

1 Introduction

L^AT_EX is not just a system for typesetting mathematics. Its applications span the one-page memorandum, business and personal letters, newsletters, articles, and books covering the whole range of the sciences and humanities, ... right up to full-scale expository texts and reference works on all topics. Nowadays, versions of L^AT_EX exist for practically every type of computer and operating system. This book provides a wealth of information about its many present-day uses but first provides some background information.

The first section of this chapter looks back at the origins and subsequent development of L^AT_EX. The second section gives an overview of the file types used by a typical current L^AT_EX system and the role played by each. Finally, the chapter offers some guidance on how to use the book.

2 The Structure of a L^AT_EX Document

One of the ideas behind L^AT_EX is the separation between layout and structure(as far as possible), which allows the users to concentrate on content rather than having to worry about layout issues. This chapter explains how this general principle is implemented by L^AT_EX.

The first section of this chapter shows how document class files, packages, options, and preamble commands can affect the structure and layout of a document. The logical subdivisions of a document are discussed in general, before explaining in more detail how sectioning commands and their arguments define a hierarchical structure, how they generate numbers for titles, and how they produce running heads and feet. Different ways of typesetting section titles are presented with the help of examples. It is also shown how the information that is written to the table of contents can be controlled and how the look of this table, as well as that of the list of tables and figures, can be customized. The final section introduces L^AT_EX commands for managing cross-references and their scoping rules.

3 Basic Formatting Tools

The way information is presented visually can influence, to a large extent, the message as it is understood by the reader. Therefore, it is important that you use the best possible tools available to convey the precise meaning of your words. It must, however, be emphasized that visual presentation forms should aid the reader in understanding the text, and should not distract his or her attention. For this reason, visual consistency and uniform conventions for the visual clues are a must, and the way given structural elements are highlighted should be the same throughout a document. This constraint is most easily implemented by defining a specific command or environment for each document element that has to be treated specially and by grouping these commands and environments in a package file or in the document preamble. By using exclusively these commands, you can be sure of a consistent presentation form.

This chapter explains various ways for highlighting parts of a document. The first part looks at how short text fragment or paragraphs can be made to stand out and describes tools to manipulate such elements.

The second part deals with the different kind of “notes”, such as footnotes, marginal notes, and endnotes, and explains how they can be customized to conform different style, if necessary.

Typesetting lists is the subject of the third part. First, the various parameters and commands controlling the standard L^AT_EX lists, `enumerate`, `itemize`, and `description`, are discussed. Then, the extensions provided by the `paralist` package and the concept of “headed list” exemplified by the `amsthm` package are presented. These will probably satisfy the structure and layout requirements of most readers. If not, then the remainder of this part introduces the generic `list` environment and explains how to build custom layouts by varying the values of the parameters controlling it.

The fourth part explains how to simulate “verbatim” text. In particular, we have a detailed look at the powerful package `fancyvrb` and `listings`.

The final part presents packages that deal with line numbering, handling of columns, such as parallel text in two columns, or solving the problem of producing multiple columns.

4 The Layout of the Page

In this chapter we will see how to specify different page layouts. Often a single document requires several different page layout. For instance, the layout of the first page of a chapter, which carries the chapter title, is generally different from that of the other pages in that chapter.

We first introduce L^AT_EX's dimensional parameters that influences the page layout and describe ways to change them and visualize their values. This is followed by an in-depth discussion of the package `typearea` and `geometry`, both of which provide sophisticated ways to implement page layout specifications. The third section deals with the L^AT_EX concepts used to provide data for running headers and footers. This is followed by a section that explains how to format such elements, including many examples deploying the `fancyhdr` package and others. The fifth section then introduces commands that help in situations when the text does not fit into the layout and manual intervention is required. The chapter concludes with a brief look at two generic classes that go a long way toward providing almost full control over the page layout specification process.

5 Tabular Material

Data is often most efficiently presented in tabular form. T_EX uses powerful primitive for arranging material in rows and columns. Because they implement only a low-level, formatting-oriented functionality, several macro package have been developed that build on those primitive a higher-level command language and a more user-friendly interface.

In L^AT_EX, two type of environment for constructing tables are provided. Most commonly the `tabular` environment or its math-mode equivalent, the `array` environment, is used. However, in some circumstances the `tabbing` environment might prove useful.

Tables typically form large units of the document that must be allowed to “float” so that the document may be paginated correctly. The environments described in this chapter are principally concerned with the table layout. To achieve correct pagination they will often be used within the `table` environment described in Chapter 6. An exception is the environments for multipage

tables described in Section 5.4, which should never be used in conjunction with the \LaTeX float mechanism. Be careful, however, not to confuse the `tabular` environment with the `table` environment. The former allows material to be aligned in columns, while the latter is a logical document element identifying its contents as belonging together and allowing the material to be floated jointly. In particular, one `table` environment can contain several `tabular` environments.

After taking a quick look at the `tabbing` environment, this chapter describes the extension to \LaTeX 's basic `tabular` and `array` environment provided by the `array` package. This package offers increased functionality, especially in terms of a more flexible positioning of paragraph material, a better control of inter-column and inter-row spacing, and the possibility of defining new preamble specifiers. Several packages build on the primitives provided by the `array` package to provide specific extra functionality. By combining the features in these packages, you will be able to construct complex tables in a simple way. For example, the `tabularx` and `tabulary` packages provide extra column types that allow table column widths to be calculated automatically.

Standard \LaTeX environments do not produce tables that may be broken over a page. We give several examples of multipage tables using the `supertabular` and `longtable` environments provided by the similarly named packages.

We then briefly look at the use of color in tables and at several packages that give finer control over rules, and spacing around rules, in tables. Next, we discuss table entries spanning multiple rows, created by `multirow` package, and the `dcolumn` package, which provides a mechanism for aligning columns of figures on a decimal points.

We also discuss the use of footnotes in tables. The `threepartable` package provides a convenient mechanism to have table notes and captions combined with a tabular layout.

The final section gives some practical advice on handling nested tables and large entries spanning multiple columns.

Mathematically oriented readers should consult the chapter on advanced mathematics, especially Section 8.2 on page 468, which discusses the alignment structures for equations. Further examples of table layouts may be found in the section on graphics packages, Section 10.3 on page 628.

6 Mastering Floats

Documents would be easier to read if all the material that belong together was never split between pages. However, that is often technically impossible and \TeX will, by default, split textual material between two pages to avoid partially filled pages. Nevertheless, when this outcome is not desired(as with figures and tables), the material must be “floated” to a convenient place, such as the bottom or the top of the current or next page, to prevent half-empty pages.

This chapter shows how “large chunks” of material can be kept conveniently on the same page by using a float object. We begin by introducing the parameters that defines how \LaTeX typesets its basic **figure** and **table** float environments, and we describe some of the packages that make it easy to control float placement (Section 6.2). We then continue by explaining how you can define and use your own floating environments (Section 6.3.1), or conversely, how capturing commands can be used to enter information into the list of figures and tables for nonfloating materials(Section 6.3.2). Then methods for rotating the content of a float are described(Section 6.3.3).

It is often visually pleasing to include a “picture” inside a paragraph, with the text wrapping around it. Various packages have been written to achieve this goal more or less easily ; in Section 6.4 we look at two of them in some detail.

The final section addresses the problem of customizing captions. There is a recognized need to be able to typeset the description of the contents of figures and tables in many different ways. This includes specifying sub-figures and sub-tables, each with its own caption and label, inside a large float.

Many float-related packages have been developed over the years and we cannot hope to mention them all here. In fact, the packages that we describe often feature quite a few more commands than we are able to illustrate. Our aim is to enable you to make an educated choice and to show how a certain function can be obtained in a given framework. In each case consulting the original documentation will introduce you to the full possiblities of a given package.

7 Fonts and Encodings

Half of the job of \TeX (\LaTeX) as a typesetting system is to process the source document and to calculate from it the character's positions on the output page. But \TeX (\LaTeX) has only a primitive knowledge about these characters, which it basically regards as black boxes having a width, height, and depth. For each font these dimensions are stored in a separate external files, the so-called \TeX font metric or `.tfm` file.

The character shapes that correspond to such a `.tfm` file come into play at a later stage, after \TeX (\LaTeX) has produced its `.dvi` files. Character placement in the `.pk` file or in outline descriptions(e.g. PostScript) are combined by a driver program that produces the character image on the output medium. Usually one driver program is needed for every output medium—for screen representation, a low-resolution laser printer, or other device. With \TeX variants such as \pdfTeX or \VTeX that bypass the production of `.dvi` output and instead directly generate PDF or PostScript output, the situation is slightly different(but, as far as \LaTeX concerned, similar). In that case the character shapes are “added” when the underlying formatter produces the final output format. That is, the driver program is internal, but the basic concepts are identical.

8 Higher Mathematics

Basic \LaTeX offers excellent mathematical typesetting capabilities for straightforward documents. However, when complex displayed equations or more advanced mathematical constructs are heavily used, something more is needed. Although it is possible to define a new commands or environments to ease the burden of typing in formulas, this is not the best solution. The American Mathematical Society(AMS) provides a major package, `amsmath`, which make the preparation of mathematical documents much less time-consuming and more consistent. It forms the core of a collection of packages known as \AMS-LaTeX and is the major subject of this chapter. A useful book by George Grätzer also covers these packages in detail.

This chapter describes briefly, and provides examples of, a subsequential number of the many features of these packages as well as few closely related

packages; it also gives a few pointers to other relevant packages. In addition, it provides some essential background on mathematical typesetting with \TeX . Thus, it covers some of standard \LaTeX 's features for mathematical formulas, though these are not the main aims of this chapter.

It is also definitely not a comprehensive manual of good practice for typesetting mathematics with \LaTeX . Indeed, many of the examples are offered up purely for illustration purposes and, therefore, present neither good design, nor good mathematics, nor necessarily good \LaTeX coding.

Advice on how to typeset mathematics according to late 20th-century U.S. practice can be found in Ellen Swanson's *Math into Type*. Many details concerning how to implement this advice using \TeX or, equally, standard \LaTeX appear in Chapters 16-18 of Donald Knuth's *The \TeX book*.