



# L<sup>A</sup>T<sub>E</sub>X Tutorials

A PRIMER

Indian T<sub>E</sub>X Users Group

Trivandrum, India

2003 September

L<sup>A</sup>T<sub>E</sub>X TUTORIALS — A PRIMER  
Indian T<sub>E</sub>X Users Group

EDITOR: E. Krishnan  
COVER: G. S. Krishna

Copyright ©2002, 2003 Indian T<sub>E</sub>X Users Group  
Floor III, SJP Buildings, Cotton Hills  
Trivandrum 695014, India  
<http://www.tug.org.in>

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, version 1.2, with no invariant sections, no front-cover texts, and no back-cover texts. A copy of the license is included in the end.

This document is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

Published by the Indian T<sub>E</sub>X Users Group

Online versions of this tutorials are available at:  
<http://www.tug.org.in/tutorials.html>

## PREFACE

*The ideal situation occurs when the things that we regard as beautiful are also regarded by other people as useful.*

— Donald Knuth

For us who wrote the following pages, T<sub>E</sub>X is something beautiful and also useful. We enjoy T<sub>E</sub>X, sharing the delights of newly discovered secrets amongst ourselves and wondering ever anew at the infinite variety of the program and the ingenuity of its creator. We also lend a helping hand to the new initiates to this art. Then we thought of extending this help to a wider group and The Net being the new medium, we started an online tutorial. This was well received and now the Free Software Foundation has decided to publish these lessons as a book. It is a fitting gesture that the organization which upholds the rights of the user to study and modify a software publish a book on one of the earliest programs which allows this right.

Dear reader, read the book, enjoy it and if possible, try to add to it.

The TUG<sub>India</sub> Tutorial Team



# CONTENTS

<b>I. The Basics</b> . . . . .	<b>7</b>
I.1 What is L <sup>A</sup> T <sub>E</sub> X? – 7 • I.2 Simple typesetting – 8 • I.3 Fonts – 13 • I.4 Type size – 15	
<b>II. The Document</b> . . . . .	<b>17</b>
II.1 Document class – 17 • II.2 Page style – 18 • II.3 Page numbering – 19 • II.4 Formatting lengths – 20 • II.5 Parts of a document – 20 • II.6 Dividing the document – 21 • II.7 What next? – 23	
<b>III. Bibliography</b> . . . . .	<b>27</b>
III.1 Introduction – 27 • III.2 natbib – 28	
<b>IV. Bibliographic Databases</b> . . . . .	<b>33</b>
IV.1 The Bib <sub>T</sub> E <sub>X</sub> program – 33 • IV.2 Bib <sub>T</sub> E <sub>X</sub> style files – 33 • IV.3 Creating a bibliographic database – 34	
<b>V. Table of contents, Index and Glossary</b> . . . . .	<b>39</b>
V.1 Table of contents – 39 • V.2 Index – 41 • V.3 Glossary – 44	
<b>VI. Displayed Text</b> . . . . .	<b>47</b>
VI.1 Borrowed words – 47 • VI.2 Poetry in typesetting – 48 • VI.3 Making lists – 48 • VI.4 When order matters – 51 • VI.5 Descriptions and definitions – 54	
<b>VII. Rows and Columns</b> . . . . .	<b>57</b>
VII.1 Keeping tabs – 57 • VII.2 Tables – 62	
<b>VIII. Typesetting Mathematics</b> . . . . .	<b>77</b>
VIII.1 The basics – 77 • VIII.2 Custom commands – 81 • VIII.3 More on mathematics – 82 • VIII.4 Mathematics miscellany – 89 • VIII.5 New operators – 101 • VIII.6 The many faces of mathematics – 102 • VIII.7 And that is not all! – 103 • VIII.8 Symbols – 103	
<b>IX. Typesetting Theorems</b> . . . . .	<b>109</b>
IX.1 Theorems in L <sup>A</sup> T <sub>E</sub> X – 109 • IX.2 Designer theorems—The amsthm package – 111 • IX.3 Housekeeping – 118	
<b>X. Several Kinds of Boxes</b> . . . . .	<b>119</b>
X.1 LR boxes – 119 • X.2 Paragraph boxes – 121 • X.3 Paragraph boxes with specific height – 122 • X.4 Nested boxes – 123 • X.5 Rule boxes – 123	
<b>XI. Floats</b> . . . . .	<b>125</b>
XI.1 The figure environment – 125 • XI.2 The table environment – 130	

<b>XII. Cross References in <math>\LaTeX</math></b> . . . . .	<b>135</b>
XII.1 Why cross references? – 135 • XII.2 Let $\LaTeX$ do it – 135 • XII.3 Pointing to a page—the package <code>varioref</code> – 138 • XII.4 Pointing outside—the package <code>xr</code> – 140 • XII.5 Lost the keys? Use <code>tblst.tex</code> – 140	
<b>XIII. Footnotes, Marginpars, and Endnotes</b> . . . . .	<b>143</b>
XIII.1 Footnotes – 143 • XIII.2 Marginal notes – 147 • XIII.3 Endnotes – 148	

# TUTORIAL I

---

## THE BASICS

### 1.1. WHAT IS L<sup>A</sup>T<sub>E</sub>X?

The short and simple answer is that L<sup>A</sup>T<sub>E</sub>X is a typesetting program and is an extension of the original program T<sub>E</sub>X written by Donald Knuth. But then what is a *typesetting program*?

To answer this, let us look at the various stages in the preparation of a document using computers.

1. The text is *entered* into the computer.
2. The input text is *formatted* into lines, paragraphs and pages.
3. The output text is *displayed* on the computer screen.
4. The final output is *printed*.

In most *word processors* all these operations are integrated into a single application package. But a typesetting program like T<sub>E</sub>X is concerned only with the second stage above. So to typeset a document using T<sub>E</sub>X, we type the text of the document and the necessary formatting commands in a *text editor* (such as Emacs in GNU/Linux) and then compile it. After that the document can be viewed using a *previewer* or printed using a *printer driver*.

T<sub>E</sub>X is also a *programming language*, so that by learning this language, people can write code for additional features. In fact L<sup>A</sup>T<sub>E</sub>X itself is such a (large) collection of extra features. And the collective effort is continuing, with more and more people writing extra *packages*.

#### 1.1.1. A small example

Let us see L<sup>A</sup>T<sub>E</sub>X in action by typesetting a short (really short) document. Start your favorite text editor and type in the lines below *exactly* as shown

```
\documentclass{article}
\begin{document}
This is my \emph{first} document prepared in \LaTeX.
\end{document}
```

Be especially careful with the `\` character (called the *backslash*) and note that this is different from the more familiar `/` (the *slash*) in `and/or` and save the file onto the hard disk as `myfile.tex`. (Instead of `myfile` you can use any name you wish, but be sure to have `.tex` at the end as the *extension*.) The process of compiling this and viewing the output depends on your operating system. We describe below the process of doing this in GNU/Linux.

At the shell prompt type

```
latex myfile
```

You will see a number of lines of text scroll by in the screen and then you get the prompt back. To view the output in screen, you must have the X Window running. So, start X if you have not done so, and in a terminal window, type

```
xdvi myfile
```

A window comes up showing the output below

This is my *first* document prepared in L<sup>A</sup>T<sub>E</sub>X.

Now let us take a closer look at the *source file* (that is, the file you have typed). The first line `\documentclass{article}` tells L<sup>A</sup>T<sub>E</sub>X that what we want to produce is an article. If you want to write a book, this must be changed to `\documentclass{book}`. The whole document we want to typeset should be included between `\begin{document}` and `\end{document}`. In our example, this is just one line. Now compare this line in the source and the output. The first three words are produced as typed. Then `\emph{first}`, becomes *first* in the output (as you have probably noticed, it is a common practice to *emphasize* words in print using italic letters). Thus `\emph` is a *command* to L<sup>A</sup>T<sub>E</sub>X to typeset the text within the braces in *italic*<sup>1</sup>. Again, the next three words come out without any change in the output. Finally, the input `\LaTeX` comes out in the output as L<sup>A</sup>T<sub>E</sub>X.

Thus our source is a mixture of text to be typeset and a couple of L<sup>A</sup>T<sub>E</sub>X *commands* `\emph` and `\LaTeX`. The first command changes the input text in a certain way and the second one generates new text. Now call up the file again and add one more sentence given below.

```
This is my \emph{first} document prepared in \LaTeX. I typed it
on \today.
```

What do you get in the output? What new text does the command `\today` generate?

### 1.1.2. Why L<sup>A</sup>T<sub>E</sub>X?

So, why all this trouble? Why not simply use a word processor? The answer lies in the motivation behind T<sub>E</sub>X. Donald Knuth says that his aim in creating T<sub>E</sub>X is to *beautifully* typeset *technical documents* especially those containing *a lot of Mathematics*. It is very difficult (sometimes even impossible) to produce complex mathematical formulas using a word processor. Again, even for ordinary text, if you want your document to look *really beautiful* then L<sup>A</sup>T<sub>E</sub>X is the natural choice.

## 1.2. SIMPLE TYPESETTING

We have seen that to typeset something in L<sup>A</sup>T<sub>E</sub>X, we type in the text to be typeset together with some L<sup>A</sup>T<sub>E</sub>X commands. Words must be separated by spaces (does not matter how many) and lines maybe broken arbitrarily.

The end of a paragraph is specified by a *blank line* in the input. In other words, whenever you want to start a new paragraph, just leave a blank line and proceed. For example, the first two paragraphs above were produced by the input

<sup>1</sup>This is not really true. For the real story of the `\emph` command, see the section on fonts.



We have seen that to typeset something in `\LaTeX`, we type in the text to be typeset together with some `\LaTeX` commands. Words must be separated by spaces (does not matter how many) and lines maybe broken arbitrarily.

The end of a paragraph is specified by a `\emph{blank line}` in the input. In other words, whenever you want to start a new paragraph, just leave a blank line and proceed.

Note that the first line of each paragraph starts with an *indentation* from the left margin of the text. If you do not want this indentation, just type `\noindent` at the start of each paragraph for example, in the above input, `\noindent We have seen ...` and `\noindent The end of ...` (come on, try it!) There is an easier way to suppress paragraph indentation for *all* paragraphs of the document in one go, but such tricks can wait.

### 1.2.1. Spaces

You might have noticed that even though the length of the lines of text we type in a paragraph are different, in the output, all lines are of equal length, aligned perfectly on the right and left. `TEX` does this by adjusting the space between the words.

In traditional typesetting, a little extra space is added to periods which end sentences and `TEX` also follows this custom. But how does `TEX` know whether a period ends a sentence or not? It assumes that every period *not following an upper case letter* ends a sentence. But this does not always work, for there are instances where a sentence does end in an upper case letter. For example, consider the following

Carrots are good for your eyes, since they contain Vitamin A. Have you ever seen a rabbit wearing glasses?

The right input to produce this is

```
Carrots are good for your eyes, since they contain Vitamin A\@. Have
you ever seen a rabbit wearing glasses?
```

Note the use of the command `\@` *before* the period to produce the extra space after the period. (Remove this from the input and see the difference in the output.)

On the other hand, there are instances where a period following a lowercase letter does not end a sentence. For example

The numbers 1, 2, 3, etc. are called natural numbers. According to Kronecker, they were made by God; all else being the work of Man.

To produce this (without extra space after etc.) the input should be

```
The numbers 1, 2, 3, etc.\ are called natural numbers. According to
Kronecker, they were made by God;all else being the works of Man.
```

Here, we use the command `\` (that is, a backslash and a *space*—here and elsewhere, we sometimes use `\`  to denote a space in the input, especially when we draw attention to the space).

There are other situations where the command `\`  (which always produce a space in the output) is useful. For example, type the following line and compile it.

```
I think \LaTeX is fun.
```

You get

```
I think \LaTeX is fun.
```

What happened to the *space* you typed between `\LaTeX` and `is`? You see, `\TeX` gobbles up all spaces after a command. To get the required sequence in the output, change the input as

```
I think \LaTeX\ is fun.
```

Again, the command `\` comes to the rescue.

### 1.2.2. Quotes

Have you noticed that in typesetting, opening quotes are different from closing quotes? Look at the `\TeX` output below

```
Note the difference in right and left quotes in 'single quotes' and "double quotes".
```

This is produced by the input

```
Note the difference in right and left quotes in 'single quotes'
and 'double quotes'.
```

Modern computer keyboards have a key to type the symbol ``` which produces a left quote in `\TeX`. (In our simulated inputs, we show this symbol as `'`.) Also, the key `'` (the usual ‘typewriter’ quote key, which also doubles as the apostrophe key) produces a left quote in `\TeX`. Double quotes are produced by typing the corresponding single quote twice. The ‘usual’ double quote key `"` can also be used to produce a *closing* double quote in `\TeX`.

If your keyboard does not have a left quote key, you can use `\lq` command to produce it. The corresponding command `\rq` produces a right quote. Thus the output above can also be produced by

```
Note the difference in right and left quotes in \lq single
quotes\rq\ and \lq\lq double quotes\rq\rq.
```

(Why the command `\` after the first `\rq`?)

### 1.2.3. Dashes

In text, dashes are used for various purposes and they are distinguished in typesetting by their lengths; thus short dashes are used for hyphens, slightly longer dashes are used to indicate number ranges and still longer dashes used for parenthetical comments. Look at the following `\TeX` output

```
X-rays are discussed in pages 221–225 of Volume 3—the volume on electromagnetic waves.
```

This is produced from the input

```
X-rays are discussed in pages 221--225 of Volume 3---the volume on
electromagnetic waves.
```

Note that a single dash character in the input `-` produces a hyphen in the output, two dashes `--` produces a longer dash (`-`) in the output and three dashes `---` produce the longest dash (`—`) in the output.

### 1.2.4. Accents

Sometimes, especially when typing foreign words in English, we need to put different types of accents over the letters. The table below shows the accents available in  $\LaTeX$ . Each column shows some of the accents and the inputs to generate them.

ò	\`o	ó	\'o	ô	\^o	õ	\~o
ō	\=o	ô	\.o	ö	\"o	ç	\c c
ö	\u o	ř	\v o	ő	\H o	ø	\d o
ø	\b o	ôo	\t oo				

The letters *i* and *j* need special treatment with regard to accents, since they should not have their customary dots when accented. The commands `\i` and `\j` produce dot-less *i* and *j* as *ı* and *ȷ*. Thus to get

Él está aquí

you must type

```
\{E}l est\'{a} aqu\i
```

Some symbols from non-English languages are also available in  $\LaTeX$ , as shown in the table below:

œ	\oe	Œ	\OE	æ	\ae	Æ	\AE
	\aa		\AA				
ø	\o	Ø	\O	ı	\i	ȷ	\j
ß	\ss						
ı	\i	ȷ	\j				

### 1.2.5. Special symbols

We have seen that the input `\LaTeX` produces  $\LaTeX$  in the output and `\` produces a space. Thus  $\TeX$  uses the symbol `\` for a special purpose—to indicate the program that what follows is not text to be typeset but an instruction to be carried out. So what if you want to get `\` in your output (improbable as it may be)? The command `\textbackslash` produces `\` in the output.

Thus `\` is a symbol which has a special meaning for  $\TeX$  and cannot be produced by direct input. As another example of such a special symbol, see what is obtained from the input below

```
Maybe I have now learnt about 1% of \LaTeX.
```

You only get

Maybe I have now learnt about ı

What happened to the rest of the line? You see,  $\TeX$  uses the per cent symbol `%` as the *comment* character; that is a symbol which tells  $\TeX$  to consider the text following as ‘comments’ and not as text to be typeset. This is especially useful for a  $\TeX$  programmer to explain a particularly sticky bit of code to others (and perhaps to himself). Even for ordinary users, this comes in handy, to keep a ‘to do’ list within the document itself for example.

But then, how do you get a percent sign in the output? Just type `\%` as in

Maybe I have now learnt about 1% of \LaTeX.

The symbols `\` and `%` are just two of the ten characters T<sub>E</sub>X reserves for its internal use. The complete list is

```
~ # $ % ^ & _ \ { }
```

We have seen how T<sub>E</sub>X uses two of these symbols (or is it four? Did not we use `{ }` in one of our examples?) The use of others we will see as we proceed.

Also, we have noted that `\` is produced in the output by the command `\textbackslash` and `%` is produced by `\%`. What about the other symbols? The table below gives the inputs to produce these symbols.

<code>~</code>	<code>\textasciitilde</code>	<code>&amp;</code>	<code>\&amp;</code>
<code>#</code>	<code>\#</code>	<code>-</code>	<code>\_</code>
<code>\$</code>	<code>\\$</code>	<code>\</code>	<code>\textbackslash</code>
<code>%</code>	<code>\%</code>	<code>{</code>	<code>\{</code>
<code>^</code>	<code>\textasciicircum</code>	<code>}</code>	<code>\}</code>

You can see that except for three, all special symbols are produced by preceding them with a `\`. Of the exceptional three, we have seen that `\~` and `\^` are used for producing accents. So what does `\\` do? It is used to break lines. For example,

```
This is the first line.\\ This is the second line
```

produces

```
This is the first line.
This is the second line
```

We can also give an *optional* argument to `\\` to increase the vertical distance between the lines. For example,

```
This is the first line.\\[10pt]
This is the second line
```

gives

```
This is the first line.

This is the second line
```

Now there is an extra 10 points of space between the lines (1 point is about 1/72<sup>nd</sup> of an inch).

### 1.2.6. Text positioning

We have seen that T<sub>E</sub>X aligns text in its own way, regardless of the way text is formatted in the input file. Now suppose you want to typeset something like this

```
The TEXnical Institute

Certificate

This is to certify that Mr. N. O. Vice has undergone a course at this institute
and is qualified to be a TEXnician.

The Director
The TEXnical Institute
```

This is produced by

```

\begin{center}
  The \TeX nical Institute\[\.75cm]
  Certificate
\end{center}
\noindent This is to certify that Mr. N. O. Vice has undergone a
course at this institute and is qualified to be a \TeX nician.
\begin{flushright}
  The Director\
  The \TeX nical Institute
\end{flushright}

```

Here, the commands

```
\begin{center} ... \end{center}
```

typesets the text between them exactly at the center of the page and the commands

```
\begin{flushright} ... \end{flushright}
```

typesets text flush with the right margin. The corresponding commands

```
\begin{flushleft} ... \end{flushleft}
```

places the enclosed text flush with the *left* margin. (Change the `flushright` to `flushleft` and see what happens to the output.)

These examples are an illustration of a  $\text{\LaTeX}$  construct called an *environment*, which is of the form

```
\begin{name} ... \end{name}
```

where *name* is the name of the environment. We have seen an example of an environment at the very beginning of this chapter (though not identified as such), namely the document environment.

## 1.3. FONTS

The actual letters and symbols (collectively called *type*) that  $\text{\LaTeX}$  (or any other typesetting system) produces are characterized by their *style* and *size*. For example, in this book emphasized text is given in *italic* style and the example inputs are given in typewriter style. We can also produce smaller and **bigger** type. A set of types of a particular style and size is called a *font*.

### 1.3.1. Type style

In  $\text{\LaTeX}$ , a type style is specified by family, series and shape. They are shown in the table [1.1](#).

Any type style in the output is a combination of these three characteristics. For example, by default we get roman family, medium series, upright shape type style in a  $\text{\LaTeX}$  output. The `\textit` command produces roman family, medium series, italic shape type. Again, the command `\textbf` produces roman family, boldface series, upright shape type.

We can combine these commands to produce a wide variety of type styles. For example, the input

```

\textsf{\textbf{sans serif family, boldface series, upright shape}}
\textrm{\textsl{roman family, medium series, slanted shape}}

```

Table 1.1:

	STYLE	COMMAND
FAMILY	roman	<code>\textrm{roman}</code>
	sans serif	<code>\textsf{sans serif}</code>
	typewriter	<code>\texttt{typewriter}</code>
SERIES	medium	<code>\textmd{medium}</code>
	<b>boldface</b>	<code>\textbf{boldface}</code>
SHAPE	upright	<code>\textup{upright}</code>
	<i>italic</i>	<code>\textit{italic}</code>
	<i>slanted</i>	<code>\textsl{slanted}</code>
	SMALL CAP	<code>\textsc{small cap}</code>

produces the output shown below:

**sans serif family, boldface series, upright shape**  
*roman family, medium series, slanted shape*

Some of these type styles may not be available in your computer. In that case,  $\text{\LaTeX}$  gives a warning message on compilation and substitutes another available type style which it thinks is a close approximation to what you had requested.

We can now tell the whole story of the `\emph` command. We have seen that it usually, that is when we are in the middle of normal (upright) text, it produces italic shape. But if the current type shape is slanted or italic, then it switches to upright shape. Also, it uses the family and series of the current font. Thus

```
\textit{A polygon of three sides is called a \emph{triangle} and a
  polygon of four sides is called a \emph{quadrilateral}}
```

gives

*A polygon of three sides is called a triangle and a polygon of four sides is called a quadrilateral*

while the input

```
\textbf{A polygon of three sides is called a
  \emph{triangle} and a polygon of four sides is called a
  \emph{quadrilateral}}
```

produces

A polygon of three sides is called a *triangle* and a polygon of four sides is called a *quadrilateral*

Each of these type style changing commands has an alternate form as a *declaration*. For example, instead of `\textbf{boldface}` you can also type `{\bfseries boldface}` to get **boldface**. Note that that not only the name of the command, but its usage also is different. For example, to typeset

By a **triangle**, we mean a polygon of three sides.

if you type

By a `\bfseries{triangle}`, we mean a polygon of three sides.

you will end up with

By a **triangle**, we mean a polygon of three sides.

Thus to make the declaration act upon a specific piece of text (and no more), the declaration and the text should be enclosed in braces.

The table below completes the one given earlier, by giving also the declarations to produce type style changes.

	STYLE	COMMAND	DECLARATION
SHAPE	upright	<code>\textup{upright}</code>	<code>{\upshape upright}</code>
	<i>italic</i>	<code>\textit{italic}</code>	<code>{\itshape italic}</code>
	<i>slanted</i>	<code>\textsl{slanted}</code>	<code>{\slshape slanted}</code>
	SMALL CAP	<code>\textsc{small cap}</code>	<code>{\scshape small cap}</code>
SERIES	medium	<code>\textmd{medium}</code>	<code>{\mdseries medium}</code>
	<b>boldface</b>	<code>\textbf{boldface}</code>	<code>{\bfseries boldface}</code>
FAMILY	roman	<code>\textrm{roman}</code>	<code>{\rmfamily roman}</code>
	sans serif	<code>\textsf{sans serif}</code>	<code>{\sffamily sans serif}</code>
	typewriter	<code>\texttt{typewriter}</code>	<code>{\ttfamily typewriter}</code>

These declaration names can also be used as environment names. Thus to typeset a long passage in, say, sans serif, just enclose the passage within the commands `\begin{sffamily} ... \end{sffamily}`.

## 1.4. TYPE SIZE

Traditionally, type size is measured in (printer) points. The default type that  $\text{\TeX}$  produces is of 10pt size. There are some *declarations* (ten, to be precise) provided in  $\text{\LaTeX}$  for changing the type size. They are given in the following table:

size	<code>{\tiny size}</code>	size	<code>{\large size}</code>
size	<code>{\scriptsize size}</code>	size	<code>{\Large size}</code>
size	<code>{\footnotesize size}</code>	size	<code>{\LARGE size}</code>
size	<code>{\small size}</code>	size	<code>{\huge size}</code>
size	<code>{\normalsize size}</code>	size	<code>{\Huge size}</code>

Note that the `\normalsize` corresponds to the size we get by default and the sizes form an ordered sequence with `\tiny` producing the smallest and `\Huge` producing the largest. Unlike the style changing commands, there are no *command-with-one-argument* forms for these declarations.

We can combine style changes with size changes. For example, the “certificate” we typed earlier can now be ‘improved’ as follows

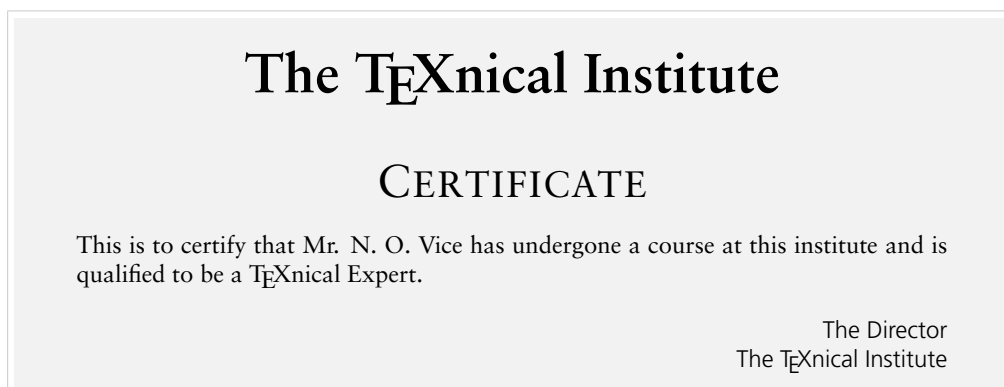
```
\begin{center}
{\bfseries\huge The \TeX nical Institute}\[1cm]
{\scshape\LARGE Certificate}
```

```
\end{center}
```

```
\noindent This is to certify that Mr. N. O. Vice has undergone a  
course at this institute and is qualified to be a \TeX nical Expert.
```

```
\begin{flushright}  
  {\sffamily The Director\  
  The \TeX nical Institute}  
\end{flushright}
```

and this produces





## TUTORIAL II

---

# THE DOCUMENT

### II.1. DOCUMENT CLASS

We now describe how an entire document with chapters and sections and other embellishments can be produced with  $\LaTeX$ . We have seen that all  $\LaTeX$  files should begin by specifying the kind of document to be produced, using the command `\documentclass{...}`. We've also noted that for a short article (which can actually turn out to be quite long!) we write `\documentclass{article}` and for books, we write `\documentclass{book}`. There are other *document classes* available in  $\LaTeX$  such as `report` and `letter`. All of them share some common features and there are features specific to each.

In addition to specifying the type of document (which we *must* do, since  $\LaTeX$  has no default document class), we *can* also specify some options which modify the default format. Thus the actual syntax of the `\documentclass` command is

```
\documentclass[options]{class}
```

Note that options are given in *square brackets* and not braces. (This is often the case with  $\LaTeX$  commands—options are specified within square brackets, after which mandatory arguments are given within braces.)

#### II.1.1. Font size

We can select the size of the font for the normal text in the entire document with one of the options

```
10pt 11pt 12pt
```

Thus we can say

```
\documentclass[11pt]{article}
```

to set the normal text in our document in 11 pt size. The default is 10pt and so this is the size we get, if we do not specify any font-size option.

#### II.1.2. Paper size

We know that  $\LaTeX$  has its own method of breaking lines to make paragraphs. It also has methods to make vertical breaks to produce different pages of output. For these breaks to work properly, it must know the width and height of the paper used. The various options for selecting the paper size are given below:

letterpaper	11×8.5 in	a4paper	20.7×21 in
legalpaper	14×8.5 in	a5paper	21×14.8 in
executivepaper	10.5×7.25 in	b5paper	25×17.6 in

Normally, the longer dimension is the vertical one—that is, the height of the page. The default is `letterpaper`.

### II.1.3. Page formats

There are options for setting the contents of each page in a single column (as is usual) or in two columns (as in most dictionaries). This is set by the options

```
onecolumn twocolumn
```

and the default is `onecolumn`.

There is also an option to specify whether the document will be finally printed on just one side of each paper or on both sides. The names of the options are

```
oneside twoside
```

One of the differences is that with the `twoside` option, page numbers are printed on the right on odd-numbered pages and on the left on even numbered pages, so that when these printed back to back, the numbers are always on the outside, for better visibility. (Note that  $\LaTeX$  has no control over the actual *printing*. It only makes the *formats* for different types of printing.) The default is `oneside` for `article`, `report` and `letter` and `twoside` for `book`.

In the `report` and `book` class there is a provision to specify the different chapters (we will soon see how). Chapters always begin on a new page, leaving blank space in the previous page, if necessary. With the `book` class there is the additional restriction that chapters begin only on odd-numbered pages, leaving an entire page blank, if need be. Such behavior is controlled by the options,

```
openany openright
```

The default is `openany` for `reportclass` (so that chapters begin on “any” *new* page) and `openright` for the `book` class (so that chapters begin only on *new* right, that is, odd numbered, page).

There is also a provision in  $\LaTeX$  for formatting the “title” (the name of the document, author(s) and so on) of a document with special typographic consideration. In the `article` class, this part of the document is printed along with the text following on the first page, while for `report` and `book`, a *separate* title page is printed. These are set by the options

```
notitlepage titlepage
```

As noted above, the default is `notitlepage` for `article` and `titlepage` for `report` and `book`. As with the other options, the default behavior can be overruled by explicitly specifying an option with the `documentclass` command.

There are some other options to the `documentclass` which we will discuss in the relevant context.

## II.2. PAGE STYLE

Having decided on the overall appearance of the document through the `\documentclass` command with its various options, we next see how we can set the style for the individual pages. In  $\LaTeX$  parlance, each page has a “head” and “foot” usually containing such information as the current page number or the current chapter or section. Just what goes where is set by the command

```
\pagestyle{...}
```

where the mandatory argument can be any one of the following *styles*

```
plain empty headings myheadings
```

The behavior pertaining to each of these is given below:

- plain** The page head is empty and the foot contains just the page number, centered with respect to the width of the text. This is the default for the `article` class if no `\pagestyle` is specified in the preamble.
- empty** Both the head and foot are empty. In particular, no page numbers are printed.
- headings** This is the default for the `book` class. The foot is empty and the head contains the page number and names of the chapter section or subsection, depending on the document class and its options as given below:

CLASS	OPTION	LEFT PAGE	RIGHT PAGE
book, report	one-sided	—	<i>chapter</i>
	two-sided	<i>chapter</i>	<i>section</i>
article	one-sided	—	<i>section</i>
	two-sided	<i>section</i>	<i>subsection</i>

- myheadings** The same as `headings`, except that the ‘section’ information in the head are not predetermined, but to be given explicitly using the commands `\markright` or `\markboth` as described below.

Moreover, we can customize the style for the *current page* only using the command `\thispagestyle{style}`

where *style* is the name of one of the styles above. For example, the page number may be suppressed for the current page alone by the command `\thispagestyle{empty}`. Note that only the *printing* of the page number is suppressed. The next page will be numbered with the next number and so on.

### II.2.1. Heading declarations

As we mentioned above, in the page style `myheadings`, we have to specify the text to appear on the head of every page. It is done with one of the commands

```
\markboth{left head}{right head}
\markright{right head}
```

where *left head* is the text to appear in the head on left-hand pages and *right head* is the text to appear on the right-hand pages.

The `\markboth` command is used with the `twoside` option with even numbered pages considered to be on the left and odd numbered pages on the right. With `oneside` option, all pages are considered to be right-handed and so in this case, the command `\markright` can be used. These commands can also be used to override the default head set by the `headings` style.

Note that these give only a limited control over the head and foot. since the general format, including the font used and the placement of the page number, is fixed by  $\LaTeX$ . Better customization of the head and foot are offered by the package `fancyhdr`, which is included in most  $\LaTeX$  distributions.

## II.3. PAGE NUMBERING

The style of page numbers can be specified by the command

```
\pagenumbering{...}
```

The possible arguments to this command and the resulting style of the numbers are given below:

arabic	Indo-Arabic numerals
roman	lowercase Roman numerals
Roman	upper case Roman numerals
alph	lowercase English letters
Alph	uppercase English letters

The default value is `arabic`. This command resets the page *counter*. Thus for example, to number all the pages in the ‘Preface’ with lowercase Roman numerals and the rest of the document with Indo-Arabic numerals, declare `\pagenumbering{roman}` at the beginning of the Preface and issue the command `\pagestyle{arabic}` immediately after the first `\chapter` command. (The `\chapter{...}` command starts a new chapter. We will come to it soon.)

We can make the pages start with any number we want by the command

```
\setcounter{page}{number}
```

where *number* is the page number we wish the current page to have.

## II.4. FORMATTING LENGTHS

Each page that  $\text{\LaTeX}$  produces consists not only of a *head* and *foot* as discussed above but also a *body* (surprise!) containing the actual text. In formatting a page,  $\text{\LaTeX}$  uses the width and heights of these parts of the page and various other lengths such as the left and right margins. The values of these lengths are set by the paper size options and the page format and style commands. For example, the page layout with values of these lengths for an odd page and even in this book are separately shown below.

These lengths can all be changed with the command `\setlength`. For example,

```
\setlength{\textwidth}{15cm}
```

makes the width of text 15 cm. The package `geometry` gives easier interfaces to customize page format.

## II.5. PARTS OF A DOCUMENT

We now turn our attention to the contents of the document itself. Documents (especially longer ones) are divided into chapters, sections and so on. There may be a title part (sometimes even a separate title page) and an abstract. All these require special typographic considerations and  $\text{\LaTeX}$  has a number of features which automate this task.

### II.5.1. Title

The ‘title’ part of a document usually consists of the name of the document, the name of author(s) and sometimes a date. To produce a title, we make use of the commands

```
\title{document name}
\author{author names}
\date{date text}
```

```
\maketitle
```

Note that after specifying the arguments of `\title`, `\author` and `\date`, we must issue the command `\maketitle` for this part to be typeset.

By default, all entries produced by these commands are centered on the lines in which they appear. If a title text is too long to fit in one line, it will be broken automatically. However, we can choose the break points with the `\` command.

If there are several authors and their names are separated by the `\and` command, then the names appear side by side. Thus