col>...
	  <row>*text*<col>*text*<col>...
	    :
	</table>
		\begin{table}
		  \caption{*caption text*}
		  \begin{center}
		    \begin{tabular}{|c|...}\hline
		      text& *text*& ...\\ \hline
		      text& *text*& ...\\ \hline
		        :
		    \end{tabular}
		  \end{center}
		\end{table}

Footnote	<footnote>*text*</footnote>	\footnote{*text*}
Unformatted text	<code>*text*</code>	\begin{verbatim}*text*\end{verbatim}
Cross references	<label>[*label*] <ref>[*label*]	\label{*label*} \ref{*label*}, \pageref{*label*}
Comments	<comment>*text*</comment>	%*text*⟨end-of-line⟩
User macro	<define><*name*>*definition*⟨end-of-line⟩	\newcommand{*name*}{*definition*}

Table 1: StarTEX command overview

Cross references StarTeX uses more or less the same mechanisms for cross references as LaTeX. Interesting sections, figures and tables are given a label using the <label> command, which may then be referenced using the <ref> command.

The appearance of the reference is defined by the document style, but will normally contain the page number if the reference is a different page; there is thus no need for a \pageref command. (This is similar to the varioref package (Mittelbach, 1995).

Mathematical formulae One of the most important reasons for choosing a typesetting system based on TeX is its ability to typeset mathematical formulae. All the math mode commands available in (LA)TeX are implemented in StarTeX, and most of them use a notation similar to HTML version 3.0. For example, the formula

$$\int_1^\infty \frac{f(x)}{1+x}\partial x$$

is typed as

```
<displaymath>
  <int><sub>1</sub><sup><infinity></sup>
    <frac>f(x)<over>1+x</frac>
    <partial>x
</displaymath>
```

User-defined macros It was decided to allow the users to define their own commands, but with the following restrictions:

- The macros may not have parameters.
- No macros may be redefined.

The StarTeX notation

<define>*<name>definition*⟨end-of-line⟩

was chosen to make error recovery easier. There is now no chance of a runaway definition, like you would get in (LA)TeX if you forgot a final }.

Various other commands In table 3 are shown the few remaining StarTeX commands.

An example In figure 1 is shown an example document using some of the StarTeX commands.

Other design decisions

Error recovery As mentioned previously, StarTeX can employ the <xxx> notation to detect errors and provide some error recovery. For instance, it keeps track of both the current and the outer environments, and which commands should be used to exit those environments. This means that it can detect and remedy the following situations:

Symbol	StarTeX code
$<$	<lt>
$>$	<gt>
–	<-->
—	<--->
⟨a tie⟩	<~>
...	<...>
⟨today's date⟩	<today>
⟨the present time⟩	<now>
TeX	<tex>
LaTeX	<latex>
StarTeX	<startex>

Table 3: The remaining StarTeX commands

- A missing terminator </xxx> will be detected when the outer environment is finished. In this case, both environments will be exited, and you would get an error message like

```
** StarTeX error detected on line 7:
   <i> on line 7 terminated by </b>.
   An extra </i> has been inserted.
```

- A superfluous terminator </xxx> will be recognized as such, and ignored, and the user would be notified with the following error message:

```
** StarTeX error detected on line 15:
   <body> on line 1 terminated by </b>.
   The </b> will be ignored.
```

Paragraph parameters LaTeX is a program for quality typesetting, and this is reflected in the standard parameters for paragraph breaking. Even paragraphs that look quite good to an untrained eye may produce messages about under- or over-full boxes. When LaTeX is unable to find a set of breaks it regards as acceptable, the result may be truly horrible, with words sticking into the margin, or all excess space put into the first line. This occurs quite often in Norwegian which has many long compound words. An experienced LaTeX user will easily detect the problem word and fix that or rephrase the text, but novice users seldom understand these messages and tend to ignore them.

All the messages about over-full and under-full boxes create another problem for the LaTeX novices. Since many of them use tools (like AUC-TeX (Thorup, 1996)) that run LaTeX in non-stop mode, they get pages and pages of serious error messages intertwined with innocuous warnings, so they tend to just ignore all the messages as long as the printed result looks acceptable to them.

StarTeX sets its standard parameters for very loose typesetting with high values for \tolerance and \emergencystretch. The reasons for this are:

```
<body>
<title> <startex><--->A <tex> for beginners </title>
<author> Dag Langmyhr<p> Department of Informatics<p>
  University of Oslo<p> <tt>dag@ifi.uio.no</tt>
</author>
<info> <today> </info>

<abstract> This document describes <startex>, a special <tex>
  format for students writing their first project report.
</abstract>

<h1> The basic philosophy of <Startex> </h1>
<Startex> was designed for novice <tex> users. It employs a
different notation and a different set of commands from <latex>,
and the idea is that this makes it more user-friendly for these
users than plain <tex> or <latex>.

<p>
The notation used in <startex> resembles HTML and some of the
commands are the same, but the philosophy of the two is
different. HTML was designed to display hypertext information
on a computer screen, while <startex> is used to produce a
student report on paper.
</body>
```

Figure 1: An example StarTEX document

- If a good set of paragraph breaks exists, TEX will still choose that.
- Since the users tend to ignore messages about bad breaks, it is better to have a loosely broken paragraph than the very bad result you may get when TEX has to give up.
- The results achieved this way are at least as good as those produced by other typesetting and text-processing software.

This solution does not solve the problem of obtaining good paragraph breaks, but experience so far has shown that it goes a long way.

Concluding remarks

StarTEX has been completed and is being introduced to the students the coming term. It has — in my opinion – achieved most of the specified goals, but not all.

- It is quite small, consisting of fewer than one thousand lines of TEX code plus documentation. Whether the code is easy to understand is for others to judge.
- It is moderately robust. Most simple errors are handled by StarTEX, but grave ones still confuse it.
- It is reasonably fast; the 2 1/2 page example document mentioned at the beginning of this article is processed in 0.9 and 1.6 seconds, respectively.

Even though the users are taught a different format with a different command syntax, I believe StarTEX will serve as a suitable introduction to LATEX and document processing, because it provides training in the *concepts* of LATEX and structural mark-up.

(An analogy from computer science: The programming language C is widely used, and most programmers should know it. It is, however, a language for experts, so a common view is that students should first learn the concepts of programming in a different language before being exposed to C.)

The invention of StarTEX is not intended as any kind of criticism against LATEX, which is still our main tool for larger documents and for the more experienced users. The aim of StarTEX is to help one specific group of users, and provide them with a gentler introduction into the world of (LA)TEX.

On the other hand, StarTEX can be regarded as a tribute to TEX which so easily allows one to produce a different user interface to its powerful mechanisms.

Why not use HTML? Some users have asked why we do not use HTML when the notation is so similar. There are several reasons for that:

- There is no yet final definition of HTML. There are several versions available, in addition to the inventions of various software companies. Nobody knows what HTML will look like a few years from now.
- HTML is growing very complex, with many constructs of little interest to the student writing a report.
- It is difficult to write a robust parser of HTML in TEX.

Availability If anyone is interested in obtaining a copy of StarTEX, they can find it available for anonymous FTP on ftp.ifi.uio.no in the directory pub/tex/startex.

References

Knuth, Donald E. *The TEXbook*. Addison-Wesley, 1991.

Lamport, Leslie. *LATEX user's guide and reference manual*. Addison-Wesley, 1994.

Mittelbach, Frank. "The `varioref` package". Part of the LATEX 2$_\varepsilon$ distribution.

Raggett, Dave. "HyperText markup language specification version 3.0 draft". Available at http://www.w3.org/pub/WWW/MarkUp/html3/.

Spivak, Michael. *The joy of TEX*. American Mathematical Society, 1986. The guide to AMS-TEX.

Taylor, Philip. "TEX: an unsuitable language for document markup?". Talk given at the EuroTEX 1995 conference; does not appear in the proceedings.

Thorup, Kresten Krab. "AUC TEX". An Emacs mode for editing (LA)TEX code; available from http://www.iesd.auc.dk/~amanda/auctex/.