LaTeX

From TeX to LaTeX

Maria Luisa Luvisetto and Enzo Ugolini

1 Introduction

Our Institute is a very old TeX site (since 1982) and users have basic or good knowledge of TeX, but would also like to use LaTeX without the need of reading manuals and documentation. For such users we have prepared a fast reference with guidelines for article setup (title, authors, page numbers, etc.), page setup (section, subsections), font selection, mathematics, tabular information (item, subitem, tables, etc.), index and bibliographic reference, comparing TeX with LaTeX commands. Tools are provided to help in editing the document and inserting complex elements, such as tables.

2 LaTeX Syntax

LaTeX defines the *logical design* of a document and provides *environments* for title, author, abstract, and the like. Environments are structures starting with \begin and ending with the corresponding \end statement. Each document unit is enclosed in a structure, the whole document is delimited by \begin{document} and \end{document}, where begin and end have the same function as {...} in TeX. A LaTeX input looks like the following:

```
\documentstyle[12pt]{article}
%
% preamble section
% add other options in [...]
% add size and paging options here, if any
% add definitions
%
\title{Any Title}
\author{Any N. Author}
%
\begin{document}
% document structure init
\maketitle
% produce title from definition
%
 \begin{abstract}
   ... abstract text ...
 \end{abstract}
%
 \section{First}
  ... section text ...
%
 ...
 ...
%
 \section{Last}
  ... section text ...
%
 \begin{thebibliography}
  \bibitem{bib:one} ... bib text ...
  ...
  \bibitem{bib:end} ... bib text ...
 \end{thebibliography}
%
 \tableofcontents
%
\end{document}
```

We will limit our description to *articles* and to basic typesetting, but LaTeX can refer to any other style your local installation supports. In general, each document is made at least of a style definition command (preamble), the document structure and some text (the document body), an optional abstract, a few sections, reference information and a table of contents, with reference and index at the end of the document. Macro definitions and style changes are declared in the preamble.

LaTeX changes font size for titles and authors, automatically centers both title and authors, adds document date, numbers sections and tabular information, etc. For articles, the sectioning commands are: \section \subsection \subsubsection \appendix

The input format is similar to TeX, with the escape character \ and the same special character set (# $ % & ~ _ ^ \ { }). Basic TeX macros are common to LaTeX. Obviously structures (\begin ... \end) and braces ({...}) must be balanced. Environments can have optional parameters that are enclosed in brackets. Options must be specified immediately after the environment call, multiple options are separated by commas, no blank is allowed inside the brackets.

As in TeX, the document can be split into multiple input files; these are included unconditionally using \input{*file-name*} or conditionally using \include{*file-name*} to include **only** the files named in the preamble through the command:

\includeonly{*file-1,file-2,file-3...*}

If the preamble does not contain an \includeonly command all files are included. If \includeonly has an empty argument list, no file is included.

3 Fonts

Font selection in LaTeX is almost the same as in TeX. The default active font is *roman*, the default inactive font is *italic*. Font size is selected at document level; the default is 10 pt.

Furthermore, LaTeX provides a useful tool to emphasize text elements, the \em environment. The

`\em` command switches the default font from the active one to the inactive; thus if the current font is **roman**, the emphasized text is printed in *italic*, and vice versa. For entire sentences or paragraphs, the emphasized mode is declared as a structure, i.e. `\begin{em} ... \end{em}`. As a consequence of the switching feature, in a long emphasized text typeset in italic, a shorter fragment can be emphasized in roman, as in the following example:

A long emphasized text can include emphasized strings *written in* roman, *inside an italic sentence.*

The above fragment is produced by the following source code:

```
\begin{em} A long emphasized text can
include {\em emphasized strings} written
in {\em roman}, inside an italic
sentence.\end{em}
```

Other predefined fonts are:

bf	bold	**Bold font**
sf	sans serif	Sans Serif font
sl	slanted	*Slanted font*
sc	small caps	SMALL CAPS FONT
tt	type writer	`Typewriter font`

Fonts are declared as in TEX (i.e. `{\bf` *text*`}`). Accents and symbols are typeset as in TEX. Other fonts are defined for mathematical use, like greek letters (limited set), calligraphic ones (only uppercase), plus the mathematical italic (`\mit`) that is the default type for maths and mathematical bold (`\boldmath`) fonts.

In technical manuals, especially when reporting computer programs, it is required not only to use a non-proportional font as `\tt`, but also to reproduce the text as it stands, including producing a mark such as ␣ for *required* spaces. For this purpose LATEX has four commands:

```
\begin{verbatim} ... \end{verbatim},
\begin{verbatim*} ... \end{verbatim*},
\verb, \verb*
```

The `*` commands typeset ␣ for blanks. The `verbatim` environment typesets the text on a new line; thus it is used for long insertions. The `\verb` command is used for short strings inside the current line. The text is delimited by any pair of identical characters such as `!` as in the following example:

To display blanks in computer programs type `\verb*!int 1,k;!`, *the typeset result will be* `int␣1,k;`

Font size can be changed with the commands `\tiny` or `\small` for smaller fonts, or `\Large` or`\Huge` for bigger fonts, followed by the font specification if different from roman, so we have `\large\bf` for large bold letters. Examples of the `\tiny`, `\Huge` and `\large\bf` follow:

Font Font **Font**

Users can define other fonts not provided in the default set, as in TEX, with the command:

```
\newfont{\symbf}{cmsy10 scaled\magstep1}
```

where `\symbf` is name of the new font that is available in the `cmsy10` font description file in an enlarged size. To typeset some text in the new font, just call it as any other font: `{\symbf \symbol{26}}` to write the symbol with character code 26 (i.e. $\subset$).

4 Notes

LATEX provides two types of notes: *footnotes* and *marginal notes.* The syntax for footnotes is similar but not identical to TEX. In TEX footnote numbering is required (number and text enclosed in braces) and number generation is not handled, thus the user must take care of footnote numbering.

In LATEX the number is a positive integer automatically stepped for the next footnote command. The numbering can be changed at user's will as an option. The syntax is: `\footnote [`*num*`]{`*text*`}` where *num* is the optional number for the following footnote text.

Marginal notes are not numbered and are placed in the free paper margin with the first line even with the line of text in which the note is inserted, as happens here. The note is placed according to the style in use: it is placed on the right for one-sided documents, on the outside margin for two-sided printing, in the nearest margin for multicolumn style. *note*

The marginal note is produced by `\marginpar` and different text can be produced for left and right margin as an option. The syntax is:
`\marginpar[`*left_text*`]{`*right_text*`}`

5 Item Lists

LATEX has extensive capabilities to handle tabular information, both in the form of tables (see Section 6) and in the form of aligned text.

LATEX defines a set of structures to format lists of items or quotations. The structures are: `quote`, `quotation`, `itemize`, `enumerate`, `description`. Inside the structure, each entry is declared through the `\item` command.

The `quote` environment is used for short quotations, the `quotation` environment for longer ones. The “quoted” text is indented, as shown here.

> LATEX is able to handle quotations.
> Each quotation is indented.

The above example is produced by:

```
\begin{quote}
\LaTeX{} is able to handle quotations.

Each quotation is indented.
\end{quote}
```

More useful in technical documents are the `itemize` and `enumerate` environments described in the following example.

- Each item in a list is marked by a *bullet.*
- Item lists can be nested.
    1. Items in enumerated lists are labelled by numerals.
    2. Lists can contain two or more items.
    3. If there is only one item, there is logically no list.
- Blank lines are ignored.
- In the input file, indent lines to show the item list structure.

Item lists can be nested in complex manners, as shown by the above example produced by the following code:

```
\begin{itemize}
 \item Each item in a ...
 \item Item lists can be ...
  \begin{enumerate}
   \item Items in ...
   \item Lists can ...
   \item If there is ...
  \end{enumerate}
 \item Blank lines are ...
 \item In the input file, ...
\end{itemize}
```

Another very useful feature to format item lists is the `declaration` environment. In this environment an item has a name, which is typeset in boldface, and is followed by its description, which is typeset as itemized text:

X nX delete one or 'n' characters starting at cursor position.

dnG delete all lines starting with the current line up to line 'n'.

The commands for the above example are:

```
\begin{description}
 \item[X nX] delete one or 'n' char...
 \item[dnG]  delete all lines starting ...
\end{description}
```

6 Tables

In TEX tables are handled by the "settabs" commands, that can either create fixed width columns or use a template to describe the table fields. LATEX has two environments for tables: the `\tabbing` and the `tabular` environments. The first one emulates TEX commands with template description, while the second one enables the user to create very complex tables in an easy way. Furthermore LATEX code for tables is much more readable than TEX one.

The `tabbing` environment handles tables of any length that span across pages. The tab stops can be set either in a prototype or as columns are typed. The tabbing information is a structure started by `\begin{tabbing}` and ended by `\end{tabbing}`. The command `\=` sets the tab stop, `\>` moves to the next stop. The element of the first column has no tab information, and the line is ended by `\\`. If the line represents a prototype, it ends with `\kill`.

As a tabbing example consider the following table describing how to type some LATEX special characters in normal text:

`\{`	{	open brace
`\}`	}	close brace
`\$`	$	dollar sign
`\_>`	_	underscore
`\%`	%	percent

This table makes use of a prototype line and is generated by the following code:

```
\begin{center}
 \begin{minipage}{\hsize}
  \begin{tabbing}
  xxxx \= xxxxxxxx \= \kill
  \verb!\{!  \> \{  \> open brace  \\
  \verb!\}!  \> \}  \> close brace \\
  \verb!\$!  \> \$  \> dollar sign \\
  \verb!\_>! \> \_  \> underscore \\
  \verb!\%!  \> \%  \> percent
  \end{tabbing}
 \end{minipage}
\end{center}
```

The `tabular` environment creates tables that are essentially boxes, that behave like figures and can float around the page but cannot span pages. Frequently these tables are enclosed in drawn rectangles containing vertical and horizontal lines to separate the columns. The tab stops are handled automatically and specified by `&`, the position of the items in the column is defined within the tabular environment by one of the characters `l r c` to respectively align on the left, on the right or center the item, and the line is ended by `\\`.

A vertical line is drawn with `|` declared in the tabular specification; the command `\hline` after `\\` draws a horizontal line across the full width of the table. The command `\cline{`$i-j$`}` draws a horizontal line across columns i through j, inclusive.

When an argument spans multiple columns, it is produced by the `\multicolumn` command with the following syntax:

`\multicolumn{`*n*`}{`*pos*`}{`*item*`}`

where n is the number of columns to be spanned, *pos* defines the position: `l` (for left), `r` (for right), `c` (for centre), and *item* is the text to be typeset.

As an example consider the following table:

Cray Total Gain (millisec)			
Level	Time	Gain	CDC/Cray
0	33.37	–	1.36
1	29.86	10.5	1.52
2	24.19	19.0	1.87
3	21.92	9.4	2.07

that was produced by the following code:

```
\begin{center}
\begin{tabular}{|c|c|c|c|c|} \hline
\multicolumn{4}{|c|}{Cray ...}\\ \hline
Level & Time  & Gain & CDC/Cray \\ \hline
0     & 33.37 &  --  & 1.36 \\ \hline
1     & 29.86 & 10.5 & 1.52 \\ \hline
2     & 24.19 & 19.0 & 1.87 \\ \hline
3     & 21.92 &  9.4 & 2.07 \\ \hline
\end{tabular}
\end{center}
```

To get an idea of the easy tabular environment provided by LaTeX, note that the TeX code used to produce the same table is made of 22 lines, each longer and more complex.

7 Mathematics

Mathematical formulas can appear as *in-text* elements (`math` environment) or as displayed formulas (`displaymath` environment). Numbered displayed formulas are produced in the `equation` environment. The commands to select the environments are:

in-text maths: `$...$` or `\(...\)` or
`\begin{math} ... \end{math}`

displayed maths: `\[...\]` or
`\begin{displaymath} ... \end{displaymath}`

equation: `\begin{equation} ... \end{equation}`

Most math elements and symbols are made as in TeX. The unchanged items are: subscripts and superscripts, greek and calligraphic letters, math spacing, symbols, ellipsis, etc. Most math commands are identical to TeX, such as `\overline` and `\underline`, `\vec`, etc. The same applies to font selection for math, text and scripts.

Fractions are handled in an easy way by the `\frac` command that has two arguments: numerator and denominator.

$$x = \frac{y+z}{y^{2}+z^{2}}$$

```
\[x = \frac{y+z}{y^{2}+z^{2}} \]
```

$$y = \frac{a+b}{1+\frac{a}{a^2+b^2}}$$

```
\[\frac{a+b}{1+\frac{a}{a^2+b^2}}\]
```

Arrays are produced with the `array` environment, which is similar to the tabular one. The declaration specifies the size (number of columns) and the alignment of items: `l r c` for left, right or center. New items are begun with `&`, rows are ended with `\\`

$$A = \left(\begin{array}{ccc} x-1 & 1 & 0 \\ 0 & x-1 & 1 \\ 0 & 0 & x-1 \end{array} \right)$$

The above array is typeset with the following commands. Note that array syntax in LaTeX is very similar to TeX.

```
\[ A = \left( \begin{array}{ccc}
x-1 &1    &0    \\
0   &x-1  &1    \\
0   &0    &x-1
\end{array} \right) \]
```

The delimiters for arrays are typeset in the same way as TeX: the commands `\left` or `\right` to specify the left or right delimiter followed by the delimiter itself `() [] | {}`. The `\left` and `\right` commands must come in matching pairs, but the delimiters do not need to match in any form.

Long or multiple formulas are displayed with the `eqnarray` environment, that enables line numbering and equation splitting. Rows are separated by `\\`, items by `&`; numbering is disabled by `\nonumber`. The following example shows the use of `eqnarray`.

$$\begin{aligned} x &= a+y-c^2 && (1) \\ y &= a^2+b^2-x- \\ & \quad xy+c^3-p && (2) \end{aligned}$$

```
\begin{eqnarray}
x & = & a + y - c^2 \\
y & = & a^2 + b^2 - x -\nonumber \\
  &   & xy + c^3 -p
\end{eqnarray}
```

When symbols must be typeset one above another, use the command `\stackrel`:

$$A \stackrel{a'}{\rightarrow} B$$

```
\[ A \stackrel{a'} {\rightarrow} B \]
```

Theorems can receive a name, a label and a number when defined by the \newtheorem command that takes two arguments: the name and the label. The theorem text is emphasized. The numbering is automatic and can be computed within the specified sectional unit using the optional argument. The sectional unit can be one predefined by LaTeX such as `chapter`, `section`, etc., or the name of a user defined theorem, so that all theorems of the same type are numbered in the same sequence.

```
\newtheorem{guess}{Conjecture}[section]
% define conjecture in section
 ....
\begin{guess} This is a guess. \end{guess}
```

Conjecture 7.1 *This is a guess.*

8 Definitions

LaTeX provides tools to define new commands (TeX macros). The new commands are defined by \newcommand, followed by the name, the *optional* arguments and finally the definition. The name **must** be prefixed by \; when used, arguments must be enclosed in braces. In the following example is shown LaTeX syntax to define and call the macro \abx and the typeset formula thus generated.

```
\newcommand{\abx}[2]{$#1x+#2$} \abx{5a}{b}
```

$5ax + b$

In a similar way, to change style, fonts, emphasis, etc., the user can define a new environment with \newenvironment.

Commands can be defined anywhere, but the definitions must appear before their use.

9 Graphics – Floating Objects

LaTeX has a limited capacity for creating graphic objects and is able to move figures around in a *floating* way to avoid splitting between pages. Figures can receive a caption; in this case a caption number is produced (for an example see Figure 1). Suppose that you must insert a figure 5cm tall in your text; the LaTeX code is:

```
\begin{figure}
  \vspace{5cm}  % leave space for figure
  \caption{Fractal image.}
\end{figure}
```

In the above example we leave blank space to insert the picture at a later time with cut and paste methods. For simple graphics LaTeX is able to draw axes, lines, circles. The coordinate system is expressed in \unitlength with default value 1 point (nearly 1/72 inch).

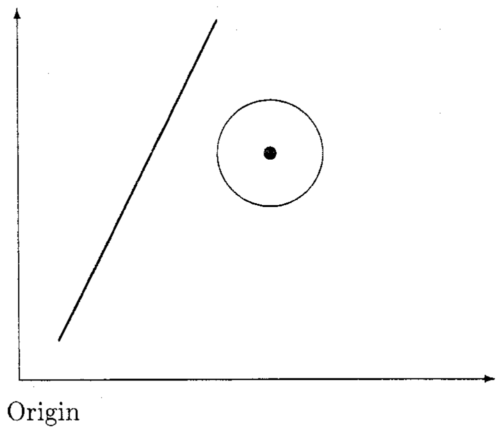

Figure 1: Graphic example

A picture is created with the `picture` environment by specifying the picture's $x - y$ dimensions and, optionally, origin, enclosed in parentheses:

```
\begin{picture}(200,150)(20,10)
```

Any graphic and/or text information is positioned in the picture with the \put command, which is followed by the coordinates in parentheses and by the object to put in braces: `\put(0,-10){Origin}`. Objects are ordinary text, straight lines, arrows, circles, ovals.

The line and arrow syntax is the same; the commands are \line and \vector. The command arguments are slope expressed as $(\Delta x, \Delta y)$ and length. For \circle the argument is the diameter. For \oval the arguments are width and height plus an optional argument to draw only half or a quarter of the complete oval. Lines can have two standard thicknesses: \thinlines (default) and \thicklines. Both declarations are used in the example.

The source code example for a simple drawing follows. LaTeX output is shown in Figure 1. Note that LaTeX requires the use of parentheses () when describing graphic objects.

```
\begin{figure}
 \begin{picture}(200,150)
  \put(0,-10){Origin}
  \put(5,5){\vector(1,0){180}} % x-axis
  \put(5,5){\vector(0,1){140}} % y-axis
  \thicklines
  \put(20,20){\line(1,2){60}}  %draw line
  \thinlines
  \put(100,90){\circle{40}}   %draw circle
  \put(100,90){\circle*{5}}   %fill center
 \end{picture}
 \caption{Graphic example}\label{fig:ex}
\end{figure}
```

Using the same basic criteria, virtual boxes can be created to split the physical page into sub-areas called *boxes*. Text can be placed inside the boxes at center (default), left (`l`) or right (`r`). There are two commands to handle such boxes: `\makebox` and `\framebox`; the second one draws a frame around the box. Both commands have the same syntax: two optional arguments for width and position, and the text to be framed: `\framebox[`*width*`][`*pos*`]{`*text*`}`. Both commands can define box size and text position as optional arguments. The text to be typeset is enclosed in braces. The syntax and typeset results are shown in the following example:

$\boxed{\text{ framebox }}$	`\framebox[2cm]{framebox}`
$\boxed{\text{framebox }\quad}$	`\framebox[2cm][l]{framebox}`

Besides the above commands, the `minipage` environment enables the user to split the typeset information into variable size paragraphs of specified width and position typed as multicolumns side by side inside the current environment. The `minipage` environment is used in the above example with the following commands, in which boxes are created in a nested way. The first `minipage` is 2.5cm wide and the second one is wider (4.0cm). Both minipages are typeset with the top line at the current text position (`[t]`). Note that the two `minipage` environments are typeset side by side as normal text with a `\quad` horizontal space.

```
\noindent
\begin{minipage}[t]{2.5cm}
 \framebox[2cm]{framebox}
 \framebox[2cm][l]{framebox}
\end{minipage}
\quad   % align second minipage
\begin{minipage}[t]{4.0cm}
\verb!\framebox[2cm]{framebox}!
\verb!\framebox[2cm][l]{framebox}!
\end{minipage}
```

Text can be positioned in the center, left or right of the page using the `center`, `flushleft` or `flushright` environments. To start new lines in such environments use `\\` as in tabular and array structures.

10 Reference – Index – Bibliography

LaTeX provides an easy way to create cross-references linking the various elements of the document, such as figures, equations, sections. Each element can receive a name through the `\label` command. Any string can be assigned to the name, suggested naming conventions are `eq:euler`, `sect:syntax` and the like to create mnemonic names related to structures and thus more easily identifiable.

Once an element is named, it is referenced with the `\ref` command. The name can be defined in any place in the source code (before or after being referenced), but it should be typed immediately after the referenced item, i.e. if the user wants to label a caption to refer it by figure number, the `label` statement must be typed after the caption title: `\caption{Graphic example}\label{fig:ex}`.

LaTeX writes temporary files to handle references that are resolved on the next run, thus it **must** be run twice to typeset the updated reference information. If the temporary files are missing or possibly not up to date, a warning message is written. As an example consider the following fragment:

Equation 3 is very famous.

...

Energy equation is

$$E = mc^2 \tag{3}$$

produced by:

```
Equation \ref{eq:ck} is very famous.
Energy equation is \begin{equation}
E = mc^2  \label{eq:ck} \end{equation}
```

References can be set also on any page using `\label` to name the text and the command `\pageref` to get the page number of the named text, the source code looks like the following:

```
see page~\pageref{fonts} for more details.
 ...
Predefined \label{fonts} fonts are:
```

Keep label definitions to a reasonably short size to avoid LaTeX problems with internal space. Avoid defining labels that are never used or used too seldom. Keep a list of used labels and their meanings to produce a readable and mantainable input file. More than forty labels can cause problems.

Finally, LaTeX can produce a table of contents and bibliographic reference with auto-labels. The style of this information is, as always, related to the document style. Its position inside the document is determined by the place in the input file: at the beginning if the command is typed before `\maketitle`, at the end if it is typed before `\end{document}`.

The table of contents is produced by the command `\tableofcontents`; other index information can be produced for tables and pictures using `\listoffigures` and `\listoftables`.

To produce a bibliography, the user defines entries in the `thebibiography` structure, which can have as optional argument the definition of the widest label in the item list. Each item is inserted with `\bibitem`, which has the following arguments:

an optional label that overides the default numbering scheme, the key-name for citations and the entry text. The items are referenced with the \cite command.

An example of bibliographic data is given by the following environment definition:

```
\begin{thebibliography}
\bibitem{bib:la} L. Lamport. \LaTeX:
  {\em User's Guide and Reference ... }
 ....
\bibitem{bib-my}M. L. Luvisetto, ...
{\em Introduzione ... }
\end{thebibliography}
```

and are called in any place as shown:

```
 ... for more information see
\cite{bib:la,bib:le} and ...
\cite{bib-my}.
```

11 Useful Tools

At our site, many researchers have a workstation, but a language sensitive editor is provided only on some machines. To help our users in typing documents we have created a set of files containing a template of the most common environments. The files are defined at system level: as logicals under VMS, as symbolic links under Unix, so they can be inserted into a document during the editing session.

As most users are not computer professionals, their knowledge of the editors is a basic one. Therefore we have developed the utility program PreLaTeX that, when run on a new file, interactively asks for title, authors, etc., and produces a template file similar to the one listed at the beginning of this article. The program asks also for section names and bibliography; thus the user can create a skeleton of the document at his first session.

During the editing session, LaTeX environments such as:

```
quotation itemize enumerate
description tabbing tabular
array minipage figure
```

can be inserted in template form by inserting one of the template files.

The file contains the name of the environment, the optional arguments, a title to declare its usage and empty lines to produce the wanted information. By following the template it is easy to create tables, arrays, item lists, etc., even for newcomers. In a few hours, inexperienced users are able to produce simple documents. The file names and usage are described in help files. For people using workstations, it is simple to keep this information in a separate window ready for use.

An example of a template file is given below for the tabbing environment.

```
\begin{tabbing}
% create table, set tab with \=
% recall tab with \> end with \\
% set template fields (f)
% insert fields between tabs
%f1 \= f2 \= f3 \= f4 \= \kill
    \>    \>    \>    \> \\
    \>    \>    \>    \> \\
 ......
\end{tabbing}
```

Thus the user needs only to fill in the columns with his own values.

PreLaTeX can be run on an existing file. In this case, it produces a list of all sections together with their headings, a list of all ref, pageref and label commands and, at the user's request, a list of all figures, captions, equations and the like. If the bibliography data are present they are listed at the end, together with cite commands. This feature is found very helpful in checking cross-references, especially in the final steps of the document preparation.

PreLaTeX is not static. Planned extensions include a reformatting option that would change the columnar alignment in the source of tables, arrays, etc., to more closely resemble the output. A source in this form should be easier to read and maintain.

The above software is free and available from the authors, who can be contacted at the cited e-mail address.

◇ Maria Luisa Luvisetto and Enzo Ugolini
Istituto Nazionale di Fisica Nucleare
Viale Ercolani 8
40138, Bologna, Italy
Internet: Luvisetto@CNAF.INFN.IT
Internet: Ugolini@CNAF.INFN.IT