The Not So Short Introduction to LaTeX

Or LaTeX in 335 minutes

Tobias Oetiker • Marcin Serwin

Hubert Partl • Irene Hyna • Elisabeth Schlegl

Nightly version 7.0@81a652a67041cd73809ada1a11c5968600594c58,
August 30, 2023

Copyright © 1995–2022 Tobias Oetiker, Marcin Serwin, Hubert Partl, Irene Hyna, Elisabeth Schlegl and Contributors.

This document is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

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*. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this document. If not, see https://www.gnu.org/licenses/.

The illustration on the first page was created for this booklet by Roman Schmid (@bummzack on GitHub). It is dedicated to the public domain via CC0 1.0 Universal Public Domain Dedication. This means that you can use it freely in your own work without any licensing issues. See https://creativecommons.org/publicdomain/zero/1.0/ for more details.

Additionally, the code examples (text written using `monospace font` in this document) are dedicated to the public domain via CC0 1.0 Universal Public Domain Dedication. This means that you can use them freely in your own work without any licensing issues. See https://creativecommons.org/publicdomain/zero/1.0/ for more details.

The full text of the GNU General Public License can be found in Appendix C on page 285.

Thank you!

Much of the material used in this introduction comes from an Austrian introduction to LaTeX 2.09 written in German by:

Hubert Partl <partl@mail.boku.ac.at>
Zentraler Informatikdienst der Universität für Bodenkultur Wien

Irene Hyna <Irene.Hyna@bmwf.ac.at>
Bundesministerium für Wissenschaft und Forschung Wien

Elisabeth Schlegl <noemail>
in Graz

If you are interested in the German document, you can find a version updated for LaTeX 2_ε by Jörg Knappen at CTAN://info/lshort/german

The following individuals helped with corrections, suggestions, and material to improve this paper. They put in a big effort to help me get this document into its present shape. I would like to sincerely thank all of them. Naturally, all the mistakes you'll find in this book are mine. If you ever find a word that is spelled correctly, it must have been one of the people below dropping me a line.

If you want to contribute to this booklet, you can find all the source code on https://github.com/oetiker/lshort. Your pull-requests will be appreciated.

Eric Abrahamsen, Lenimar Nunes de Andrade, Eilinger August, Rosemary Bailey, Barbara Beeton, Marc Bevand, Connor Blakey, Salvatore Bonaccorso, Pietro Braione, Friedemann Brauer, Markus Brühwiler, Jan Busa, David Carlisle, Neil Carter, Carl Cerecke, Mike Chapman, Pierre Chardaire, Xingyou Chen, Christopher Chin, Diego Clavadetscher, Wim van Dam, Benjamin Deschwanden Jan Dittberner, Michael John Downes, Matthias Dreier, David Dureisseix, Hans Ehrbar, Elliot, Rockrush Engch, William Faulk, Robin Fairbairns, Johan Falk, Jörg Fischer, Frank Fischli, Daniel Flipo, Frank, Mic Milic Frederickx, David Frey, Erik Frisk, Hans Fugal, Robert Funnell, Greg Gamble, Andy Goth, Cyril Goutte, Kasper B. Graversen, Arlo Griffiths, Alexandre Guimond, Neil Hammond, Christoph Hamburger, Rasmus Borup Hansen, Joseph Hilferty, Daniel Hirsbrunner, Martien Hulsen, Björn Hvittfeldt, Morten Høgholm, Werner Icking, Eric Jacoboni, Jakob, Alan Jeffrey, Martin Jenkins, Byron Jones, David Jones, Johannes-Maria Kaltenbach, Nils Kanning, Andrzej Kawalec, Christian Kern, Alain Kessi, Axel Kielhorn, Sander de Kievit, Kjetil Kjernsmo, Tobias Klauser, Jörg Knappen, Michael Koundouros, Matt Kraai, Tobias Krewer, Flori Lambrechts, Mike Lee, Maik Lehradt, Rémi Letot, Axel Liljencrantz, Jasper Loy, Johan Lundberg, Martin Maechler, Alexander Mai, Claus Malten, Kevin Van Maren, Pablo Markin, I. J. Vera Marún, Hendrik Maryns, Chris McCormack, Aleksandar S. Milosevic, Henrik Mitsch, Stefan M. Moser, Armin Müller, Philipp Nagele, Richard Nagy, Manuel Oetiker, Urs Oswald, Hubert Partl, Marcelo Pasin, Martin Pfister, Lan Thuy Pham, Breno Pietracci, Demerson Andre Polli, Maksym Polyakov, Nikos Pothitos, John Refling, Mike Ressler, Brian Ripley, Kurt Rosenfeld, Bernd Rosenlecher, Chris Rowley, Young U. Ryu, Risto Saarelma, András Salamon, José Carlos Santos, Christopher Sawtell, Gilles Schintgen, Craig Schlenter, Hanspeter Schmid, Baron Schwartz, John Scott, Jordi Serra i Solanich, Miles Spielberg, Susan Stewart, Matthieu Stigler, Geoffrey Swindale, Laszlo Szathmary, Boris Tobotras, Josef Tkadlec, Scott Veirs, Didier Verna, Carl-Gustav Werner,

Fabian Wernli, Matthew Widmann, David Woodhouse, Chris York,
Rick Zaccone, Fritz Zaucker, and Mikhail Zotov.

Preface

LaTeX [42] is a typesetting system that is very suitable for producing scientific and mathematical documents of high typographical quality. It is also suitable for producing all sorts of other documents, from simple letters to complete books. LaTeX uses TeX [39] as its formatting engine.

This short introduction will teach you all you need to get going with LaTeX. Refer to [42, 48] for a complete description of the LaTeX system.

This introduction is split into 7 chapters:

Chapter 1 tells you about the basic structure of LaTeX documents. You will also learn a bit about the history of LaTeX. After reading this chapter, you should have a rough understanding of how LaTeX works.

Chapter 2 goes into the details of typesetting your documents. It explains most of the essential LaTeX commands and environments. After reading this chapter, you will be able to write your first documents, with itemized lists, tables, graphics and floating bodies.

Chapter 3 explains how to typeset formulae with LaTeX. Many examples demonstrate how to use one of LaTeX's main strengths.

Chapter 4 shows how to create bibliographies for your publications.

Chapter 5 explains the secrets of indexes and some finer points about creating PDFs.

Chapter 6 shows how to use LaTeX for creating graphics. Instead of drawing a picture with some graphics program, saving it to a file

and then including it into LaTeX, you describe the picture you want and have LaTeX draw it for you.

Chapter 7 contains some potentially dangerous information about how to alter the standard document layout produced by LaTeX. You will learn how to change things around in such a way that the beautiful output of LaTeX turns ugly or stunning, depending on your abilities.

It is important to read the chapters in order—the book is not that big, after all. Be sure to carefully read the examples, because a lot of the information is in the examples placed throughout the book.

LaTeX is available for most computers, from the PC and Mac, to large UNIX and VMS systems. On many university computer clusters, you will find that a LaTeX installation is available, ready to use. Information on how to access the local LaTeX installation should be provided in the *Local Guide* [1]. If you have problems getting started, ask the person who gave you this booklet. The scope of this document is *not* to tell you how to install and set up a LaTeX system, but to teach you how to write your documents so that they can be processed by LaTeX.

If you need to get hold of any LaTeX related material, have a look at one of the Comprehensive TeX Archive Network (CTAN) sites. The homepage is at `http://www.ctan.org`.

You will find other references to CTAN throughout the book, especially pointers to software and documents you might want to download. Instead of writing down complete URLs, I just wrote `CTAN:` followed by whatever location within the CTAN tree you should go to.

If you want to run LaTeX on your own computer, take a look at what is available from `CTAN://systems`.

If you have ideas for something to be added, removed or altered in this document, please let me know. I am especially interested in feedback from LaTeX novices about which bits of this intro are easy to understand and which could be explained better.

Tobias Oetiker <tobi@oetiker.ch>

OETIKER+PARTNER AG
Aarweg 15
4600 Olten
Switzerland

The current version of this document is available on
CTAN://info/lshort

Contents

Thank you! **i**

Preface **v**

1 LaTeX Basics **1**

1.1 A Bit of History 1
1.1.1 TeX 1
1.1.2 Other TeX engines 2
1.1.3 LaTeX 3
1.2 Basics 3
1.2.1 Author, Book Designer, and Typesetter 3
1.2.2 Layout Design 4
1.2.3 Advantages and Disadvantages 5
1.3 LaTeX Input Files 6
1.3.1 Spaces 6
1.3.2 Comments 7
1.3.3 Special Characters 7
1.3.4 LaTeX Commands 8
1.3.5 Groups 8
1.3.6 Optional parameters 9
1.4 Input File Structure 9
1.5 A Typical Command Line Session 10
1.6 Logical Structure of Your Document 13
1.6.1 A Neverending Story of Problems with WYSIWYG Editors 13
1.6.2 Your First Text Command 14

1.6.3 Your First Environment 15
1.6.4 Summary 16
1.7 Packages 16
1.8 The Structure of Text and Language 17
1.9 Files You Might Encounter 20

2 Real World LaTeX 23
2.1 Line Breaking and Page Breaking 23
2.1.1 Justified Paragraphs 23
2.1.2 Hyphenation 25
2.2 Ready-Made Strings 26
2.3 Dashes and Hyphens 27
2.4 Slash (/) 27
2.5 Ellipsis (...) 27
2.6 Ligatures 28
2.7 Abstract 28
2.8 International Language Support 29
2.8.1 Entering Characters 29
2.8.2 Polyglossia Usage 30
2.8.3 Right to Left (RTL) languages 35
2.8.4 Chinese, Japanese and Korean (CJK) 36
2.9 Simple Commands 37
2.10 The Space Between Words 39
2.11 Titles, Chapters, and Sections 40
2.12 Cross References 42
2.13 Footnotes 43
2.14 Lists 43
2.15 Non-Justified Text 44
2.16 Quotations 45
2.16.1 Formal Quotes 45
2.16.2 Foreign Quotes 47
2.16.3 Long Quotations and Poetry 48
2.17 Code Listings 49
2.17.1 Verbatim 49
2.17.2 The `listings` Package 51
2.17.3 The minted Package 59

2.18 Tables 63
2.18.1 Basic Tables 64
2.18.2 Long Tables 71
2.18.3 Advanced Tables and Non-Tables 75
2.19 Including Graphics and Images 79
2.20 Floating Bodies 82
2.20.1 Createing your own Float-Types 87
2.20.2 The `longtable` Environment 87
2.21 Big Projects 89

3 Typesetting Mathematical Formulae 93
3.1 Modern Mathematics 93
3.2 Single Equations 94
3.2.1 Math Mode 96
3.3 Building Blocks of Mathematical Formulae 97
3.3.1 Basic Arithmetic 97
3.3.2 Logic and Set Theory 99
3.3.3 Greek Letters 100
3.3.4 Mathematical Fonts and How To Use Them 101
3.3.5 Big Operators 104
3.3.6 Math Accents 107
3.3.7 Delimiters 108
3.4 Multiline Equations 111
3.4.1 Long Equations 112
3.4.2 Multiple Unaligned Equations 113
3.4.3 Multiple Aligned Equations 114
3.4.4 Equations as Building Blocks 115
3.4.5 IEEEeqnarray Environment 117
3.5 Units 120
3.5.1 Pitfalls of Naïvely Entered Units 120
3.5.2 Basic Commands of the siunitx Package 121
3.5.3 Other siunitx Commands 125
3.5.4 Table Columns with Numbers 129
3.6 Matrices and the like 132
3.7 Spacing in Math Mode 139
3.7.1 Mathematical Object Classes 139

3.7.2 Manual Spacing 141
3.7.3 Phantoms 142
3.8 Theorems and Proofs 143
3.9 Fiddling with Math Styles 146
3.10 Dots 148
3.11 More About Fractions 149

4 Bibliographies 153
4.1 `thebibliography` environment 153
4.2 biblatex with biber 155
4.2.1 Database files 155
4.2.2 Using biblatex 156
4.2.3 Controlling the bibliography 158
4.2.4 Citing commands 163
4.2.5 More about entries 165

5 Specialities 167
5.1 Indexing 167
5.2 Installing Extra Packages 169
5.3 LaTeX and PDF 170
5.3.1 Hypertext Links 170
5.3.2 Document Metadata 174
5.3.3 Problems with Outline 175
5.4 Creating Presentations 176
5.4.1 Basic Usage 177
5.4.2 Overlay Specification 180
5.4.3 Customisation 183
5.4.4 Handouts 186

6 Graphics in Your Document 191
6.1 Overview 191
6.2 Basic Usage 193
6.3 Curves and Shapes 196
6.4 Customizing Paths and Nodes 199
6.5 Coordinates 203
6.6 Reusing Pictures 206
6.7 Libraries 209

7 Customising LaTeX 213
7.1 New Commands, Environments and Packages 213
7.1.1 New Commands 214
7.1.2 New Environments 218
7.1.3 Copying commands 220
7.1.4 Command-line LaTeX 221
7.1.5 Your Own Package 222
7.2 Fonts and Sizes 224
7.2.1 Font Changing Commands 224
7.2.2 Danger, Will Robinson, Danger 229
7.2.3 Advice 229
7.3 Custom Fonts with fontspec 230
7.3.1 Main Document Fonts 230
7.3.2 Specifying Fonts via Filenames 232
7.3.3 Defining New Fonts 234
7.3.4 Math Fonts 235
7.4 Colours 237
7.4.1 Coloured Text 237
7.4.2 Models 239
7.4.3 Defining Your Own Colours 241
7.4.4 Colourful Pages and Boxes 242
7.5 Lengths and Spacing 242
7.5.1 LaTeX Units 242
7.5.2 Horizontal Space 243
7.5.3 Vertical Space 247
7.5.4 Length Variables 248
7.6 The Layout of the Document 252
7.6.1 Document Class Options 252
7.6.2 Page Styles 252
7.6.3 Line Spacing 254
7.6.4 Paragraph Formatting 256
7.6.5 Page Layout 257
7.7 Fancy Headers 260
7.7.1 Basic commands 260
7.7.2 Contents of the headers 263
7.7.3 Advanced commands 265

7.8 Boxes 269
7.9 Rules 272

A Installing LaTeX **273**
A.1 What to Install 273
A.2 Cross Platform Editor 274
A.3 TeX on macOS 274
A.3.1 TeX Distribution 274
A.3.2 macOS TeX Editor 274
A.3.3 Treat yourself to PDFView 275
A.4 TeX on Windows 275
A.4.1 Getting TeX 275
A.4.2 A LaTeX editor 275
A.4.3 Document Preview 275
A.4.4 Working with graphics 276
A.5 TeX on Linux 276

B Things You Shouldn't Use **277**
B.1 ... for Display Math 277
B.2 ... for Inline Math 278
B.3 ... for Typesetting Math 278
B.4 ... for Defining New Commands 279
B.5 ... for Copying Commands 279
B.6 ... for Aligning Equations 280
B.7 ... for Changing Fonts 280
B.8 ... for Changing Text Alignment 281
B.9 ... for Typesetting Quotations 281
B.10 ... for Printing Verbatim 282
B.11 ... for Adding a Bibliography 282
B.12 ... for Creating Relations and Operators 283
B.13 ... for Changing Math Font 283
B.14 ... for Spacing 284

C GNU General Public License, Version 3 **285**

Bibliography **303**

Index **310**

List of Figures

List of Tables

1.1 Document Classes. 10
1.2 Examples of LaTeX packages. 18

2.1 Accents and Special Characters. 31
2.2 Float Placing Permissions. 85

3.1 Available variants of Greek letters. 102
3.2 Commands that change the font of mathematical symbols. An empty example field indicates that the Unicode does not define glyphs in the given set. 103
3.3 All functions predefined by LaTeX. 104
3.4 Commands to influence mathematical object classes . . . 140
3.5 Commands for manual math spacing 142
3.6 Math style commands available in LaTeX 147

5.1 Index Key Syntax Examples. 169

7.1 Default font changing commands of LaTeX. 227
7.2 Commands changing font size. 227
7.3 Absolute point sizes in standard classes. 228
7.4 Basic colours predefined by the xcolor package. 238
7.5 LaTeX Units. 244
7.6 Document Class Options. 253
7.7 The Predefined Page Styles of LaTeX. 254
7.8 Possible argument of the `\pagenumbering` command. . . . 255

Chapter 1

LaTeX Basics

The first part of this chapter presents a short overview of the philosophy and history of LaTeX. The second part focuses on the basic structures of a LaTeX document. After reading this chapter, you should have a rough knowledge of how LaTeX works, which you will need to understand the rest of this book.

1.1 A Bit of History

1.1.1 TeX

TeX is a computer program created by Donald E. Knuth [39]. The original program was aimed at typesetting text and mathematical formulae. Knuth started writing the TeX typesetting engine in 1977 to explore the potential of digital printing equipment. These new devices were beginning to infiltrate the publishing industry at that time. His goal was to reverse the trend of deteriorating typographical quality that he saw affecting his own books and articles. The first stable version of TeX was released in 1982. Version 3.0 was released in 1989 to better support 8-bit characters and multiple languages. Knuth considered the TeX-design to be complete with the release of Version 3. TeX is renowned for being extremely stable, for running on many kinds of computers, and for being virtually bug free. The version number of TeX is converging to the π constant and is now at 3.141592653.

TeX is pronounced "Tech", with a "ch" as in the German word "Ach"[1] or in the Scottish "Loch"' The "ch" originates from the Greek alphabet where X is the letter "ch" or "chi". TeX is also the first syllable of the Greek word technique. In an ASCII environment, TeX becomes `TeX`.

1.1.2 Other TeX engines

While the original TeX engine is still fully functional, and can be used to typeset documents, some of the technical solutions it uses are now dated. Over the years, the TeX engine has been extended, introducing many new features. Due to the way TeX is licensed, anyone is free to produce enhanced versions of TeX, but they must not call the program TeX anymore. The first widely popular enhanced version of TeX was ε-TeX [79].

The original program produced `.dvi` files which were meant only to be sent to a printer. With the proliferation of high resolution displays, it became more common to read documents directly on-screen without printing them. This prompted creation of another extension, called pdfTeX, which could produce standard PDF files. Yet another problem was the original font format, which was not compatible with modern font formats. This was in turn solved in XeTeX.

Today, four TeX engines are actively maintained: the original TeX, pdfTeX, XeTeX and LuaTeX. This book recommends using either XeTeX or LuaTeX. The examples presented should produce the same results on both engines (except where otherwise noted). The basic examples will work with pdfTeX[2] too, but we suggest to switch to XeTeX or LuaTeX from the start, to avoid complications down the road as you explore more advanced concepts.

[1] In German there are actually two pronunciations for "ch" and one might assume that the soft "ch" sound from "Pech" would be a more appropriate. When asked about this by one of the German Wikipedia contributors, Knuth wrote: "I do not get angry when people pronounce TeX in their favorite way…and in Germany many use a soft ch because the χ follows the vowel e, not the harder ch that follows the vowel a. In Russia, 'tex' is a very common word, pronounced 'tyekh'. But I believe the most proper pronunciation is heard in Greece, where you have the harsher ch of ach and Loch." ([38])

[2] The original TeX will not work at all, because LaTeX requires an ε-TeX-enabled engine since 2017 [56].

1.1.3 LaTeX

LaTeX is a set of macros[3] for the TeX engine. LaTeX was originally developed by Leslie Lamport for his own use. After some consideration, he decided to make them more general so that others could use them for their own projects. Thus, in 1985, the first version of LaTeX — named LaTeX 2.09 — was released [42].

The original LaTeX became quite popular and promoted the creation of many extension packages. Unfortunately, some of the more popular extensions were not compatible with each other. LaTeX 2_ε managed to unify many of the extensions, and also provided an extension packaging system, dealing with third party extensions in a standardised way.

The same year LaTeX 2_ε was released, the LaTeX3 project was started. Its aim was to create improved standards for writing LaTeX documents, fixing some of the mistakes that were made when defining the initial LaTeX macros. While at the beginning, it was planned to release LaTeX3 as a standalone system that was not backward compatible with LaTeX 2_ε, in the end, the consensus was that abandoning the huge collection of third party packages written for LaTeX 2_ε, would be a mistake. Thus, the development team decided that LaTeX3 would be slowly backported into LaTeX 2_ε format, while avoiding breaking changes as much as reasonably possible [47].

LaTeX is pronounced "Lay-tech" or "Lah-tech." If you refer to LaTeX in an ASCII environment, you type `LaTeX`. LaTeX 2_ε is pronounced "Lay-tech two e" and typed `LaTeX2e`. LaTeX3 is pronounced "Lay-tech three" and typed `LaTeX3`.

1.2 Basics

1.2.1 Author, Book Designer, and Typesetter

To publish something, authors give their typed manuscript to a publishing company. One of their book designers then decides the layout of the document (column width, fonts, space before and after headings, …). The

[3] Macros are short names for long lists of instructions, which are created to avoid retyping the instructions each time they are needed.

book designer writes his instructions into the manuscript and then gives it to a typesetter, who typesets the book according to these instructions.

A human book designer tries to find out what the author had in mind while writing the manuscript. He decides on chapter headings, citations, examples, formulae, etc., based on his professional knowledge and from the contents of the manuscript.

In a LaTeX environment, LaTeX takes the role of the book designer and uses TeX as its typesetter. But LaTeX is "only" a program and therefore needs more guidance. The author has to provide additional information to describe the logical structure of his work. This information is written into the text as "LaTeX commands."

This is quite different from the WYSIWYG[4] approach that most modern word processors, such as *MS Word* or *LibreOffice*, take. With these applications, authors specify the document layout interactively while typing text into the computer. They can see on the screen how the final work will look when it is printed.

When using LaTeX, it is not normally possible to see the final output while typing the text, but the final output can be previewed on the screen after processing the file with LaTeX. Corrections can then be made before actually sending the document to the printer.

1.2.2 Layout Design

Typographical design is a craft. Unskilled authors often commit serious formatting errors by assuming that book design is mostly a question of aesthetics—"If a document looks good artistically, it is well-designed." But as a document has to be read and not hung up in a picture gallery, the readability and comprehensibility is much more important than the beautiful look of it. Examples:

- The font size and the numbering of headings have to be chosen to make the structure of chapters and sections clear to the reader.

- The line length has to be short enough to not strain the eyes of the reader, while long enough to fill the page beautifully.

[4] What You See Is What You Get.

With WYSIWYG systems, authors tend to generate aesthetically pleasing documents with very little, or inconsistent, structure. LaTeX prevents such problems by forcing the author to declare the *logical* structure of his document. LaTeX then chooses the most suitable layout.

1.2.3 Advantages and Disadvantages

When people from the WYSIWYG world meet people who use LaTeX, they often discuss "the advantages of LaTeX over a normal word processor", or the opposite. The best thing to do when such a discussion starts, is to keep a low profile, since such discussions like to get out of hand. But sometimes there is no escaping …

So here is some ammunition. The main advantages of LaTeX over normal word processors are the following:

- Professionally crafted layouts are available, which make a document really look as if "printed".
- The typesetting of mathematical formulae is supported out of the box.
- Users only need to learn a few easy-to-understand commands that specify the logical structure of a document. They almost never need to tinker with the actual layout of the document.
- Even complex structures, such as footnotes, references, table of contents, and bibliographies, can be generated easily.
- Free add-on packages exist for many typographical tasks not directly supported by basic LaTeX. For example, packages are available to include PostScript graphics or to typeset bibliographies conforming to exact standards. Many of these add-on packages are described in *The LaTeX Companion* [48].
- LaTeX encourages authors to write well-structured texts, because this is how LaTeX works—by specifying structure.
- TeX, the formatting engine of LaTeX, is highly portable and free. Therefore, the system runs on almost any hardware platform available.

LaTeX also has some disadvantages, and I guess it's a bit difficult for me to find any sensible ones, though I am sure other people can tell you hundreds `;-)`.

- LaTeX does not work well for people who have sold their souls …
- Although some parameters can be adjusted within a predefined document layout, the design of a whole new layout is difficult and takes a lot of time.
- It is very hard to write unstructured and disorganised documents.
- Your hamster might, despite some encouraging first steps, never be able to fully grasp the concept of Logical Markup.

1.3 LaTeX Input Files

The input for LaTeX is a plain text file. On Unix-like systems text files are pretty common. On Windows, one could use Notepad to create a text file. It contains the text of the document, as well as the commands that tell LaTeX how to typeset the text. For beginners, it is recommended to use a LaTeX IDE.[5]

1.3.1 Spaces

"Whitespace" characters, such as blank or tab, are treated uniformly as "space" by LaTeX. Several consecutive whitespace characters are treated as *one* "space". Whitespace at the start of a line is generally ignored, and a single line break has the same effect as "whitespace".

An empty line between two lines of text defines the end of a paragraph. Multiple empty lines are treated the same as *one* empty line. The text below is an example. On the left-hand side is the text from the input

[5] Some examples of these are listed in Appendix A.

file, and on the right-hand side is the formatted output.

```
It does not matter whether you
enter one or many     spaces
after a word.

An empty line starts a new
paragraph.
```

It does not matter whether you enter one or many spaces after a word.
An empty line starts a new paragraph.

1.3.2 Comments

When LaTeX encounters a % character while processing an input file, it ignores the rest of the present line, the line break, and all whitespace at the beginning of the next line.

This can be used to write notes into the input file, which will not show up in the printed version.

```
This is an % stupid
% Better: instructive <----
example: Supercal%
              ifragilist%
    icexpialidocious
```

This is an example: Supercalifragilisticexpialidocious

The % character can also be used to split long input lines where no whitespace or line breaks are allowed.

1.3.3 Special Characters

The following symbols are reserved characters that have a special meaning under LaTeX. If you enter them directly in your text, they will normally not print, but rather coerce LaTeX to do things you did not intend.

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

As you will see, these characters can be used in your documents all the same by using a prefix backslash:

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

$ % ^ & _ { } ~ \

Many more other symbols and many more can be printed with special commands in mathematical formulae or as accents. The backslash character \ can *not* be entered by adding another backslash in front of it (`\\`); this sequence is used for line breaking. Use the `\textbackslash` command instead.

1.3.4 LaTeX Commands

LaTeX commands are case-sensitive, and take one of the following two formats:

- They start with a backslash \ and then have a name consisting of letters only. Command names are terminated by a space, a number or any other 'non-letter', for example: `\emph`, `\begin`, `\LaTeX`.
- They consist of a backslash and exactly one non-letter, for example: `\\`, `\{`, `\"`.

Many commands also exist in a 'starred variant' where a star is appended to the command name.

1.3.5 Groups

Many commands act on parameters. Parameters are the first things the command encounters after its name ends in the source file. If the command name consists of letters, then it ignores following spaces. For example, if command 'foo' accepts two arguments, then

```
\foo bar
```

will be read as command 'foo' with first argument 'b' and second argument 'a' followed by the letter 'r'. In order to pass more than one letter as a parameter, groups are used.

Groups are delimited by `{` and `}`. They tell LaTeX to treat the content between them as a single unit. For example,

```
\foo{bar}{baz}qux
```

will be interpreted as command ‘foo’ with first argument ‘bar’ and second argument ‘baz’ followed by text ‘qux’. Always using groups when passing parameters makes the source code easier to read.

Commands that do not take any parameters still ignore any spaces after them. The easiest way to stop this behaviour, is to follow them by an empty group.

```
New \TeX users may miss
the whitespace
after a command. % renders wrong
Experienced \TeX{} users are
\TeX nicians, and know how to use
whitespace. % renders correct
```

New TEXusers may miss the whitespace after a command. Experienced TEX users are TEXnicians, and know how to use whitespace.

1.3.6 Optional parameters

Many LATEX commands also accept optional parameters. The optional parameters are normally enclosed within square brackets `[ ]`, and usually come right after the command name. The following notation will often be used within this book to denote a command with one optional parameter and one required parameter:

`\command[`⟨*optional parameter*⟩`]{`⟨*parameter*⟩`}`

1.4 Input File Structure

When LATEX processes an input file, it expects it to follow a certain structure. Input files must start with the command

`\documentclass{`⟨*class*⟩`}`

The ⟨*class*⟩ argument specifies what sort of document you intend to write. Available classes are listed in Table 1.1. Usually the article class is sufficient.

Now begins an area of the input file that is called the *preamble*. Inside it, you add commands to influence the style of the whole document, or load packages that add new features to the LATEX system. To load such a package, you use the command

Table 1.1: Document Classes.

Class	Description
article	for articles in scientific journals, short reports, and any other short document.
proc	a class for proceedings based on the article class.
report	for longer reports. containing several chapters, small books, PhD theses, ...
book	for real books.
beamer	for presentations.
letter	for letters.

```
\usepackage{⟨package⟩}
```

When the preamble is finished, you start the *body* of the text with the command

```
\begin{document}
```

Inside the body, you enter the text mixed with some useful LaTeX commands. Mark the end of the document with the

```
\end{document}
```

command, which tells LaTeX to call it a day. Anything that follows this command will be ignored by LaTeX.

Listing 1.1 shows the contents of a minimal LaTeX file. A slightly more complicated input file is given in Listing 1.2.

1.5 A Typical Command Line Session

I bet you must be dying to try out the neat small LaTeX input file shown on page 11. Here is some help: LaTeX itself comes without a GUI or fancy buttons to press. It is just a program that crunches away at your input file. Some LaTeX installations feature a graphical front-end where there is a LaTeX button to start compiling your input file. On other systems, there might be some typing involved, so here is how to coax

```
\documentclass{article}

\begin{document}
Small is beautiful.
\end{document}
```

Small is beautiful.

1

Listing 1.1: A Minimal LaTeX File.

LaTeX into compiling your input file on a text based system. Please note: this description assumes that a working LaTeX installation already sits on your computer. If this is not the case, you may want to look at Appendix A on page 273 first.

1. Edit/Create your LaTeX input file. This file must be plain text. On Unix-like systems, most of the editors will create just that. On Windows, you might want to make sure that you save the file in *Plain Text* format. When picking a name for your file, make sure it bears the extension `.tex`.

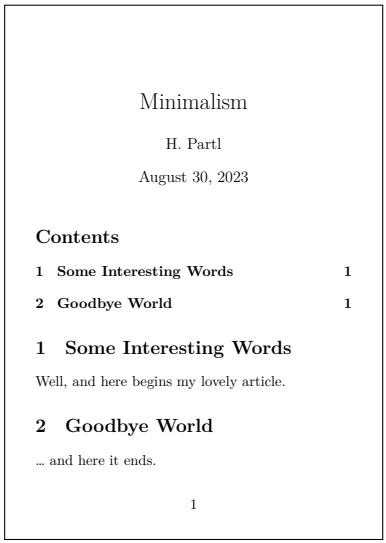
2. Open a shell or cmd window, `cd` to the directory where your input file is located and run LaTeX on your input file using either

   ```
   xelatex foo.tex
   ```

   or

   ```
   lualatex foo.tex
   ```

```latex
\documentclass[a4paper,11pt]{article}

\author{H.~Partl}
\title{Minimalism}

\begin{document}
\maketitle
\tableofcontents

\section{Some Interesting Words}
Well, and here begins my lovely
article.

\section{Goodbye World}
\ldots{} and here it ends.

\end{document}
```



Minimalism

H. Partl

August 30, 2023

Contents

1 Some Interesting Words **1**

2 Goodbye World **1**

1 Some Interesting Words

Well, and here begins my lovely article.

2 Goodbye World

… and here it ends.

1

Listing 1.2: Example of a realistic journal article. Note that all the commands you see in this example will be explained later.

If successful, you will end up with a `.pdf` file. It may be necessary to run LaTeX several times to get the table of contents and all internal references right. When your input file has a bug, LaTeX will tell you about it and stop processing your input file. Type `ctrl-D` to get back to the command line.

1.6 Logical Structure of Your Document

In subsection 1.2.2, we mentioned that one of the differences between LaTeX and WYSIWYG editors is that in LaTeX you write the document by specifying its logical structure. This section will explore this idea in more detail by presenting a problem and demonstrating how it can be solved with logical markup. If you are familiar with the idea (for example, you have worked with HTML and CSS), you can safely skip this section.

1.6.1 A Neverending Story of Problems with WYSIWYG Editors

Let us imagine that you are writing a novel in your favourite WYSIWYG editor. In this book, there are two parallel storylines happening in different dimensions. In order to communicate to the reader which of the dimensions is currently described, you have decided to use different colours for them. Thus, your book may look like this:

> "No, thou canst leave me!" shouted Peredur to the Launcelot.
>
> "I have to" he replied. "Destiny calls upon me. But we shall meet again. I promise."
>
> Shao felt inexplicable sadness, as if they had just lost something or someone important to them.
>
> "Are you okay?" asked Ashby while eating her slurry. "You don't look well."
>
> "I'm fine, just tired."

After you finish your first draft, you decided to email it over to your friend so they can share their opinion about it. However, it turns out

that your friend's printer can only print black and white, so they cannot print the file you sent them.

After some consideration, you decided to simply use cursive font for one dimension, while keeping upright font for the other. After some time of manually changing each paragraph of your book to match the correct font, you remembered that you have already used cursive font for emphasis in some cases, such as "Are you gonna eat *that*?". Continuing with your current approach you, would also have to check each paragraph for emphasis and change it to something else before changing the font to cursive.

The above problems with changing the style of paragraphs are caused by the fact that your WYSIWYG editor doesn't know that all green paragraphs somehow represent common concept (events in one dimension). It just remembers that you want these words in green, those in red and that one in cursive. Thus, changing it to a different style means going over everything again, and manually changing the style of each and every paragraph and word.

Wouldn't it be nice if you could somehow communicate to your editor "these paragraphs are happening in dimension A", and then simply decide that all such paragraphs are green or use a different font? This is exactly what logical markup does.

1.6.2 Your First Text Command

To see how this example would play out differently in LaTeX, let's introduce our first text command:

`\emph{`⟨*text*⟩`}`

It stands for "emphasise" and does just that: it emphasises the ⟨*text*⟩ it received as a parameter.

```
Are you gonna eat \emph{that}?
```

Are you gonna eat *that*?

If you write the code on the left in the body of your document (you can use Listing 1.1 as a template), you will get the output on the right in the PDF file you produce.

As you can see, LaTeX typesets the emphasised text in cursive font by default. It is important to understand that the `\emph` command *does not* mean "write this text in cursive". It is much smarter than that. To illustrate this, let's see how `\emph` behaves when used inside text that is already in cursive

```
Are you gonna eat \emph{that}?
```

Are you gonna eat that*?*

As you can see, in this case it changed the font back to upright.

Remember that you should only use the `\emph` command to emphasise text and nothing else, even if the resulting output would be the same (for example cursive font). If you stick to this rule, then if you later decide to use a different form of emphasis, you can simply change the definition of the `\emph` command and other things that just *happened* to be in cursive will be unaffected.

```
Are you gonna eat \emph{that}?
```

Are you gonna eat **that**?

1.6.3 Your First Environment

If you have played with the `\emph` command a bit, you may have noticed that trying to write several paragraphs inside it results in an error. This is the case for most LaTeX commands. The reason is that putting a lot of text inside their parameters could result in poor performance and excessive memory consumption. In order to overcome that, LaTeX uses a concept of *environments*.

Environments are started using the `\begin` command and ended using the `\end` command. You have already seen one environment — the `document` environment that holds the body of the document. This one is present exactly once in every LaTeX document. To explore the concept a bit, let's introduce another one that is not very useful in practice, but

is easy to understand: the `em` environment, short for "emphasise".[6]

```
\begin{em}
  This paragraph is emphasised.

  This one is \emph{too}.
\end{em}
```

This paragraph is emphasised.
This one is too.

1.6.4 Summary

Using logical markup, we can embed the logical structure of our document inside the document itself. Instead of saying "write this in green and write this word in cursive", we say "this text happens in dimension A and this word is emphasised". The style of all text in 'dimension A', or of emphasised words, can be decided later and easily changed.

If you started writing your hypothetical novel using LaTeX instead of a WYSIWYG editor, and used custom environments for typesetting events in different dimensions, and `\emph` for emphasis, then changing it to a black and white version would come down to simply changing the definition of the custom environments.

Before learning how to define your own commands and environments, this book will introduce you to many of the standard ones that are provided either by LaTeX itself or third-party packages.

1.7 Packages

While writing your document, you will probably find that there are some areas where basic LaTeX cannot solve your problem. If you want to include graphics, coloured text or source code from a file in your document, then you need to enhance the capabilities of LaTeX. Such enhancements are called packages. Packages are activated with the

`\usepackage[`⟨*options*⟩`]{`⟨*package*⟩`}`

command, where ⟨*package*⟩ is the name of the package and ⟨*options*⟩ is a list of keywords that trigger special features in the package. The

[6] It is not useful because there is very little reason to emphasise more than one paragraph. In order to make the emphasis effective, it should be used with restraint. After all, if everything is emphasised then nothing is.

`\usepackage` command goes into the preamble of the document. See Section 1.4 for details.

This book will describe some packages that the authors thought were especially useful and should be installed along with your LaTeX distribution. See Table 1.2 for some examples. The versions installed on your system may be different than the ones described in this book, which in turn may lead to differences in the produced output. Along with each package's description we will also point to its entry in our bibliography. In the bibliography entry, you will find information about the package version that was used when writing this booklet. You can check the versions of all packages used in a document by looking at the `.log` file that is produced when compiling it. If the package versions aren't very different (usually the first number is the most important) then you should be fine.

Modern LaTeX distributions come with many packages preinstalled. If you are working on a Unix-like system, try using the command `texdoc` for accessing package documentation. Alternatively you can search for the package on https://www.ctan.org/, and its documentation should be present under the 'Documentation' field.

1.8 The Structure of Text and Language

By Hanspeter Schmid <hanspi@schmid-werren.ch>

The main point of writing a text, is to convey ideas, information, or knowledge to the reader. The reader will understand the text better if these ideas are well-structured, and will see and feel this structure much better if the typographical form reflects the logical and semantic structure of the content.

As we have seen, LaTeX is different from other typesetting systems in that you just have to tell it the logical and semantic structure of a text. It then derives the typographical form of the text according to the "rules" given in the document class file and in various style files.

The most important text unit in LaTeX (and in typography) is the paragraph. We call it "text unit" because a paragraph is the typographical form that should reflect one coherent thought, or one idea. Therefore, if a new thought begins, a new paragraph should begin, and if not, only line

Table 1.2: Examples of LaTeX packages.

Package	Description
amsmath [55]	Provides additional commands for typesetting mathematical symbols, and environments for aligning equations. Described in Chapter 3.
polyglossia [15]	Makes it easy to write LaTeX documents in languages other than English, or even in multiple languages. Described in Section 2.8.
booktabs [17]	Provides commands for producing beautifully formatted tables for your document. Described in Section 2.18.
biblatex [44]	Provides commands for automatically specifying and producing a bibliography for your document. Described in Chapter 4.
makeidx [76]	Provides commands for producing indexes. Described in Section 5.1.
fancyhdr [50]	Let's you easily customise page headers and footers. Described in Section 7.7.
beamer [74]	Provides a document class that changes output to produce presentations and provides command to typeset slides. Described in Section 5.4.

breaks should be used. If in doubt about paragraph breaks, think about your text as a conveyor of ideas and thoughts. If you have a paragraph break, but the old thought continues, it should be removed. If some totally new line of thought occurs in the same paragraph, then it should be broken.

Most people completely underestimate the importance of well-placed paragraph breaks. Many people do not even know what the meaning of a paragraph break is, or, especially in LaTeX, introduce paragraph breaks without knowing it. The latter mistake is especially easy to make if equations are used in the text. You have already learned in subsection 1.3.1 that paragraph breaks are introduced by leaving an empty line in the source code. Look at the following examples, and figure out why sometimes empty lines (paragraph breaks) are used before and after the equation, and sometimes not. (Ignore the contents of the equations themselves as they are not important.)

```
% Example 1
\ldots when Einstein introduced
his formula
\begin{equation}
  e = m \cdot c^2 \; ,
\end{equation}
which is at the same time the
most widely known and the least
well understood physical formula.
```

...when Einstein introduced his formula

$$e = m \cdot c^2 \; , \tag{1}$$

which is at the same time the most widely known and the least well understood physical formula.

```
% Example 2
\ldots from which follows
Kirchhoff's current law:
\begin{equation}
  \sum_{k=1}^{n} I_k = 0 \; .
\end{equation}

Kirchhoff's voltage law can
be derived \ldots
```

...from which follows Kirchhoff's current law:

$$\sum_{k=1}^{n} I_k = 0 \; . \tag{1}$$

Kirchhoff's voltage law can be derived ...

```
% Example 3
\ldots which has several
advantages.

\begin{equation}
  I_D = I_F - I_R
\end{equation}
is the core of a very different
transistor model. \ldots
```

…which has several advantages.

$$I_D = I_F - I_R \tag{1}$$

is the core of a very different transistor model. …

The next smaller text unit is the sentence. In English texts, there is a larger space after a period that ends a sentence than after one that ends an abbreviation. LaTeX tries to figure out which one you wanted to have. If LaTeX gets it wrong, you must tell it what you want. This is explained later in the next chapter.

The structuring of text even extends to parts of sentences. Most languages have very complicated punctuation rules, but in many languages (including German and English), you will get almost every comma right if you remember what it represents: a short stop in the flow of language. If you are not sure about where to put a comma, read the sentence aloud and take a short breath at every comma. If this feels awkward at some place, delete that comma; if you feel the urge to breathe (or make a short stop) at some other place, insert a comma.

Finally, the paragraphs of a text should also be structured logically at a higher level, by putting them into chapters, sections, subsections, and so on.

1.9 Files You Might Encounter

When you work with LaTeX, you will soon find yourself in a maze of files with various extensions and probably no clue. The following list explains the various file types you might encounter when working with TeX. Please note that this table does not claim to be a complete list of extensions, but if you find one missing that you think is important, please drop me a line.

`.tex` LaTeX or TeX input file. Can be compiled with `latex`.

`.sty` LaTeX Macro package. Load this into your LaTeX document using the `\usepackage` command.

`.dtx` Documented TeX. This is the main distribution format for LaTeX style files. If you process a `.dtx` file you get documented macro code of the LaTeX package contained in the `.dtx` file.

`.ins` The installer for the files contained in the matching `.dtx` file. If you download a LaTeX package from the net, you will normally get a `.dtx` and a `.ins` file. Run LaTeX on the `.ins` file to unpack the `.dtx` file.

`.cls` Class files define what your document looks like. They are selected with the `\documentclass` command.

`.fd` Font description file, telling LaTeX about new fonts.

The following files are generated when you run LaTeX on your input file:

`.log` Gives a detailed account of what happened during the last compiler run.

`.toc` Stores all your section headers. It gets read in for the next compiler run and is used to produce the table of contents.

`.lof` This is like `.toc`, but for the list of figures.

`.lot` Same again, for the list of tables.

`.aux` Another file that transports information from one compiler run to the next. Among other things, the `.aux` file is used to store information associated with cross-references.

`.idx` If your document contains an index, LaTeX stores all the words that go into the index in this file. Process this file with `makeindex`. Refer to Section 5.1 on page 167 for more information on indexing.

`.ind` The processed `.idx` file, ready for inclusion into your document on the next compile cycle.

`.ilg` Log file, telling what `makeindex` did.

Chapter 2

Real World LaTeX

After reading the previous chapter, you should have some general idea about LaTeX. This chapter will fill in the remaining structure you will need to know in order to produce real world documents.

2.1 Line Breaking and Page Breaking

2.1.1 Justified Paragraphs

Books are often typeset with each line having the same length. LaTeX inserts the necessary line breaks and spaces between words by optimising the contents of a whole paragraph. If necessary, it also hyphenates words that would not fit comfortably on a line. How the paragraphs are typeset depends on the document class. Normally the first line of a paragraph is indented, and there is no additional space between two paragraphs. Refer to subsection 7.6.4 for more information.

In special cases it might be necessary to order LaTeX to break a line.

`\\[`⟨*length*⟩`]` or `\newline`

starts a new line without starting a new paragraph. The optional ⟨*length*⟩

argument adds additional space after the line.

`\\*[⟨length⟩]`

additionally prohibits a page break after the forced line break.

`\newpage`

starts a new page.

```
\linebreak[⟨n⟩]
\nolinebreak[⟨n⟩]
\pagebreak[⟨n⟩]
\nopagebreak[⟨n⟩]
```

suggest places where a break may (or may not) happen. They enable the author to influence their actions with the optional argument n, which can be set to a number between zero and four. By setting n to a value below 4, you leave LaTeX the option of ignoring your command if the result would look very bad. Do not confuse these "break" commands with the "new" commands. Even when you give a "break" command, LaTeX still tries to even out the right border of the line and the total length of the page. As described in the next paragraph, this can lead to unpleasant gaps in your text. If you really want to start a "new line" or a "new page", then use the corresponding command. Guess their names!

```
Start a new line here,\\
and also here but make
it bigger.\\[1cm]
Do a linebreak here\linebreak
and maybe here, \linebreak[1]
but no pressure.
Linebreak here\linebreak[3]
would be \emph{really} cool.
```

Start a new line here,
and also here but make it bigger.

Do a linebreak here and maybe here, but no pressure. Linebreak here would be *really* cool.

LaTeX always tries to produce the best line breaks possible. If it cannot find a way to break the lines in a manner that meets its high

standards, it lets one line stick out on the right of the paragraph. LaTeX then complains ("overfull hbox") while processing the input file. This happens most often when LaTeX cannot find a suitable place to hyphenate a word.[1] Instruct LaTeX to lower its standards a little by giving the `\sloppy` command. It prevents such over-long lines by increasing the inter-word spacing—even if the final output is not optimal. In this case a warning ("underfull hbox") is given to the user. In most such cases the result doesn't look very good. The command `\fussy` brings LaTeX back to its default behaviour.

2.1.2 Hyphenation

LaTeX hyphenates words whenever necessary. If the hyphenation algorithm does not find the correct hyphenation points, remedy the situation by using the following commands to tell TeX about the exception.

The command

`\hyphenation{`⟨*word list*⟩`}`

causes the words listed in the argument to be hyphenated only at the points marked by '`-`'. The argument of the command should only contain words built from normal letters, or rather glyphs that are considered to be normal letters by LaTeX. The hyphenation hints are stored for the language that is active when the hyphenation command occurs. This means that if you place a hyphenation command into the preamble of your document, it will influence the English language hyphenation. If you place the command after the `\begin{document}` and you are using a package for national language support, like polyglossia, then the hyphenation hints will be active in the language activated through polyglossia.

The example below allows the word "locomotion" to be hyphenated, as well as "Locomotion", and it prevents "FORTRAN", "Fortran" and "fortran" from being hyphenated at all. No special characters or symbols are allowed in the argument.

[1] Although LaTeX gives you a warning when that happens (`Overfull \hbox`) and displays the offending line, such lines are not always easy to find. If you use the option `draft` in the `\documentclass` command, these lines will be marked with a thick black line on the right margin.

```
\hyphenation{FORTRAN Lo-co-mo-tion}
```

The command `\-` inserts a discretionary hyphen into a word. This also becomes the only point hyphenation is allowed in this word. This command is especially useful for words containing special characters, because LaTeX does not automatically hyphenate them.

```
I think this is: su\-per\-%
cal\-i\-frag\-i\-lis\-tic\-%
ex\-pi\-al\-i\-do\-cious
```

I think this is: supercalifragilistic-expialidocious

Several words can be kept together on one line with the command

`\mbox{`⟨*text*⟩`}`

It causes its argument to be kept together under all circumstances.

```
My phone number will
change soon. It will
be \mbox{0116 291 2319}.

The parameter
\mbox{[filename]} should
contain the name of the file.
```

My phone number will change soon. It will be 0116 291 2319. The parameter [filename] should contain the name of the file.

2.2 Ready-Made Strings

In some of the examples on the previous pages, you have seen some very simple LaTeX commands for typesetting special text strings:

Command	Example	Description
`\today`	August 30, 2023	Current date
`\TeX`	TeX	Your favourite typesetter
`\LaTeX`	LaTeX	The Name of the Game
`\LaTeXe`	LaTeX 2_ε	The current incarnation

2.3 Dashes and Hyphens

LaTeX knows four kinds of dashes. You access three of them with different numbers of consecutive dashes. The fourth sign is actually not a dash at all—it is the mathematical minus sign. It is typed automatically when inside math mode, as described later in Chapter 3.

```
daughter-in-law, X-rated\\
pages 13--67\\
yes---or no?
```

daughter-in-law, X-rated
pages 13–67
yes—or no?

The names for these dashes are: '-' hyphen, '–' en-dash and '—' em-dash. Hyphens are used when writing compound words (and inserted automatically by LaTeX when splitting a single word), en-dashes are used for writing a range of numbers, and em-dashes are used to mark an interruption in speech or an abrupt change of thought.

2.4 Slash (/)

In order to typeset a slash between two words, one can simply type, for example, `read/write`, but this makes LaTeX treat the two words as one. Hyphenation is disabled by the slash character (see subsection 2.1.2), so there may be 'overfull' errors. To overcome this, use `\slash`. For example, type

```
read\slash write
```

which allows hyphenation. Normal '/' characters may be still used for inline fractions. The typesetting of units, such as MiB/s, will be described in Section 3.5.

2.5 Ellipsis (...)

On a typewriter, a comma or a period takes the same amount of space as any other letter. In book printing, these characters occupy only a little space and are set very close to the preceding letter. Therefore,

entering 'ellipsis' by just typing three dots would produce the wrong result. Instead, there is a special command for these dots. It is called

`\ldots` (low dots)

```
Not like this ...
but like this:\\
New York, Tokyo,
Budapest, \ldots
```

Not like this ... but like this:
New York, Tokyo, Budapest, …

2.6 Ligatures

Some letter combinations are typeset not just by setting the different letters one after the other, but by actually using special symbols.

ff fi fl ffi … instead of ff fi fl ffi …

These so-called ligatures can be prohibited by inserting an `\mbox{}` between the two letters in question. This might be necessary with words built from two words.

```
Not shelfful\\
but shelf\mbox{}ful
```

Not shelfful
but shelfful

2.7 Abstract

In scientific publications, it is customary to start with an abstract which gives the reader a quick overview of what to expect. LaTeX provides the `abstract` environment for this purpose. Normally `abstract` is used in documents typeset with the article document class.

```
\begin{abstract}
  This paper will talk about
  abstracts.
\end{abstract}

Abstracts are very important \ldots
```

Abstract

This paper will talk about abstracts.

Abstracts are very important …

2.8 International Language Support

By Axel Kielhorn <A.Kielhorn@web.de>

When you write documents in languages other than English, there are three areas where LaTeX has to be configured appropriately:

1. All automatically generated text strings ("Table of Contents", "List of Figures", ...) have to be adapted to the new language.
2. LaTeX needs to know the hyphenation rules for the current language.
3. Language-specific typographic rules. For example, in French there is a mandatory space before each colon character (:).

The package polyglossia [15] is a replacement for the venerable babel package. It takes care of the hyphenation patterns and automatically-generated text strings in your documents. Polyglossia works only with the XeTeX and LuaTeX engines, so if you are using pdfTeX you'll have to stick with babel.

2.8.1 Entering Characters

Early versions of TeX were only intended to support the English language, and thus assumed that source files were ASCII encoded; that is, in a format that only supports the Latin alphabet plus few symbols common on computers. With the popularization of LaTeX outside the English-speaking world, the situation has slowly improved. These days TeX engines speak UTF-8 natively, which considerably eases the usage of LaTeX when typesetting non-English documents. Thus, inputting non-English characters is as easy as:

```
Hôtel, naïve, élève, \\
smørrebrød, ¡Señorita!, \\
Schönbrunner Schloß, Żółć
```

Hôtel, naïve, élève,
smørrebrød, ¡Señorita!,
Schönbrunner Schloß, Żółć

If your keyboard does not have all the special characters, you can also enter accents and special characters via dedicated commands. The commands for adding accents to characters are listed in Table 2.1. Keep

in mind that while older TeX engines worked by combining accents with existing letters manually, nowadays they simply look up the correct glyph in the font. This means that accents and characters can no longer be freely combined, and output is produced only if the font supports the combination. The example above would look like this if entered on an english keyboard:

```
H\^otel, na\"{\i}ve, \'el\`eve, \\
sm{\o}rrebr{\o}d, !`Se\~norita!, \\
Sch\"onbrunner Schlo\ss, \.Z\'o\l\'c
```

Hôtel, naïve, élève,
smørrebrød, ¡Señorita!,
Schönbrunner Schloß, Żółć

The Unicode standard defines over 100 000 glyphs for use across many languages and specialized disciplines. Currently, no font implements the whole set, so if you are working with a lot of non-Latin characters you will, sooner or later, find one that is not supported by your currently selected font. In such cases, LaTeX will not typeset the requested character, and prints a "Missing character" warning in its log. Because such warnings are easy to overlook, we recommend you put

```
\tracinglostchars=3
```

somewhere in the preamble of the document. This command will elevate the warnings to error status, so they will be much harder to miss.

2.8.2 Polyglossia Usage

Writing in different languages is easy. Just load the polyglossia package and specify the languages in the preamble using

```
\setdefaultlanguage[⟨options⟩]{⟨language⟩}
\setotherlanguage[⟨options⟩]{⟨language⟩}
```

where ⟨*language*⟩ is either the name of the language to use, such as `gaelic` or `japanese`, or its BCP-47 tag such as `de-CH` or `ru-luna1918`. When specifying language via its name you may pass additional ⟨*options*⟩ to decide between its variants, for example `variant=british` or `script=Arabic`. Note that when loading a language, either via tag or with options, all variants are loaded and the tag/options specified are

Table 2.1: Accents and Special Characters.

Code	Result	Code	Result	Code	Result	Code	Result
`\`o`	ò	`\'o`	ó	`\^o`	ô	`\~o`	õ
`\=o`	ō	`\.o`	ȯ	`\"o`	ö	`\c c`	ç
`\u o`	ŏ	`\v o`	ǒ	`\H o`	ő	`\k a`	ą
`\d o`	ọ	`\b o`	ṏ	`\t oo`	o͡o	`\r o`	ů
`\oe`	œ	`\OE`	Œ	`\ae`	æ	`\AE`	Æ
`\aa`	å	`\AA`	Å				
`\o`	ø	`\O`	Ø	`\l`	ł	`\L`	Ł
`\i`	ı	`\j`	ȷ	`!``	¡	`?``	¿

just the defaults. For a full list of supported languages and variants, see the polyglossia package documentation.

To write a paragraph in German, you can use the `german` environment:

```
% In the preamble
\setdefaultlanguage{english}
\setotherlanguage{german}
% ...
Today is not \today.

\begin{german}
  Heute ist nicht \today.
\end{german}
```

Today is not January 19, 3022.
Heute ist nicht 19. Januar 3022.

You can also use `lang` environment, which accepts the ⟨*language*⟩ as its first argument. This is especially useful if you prefer to specify the language via a BCP-47 tag, since environment names cannot contain the '`-`' symbol.

```
\begin{lang}{de-AT}
  Heute ist nicht \today.
\end{lang}
```

Heute ist nicht 19. Jänner 3022.

You can also pass ⟨*options*⟩ to the environment.

```
\begin{german}[
  script=blackletter
]
  Heute ist nicht \today.
\end{german}
```

Heute ist nicht 17. Auguſt 3023.

If you just need a word or short phrase in a foreign language, you can use either the

\text*language*[⟨*options*⟩]{⟨*text*⟩}

or the

\textlang[⟨*options*⟩]{⟨*language*⟩}{⟨*text*⟩}

command:

```
In Austria they write
\textlang[variant=austrian]{de}{Jänner}
instead of \textgerman{Januar}. And in the olden
days \textlang{de}{August} was written
\textgerman[script=blackletter]{Auguſt}.
```

In Austria they write Jänner instead of Januar. And in the olden days August was written Auguſt.

For languages that are not very different from English, the results are not very impressive. We get correct hyphenation, and some context-aware commands, such as \today, adjust their output. However, the further we stray from English, the more useful these commands become.

Sometimes the font used in the main document does not contain glyphs required by the second language. The default Latin Modern font, for example, does not contain Cyrillic letters. The solution is to define a custom font that will be used for that language.

To set the fonts, use the preamble command

```
\newfontfamily{\⟨familyname⟩}[⟨options⟩]{⟨font⟩}
```

from the fontspec package,[2] which is described in much more detail in Section 7.3. When changing the text language, polyglossia checks whether a font family named *language*font exists, and switches to that if it's available.

If you are happy with the default Latin Modern font, you may want to try the "CMU" font, which contains some additional Cyrillic or Greek glyphs while looking nearly identical to Latin Modern.[3] If you have the CMU font installed in your system (unlikely), or if you are using LuaLaTeX, then you can simply write

```
\newfontfamily\greekfont{CMU Serif}
```

If you are using XeLaTeX and don't have the font installed, then you need to specify the font by its file name.

```
\newfontfamily\greekfont[
  Extension=.otf, UprightFont=*rm, ItalicFont=*ti,
  BoldFont=*bx, BoldItalicFont=*bi,
]{Cm}
```

With the appropriate fonts loaded, you can now write:

[2] fontspec is loaded automatically by polyglossia, so you don't need to add it to the preamble of your document.

[3] Both fonts are actually based on the Computer Modern font, designed by Donald Knuth for the first TeX versions.

```
% In the preamble (LuaLaTeX)
\setotherlanguage{greek}
\newfontfamily\greekfont{CMU Serif}
\setotherlanguage{russian}
\newfontfamily\russianfont{CMU Serif}
% ...

\textrussian{Правда} is a Russian newspaper.
\textlang{greek}{ἀλήθεια} is truth or disclosure in
philosophy.
```

> Правда is a Russian newspaper. ἀλήθεια is truth or disclosure in philosophy.

LaTeX actually uses three different fonts for typesetting documents. The above commands only set the default serif font used for the body text. The other fonts are sans serif (used in presentation slides) and monospace (used when displaying code). In order to adjust the fonts used for a particular language, define the *language*fontsf family for sans serif, and *language*fonttt for monospace. For example, to define CMU as the font for Greek in all three fonts when using LuaLaTeX, you would write

```
\newfontfamily\greekfont{CMU Serif}
\newfontfamily\greekfontsf{CMU Sans Serif}
\newfontfamily\greekfonttt{CMU Typewriter Text}
```

or in XeLaTeX

```
\newfontfamily\greekfont[
  Extension=.otf, UprightFont=*rm, ItalicFont=*ti,
  BoldFont=*bx, BoldItalicFont=*bi,
]{cmun}
\newfontfamily\greekfontsf[
  Extension=.otf, UprightFont=*ss, ItalicFont=*si,
  BoldFont=*sx, BoldItalicFont=*so,
]{cmun}
\newfontfamily\greekfonttt[
  Extension=.otf, UprightFont=*btl, ItalicFont=*bto,
```

```
    BoldFont=*tb, BoldItalicFont=*tx,
  ]{cmun}
```

The commands above redefine the fonts for languages used as 'other' in polyglossia. If you want to influence the font of the main document language, use \setmainfont, \setsansfont and \setmonofont. These work the same way, but without the ⟨*familyname*⟩ argument.

```
\setmainfont{CMU Serif}
\setsansfont{CMU Sans Serif}
\setmonofont{CMU Typewriter Text}
```

If you want to learn even more about fonts, have a look at Section 7.3.

2.8.3 Right to Left (RTL) languages

Some languages are written left to right, others are written right to left (RTL). In order to support RTL languages, polyglossia needs the bidi [33] package.[4] The bidi package should be the last package you load; even after hyperref, which is otherwise usually the last package. (Since polyglossia loads bidi, this means that polyglossia should be the last package loaded.)

To typeset Arabic, you can use the Iran Nastaliq font [72] developed by the SCICT and updated by Mohammad Saleh Souzanchi.

```
\documentclass{article}

\usepackage{polyglossia}
\setdefaultlanguage{arabic}
\newfontfamily\arabicfont{Iran Nastaliq}

\begin{document}
هذان صح رب ي
\end{document}
```

هذان صح رب ي

١

The package xepersian [34][5] offers support for the Persian language. It supplies Persian LaTeX-commands that allows you to enter commands

[4] bidi supports only the XeTeX engine. If you use LuaTeX, polyglossia uses the luabidi [35] package, but keep in mind that it is much more limited than bidi.

[5] Works only with XeLaTeX.

like `\section` in Persian, which makes this really attractive to native speakers. xepersian is the only package that supports kashida with XeLaTeX. A package for Syriac, which uses a similar algorithm, is under development.

The arabxetex [13] package supports several languages with an Arabic script:

- arab (Arabic)
- persian
- urdu
- sindhi
- pashto
- ottoman (turk)
- kurdish
- kashmiri
- malay (jawi)
- uighur

It offers a font mapping that enables XeLaTeX to process input using the ArabTeX ASCII transcription.

There is no package available for Hebrew, because none is needed. The Hebrew support in polyglossia should be sufficient, but you do need a suitable font with real Unicode Hebrew. An extension to the default Latin Modern font, called *New Computer Modern* [82], adds, among other things, a full Hebrew character set. It's distributed with TeXLive, so there is a good chance you already have it. Another Hebrew font, available under the SIL Open Font License, is Ezra SIL [78].

2.8.4 Chinese, Japanese and Korean (CJK)

The package xeCJK [71][6] takes care of font selection and punctuation for these languages.

[6] Works only with XeLaTeX.

2.9 Simple Commands

You may find yourself retyping the same name/logo/complicated text in multiple places of your documents. This is tedious and bears the risk of introducing typos. Brand logos also often have special casing rules, such as all uppercase or starting with a lowercase letter even at the beginning of the sentence.

We have already seen a solution for this problem—commands. For example, to refer to LaTeX, which uses a rather complicated logo within text, you don't need to insert pictures or manually space letters—just use the `\LaTeX` command. In LaTeX, you can define your own commands to replace complicated sequences producing given text.

For example, let's imagine that you are writing an article about Polish notation. In doing so, you may have to refer to the the inventor of the notation, Jan Łukasiewicz. His last name may not be easy to remember or type for someone who does not speak Polish. It also starts with letter 'Ł', which may not be easily accessible on your keyboard, so you will have to use `\L`, making the code less readable.

LaTeX enables you to define your own commands using the

`\NewDocumentCommand{\`⟨*name*⟩`}{}{`⟨*definition*⟩`}`

command. The ⟨*name*⟩ is the name of the command that you will use in the code, while the ⟨*definition*⟩ is the output it will produce. The second argument is left empty for now, but will be explored in subsection 7.1.1.

```
\NewDocumentCommand{\lukas}{}{%
  Łukasiewicz%
}

Polish notation (also known as
\lukas{} notation, due to its
inventor, Jan \lukas) is a
mathematical notation \ldots{}
```

> Polish notation (also known as Łukasiewicz notation, due to its inventor, Jan Łukasiewicz) is a mathematical notation …

In the example above, we have used `%` comments to avoid introducing spaces. If you need a refresher on this technique, refer to subsection 1.3.2.

While the 'Ł' character is available in LaTeX from the `\L` command, this is not the case for all unicode characters. For example, the default

font contains a glyph for the commercial minus sign (⅍), used in some German speaking countries, but LaTeX does not define a command to enter it. If you want to avoid continuously copy-pasting it into your code, you can set up a command to typeset it for you.

```
\NewDocumentCommand{%
  \comminus}{}{⅍}

Now I can type ⅍ or \comminus.
```

Now I can type ⅍ or ⅍.

Yet another use is to define a command for a simple string you may want to change later. For example, let's imagine you are writing a novel, but are not so sure about your protagonist's name. You can define a command for it, and in the event you change your mind, you need only modify it in one place instead of hunting down every single instance throughout the book.

```
\NewDocumentCommand{%
  \hero}{}{Launcelot}

\hero{} unsheathed his sword.
He  had a bad feeling about
this place from the start.
```

Launcelot unsheathed his sword. He had a bad feeling about this place from the start.

The `\NewDocumentCommand` checks if a command with the given name is already defined, and raises an error if it is. This means you can freely use it to define your own commands without worrying about breaking internal commands used by LaTeX or its packages. But sometimes, redefining existing commands is actually what you want. For example, many packages allow customisation in this way. In this, case you should use `\RenewDocumentCommand`. It works the same way as the former command, but raises an error if a command with the given name is *not* already defined.

```
\RenewDocumentCommand{%
  \ldots}{}{...}

I like my ellipsis looking
uglier \ldots
```

I like my ellipsis looking uglier ...

These commands can be used with nearly all features of LaTeX. Although we have only dealt with text here, the same commands may be used to insert pictures or mathematical expressions. They are especially useful in the latter case, since mathematical symbols are often repurposed to mean different things, and consistency of their usage is very important.

Note that these commands are actually much more powerful than described here. Their full capabilities are discussed in subsection 7.1.1.

2.10 The Space Between Words

To produce a straight right-margin in its output, LaTeX inserts varying amounts of space between words. It inserts slightly more space at the end of a sentence, as this makes the text more readable. LaTeX assumes that sentences end with periods, question marks or exclamation marks. If a period follows an uppercase letter, this is not taken as the end of a sentence, since periods after uppercase letters normally occur in abbreviations.

Any exception from these assumptions has to be specified by the author. A backslash in front of a space generates a space that will not be enlarged. A tilde '~' character generates a non-breaking space (prohibits a line break). The command `\@` in front of a period specifies that this period terminates a sentence even when it follows an uppercase letter.

```
Mr.~Smith was happy to see her\\
cf.~Fig.~5\\
I like BASIC\@. What about you?
```

Mr. Smith was happy to see her
cf. Fig. 5
I like BASIC. What about you?

The additional space after periods can be disabled with the command

```
\frenchspacing
```

which tells LaTeX *not* to insert more space after a period than after an ordinary character. This is very common in non-English languages (polyglossia sets it automatically, based on the main language).

2.11 Titles, Chapters, and Sections

To help the reader find his or her way through your work, you should divide it into chapters, sections, and subsections. LaTeX supports this with special commands that take the section title as their argument. It is up to you to use them in the correct order.

The following sectioning commands are available for the `article` class:

```
\section{⟨title⟩}
\subsection{⟨title⟩}
\subsubsection{⟨title⟩}
\paragraph{⟨title⟩}
\subparagraph{⟨title⟩}
```

If you want to split your document into parts without influencing the section or chapter numbering use

```
\part{⟨title⟩}
```

When you work with the `report` or `book` class, an additional top-level sectioning command becomes available:

```
\chapter{⟨title⟩}
```

As the `article` class does not know about chapters, it is quite easy to add articles as chapters to a book. The spacing between sections, the numbering, and the font size of the titles will be set automatically by LaTeX.

LaTeX creates a table of contents by taking the section headings and page numbers from the last compile cycle of the document. The command

```
\tableofcontents
```

expands to a table of contents at the place it is issued. A new document has to be compiled twice to get a correct table of contents. Sometimes it might be necessary to compile the document a third time. LaTeX will tell you when this is necessary.

All sectioning commands listed above also exist as starred versions. This generates section headings that do not show up in the table of contents and are not numbered. The command `\section{Help}`, for example, would become `\section*{Help}`.

Normally, the section headings show up in the table of contents exactly as they are entered in the text. Sometimes this is not possible, because the heading is too long to fit into the table. The entry for the table of contents can then be specified as an optional argument in front of the actual heading.

```
\tableofcontents

\section[Title for the
  table of contents]{A long
  and especially boring title,
  shown in the text}
```

Contents

1 Title for the table of contents 1

1 A long and especially boring title, shown in the text

The title of the whole document is generated by issuing a

`\maketitle`

command. The contents of the title have to be defined by the commands

`\title{`⟨*title*⟩`}`, `\author{`⟨*author*⟩`}` and optionally `\date{`⟨*date*⟩`}`

before calling `\maketitle`. In the argument to `\author`, you can supply several names separated by `\and` commands.

An example of some of the commands mentioned above can be found in Listing 1.2 on page 12.

Apart from the sectioning commands explained above, LaTeX defines four additional commands for use with the `book` class. They are useful for dividing your publication. The commands alter chapter headings and page numbering to work as you would expect in a book:

`\frontmatter` should be the very first command after the start of the document body (`\begin{document}`). It will switch page numbering to Roman numerals and sections will be non-enumerated as if you were using the starred sectioning commands (for example `\chapter*{Preface}`) but the sections will still show up in the table of contents.

`\mainmatter` comes right before the first chapter of the book. It turns on Arabic page numbering and restarts the page counter.

`\appendix` marks the start of additional material in your book. After this command chapters will be numbered with letters. Unlike others this method can also be used inside articles. In this case, it changes the numbering of sections.

`\backmatter` should be inserted before the very last items in your book, such as the bibliography and the index. In the standard document classes, this has no visual effect.

2.12 Cross References

In books, reports and articles, there are often cross-references to figures, tables and special segments of text. LaTeX provides the following commands for cross referencing

`\label{`⟨*marker*⟩`}`, `\ref{`⟨*marker*⟩`}` and `\pageref{`⟨*marker*⟩`}`

where ⟨*marker*⟩ is an identifier chosen by the user. LaTeX replaces `\ref` by the number of the section, subsection, figure, table, or theorem after which the corresponding `\label` command was issued. `\pageref` prints the page number of the page where the `\label` command occurred.[7] As with section titles and page numbers for the table of contents, the

[7] Note that these commands are not aware of what they refer to. `\label` just saves the last automatically generated number.

numbers from the previous compile cycle are used.

```
A reference to this subsection%
\label{my-marker} looks like:
see Section~\ref{my-marker}
on page~\pageref{my-marker}.
```

A reference to this subsection looks like: see Section 2.12 on page 43.

2.13 Footnotes

With the command

`\footnote{`⟨*footnote text*⟩`}`

a footnote is printed at the foot of the current page. Footnotes should always be put after the word or sentence they refer to. Footnotes referring to a sentence or part of it should therefore be put after the comma or period.

```
Footnotes\footnote{This is
  a footnote.} are often used
by people using \LaTeX.
```

Footnotes[1] are often used by people using LaTeX.

[1] This is a footnote.

2.14 Lists

The `itemize` environment is suitable for simple lists, the `enumerate` environment for enumerated lists, and the `description` environment for

descriptions.

```
\begin{enumerate}
\item You can nest the list
environments to your taste:
\begin{itemize}
\item But it might start to
look silly.
\item[-] With a dash.
\end{itemize}
\item Therefore remember:
\begin{description}
\item[Stupid] things will not
become smart because they are
in a list.
\item[Smart] things, though,
can be presented beautifully
in a list.
\end{description}
\end{enumerate}
```

1. You can nest the list environments to your taste:
   - But it might start to look silly.
   - With a dash.
2. Therefore remember:

   Stupid things will not become smart because they are in a list.

   Smart things, though, can be presented beautifully in a list.

2.15 Non-Justified Text

By default, LaTeX justifies text inside a document. While this is desirable for the main text there may be occasions where you want the text to be left- or right-aligned. This is where the ragged2e [67] package is useful. It defines environments

```
FlushLeft
FlushRight
Center
```

which allow you to achieve left-aligned, right-aligned and centred text.

```
\begin{FlushLeft}
  This text is\\ left-aligned.
  \LaTeX{} is not trying to make
  each line the same length.
\end{FlushLeft}
```

This text is
left-aligned. LaTeX is not trying to make each line the same length.

```
\begin{FlushRight}
  This text is right-\\aligned.
  \LaTeX{} is not trying to make
  each line the same length.
\end{FlushRight}
```

This text is right-aligned. LaTeX is not trying to make each line the same length.

```
\begin{Center}
  At the centre\\of the earth
\end{Center}
```

At the centre
of the earth

2.16 Quotations

You should *not* use the '"' character for quotation marks as you would in a WYSIWYG program. In publishing, there are special opening and closing quotation marks. In LaTeX, the csquotes [43] package provides the

`\enquote{⟨text⟩}`

command that automatically encloses the ⟨*text*⟩ within the right quotes.

```
\enquote{Please press
  the \enquote{x} key.}
```

"Please press the 'x' key."

If you want to skip directly to the inner form of quotation, use the starred version of the command:

```
The \enquote*{x} key is here.
```

The 'x' key is here.

2.16.1 Formal Quotes

The above commands are useful when you want to use, for example, scare quotes. If you are actually quoting someone else, it is better to use the `\textquote` command. It accepts an optional argument with the source of the quote.

```
\textquote[A.~Einstein]{Why
  is it nobody understands
  me and everybody likes me?}
```

"Why is it nobody understands me and everybody likes me?" (A. Einstein)

For longer quotations you may want to use the `displayquote` environment.

```
As Einstein once said
\begin{displayquote}[%
  Albert Einstein]
  Whoever undertakes to
  set himself up as a judge
  of Truth and Knowledge is
  shipwrecked by the laughter
  of the gods.
\end{displayquote}
```

As Einstein once said

> Whoever undertakes to set himself up as a judge of Truth and Knowledge is shipwrecked by the laughter of the gods. (Albert Einstein)

In academic writing, there may be strict rules present when quotes should be typeset inline, and when they should be typeset in display style, based on their length. This can be achieved automatically with the `\blockquote` command. It will scan the text inside and typeset the quote accordingly.

```
\blockquote[Me]{A short quote}

\blockquote[Also Me]{A very long
  quote that spans over multiple
  lines and should be typeset in
  display quote style
  according to my publisher.}
```

"A short quote" (Me)

> A very long quote that spans over multiple lines and should be typeset in display quote style according to my publisher. (Also Me)

By default, `\blockquote` will switch to display quote style if the quote spans more than three lines or contains more than one paragraph. The number of lines may be changed using the `threshold` package options. If you prefer to count words instead of lines, you may change the `thresholdtype` from lines to words and set `threshold` accordingly. If you want to ignore paragraphs, and rely only on automatic line/word counting, set the `parthreshold` to false.

The quote attribution shown in the previous examples is especially useful when combined with bibliography commands described in Chapter 4.

2.16.2 Foreign Quotes

The csquotes package has excellent integration with polyglossia. Thus, simply changing the document's main language automatically adapts the quotation marks used.

```
% In preamble
\setmainlanguage{french}
% ...

\textquote[Antoine de
  Saint Exupéry]{Toute
  nation est égoïste.
  Toute nation considère
  son égoïsme comme sacré.}
```

«Toute nation est égoïste. Toute nation considère son égoïsme comme sacré.» (Antoine de Saint Exupéry)

If you load the package with the `autostyle` option, it will also adapt the quotation marks to the language of the surrounding text.

```
% In preamble
\usepackage[autostyle]{csquotes}
% ...

\enquote{English quote}

\begin{german}
  \enquote{Deutsches zitat}
\end{german}
```

"English quote"
„Deutsches zitat"

If frequently quoting in foreign languages, you may get tired of first switching to a given language with a polyglossia command, and then using `\textquote`. In order to avoid this, csquotes defines an additional `\foreignquote` command, which takes the language as its first language.

```
As Dostoevsky once wrote
\foreignquote{russian}{%
  Мир спасёт красота}.
```

As Dostoevsky once wrote «Мир спасёт красота».

The commands \foreigntextquote and \foreignblockquote, and the environment foreigndisplayquote, are also available, and work like those described in the previous section.

Some languages use multiple variants of quotes. To change the variant used by the given language, add option ⟨*language*⟩=⟨*variant*⟩ when loading the csquotes package.

```
% In preamble
\usepackage[
  autostyle,
  german=guillemets,
]{csquotes}
% ...

\foreignquote{german}{%
  Deutsches zitat}
```

»Deutsches zitat«

2.16.3 Long Quotations and Poetry

LaTeX provides two additional environments: quotation and verse. The quotation environment is useful for longer quotes running over several paragraphs, because it indents the first line of each paragraph. The verse environment is useful for poems where the line breaks are important. The lines are separated by issuing a \\ at the end of a line, and an empty line after each verse. Note that these environments are not defined by csquotes, so they do not accept the optional attribution argument, and are not context sensitive.

```
I wanted to quote my
favourite speech:
\begin{quotation}
  This is a very
  long speech.

  It spans over multiple
  paragraphs.
\end{quotation}
```

I wanted to quote my favourite speech:

> This is a very long speech.
>
> It spans over multiple paragraphs.

```
I know only one English
poem by heart. It is
about Humpty Dumpty.
\begin{verse}
  Humpty Dumpty sat
  on a wall:\\
  Humpty Dumpty had
  a great fall.\\
  All the King's horses
  and all the King's men\\
  Couldn't put Humpty
  together again.
\end{verse}
```

I know only one English poem by heart. It is about Humpty Dumpty.

> Humpty Dumpty sat on a wall:
> Humpty Dumpty had a great fall.
> All the King's horses and all the King's men
> Couldn't put Humpty together again.

2.17 Code Listings

When writing about LaTeX, or other programming languages, you often need to insert short code snippets like this `\LaTeX{}`. While you could escape all the characters, this would quickly become rather tiresome, especially for longer pieces of code where correct spacing is crucial for readability (multiple spaces being collapsed by LaTeX to singles). We will present three solutions to this problem.

2.17.1 Verbatim

LaTeX itself comes with the `\verb` command. It is unusual in that it does not use groups for getting its first argument. Instead, you pass the argument between a chosen delimiter that can be any character except a letter, `*` or space. For example, to use `|` as the delimiter (as is typical), you type

`\verb|`⟨*text*⟩`|`

Any text that is enclosed between these delimiters will be directly printed, as if typed on an old typewriter (in a monospaced font), with all spaces,

and without any LaTeX commands being executed.

```
Use \verb|\LaTeX{}| to
print \LaTeX.

We normally use \verb+|+ but
when it's not available we
use \verb|+| to delimit the
argument of the \verb|\verb|
command.
```

Use `\LaTeX{}` to print LaTeX.
We normally use `|` but when it's not available we use `+` to delimit the argument of the `\verb` command.

The starred version of the `\verb` command replaces spaces with the '␣' symbol.

```
\verb*|\TeX user| will
produce \TeX user.

To get proper spacing use
\verb*|\TeX{} user|.

Several spaces are treated
by \LaTeX{} as one space,
so \verb*|a     b| will
produce a     b.
```

`\TeX␣user` will produce TeXuser.
To get proper spacing use `\TeX{}␣user`.
Several spaces are treated by LaTeX as one space, so `a␣␣␣␣␣b` will produce a b.

For longer text, you may want to use the `verbatim` environment from the verbatim [66] package.

```
\begin{verbatim}
\documentclass{article}

\begin{document}
Small is beautiful.
\end{document}
\end{verbatim}
```

```
\documentclass{article}

\begin{document}
Small is beautiful.
\end{document}
```

Its starred version will draw the '␣' symbol instead of spaces.

For code snippets, it may be useful to store them in separate files, outside of the LaTeX document. This allows you to edit them in your editor

of choice, and makes the LaTeX code less cluttered. To accomplish this, use the \verbatiminput{⟨*file*⟩} command from the verbatim package.[8]

```
\verbatiminput{hello.c}
```

```
// A simple program
#include "stdio.h"

int main() {
  printf("Hello, World!\n");
  return 0;
}
```

\verbatiminput* also exists and works exactly as you would predict.

The verbatim environment, and the \verb and \verbatiminput commands, may not be used within parameters of other commands, so the below code will result in an error.

```
Text\footnote{\verb|abc| cannot be used here.}
```

If this is a problem for you, look at the fancyvrb [88] package.

2.17.2 The listings Package

While the verbatim environment is fine for small snippets, they are very crude and look basic. In order to make them a bit more fancy, you can

[8] This and the following examples use code snippets that can be found at https://github.com/oetiker/lshort/tree/master/src/examples.

use the listings [23] package. It defines the

`\lstinline` command that works like `\verb`
`lstlisting` environment that works like `verbatim`
`\lstinputlisting` command that works like `\verbatiminput`

```
\lstinline|\LaTeX{}|

\begin{lstlisting}
Here is some \LaTeX{} code.
\end{lstlisting}

\lstinputlisting{hello.c}
```

```
\LaTeX{}

Here is some \LaTeX{} code.

// A simple program
#include "stdio.h"

int main() {
  printf("Hello, World!\n");
  return 0;
}
```

As you can see, by default it isn't much different from the `verbatim environment`. The big difference is that every command accepts optional argument that allows you to customise the output. The argument accepts a comma-delimited key-value list. For example, to automatically highlight the code, we may pass the `language` key.

```
\lstinputlisting[language=C]{hello.c}
```

```
// A simple program
#include "stdio.h"

int main() {
  printf("Hello, World!\n");
  return 0;
}
```

For many, the default style of the listings may look a bit strange. No worries, this is easily fixed. The output can be customized using the following keys:

`basicstyle`, `keywordstyle`, `identifierstyle` and `commentstyle`

Note that you will need to use some font changing commands, when configuring the styles. More about font changing command can be found in Section 7.2.

```
\lstinputlisting[
  language=SQL,
  basicstyle=\ttfamily,
  commentstyle=\color{gray},
  keywordstyle=\itshape,
]{employee.sql}
```

```
-- A simple query
SELECT * FROM  employees
    WHERE salary > 2000.00
    ORDER BY last_name;
```

If you intend to use similar options for many listings in the document, you can use the `\lstset` command to set them beforehand. In this way,

you will avoid repeating them every time you want to typeset some code.

```
\lstset{
  language=C,
  numbers=left,
  breaklines,
  basicstyle=\ttfamily,
  commentstyle=\color{gray},
  prebreak=\textrightarrow,
}

\lstinputlisting{hello.c}

The statement
\lstinline|int x = 1;|
assigns 1 to variable
\lstinline|x|.
```

```
1 // A simple program
2 #include "stdio.h"
3
4 int main() {
5   printf("Hello,␣World→
       !\n");
6   return 0;
7 }
```

The statement `int x = 1;` assigns 1 to variable `x`.

If a language has several dialects, you may specify which inside square brackets before the language name. You must enclose the value in curly brackets.

```
\lstinputlisting[language={[LaTeX]TeX}]{hello.tex}
```

```
\documentclass{article}

\begin{document}
Hello, World!
\end{document}
```

When presenting code, it is often that we want to point to a specific line. To save the reader from counting which line we mean, it is possible to print line numbers by using the `numbers` key. Its possible values are

`none`, `left` and `right`.

```
\lstinputlisting[language=C, numbers=left]{hello.c}
```

```
1 // A simple program
2 #include "stdio.h"
3
4 int main() {
5   printf("Hello,␣World!\n");
6   return 0;
7 }
```

The appearance of numbers may be further customised with the following keys:

`firstnumber` allows you to specify the first number. Besides numbers, this key can also contain two special values: `last` and `auto` (the default). `last` continues numbering from the last listing, while `auto` continues it from the last listing with the same `name` argument, or starts over if no `name` is present.

`stepnumber` prints only every n-th number. For example, if you pass `stepnumber=7`, then only line numbers $1, 8, 15, 22, \ldots$ will be printed.

`numberblanklines` controls whether or not line numbers are printed on empty lines. Either `true` or `false`.

`numberstyle` allows you to customise the font for printing numbers. It accepts switch commands, as described in Section 7.2.

```
\lstinputlisting[
  language=C,
  numbers=left,
  stepnumber=2,
  numberblanklines=false,
  firstnumber=4,
  numberstyle=\tiny,
]{hello.c}
```

```
4  // A simple program
   #include "stdio.h"

   int main() {
8    printf("Hello,␣World!\n");
     return 0;
10 }
```

If you have a long file and want to show it piecewise, you can use the `firstline` and `lastline` keys.

```
The \lstinline|main|
function body
consists of:
\lstinputlisting[
  language=C,
  firstline=5,
  lastline=6,
]{hello.c}
```

The `main` function body consists of:

```
printf("Hello, World!\n");
return 0;
```

If you are trying to typeset long lines, you may order the listings to automatically break them by using the `breaklines` key.

```
\lstinputlisting[
  language=python,
  numbers=left,
  breaklines,
]{factorial.py}
```

```
1 import math
2
3 def oneline_factorial(n):
4     return math.prod(i for
          i in range(1, n + 1)
          )
5
6 print(oneline_factorial(5))
```

You can customise the indentation width with the `breakindent` key. It may be useful to indicate that a line break has occurred by using the `prebreak` and `postbreak` keys, which print their value respectively before and after an artificial line break. Listing 2.1 shows an example of this.

One of the neat features of the listings package is that you can evaluate LaTeX code inside the listing. The easiest way is to pass the `texcl` key,

```
\lstinputlisting[
  language=python,
  numbers=left,
  breaklines,
  breakindent=1cm,
  prebreak=\textrightarrow,
  postbreak=\textleftarrow,
]{factorial.py}
```

```
1  import math
2
3  def oneline_factorial(n):
4      return math.prod(i for→
           ← i in range(1, n →
           ←+ 1))
5
6  print(oneline_factorial(5)→
       ←)
```

Listing 2.1: An example of marking the artificial linebreaks in `listing` package.

which enables LaTeX syntax within comments.

```
\begin{lstlisting}[
  texcl,
  language=haskell,
]
-- I can use \LaTeX{} here.
-- The cost is in \texteuro{}
cost x = show (
    foldr (+) 0 x
  ) ++ " eur"
\end{lstlisting}
```

```
-- I can use LaTeX here.
-- The cost is in €
cost x = show (
    foldr (+) 0 x
  ) ++ "␣eur"
```

This feature is especially useful in combination with the `\label` command. We can use it to point to a specific line of code without hardcoding it into the document.

```
\begin{lstlisting}[
  texcl,
  numbers=left,
  language=haskell,
]
cost x = show (
    foldr (+) 0 x
  ) ++ " eur" -- \label{concat}
\end{lstlisting}
The \lstinline|++| in Haskell
as seen in line~\ref{concat}
means string concatenation.
```

```
1  cost x = show (
2      foldr (+) 0 x
3    ) ++ "␣eur" --
```

The ++ in Haskell as seen in line 3 means string concatenation.

This has the downside that it introduces empty comments that are just there to include label. To fix this we may use the `escapeinside` key that accepts sets the two delimiters between which LaTeX code can be

typed. The delimiters themselves will not be printed.

```
\begin{lstlisting}[
  escapeinside={(*}{*)},
  numbers=left,
  language=haskell,
]
cost x = show ( (*\label{show}*)
    foldr (+) 0 x
  ) ++ " eur" (*(\texteuro)*)
\end{lstlisting}
The \lstinline|show| in Haskell
as seen in line~\ref{show}
converts its argument to a string.
```

```
1  cost x = show (
2      foldr (+) 0 x
3    ) ++ "␣eur" (€)
```

The `show` in Haskell as seen in line 1 converts its argument to a string.

2.17.3 The minted Package

While the listings package allows you to typeset pretty decent looking code snippets, you may still find it lacking in some respects. The most obvious problem is the syntax highlighting: you may find that your language of choice is not supported. You can write a custom style for the language, but it relies on a simple keyword search, so this is an option only for simple languages.

If this is a problem for you, it may be worth considering the minted [63] package. It relies on the Pygments [7] program to highlight code. In contrast to the listings package, this program uses pushdown automata to colour the code, which may produce much better results. Moreover, the predefined styles and language support are much richer than those offered by the listings package. This is the package that is used to colour examples throughout this booklet.

Due to the fact that the minted package relies on an external program, there are two things that you need to do before using it:

1. Ensure that the Pygments program is installed. If you are using a Unix-like system it is probably enough to execute
   `pip install Pygments`
   You can find more information at their official website: https://pygments.org and in the minted [63] package documentation.

2. Add the `--shell-escape` flag when the compiling LaTeX document, for example:
   `xelatex --shell-escape document.tex`

It is important to understand what the cryptic flag mentioned in the second point actually means. It allows the LaTeX document to call any external program via the shell. Let me repeat this to make the point clear: *Enabling this flag allows the LaTeX document to call any program via the shell.* This means that a maliciously crafted document could, for example, delete all files from your computer. Only enable this flag if the document comes from a trusted source.

Once you meet these requirements, you can use the package in a manner similar to the previous approach with listings. It provides

```
\mintinline[⟨options⟩]{⟨language⟩}{⟨code⟩}
\inputminted[⟨options⟩]{⟨language⟩}{⟨code⟩}
\begin{minted}[⟨options⟩]{⟨language⟩}
```

In contrast to listings, the ⟨*language*⟩ is a required parameter. The command `\mintinline` works with both normal groups and custom delimiters.

```
\mintinline{latex}|\LaTeX{}|

\begin{minted}{python}
x = 0
for i in range(10):
  x += 2**i
print(f"Result: {x}")
\end{minted}

\inputminted{rust}{age.rs}
```

```
\LaTeX{}

x = 0
for i in range(10):
  x += 2**i
print(f"Result: {x}")

fn main() {
  let age = 25;
  println!("I am {} years old.",
    age);
}
```

minted's capabilities are comparable to those of the listings package. Some of the options have synonyms that makes their usage identical to the listings package. For example, to enable line numbers you can use

either the `linenos` key (which is a boolean value) or the `numbers` key (which accepts `left`, `right`, `none` or `both`).

```
\inputminted[
  linenos,
]{css}{review.css}
```

```
1 .review {
2     font-size: large;
3     background-color: darkred;
4     color: aliceblue;
5     display: flex;
6     align-items: center;
7 }
```

You can change the line numbering using `firstnumber` and `stepnumber`. Additionally you may highlight some lines using the `highlightlines` key to draw attention to them. Note that `highlightlines` uses the transformed line numbers.

```
\inputminted[
  linenos,
  firstnumber=5,
  stepnumber=3,
  highlightlines={6, 10-12}
]{coq}{em.v}
```

```
  Theorem em_irr:
6   forall A: Prop, ~~(A \/ ~A).
  Proof.
    unfold not.
9   intros.
    apply H.
    right.
12  intros.
    apply H.
    left.
15  exact H0.
  Qed.
```

As in the listings package, `breaklines` enables automatic breaking of long lines, however it will only break the lines at whitespace characters. If you want the breaks to occur anywhere, use the `breakanywhere` key. The breaking behaviour may be fine-tuned by using the keys `breakbytoken`,

`breakbytokenanywhere`, `breakbefore` and `breakafter`.

```
\inputminted[
  linenos,
  breaklines,
]{haskell}{quicksort.hs}
```

```haskell
quicksort :: Ord a => [a] ->
  ↪ [a]
quicksort [] = []
quicksort (p:xs) = quicksort
  ↪ (filter (< p) xs) ++ [p]
  ↪ ++ quicksort (filter (>=
  ↪ p) xs)
```

As you can see, by default it inserts "↪" to indicate that an automatic line break occurred. You can customise the printed symbols with the `breaksymbolleft` and `breaksymbolright` keys.

In order to use LaTeX inside the comments, you can use either the `texcl` or its synonym `texcomments`. You can also add custom LaTeX escape delimiters with `escapeinside`, but these must be single tokens (unlike in listings).

```
\begin{minted}[
  linenos,
  texcomments,
  escapeinside=||,
]{html}
<!DOCTYPE html>
<html>
  <head> |\label{head}|
    <title>On |\LaTeX{}|</title>
  </head>
  <body>
    <div class="review">
      <!-- \LaTeX{} review -->
      <p>It's awesome!</p>
    </div>
  </body>
</html>
\end{minted}
The \mintinline{html}{<head>}
element in line~\ref{head}
contains page metadata.
```

```html
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>On LaTeX</title>
5    </head>
6    <body>
7      <div class="review">
8        <!-- LaTeX review -->
9        <p>It's awesome!</p>
10     </div>
11   </body>
12 </html>
```

The `<head>` element in line 3 contains page metadata.

The Pygments program comes with numerous predefined styles for typesetting your code. In order to choose between them use the `style` key.

```
\inputminted[
  style=bw,
]{bash}{echo.sh}
\inputminted[
  style=sas,
]{bash}{echo.sh}
\inputminted[
  style=xcode,
]{bash}{echo.sh}
```

```
#!/bin/bash
echo LaTeX is awesome!

#!/bin/bash
echo LaTeX is awesome!

#!/bin/bash
echo LaTeX is awesome!
```

To avoid having to set the options for every listing and code snippet, you can use `\setminted` to define the default values.

```
\setminted{
  linenos,
  breaklines,
  style=xcode,
}
\inputminted{c++}{goodbye.cpp}
```

```
1  #include <iostream>
2
3  int main() {
4    std::cout << "Goodbye!"
     ↪  << std::endl;
5    return 0;
6  }
```

2.18 Tables

In LaTeX, the environment to typeset tables (and more) is called `tabular`. While it can be used on its own, the resulting table layouts look extremely old fashioned. We recommend adding the booktabs [17] package, which provides several commands that let you typeset beautiful modern looking tables.

Note that the booktabs [17] package does not modify the tabular environment, it just adds some extra commands for spacing and adding lines to your tables. If you write for a publication which has other

requirements, like not using any extra packages, simply do not use the extra commands from booktabs [17].

To achieve some more advanced table layouts, you will also learn about the longtable [11], array [46] and multirow [51] packages in this section. They are so useful, that you'll probably want to put them all into the preamble of your document right away.

In addition to a description of its commands, the booktabs documentation [17] contains guidelines on typesetting professional-looking tables. We have taken these guidelines to heart when writing this booklet. You will find some of them in this section.

2.18.1 Basic Tables

The environment `tabular` has the following form

```
\begin{tabular}[⟨pos⟩]{⟨colspec⟩}
```

The ⟨*colspec*⟩ argument, which stands for column specifiers, defines the format of the columns in the table. Use an `l` for a column of left-aligned text, `r` for right-aligned text, and `c` for centred text. Within the environment, use `&` to begin the next cell within the current row, and `\\` to begin the next row.

```
\begin{tabular}{lcr}
  left & centre & right \\
  1    & 2      & 3     \\
\end{tabular}
```

left	centre	right
1	2	3

Note that the text inside the cells will not be wrapped. If you want the column to contain justified text with line breaks, use the `p{`⟨*width*⟩`}` column specifier, where ⟨*width*⟩ is the width of the column.

```
\begin{tabular}{lp{3cm}}
  left & Very long paragraph
         that gets broken into
         multiple lines. \\
  1    & Another one,
         but shorter. \\
\end{tabular}
```

left	Very long paragraph that gets broken into multiple lines.
1	Another one, but shorter.

The optional ⟨*pos*⟩ argument specifies the vertical position of the table with respect to the baseline of the text. There are three possible alignments: `c` centres the table (the default), `t` matches the baseline of the top row, and `b` matches the baseline of the bottom row.

```
text
\begin{tabular}{ll} 1 & 2 \\ 3 & 4 \\ \end{tabular}
text
\begin{tabular}[t]{ll} 1 & 2 \\ 3 & 4 \\ \end{tabular}
text
\begin{tabular}[b]{ll} 1 & 2 \\ 3 & 4 \\ \end{tabular}
text
```

							1	2		
text	1	2	text	1	2	text	3	4	text	
	3	4		3	4					

What we have seen so far, allows aligning some items in rows and columns, but real tables need visible headings. To insert them, use the commands `\toprule`, `\midrule` and `\bottomrule` from the booktabs package. All of these accept an optional argument that specifies their thickness, but usually the default settings are just fine. See Listing 2.2 for an example of using these commands.

With these commands you are already able to produce simple, yet nicely looking tables. You may be are wondering why we have not mentioned how to add vertical lines between the columns. Well, we did not mention them because the first rule of producing professional-looking tables is that you *must not* use vertical lines.

The second rule is to never use double lines such as these shown in Listing 2.3. The default lines are already of different weight in order to signify their meaning. Using more lines than is necessary means cluttering the space without adding information. If you stick to these two rules, your tables will look quite good already.

To make the tables look even more sleek, you may want to remove the padding in the first and last column. To control the space between the columns use the `@{`⟨*sep*⟩`}` column specifier, where ⟨*sep*⟩ is either text or space. Space of arbitrary length can be inserted using the

```
\begin{tabular}{lcl}
  \toprule
  Alignment & Letter & Niceness  \\
  \midrule
  Left      & l      & Very nice \\
  Centre    & c      & Very nice \\
  Right     & r      & Very nice \\
  \bottomrule
\end{tabular}
```

Alignment	Letter	Niceness
Left	l	Very nice
Centre	c	Very nice
Right	r	Very nice

Listing 2.2: A simple table using the `booktabs` commands.

`\hspace{`⟨*width*⟩`}` command. The contents of the ⟨*sep*⟩ arguments will be put between cells in the relevant column.

```
\begin{tabular}{
  @{a} c @{\hspace{1cm}} c @{|} c @{ b}
}
  1 & 2 & 3 \\
  4 & 5 & 6\\
  7 & 8 & 9\\
\end{tabular}
```

a1	2	3 b
a4	5	6 b
a7	8	9 b

If you leave the ⟨*sep*⟩ empty it will suppress the padding between the columns. See Listing 2.4 for an example.

Sometimes it may make sense to group columns under one heading. To achieve this, we need two things: a way to group multiple heading columns, and to have a line covering the same columns (sitting underneath the merged heading cell). Use the

`\multicolumn{`⟨*ncols*⟩`}{`⟨*colspec*⟩`}{`⟨*text*⟩`}`

command to merge the cells. The ⟨*ncols*⟩ argument indicates how many

```latex
\begin{tabular}{lll}
  \toprule[0.1cm]
  \toprule
  Person      & Face & Table    \\
  \midrule
  \midrule
  Me          & :(   & Not nice \\
  You         & :[   & Awful    \\
  Your reader & :<   & Terrible \\
  \bottomrule
  \bottomrule[0.1cm]
\end{tabular}
```

Person	Face	Table
Me	:(	Not nice
You	:[	Awful
Your reader	:<	Terrible

Listing 2.3: An anti-example of using double rules inside a table.

```latex
\begin{tabular}{@{}lll@{}}
  \toprule
  Person      & Face & Table       \\
  \midrule
  Me          & :)   & Nice        \\
  You         & :]   & Sleek       \\
  Your reader & :>   & Informative \\
  \bottomrule
\end{tabular}
```

Person	Face	Table
Me	:)	Nice
You	:]	Sleek
Your reader	:>	Informative

Listing 2.4: An example of using `@{}` column specifier to suppress outer separators.

columns should the ⟨*text*⟩ span, while the ⟨*colspec*⟩ is the column specification for the new content—the same as when starting a `tabular` environment. See Listing 2.5 for an example.

To get a horizontal line to span multiple columns, use the command

`\cmidrule[`⟨*dim*⟩`](`⟨*trim*⟩`){`⟨*a-b*⟩`}`

While the command uses nonstandard syntax for ⟨*trim*⟩, this is just an optional argument with different pair of delimiters.[9] The ⟨*dim*⟩ allows us to specify the thickness of the line. The ⟨*trim*⟩ argument accepts any combination of `r`, `r{`⟨*dim*⟩`}`, `l` or `l{`⟨*dim*⟩`}`. This allows you trim the rule from right or left, either by the package default or the specified ⟨*dim*⟩. It is usually recommended to trim the rules from the side where they touch other columns. The command's only required argument is the span of columns to draw the line over (⟨*a-b*⟩). See Listing 2.6 for an example.

[9] This syntax allows specification of the second optional argument without the first, which would be impossible if square brackets were used for both.

```
\begin{tabular}{@{}lll@{}}
  \toprule
  Person      & Face & Table                      \\
  \midrule
  Me          & :)   & Nice                       \\
  You         & :]   & Sleek                      \\
  Your reader & \multicolumn{2}{c}{Not available} \\
  \bottomrule
\end{tabular}
```

Person	Face	Table
Me	:)	Nice
You	:]	Sleek
Your reader	Not available	

Listing 2.5: An example of using the `\multicolumn` command in a table.

If you need a cell to span multiple rows instead of columns, then you have to use package multirow. Its main command is a little more complicated

`\multirow[`⟨*vpos*⟩`]{`⟨*nrows*⟩`}{`⟨*width*⟩`}[`⟨*vmove*⟩`]{`⟨*text*⟩`}`

The arguments are

⟨*vpos*⟩ is the vertical position of the text within the cell. Can be either `c` for centre (the default), `t` for top or `b` for bottom.

⟨*nrows*⟩ is the number of rows for the new cell to span.

⟨*width*⟩ is the width of the cell. Apart from a regular length, you can also pass two special arguments here: `*` for the natural length of the text and `=` for the same width as the column (this only makes sense if the column width was specified, for example via `p{3cm}`).

⟨*vmove*⟩ allows adjusting the position of the text if it sits too low or too high.

```latex
\begin{tabular}{@{}lll@{}}
  \toprule
              & \multicolumn{2}{c}{Reaction}      \\
  \cmidrule(l){2-3}
  Person      & Face & Exclamation                \\
  \midrule
  Me          & :)   & Nice                       \\
  You         & :]   & Sleek                      \\
  Your reader & \multicolumn{2}{c}{Not available} \\
  \bottomrule
\end{tabular}
```

	Reaction	
Person	Face	Exclamation
Me	:)	Nice
You	:]	Sleek
Your reader	Not available	

Listing 2.6: An example of using the `\cmidrule` command inside a table.

⟨*text*⟩ is the text to be put in the cell.

Unlike with `\multicolumn` you still have to write all the cells in remaining rows, but they should be empty. See Listing 2.7 for an example.

In general it is not a good idea to use the `\multirow` command for showing values common to multiple rows. Usually repeating the values in question makes the table more readable.

When typesetting numerical data in the table, you may want to align it by decimal point. A way to do this (and more) is described in subsection 3.5.4. The section also contains some guidelines about typesetting numerical data in tables.

2.18.2 Long Tables

Material typeset with the tabular environment always stays together on one page. This poses a problem for especially long tables. If your table is not very wide, you may get away with it by typesetting its rows side by side. You should then put a bigger space between those to indicate that these are separate; either by using `@{...}` or by putting an empty column between those as shown in Listing 2.8.

This approach will obviously not work for *really* long tables. This is where the longtable package shines. It defines a `longtable` environment that works in a similar way to the `tabular` environment, but allows page breaks inside.

`\begin{longtable}[`⟨*align*⟩`]{`⟨*colspec*⟩`}`

In contrast to `tabular`, `longtable` always starts a new paragraph and is centred by default. Thus the optional argument ⟨*align*⟩ specifies whether the table should be centred, on the left, or on the right. Use `c`, `l` and `r` to specify this. The ⟨*colspec*⟩ argument is the same as in the `tabular` environment. See Listing 2.9 for an example.

Note that the page breaks are always placed between rows. If you have a tall row, thanks to a `p{...}` specification, it will not broken (as seen in Listing 2.10).

The `longtable` environment also defines some additional commands that end table rows. One of them is `\textbackslash*` which, similarly to its use in a paragraph, prohibits a page break after the row. Other useful

```latex
\begin{tabular}{@{}lll@{}}
  \toprule
              & \multicolumn{2}{c}{Reaction}      \\
  \cmidrule(l){2-3}
  Person      & Face            & Exclamation     \\
  \midrule
  \multirow[t]{2}{*}{VIPs} & :) & Nice            \\
              & :]              & Sleek           \\
  Others      & \multicolumn{2}{c}{Not available} \\
  \bottomrule
\end{tabular}
```

	Reaction	
Person	Face	Exclamation
VIPs	:)	Nice
	:]	Sleek
Others	Not available	

Listing 2.7: An example of using `\multirow` inside a table.

```latex
\begin{tabular}{@{}cllcl@{}}
  \toprule
  Digit & Word  && Digit & Word  \\
  \midrule
  0     & Zero  && 5     & Five  \\
  1     & One   && 6     & Six   \\
  2     & Two   && 7     & Seven \\
  3     & Three && 8     & Eight \\
  4     & Four  && 9     & Nine  \\
  \bottomrule
\end{tabular}
```

Digit	Word	Digit	Word
0	Zero	5	Five
1	One	6	Six
2	Two	7	Seven
3	Three	8	Eight
4	Four	9	Nine

Listing 2.8: An example of producing table with rows next to each other.

```latex
\begin{longtable}{cl}
  \toprule
  Number & Word \\
  \midrule
  0      & Zero \\
  1      & One  \\
  2      & Two  \\
  % ...
  10     & Ten  \\
  \bottomrule
\end{longtable}
```

Number	Word
0	Zero
1	One
2	Two
3	Three
4	Four
5	Five
6	Six
7	Seven
8	Eight
9	Nine
10	Ten

Listing 2.9: An example of using the `longtable` environment.

```latex
\begin{longtable}{
  cp{2cm}
}
  \toprule
  Number & Words                \\
  \midrule
  0      & Zero                 \\
  1      & Multi paragraph
           text.

           It makes the table
           cell very tall.      \\
  2      & Two                  \\
  \bottomrule
\end{longtable}
```

Number	Words
0	Zero
1	Multi paragraph text. It makes the table cell very tall.
2	Two

Listing 2.10: An anti-example of using `longtable` assuming it will break the paragraphs inside the cells.

commands are \endhead, \endfirsthead, \endfoot and \endlastfoot. If you end a row with them, they will set the preceding rows as headers and footers of the table. The \endhead and \endfoot commands put the headers and footers on every page, while \endfirsthead and \endlastfoot put them on the first and last page. An example of using these commands is presented in Listing 2.11.

Note that while you can use \multicolumn and \multirow normally within the longtable environment, the table may get very complicated. LaTeX will then need several passes to properly calculate the column widths.

2.18.3 Advanced Tables and Non-Tables

In regular text, you can specify the amount of space between two lines as an optional argument to the \\TU\textbackslash command. Can the same be done within a table? It turns out that it can, but there is one caveat. If you use it together with a p{...} column, the resulting space will be different depending on column order.

```
\begin{tabular}{lp{1cm}}
  1 & 2\newline x \\[1cm]
  3 & 4            \\
\end{tabular}
\begin{tabular}{p{1cm}l}
  2\newline x & 1 \\[1cm]
  4           & 3 \\
\end{tabular}
```

1	2 x	2 x	1
3	4	4	3

This counterintuitive behaviour is caused by the fact that the space to add is calculated based on the last column. In order to prevent that, simply add \usepackage{array} to your preamble. Since there are no downsides to this, it is recommended to always use this package when starting a new document, to avoid breaking tables that rely on the original behaviour later. The array package also defines some additional column specifiers.

The p{...} specifier allows the insertion of text with line breaks into the table. The text always starts from the top though, so the array

```latex
\begin{longtable}{cl}
  \toprule
  \multicolumn{2}{c}{V.~Important Table} \\
  Number & Word                            \\
  \midrule \endfirsthead

  \toprule
  \multicolumn{2}{c}{VIT (continued)}    \\
  Number & Word                          \\
  \midrule \endhead

  \midrule
  \multicolumn{2}{c}{Not the end}        \\
  \bottomrule \endfoot

  \midrule
  \multicolumn{2}{c}{The end of VIT}     \\
  \bottomrule \endlastfoot

  0      & Zero   \\
  1      & One    \\
  % ...
  10     & Ten    \\
\end{longtable}
```

V. Important Table	
Number	Word
0	Zero
1	One
2	Two
3	Three
4	Four
5	Five
Not the end	

VIT (continued)	
Number	Word
6	Six
7	Seven
8	Eight
9	Nine
10	Ten
The end of VIT	

Listing 2.11: An example of `longtable` with running headers and footers.

package defines two additional specifiers: `m{...}` for vertically centred text, and `b{...}` for text placed vertically from the bottom.

```
\begin{tabular}{
  lp{1.4cm}m{1.4cm}b{1.4cm}
}
  Cell                    &
  Top matches cell.       &
  Centre matches cell. &
  Bottom matches cell. \\
\end{tabular}
```

Cell	Top matches cell.	Centre matches cell.	Bottom matches cell.

When a column must be formatted a certain way, it is inconvenient to put the same commands in every cell. Moreover, if you decide that something needs to be changed about the formatting, then you would have to edit every cell individually. To avoid that, the `array` package defines `>{`⟨*cmds*⟩`}` and `<{`⟨*cmds*⟩`}` column specifiers which can be used to put some code before and after a column.

```
\begin{tabular}{
  l
  >{\begin{em}}l<{\end{em}}
}
  text & emphasised text \\
  text & emphasised text \\
  text & emphasised text \\
\end{tabular}
```

text	*emphasised text*
text	*emphasised text*
text	*emphasised text*

While using `>{...}` and `<{...}` comes in handy if we only have one such column, it quickly becomes inconvenient when many columns of the same type appear in any tables. In that case, it may be desirable to use the command

`\newcolumntype{`⟨*newcolspec*⟩`}{`⟨*definition*⟩`}`

where ⟨*newcolspec*⟩ is a single letter used for specification of columns, and the ⟨*definition*⟩ is what should be inserted in the table when using

it.

```
\newcolumntype{e}{
  >{\begin{em}\begin{FlushLeft}}
  p{2.5cm}
  <{\end{FlushLeft}\end{em}}
}
\begin{tabular}{ee}
  This cell will
    be flushed left
    and emphasised. &
  This cell will
    be flushed left
    and emphasised.
  \tabularnewline

  This cell will
    be flushed left
    and emphasised. &
  This cell will
    be flushed left
    and emphasised.
  \tabularnewline
\end{tabular}
```

This cell will be flushed left and emphasised.	*This cell will be flushed left and emphasised.*
This cell will be flushed left and emphasised.	*This cell will be flushed left and emphasised.*

In order to create a tic-tac-toe grid, you need to put some vertical and horizontal lines of the same width. In order to do so, you may use the `|` (vertical bar) column specifier for vertical rules and `\hline` for horizontal rules.

```
\begin{tabular}{c|c|c}
  O & X & O \\
  \hline
  X & X & O \\
  \hline
  X & O & X \\
\end{tabular}
```

O	X	O
X	X	O
X	O	X

In case your grid gets more complicated, you may need the `\cline{`$\langle a\text{-}b\rangle$`}`

command, which draws a horizontal line spanning only the specified rows.

```
\begin{tabular}{|cccc|}
  \cline{4-4}
  a & a & \multicolumn{1}{c|}{a} & b \\
  \cline{2-3}
  a & \multicolumn{1}{|c}{b} & b & b \\
  \cline{2-2}
  a & a & \multicolumn{1}{|c}{b} & b \\
  \cline{1-2}
\end{tabular}
```

a	a	a	b
a	b	b	b
a	a	b	b

2.19 Including Graphics and Images

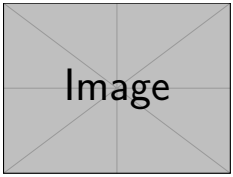
LaTeX itself does not provide facilities to include images. In order to so, it is necessary to use the graphicx [10] package. It provides the

`\includegraphics[⟨options⟩]{⟨file⟩}`

command that can be used to include the ⟨*file*⟩ in the document.[10]

```
Here is an image:
\includegraphics{example-image}
```

Here is an image:



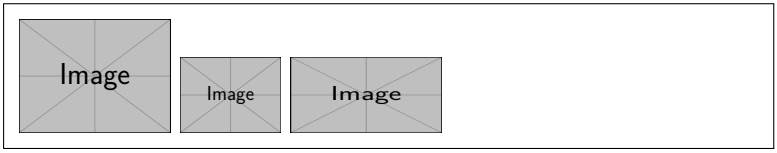
The file may be a PDF, PNG or JPEG. If you omit the extension, then LaTeX will try the following in order: `.pdf`, `.png`, `.jpg`, `.mps`, `.jpeg`, `.jbig2`, `.jb2`. Note that if your filename contains dots (other than the extension dot), then its basename has to be put in curly braces like so:

```
\includegraphics{{image.with.dots}.jpg}
```

[10] In the subsequent examples, we are using images from the mwe [64] package that should be installed with your LaTeX distribution.

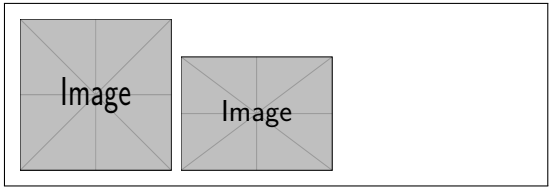
The ⟨*options*⟩ parameter is a comma delimited list of keys and values. The most basic keys are `width` and `height`. If only one is supplied, then LaTeX will calculate the other to keep the aspect ratio of the original image.

```
\includegraphics[width=2cm]{example-image}
\includegraphics[height=1cm]{example-image}
\includegraphics[width=2cm, height=1cm]{example-image}
```



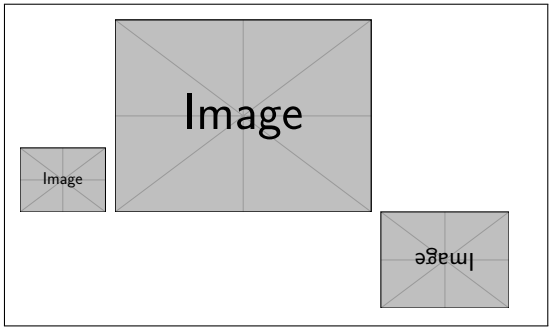
You may also pass the `keepaspectratio` key. This will change the meaning of `width` and `height` keys to the *maximum* possible width and height.

```
\includegraphics[
  width=2cm,
  height=2cm,
]{example-image}
\includegraphics[
  width=2cm,
  height=2cm,
  keepaspectratio,
]{example-image}
```



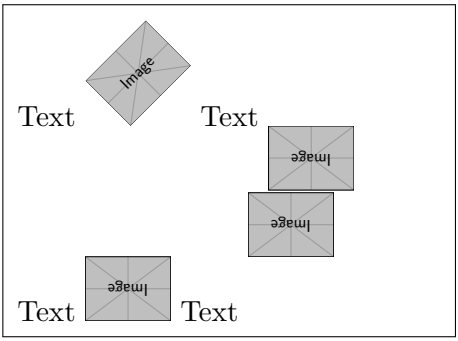
Instead of specifying width or height, you may also pass the `scale` argument to scale it by a specific amount. Passing negative values will reflect the image.

```
\includegraphics[
  scale=0.1,
]{example-image}
\includegraphics[
  scale=0.3,
]{example-image}
\includegraphics[
  scale=-0.15,
]{example-image}
```



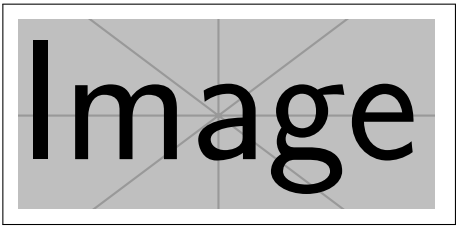
If you want to rotate the picture, you can pass the `angle` key. By default, the image is rotated about its bottom-left corner. You can change the anchor point by specifying the `origin` key. Its value may contain one or two of the following specifications: `l` for left, `r` for right, `c` for centre, `t` for top, `b` for bottom, and `B` for baseline (useful with Ti*k*Z pictures as described in Chapter 6). So, for example, `lc` will rotate the image about the centre of its left edge, while `c` will rotate it about its centre.

```
Text \includegraphics[
  angle=45,
]{example-image}
Text \includegraphics[
  angle=180,
]{example-image}  \\
Text \includegraphics[
  angle=180,
  origin=c,
]{example-image}
Text \includegraphics[
  angle=180,
  origin=lt,
]{example-image}
```



If you want to show only a part of the picture, you may specify the `clip` key and set the `viewport` key to required part of the image. The `viewport` takes as its value four numbers: the first two specify the bottom-left point of a rectangle, while the last two specify the top-right point of rectangle.

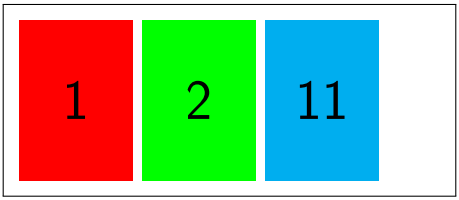
```
\includegraphics[
  clip,
  viewport=3cm 3cm 8.5cm 5.5cm,
]{example-image}
```



If you are including a PDF file, `\includegraphics` will only include its first page. If you want to specify a different page, pass it to the `page`

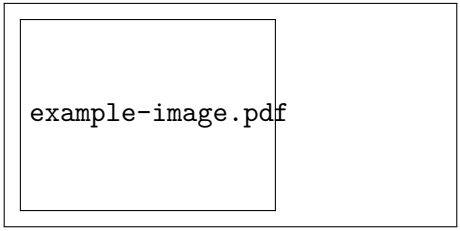
key.

```
\includegraphics[
  width=1.5cm,
]{example-image-a4-numbered}
\includegraphics[
  width=1.5cm,
  page=2,
]{example-image-a4-numbered}
\includegraphics[
  width=1.5cm,
  page=11,
]{example-image-a4-numbered}
```



If you pass the `draft` option then, instead of the actual images, only their names are printed, to save on file size and processing time. This key will be inherited from the `\documentclass` options, if specified there.

```
\includegraphics[
  scale=0.3,
  draft,
]{example-image}
```



2.20 Floating Bodies

If you try writing a longer text with tables and images, you will soon notice a problem: they cannot be broken across multiple pages (except for longtables). If you simply include them in normal text flow, this will lead to a lot of empty space if the image is large and the text preceding it would be to close to the bottom of the page. In Listing 2.12 the first page is half empty.

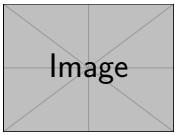
The solution to this problem is to 'float' any figure or table that does not fit on the current page to a later page, while filling the current page with body text. LaTeX offers two environments for floating bodies; one for tables and one for figures. To take full advantage of these two

```
Here is some text
to take up some space.

Below this text is an image.

\begin{Center}
  \includegraphics[
    scale=0.2
  ]{example-image}
\end{Center}

Here is more text.
```

Here is some text to take up some space.

Below this text is an image.

1



Here is more text.

2

Listing 2.12: An anti-example of half-empty page due to a large, non-floated image.

environments, it is important to understand approximately how LaTeX handles floats internally. Otherwise, floats may become a major source of frustration, because LaTeX never puts them where you want them to be.

Let's first have a look at the commands LaTeX supplies for floats. Any material enclosed in a `figure` or `table` environment will be treated as floating matter.

```
\begin{figure}[⟨placement specifier⟩]
\begin{table}[⟨placement specifier⟩]
```

The optional ⟨*placement specifier*⟩ parameter is used to tell LaTeX about the locations to which the float is allowed to be moved. A ⟨*placement specifier*⟩ is constructed by building a string of *float-placing permissions*. See Table 2.2 for a description of available specifiers. Note that the order of the specifiers does not matter.

For example, a table float could be started with the following line:

```
\begin{table}[!hbp]
```

The placement specifier `[!hbp]` allows LaTeX to place the table right here (`h`) or at the bottom (`b`) of some page or on a special floats page (`p`), and all this even if it does not look that good (`!`). If no placement specifier is given, the standard classes assume `[tbp]`.

LaTeX will place every float it encounters according to the placement specifier supplied by the author. If a float cannot be placed on the current page it is deferred either to the figures queue or the tables queue.[11] When a new page is started, LaTeX first checks if it is possible to fill a special 'float page' with floats from the queues. If this is not possible, the first float on each queue is treated as if it had just occurred in the text: LaTeX tries again to place it according to its respective placement specifiers (except `h`, which is no longer possible). Any new floats occurring in the text get placed into the appropriate queues. LaTeX strictly maintains the original order of appearance for each type of float. That's why a figure that cannot be placed pushes all subsequent figures to the end of the document. Therefore:

[11] These are FIFO—'first in first out'—queues!

Table 2.2: Float Placing Permissions.

Spec	Permission to place the float …
`h`	*here* at the very place in the text where it occurred. This is useful mainly for small floats.
`t`	at the *top* of a page
`b`	at the *bottom* of a page
`p`	on a special *page* containing only floats.
`!`	without considering most of the internal parameters[a], which could otherwise stop this float from being placed.

[a] Such as the maximum number of floats allowed on one page.

> If LaTeX is not placing the floats as you expected, it is often only one float jamming one of the two float queues.

While it is possible to give LaTeX single-location placement specifiers, this causes problems. If the float does not fit in the location specified it becomes stuck, blocking subsequent floats. In particular, you should never, ever use the [h] option—it is so bad that in more recent versions of LaTeX, it is automatically replaced by [ht].

Using floats, Listing 2.12 could be made better by rewriting it like Listing 2.13. As you can see there, LaTeX proceeded to typeset the remaining text and delayed inserting the image until the next page.

Having explained the difficult bit, there are some more things to mention about the `table` and `figure` environments. Use the

`\caption[`⟨*short caption*⟩`]{`⟨*full caption*⟩`}`

command to define a caption for the float. A running number, and the string "Figure" or "Table", will be added by LaTeX.

The two commands

`\listoffigures` and `\listoftables`

operate similar to the `\tableofcontents` command, printing a list of figures or tables, respectively. These lists will display the whole caption,

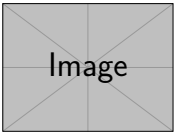
```
Here is some text
to take up some space.

Below this text is an image.

\begin{figure}
  \centering
  \includegraphics[
    scale=0.2
  ]{example-image}
\end{figure}

Here is more text.
```

Here is some text to take up some space.

Below this text is an image.

Here is more text.

1



2

Listing 2.13: An example of using `figure` float to achieve dynamic image placement inside a document.

so if you tend to use long captions you must have a shorter version of the caption for the lists. This is accomplished by entering the short version in brackets after the `\caption` command.

```
\caption[Short]{LLLLLoooooonnnnnggggg}
```

Use `\label` and `\ref` to create a reference to a float within your text. Note that the `\label` command must come *after* the `\caption` command, since you want it to reference the number of the caption. Listing 2.14 presents an example usage of these commands.

Under certain circumstances it might be necessary to use the

`\clearpage` or even the `\cleardoublepage`

command. It orders LaTeX to immediately place all floats remaining in the queues and then start a new page. `\cleardoublepage` even goes to a new right-hand page.

2.20.1 Createing your own Float-Types

If you want to have other float types besides "Table" and "Figure", use the newfloat [70] package. It provides the

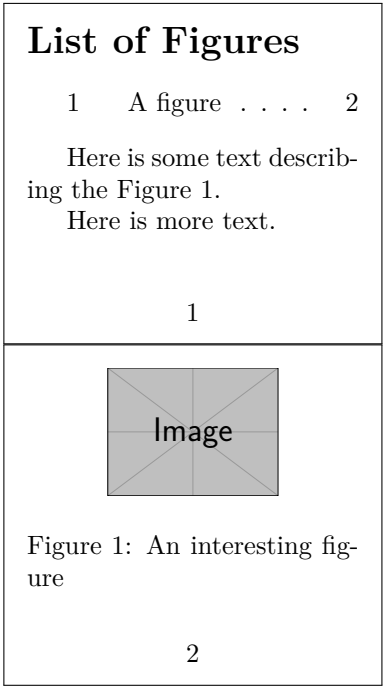
`\DeclareFloatingEnvironment[`⟨*options*⟩`]{`⟨*name*⟩`}`

For example, the `\DeclareFloatingEnvironment{cat}` will define a new environment, `cat`, which can be used like the `figure` and `table` environments, and command `\listofcats`, which works in the same way as `\listoftables`. You can customise the floats using the ⟨*options*⟩ to change the printed captions (`name`), list title (`listname`) or default placement (`placement`).

2.20.2 The `longtable` Environment

Floating bodies occupy only a single page. Thus, putting a `longtable` into one does not make sense. Still, you may want to have the `longtable` listed in the list of tables with some caption. The longtable package defines its own `\caption` command, which you can use inside the environment. Its use is similar to the standard `\caption` command, except that it

```latex
\listoffigures

Here is some text describing
the Figure~\ref{figure}.

\begin{figure}
  \centering
  \includegraphics[
    scale=0.2
  ]{example-image}
  \caption[A figure]{An
    interesting figure}%
  \label{figure}
\end{figure}

Here is more text.
```



Listing 2.14: An example of using `figure` float together with references and list of figures.

is treated as a row. If you pass an empty optional argument to the `\caption` command, it will typeset normally but it won't be put in the list of tables, which is useful if you want to have a running caption. See Listing 2.15, for an example.

2.21 Big Projects

When working on big documents, you might want to split the input file into several parts. LaTeX has two commands that help you do this.

Use the

`\include{`⟨*filename*⟩`}`

command in the document body to insert the contents of another file named *filename.tex*. Note that LaTeX will start a new page before processing the material input from *filename.tex*.

The second command can be used in the preamble. It allows you to instruct LaTeX to only input some of the `\include`d files.

`\includeonly{`⟨*filename*⟩`,` ⟨*filename*⟩`, ...}`

After this command is executed in the preamble of the document, only `\include` commands for the filenames that are listed in the argument of the `\includeonly` command will be executed.

The `\include` command starts typesetting the included text on a new page. This is helpful when you use `\includeonly`, because the page breaks will not move, even when some `\include` files are omitted. Sometimes this might not be desirable. In this case, use the

`\input{`⟨*filename*⟩`}`

command. It simply includes the file specified. No flashy suits, no strings attached.

To make LaTeX quickly check your document, use the syntonly package. This makes LaTeX skim through your document, only checking for proper syntax and usage of commands, but doesn't produce any (PDF) output. As LaTeX runs faster in this mode, you may save yourself valuable time. Usage is very simple:

```latex
\begin{longtable}{@{}cl@{}}
  \caption{Numbers}        \\
  \toprule
  Number & Word            \\
  \midrule \endfirsthead

  \caption[]{(continued)} \\
  \toprule
  Number & Word            \\
  \midrule \endhead

  \bottomrule \endfoot
  \bottomrule \endlastfoot

  0       & Zero    \\
  % ...
\end{longtable}
```

Table 1: Numbers

Number	Word
0	Zero
1	One
2	Two
3	Three

Table 1: (continued)

Number	Word
4	Four
5	Five
6	Six
7	Seven

Listing 2.15: An example of `longtable` with running caption.

```
\usepackage{syntonly}
\syntaxonly
```

When you want to produce pages, just comment out the second line (by adding a percent sign).

Chapter 3

Typesetting Mathematical Formulae

The TEX typesetting system has become nearly ubiquitous when it comes to typesetting mathematics. Many software packages even offer to export their mathematical formulae to LATEX (for example, SageMath [81]) or allow entering math expressions in LATEX or LATEX-derived syntax (for example, Wikipedia [84]). In this chapter, you will learn how to insert math formulae into your document.

3.1 Modern Mathematics

While core LATEX is able to produce high quality mathematics output out of the box, there are a few areas where useful features are missing, and some design choices have not aged well over the last 30 years.

The *American Mathematical Society* was very interested in TEX from the earliest days and sponsored the development of several well received enhancements to improve LATEX's math abilities. The enhancements are known as $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX and $\mathcal{A}\mathcal{M}\mathcal{S}$-LATEX.

These days, the $\mathcal{A}\mathcal{M}\mathcal{S}$-LATEX features, and some additional ones, are provided by the mathtools [25] package. It loads the amsmath package internally, adjusts some settings, fixes some behaviours, and adds some additional commands. We recommend loading it together with the unicode-math [61] package for an optimal math-writing experience.

The original TEX used its own font encoding for typesetting mathematics, which is still used by default in LATEX. In recent years, a new standard appeared: OpenType math extensions,[1] which makes it possible to use standardized Unicode characters inside documents. The unicode-math package adds support for these fonts in LATEX, allowing you to use more symbols and makes the commands for accessing math symbols a bit more consistent. The problem with unicode-math, is that some older math packages rely on the classic math font encoding. So if you have to use them, do not use the unicode-math package, to protect your sanity.

Throughout this section, we assume that your preamble contains these two lines in this order:

```
\usepackage{mathtools}
\usepackage[
  math-style=ISO,
  warnings-off={mathtools-colon, mathtools-overbracket},
]{unicode-math}
```

The `math-style=ISO` key fixes a small inconsistency in original TEX regarding the shape of uppercase Greek letters. The `warnings-off` key disables two warnings about overwriting some mathtools commands.[2]

3.2 Single Equations

LATEX distinguishes between two styles of typesetting formulae: *text style* and *display style*. The text style is used to typeset inline math in running text, like this: $\sum_{k=0}^{\infty} \frac{1}{k}$, while the display style is used to typeset bigger equations on their own line, like this:

$$\sum_{k=0}^{\infty} \frac{1}{k}.$$

[1] Its internal structure is actually based on TEX font tables.

[2] These are `\overbracket`, `\underbracket`, `\coloneq`, `\dblcolon`, `\coloneqq`, `\Coloneqq` and `\eqqcolon`. Disable this package option and read these warnings if you are going to use them.

To typeset an inline formula, put it between \(and \) commands, while \[and \] typesets it in display mode.

```
Not like this 3-1=0,
but like this \(3-1=2\)
or this
\[
  3 - 1 = 2
\]
```

Not like this 3-1=0, but like this $3 - 1 = 2$ or this

$$3 - 1 = 2$$

It is often the case that we want to refer to a specific formula or equation in the document. To make this possible, a special `equation` environment exists that typesets the formula in display mode and adds a number to the right. The equations can be labelled and referred to by using the commands described in Section 2.12.

```
\begin{equation}
  \label{trivial}
  2 + 2 = 4
\end{equation}
Equation~\ref{trivial}
is true.
```

$$2 + 2 = 4 \tag{1}$$

Equation 1 is true.

The commands \eqref and \tag make cross-referencing in math context even simpler. The first one is similar to the \ref command but encloses its label in parentheses to match the appearance of the equation label. The second one allows you to customize the visible label of the equation.

```
\begin{equation}
  \tag{Ingsoc's theorem}
  \label{ingsoc}
  2 + 2 = 5
\end{equation}
Equation~\eqref{ingsoc} is false.
```

$$2 + 2 = 5 \tag{Ingsoc's theorem}$$

Equation (Ingsoc's theorem) is false.

The starred version, `equation*`, is a synonym of \[.

3.2.1 Math Mode

The commands and environments mentioned in the previous section activate a special math-optimised version of the LaTeX language. It is commonly known as *math mode*. LaTeX normally operates in *text mode*. There are numerous differences between the two modes. For starters, most of the whitespace is ignored in math mode.

```
\(123xyz\) is the same as \(
  1 2 3 x y z
\).
```

$123xyz$ is the same as $123xyz$.

The spaces around symbols are derived logically from the mathematical expressions. We will talk more about those in Section 3.7. Since there is no concept of a paragraph in math formulae, leaving empty lines inside math mode is *not* allowed. Doing so will result in the "Bad math environment delimiter." error.

Another difference is the fact that all letters are treated as mathematical variables. Variables are printed using an *italic* font, and the spacing around each letter is wider than in text-mode.

```
Compare office to \(office\).
```

Compare office to $office$.

If you want to typeset normal text within a formula, you can use the `\text` command.

```
\( 1 + \text{one} = 2\)
```

$1 + \text{one} = 2$

Finally, additional commands and syntax become available to enable typesetting mathematical notation. For example, the ^ and _ characters can now be used to typeset superscripts and subscripts, respectively.

```
\( a^2 + b^2 = c^2 \) \\
\( A_x = G_{\text{foo}} \)
```

$a^2 + b^2 = c^2$
$A_x = G_{\text{foo}}$

3.3 Building Blocks of Mathematical Formulae

In this section, we describe the most important commands used in mathematical typesetting. The symbols introduced here are only a tiny pick from what is available in LaTeX world. If you need additional symbols, be sure to check out [37, 59, 52] since there is a big chance they already exist.

3.3.1 Basic Arithmetic

You can probably guess how to typeset addition, subtraction, and equality based on the previous examples. To typeset multiplication and division symbols, you can use `\times` and `\div` commands.

```
\( (4 \times 6) \div 3 = 8 \)
```

$(4 \times 6) \div 3 = 8$

If you prefer to use dots for multiplication or slash for division, you can use `\cdot` and `\divslash`, respectively. To typeset a negated equality, you can precede the equal sign with `\not`.

```
\( (5 \cdot 3) \divslash 2
  \not= 7 \)
```

$(5 \cdot 3) \mathbin{/} 2 \neq 7$

Alternatively, you can use `\neq` which will produce the same result. The `\not` notation is more general in that it works with many more symbols.

To typeset weak inequalities, you can use the `\leq` (less or equal) and `\geq` (greater or equal) commands. If you prefer slanted versions, swap them for `\leqslant` and `\geqslant`.

```
\( 1 \leq 2 \) vs.\
\(1 \leqslant 2\)
```

$1 \leq 2$ vs. $1 \leqslant 2$

Exponents are written using the superscript character ^. LaTeX will adapt the size and position of the superscript to the the height of the previous character. This is usually correct, but when parentheses are present the position may be a bit too low.

```
\( a^2 \) is fine, but what
about \( (a^2)^2 \)?
```

a^2 is fine, but what about $(a^2)^2$?

In the example above, the second superscript doesn't know about the first one inside the parenthesis, and it adapts only to the height of the closing parenthesis. To fix this, enclose the parentheses in a group.

```
\( {(a^2)}^2 \) is better.
```

$(a^2)^2$ is better.

In subsection 3.3.7 we will also talk about growing parentheses that do not have this problem.

To enter square roots use, the `\sqrt` command. It will automatically overline the expression it received as the argument.

```
From Homer's theorem it
follows that
\[
  \sqrt{a} + \sqrt{b} = \sqrt{c}
\]
so we can see that
\(c = \sqrt{2x^2+7}\).
```

From Homer's theorem it follows that

$$\sqrt{a} + \sqrt{b} = \sqrt{c}$$

so we can see that $c = \sqrt{2x^2+7}$.

The full syntax of the `\sqrt` is actually

`\sqrt[`⟨*n*⟩`]{`⟨*expr*⟩`}`

The optional argument ⟨*n*⟩ allows you to typeset ⟨*n*⟩-th root radicals.[3]

```
Find three positive integers
\(a, b\) and \(c\), such that
\[
  a = \sqrt[7]{b^7 + c^7}
\]
```

Find three positive integers a, b and c, such that

$$a = \sqrt[7]{b^7 + c^7}$$

[3]The name `\sqrt` is a bit misleading in that respect `;-)`.

Finally, typesetting built-up fractions is made possible by the

\frac{⟨*numerator*⟩}{⟨*denominator*⟩}

command.

```
And since
\[
  \frac{a}{b} \leq
  \frac{a+c}{b+d} \leq
  \frac{c}{d}
\]
it follows that
\(x = \frac{\sqrt{z+3}}{y^3}\).
```

And since

$$\frac{a}{b} \leq \frac{a+c}{b+d} \leq \frac{c}{d}$$

it follows that $x = \frac{\sqrt{z+3}}{y^3}$.

3.3.2 Logic and Set Theory

The basic logical operations can be typeset using rather self-explanatory commands: `\lnot`, `\land`, `\lor`, `\implies` and `\iff`.

```
\[
  p \land q \iff
  \lnot (p \implies \lnot q)
\]
```

$$p \land q \iff \neg(p \implies \neg q)$$

There's also a left facing implication named `\impliedBy`. Some less established logical symbols can also be found under less intuitive names such as `\veebar`, which produces $\veebar$ used by some for exclusive disjunction. If you intend to use them, make sure to use commands described in Section 2.9 to create meaningful synonyms.

The quantifier symbols can be typeset using `\forall` and `\exists` commands.

```
\[
  \lnot\forall_{q} P(q) \iff
  \exists_{q}\lnot P(q)
\]
```

$$\neg\forall_q P(q) \iff \exists_q \neg P(q)$$

To typeset sets, use the `\{` and `\}` commands.

```
\( \{1, 2, 3, \ldots, 100\} \)
```

$\{1, 2, 3, \ldots, 100\}$

To indicate set membership ("is an element of"), you can use the `\in` command.[4] Sets defined by a predicate are often written using set-builder notation with a colon. The naïve way would be to write it like this:

```
\( \{x \in X:
  \exists_n x^n = 1\} \)
```

$$\{x \in X : \exists_n x^n = 1\}$$

The result doesn't look right, though. The reason is the aforementioned automatic spacing based on mathematical meaning. The : character is used in LaTeX for ratios, such as $1 : 2$, which is why it has equal spacing on both of its sides. To typeset colons in set-builder notation, use the `\colon` command instead.

```
\(\{ X \ni x\colon
  \exists_n x^n = 1\} \)
```

$$\{X \ni x\colon \exists_n x^n = 1\}$$

The usual set inclusion symbols are accessible as `\subset`, `\subseteq` and `\subsetneq`, while their flipped versions are accessible as `\supset`, `\supseteq` and `\supsetneq`. The set union symbol is hiding behind `\cup` while set intersection is `\cap`. Set difference is just `\setminus` though. You may also find `\emptyset` useful.

```
% TODO: Waiting for unicode-math fix
\RenewDocumentCommand{\setminus}{}{\smallsetminus}
\[
  \emptyset
  \subset X \setminus X
  \subseteq X \cap \emptyset
  \subsetneq X
  \supseteq X \cup X
\]
```

$$\emptyset \subset X \setminus X \subseteq X \cap \emptyset \subsetneq X \supseteq X \cup X$$

3.3.3 Greek Letters

Every mathematician knows the Latin alphabet just isn't enough. Accessing the Greek alphabet is as easy as typing `\alpha`, `\beta`, `\gamma`, …

[4] Try to guess its left-flipped version name.

for lowercase letters and \Alpha, \Beta, \Gamma, … for uppercase letters.

```
Let \(\epsilon\) be a small
number, and let  \(\Epsilon\)
be a large number.
```

Let ϵ be a small number, and let E be a large number.

As you can see above, in most fonts (including the default LaTeX one) there is no visual difference between some uppercase Greek letters and their Latin equivalents. If you want to use them, make sure they do not clash in your document.

Some Greek letters have defined variants. These are accessible by prepending the word `var` before the letter name; \varepsilon, for example. A list of all defined variants is presented in Table 3.1.

Note that all Greek letters in math mode, the same as all the Latin letters, are considered to be mathematical variables. In particular, \pi *should not* be used to denote the famous constant.

3.3.4 Mathematical Fonts and How To Use Them

When writing mathematics, we tend to use different fonts for mathematical symbols depending on the object they represent. For example, variables are typeset using an italic font, while named sets often get a fancy double-struck (also known as 'blackboard bold') font, for example, $\mathbb{N}$. You can access various fonts by using the family of \sym... commands. For example, to typeset double-struck letters use the \symbb command.

```
\[
  \forall_{p \in \symbb{P}}
  \exists_{n \in \symbb{N}}
  p < 5 \lor p^2 = 24n + 1
\]
```

$$\forall_{p \in \mathbb{P}} \exists_{n \in \mathbb{N}} p < 5 \lor p^2 = 24n + 1$$

All font changing commands are listed in Table 3.2.

Avoid writing these commands directly in your formulas. Instead, you should create logical wrappers around them. For example, the *ISO 80000-2:2019* [30] standard states that mathematical constants should be written in a Roman (upright) font. Writing \symrm{e} every time we

Table 3.1: Available variants of Greek letters.

Command	Variant	Main
`\varepsilon`	ε	ϵ
`\varkappa`	$\varkappa$	κ
`\varphi`	φ	ϕ
`\varpi`	ϖ	π
`\varrho`	ϱ	ρ
`\varsigma`	ς	σ
`\vartheta`	ϑ	θ
`\varTheta`	$\varTheta$	Θ

want to refer to the base of the natural logarithm will make the code less readable, so we create simple wrappers for the constants we are going to use.

```
\NewDocumentCommand{\e}{}{\symrm{e}}
\NewDocumentCommand{\im}{}{\symrm{i}}
\NewDocumentCommand{\cpi}{}{\symrm{\pi}}
\[ \e^{\im\cpi} + 1 = 0 \]
```

$$\mathrm{e}^{\mathrm{i}\pi} + 1 = 0$$

Functions such as sin or log should be written in Roman style, however you should not use the `\symrm` command to do so. As the name of the command indicates, it is meant to be used for mathematical symbols. Since in "sin" s, i and n are not symbols, but components of a single operator name, there are separate commands for typesetting these. Fortunately you do not have to access them directly since many of the standard functions are already defined.

```
\( \cos(2\cpi) = \ln(\e) \)
```

$$\cos(2\pi) = \ln(\mathrm{e})$$

The full list of predefined functions is presented in Table 3.3.

Table 3.2: Commands that change the font of mathematical symbols. An empty example field indicates that the Unicode does not define glyphs in the given set.

	Example		
Command	Latin	Greek	Numerals
`\symup`	ABCabc ...	ABΓαβγ ...	123 ...
`\symbfup`	**ABCabc** ...	**ABΓαβγ** ...	**123** ...
`\symit`	*ABCabc* ...	*ABΓαβγ* ...	
`\symbfit`	***ABCabc*** ...	***ABΓαβγ*** ...	
`\symsfup`	ABCabc ...		123 ...
`\symbfsfup`	**ABCabc** ...	**ABΓαβγ** ...	**123** ...
`\symsfit`	*ABCabc* ...		
`\symbfsfit`	***ABCabc*** ...	***ABΓαβγ*** ...	
`\symtt`	ABCabc ...		123 ...
`\symbb`[a]	𝔸𝔹ℂ𝕒𝕓𝕔 ...	ℾℿ𝛄𝛑	𝟙𝟚𝟛 ...
`\symbbit`[b]	𝔻𝕕𝕖𝕚𝕛		
`\symscr`[c]	𝒜ℬ𝒞𝒶𝒷𝒸 ...		
`\symbfscr`[d]	𝓐𝓑𝓒𝓪𝓫𝓬 ...		
`\symfrak`	𝔄𝔅ℭ𝔞𝔟𝔠 ...		
`\symbffrak`	𝖆𝖇𝖈 ...		

[a] In the Greek set, only the four presented glyphs (`\Gamma`, `\Pi`, `\gamma` and `\pi`) are defined by the Unicode standard.

[b] Only the five presented glyphs (`D`, `d`, `e`, `i` and `j`) are defined by the Unicode standard.

[c] The default font does not contain lowercase glyphs. Here they are shown using a different font.

[d] Same as above.

Table 3.3: All functions predefined by LaTeX.

\arccos	\cos	\csc	\exp	\ker	\limsup
\arcsin	\cosh	\deg	\gcd	\lg	\ln
\arctan	\cot	\det	\hom	\lim	\log
\arg	\coth	\dim	\inf	\liminf	\max
\sinh	\sup	\tan	\tanh	\min	\Pr
\sec	\sin				

If these are not enough for you, use the \DeclareMathOperator command to define your own. It can be only used in the preamble.

```
% In preamble
\DeclareMathOperator{\argh}{argh}
% ...
\[ \argh(x) = \sinh(\max(x, x^2)) \]
```

$$\operatorname{argh}(x) = \sinh(\max(x, x^2))$$

3.3.5 Big Operators

At the beginning of the Section 3.2, we used the summation operator to show off some capabilities of mathematical typesetting. To typeset it, use the \sum command, its lower and upper limits are specified by the sub- and superscript operators.

```
\(
  \sum_{k=0}^{n} \frac{1}{k^2}
\)
```

$\sum_{k=0}^{n} \frac{1}{k^2}$

The appearance in text style differs massively from the one in display style.

```
\[
  \sum_{k=0}^{n} \frac{1}{k^2}
\]
```

$$\sum_{k=0}^{n} \frac{1}{k^2}$$

This is an example of what's commonly known as big operator. There are other such operators, such as `\prod` for products, or `\bigwedge` for conjunctions. Some functions listed in Table 3.3 are also big operators.

```
\[
  c = \max_{x\in X} f(x)
\]
```

$$c = \max_{x\in X} f(x)$$

You can define your own big operators by using a starred version of the `\DeclareMathOperator` command.

```
% In preamble
\DeclareMathOperator*{\nut}{Nut}
% ...
\[ \nut_y = \lim_{x \to y} \argh{x} \]
```

$$\operatorname*{Nut}_{y} = \lim_{x\to y} \operatorname{argh} x$$

If you have a long limit, it may make sense to typeset it vertically instead of horizontally. This can be done using the `\substack` command. Inside, you can place `\\` to indicate where new lines should be started.

```
\[
  \sum^n_{
    \substack{0<i<n \\
    j\subseteq i}
  }
   P(i,j) = Q(i,j)
\]
```

$$\sum^{n}_{\substack{0<i<n \\ j\subseteq i}} P(i,j) = Q(i,j)$$

The integral symbol is also a big operator, but it only shifts the lower limit left.

```
\[
  \int_a^b \liminf_{a \to x} a^2
\]
```

$$\int_a^b \liminf_{a\to x} a^2$$

Due to the fact that the symbol is highly slanted to the right, typesetting multiple integrals leads to excessive space around them. Use `\iint`,`\iiint`,`\iiiint` instead for improved spacing.

```
\[ \int\int\int\int \]
vs.\
\[ \iiiint \]
```

$$\int\int\int\int$$

vs.

$$\iiiint$$

If you want to suppress the big operators' limit-placement-behaviour and typeset normal, sub-, or superscripted text next to them, you can simply surround them with curly braces.

```
\[
  {\sum_{a}}^b
  {\int}_a^b
  {\nut^b}_a
\]
```

$$\sum_{a}^{b} \qquad \int_{a}^{b} \qquad \overset{b}{\mathrm{Nut}}_{a}$$

However, if a long limit is present this may lead to excessive spacing before the sub- or superscripts.

```
\[ {\sum_{looooooong}}^{b} \]
```

$$\sum_{looooooong}^{b}$$

To fix the issue, use the

`\sideset{`⟨*left*⟩`}{`⟨*right*⟩`}{`⟨*symbol*⟩`}`

command. It surrounds the ⟨*symbol*⟩ with left and right commands before typesetting its limits.

```
\[ \sideset{^{a}}{^{b}}{\sum}_{looooooong} \]
```

$$\sideset{^{a}}{^{b}}{\sum}_{looooooong}$$

3.3.6 Math Accents

To denote an arithmetic mean taken over a variable, a notation with a bar over the variable is used: $\bar{x}$. If you remember Table 2.1 from page 31, you may think it would be possible to write it as `\=x`. This will not work though. In fact, none of the commands from this table will work in math mode. There is a good conceptual reason for this—the accents listed there are meant to typeset non-Latin letters that are used in various languages. The character `\"u` is meant to represent u-umlaut and the correct glyph for it will be chosen if it exists in the current font. In math, we don't want an u-umlaut—we want the "double dot above" operator applied to variable u. Thus, different commands are used to achieve this.

To actually get a bar over a variable, you can use the `\bar` command. Other accents include `\hat`, `\grave`, `\acute`, `\tilde` or `\ddot`. Refer to [59] for a full list.

```
\(\bar{x}\) \(\hat{x}\)
\(\grave{x}\) \(\acute{x}\)
\(\tilde{x}\) \(\ddot{x}\)
```

$\bar{x}$ $\hat{x}$ $\grave{x}$ $\acute{x}$ $\tilde{x}$ $\ddot{x}$

Some accents also exist in a 'wide' version. These can encompass more than one character. Compare `\hat` to `\widehat` in the example below.

```
\(\hat{ABC}\) vs.\
\(\widehat{ABC}\)
```

$\hat{ABC}$ vs. $\widehat{ABC}$

The wide version of the `\bar` accent is called `\overline`. It is useful if you want to mark the repetend of a decimal fraction.

```
\(0.123\overline{456789}\)
```

$0.123\overline{456789}$

Some wide accents even have limits. For example, the `\overbrace` and `\underbrace` commands enable you to create horizontal braces with

an expression above or below them.

```
\[
  \underbrace{
    \overbrace{(a+b+c)}^6
    \times
    \overbrace{(d+e+f)}^7
  }_{\text{meaning of life}}=42
\]
```

$$\underbrace{\overbrace{(a+b+c)}^{6} \times \overbrace{(d+e+f)}^{7}}_{\text{meaning of life}} = 42$$

You can even create your own math-accents using `\overset` and `\underset`.

```
\(\overset{\text{foo}}{x}\)
\(\underset{\times}{A}\)
```

$\overset{\text{foo}}{x}$ $\underset{\times}{A}$

These commands may also be used when creating custom binary relations. Make sure that you do not overwrite existing commands!

```
\(a \overset{\text{def}}{=} b\)
vs.\ \(a \eqdef b\)
```

$a \overset{\text{def}}{=} b$ vs. $a \eqdef b$

3.3.7 Delimiters

Besides the standard parentheses, brackets and braces, LaTeX provides all sorts of symbols for delimiters, such as:

```
\[
  \langle x \rangle
  \lBrack x \rBrack
  \lfloor x \rfloor
  \lvert x \rvert
\]
```

$$\langle x \rangle \; [\![x]\!] \; \lfloor x \rfloor \; |x|$$

While all of them work fine for bungalow-style math, things get ugly

when the expression inside them starts getting tall.

```
\[
  {\lfloor
    \frac
    {{\langle\sum_{k=1}^{\infty} k^{-4}\rangle}^3}
    {\frac{{\lvert\{a,b,c\}\rvert}^8}{{(x^4)}^3}}
  \rfloor}^6
\]
```

$$\left\lfloor \frac{\left\langle \sum_{k=1}^{\infty} k^{-4} \right\rangle^3}{\frac{|\{a,b,c\}|^8}{(x^4)^3}} \right\rfloor^6$$

To fix this, you can use the `\left` and `\right` commands. They take one argument—the delimiter to extend—and adapt its size based on the content they delimit. You must close every `\left` with a corresponding `\right`, but they do not have to use the same delimiter.

```
\[
  \left(
    \frac{1}{1 +
      \frac{1}{1 +
        \frac{1}{1 +
          \frac{1}{1 +
            \sqrt{2}}}}}}
  \right]
\]
```

$$\left(\frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \sqrt{2}}}}} \right]$$

If you want to typeset a delimiter only on one side of the expression, a special value may be passed: the dot character `.`, which is treated as an invisible delimiter.

```
\[
  x = \left.
    \sum_{k=1}^{\infty}
      \frac{1}{k^2}
  \right\}
\]
```

$$x = \left. \sum_{k=1}^{\infty} \frac{1}{k^2} \right\}$$

While using the `\left` and `\right` commands directly is sometimes necessary, the simplest case of using some predefined delimiters on both sides is the most common. For this usage, the command

`\DeclarePairedDelimiter{\`⟨*name*⟩`}{`⟨*left delim*⟩`}{`⟨*right delim*⟩`}`

is really useful. It declares the ⟨*name*⟩ command which simply encloses its argument between ⟨*left delim*⟩ and ⟨*right delim*⟩, while its starred version additionally uses `\left` and `\right` commands when typesetting delimiters.

```
\DeclarePairedDelimiter{\set}{\{}{\}}
\DeclarePairedDelimiter{\size}{\lvert}{\rvert}
\DeclarePairedDelimiter{\mean}{\langle}{\rangle}
\DeclarePairedDelimiter{\floor}{\lfloor}{\rfloor}
\DeclarePairedDelimiter{\group}{(}{)}
\[
  \floor*{
    \frac
    {\mean*{\sum_{k=1}^{\infty} k^{-4}}^3}
    {\frac{\size*{\set{a,b,c}}^8}{\group*{x^4}^3}}
  }^6
\]
```

$$\left\lfloor \frac{\left\langle \sum_{k=1}^{\infty} k^{-4} \right\rangle^3}{\frac{|\{a,b,c\}|^8}{(x^4)^3}} \right\rfloor^6$$

In some special circumstances, you may also want to select the size of the delimiter yourself. You can do this by using the `\big`, `\Big`, `\bigg` and `\Bigg` commands. These may be either prepended to a delimiting symbol with "l" and "r" suffixes, or passed as an optional argument to a declared delimiter.

```
\DeclarePairedDelimiter{%
  \set}{\{}{\}}
\( \bigl[ \ddagger \Biggr) \)
\( \set[\Big]{1,\frac{3}{4}} \)
```

$$\left[\ddagger\right) \qquad \left\{1, \frac{3}{4}\right\}$$

If the \DeclarePairedDelimiter is not enough for you, there also exists a more powerful \DeclarePairedDelimiterX command. It allows you to define delimiters that are composed of more than one part (such as bra-kets) and manipulate their arguments. For a full description, check out the documentation of mathtools [25] package.

3.4 Multiline Equations

The math environments discussed so far only allow typesetting a single line equation. However, sometimes an equation might be too long to fit.

```
\begin{equation}
  a = b + c + d + e + f
  + g + h + i + j
  + k + l + m + n + o + p
\end{equation}
```

$$a = b+c+d+e+f+g+h+i+j+k+l \tag{1}$$

In this case, it is necessary to introduce line breaks inside the equation. When doing so, it is important to remember a few rules to improve the readability:

1. In general one should always break an equation *before* an equality sign or operator.

2. A break before an equality sign is preferable to a break before any operator.

3. A break before a plus- or minus-operator is preferable to a break before a multiplication-operator.

4. Any other type of break should be avoided if at all possible.

This section will introduce several environments to typeset equations with linebreaks inside them. These and more are described in more detail in the amsmath [55] package documentation.

3.4.1 Long Equations

The easiest way to display long equations is the `multline` environment. It allows introducing line breaks with the `\\` command.

```
\begin{multline}
  a + b + c + d + e
  + f + g + h + i \\
  = j + k + l + m + n
\end{multline}
```

$$\begin{multline} a + b + c + d + e \\ + f + g + h + i \\ = j + k + l + m + n \end{multline}$$

The first line in `multline` environment is aligned to the left, and the last one to the right, while all the others are centred.

```
\begin{multline}
  a + b + c + d + e \\
  + f + g + h + i \\
  = j + k + l + m + n
\end{multline}
```

$$\begin{multline} a + b + c + d + e \\ + f + g + h + i \\ = j + k + l + m + n \end{multline}$$

If you do not want some particular inner line centred, you can use `\shoveleft` and `\shoveright` commands to force the line to be left or right aligned.

```
\begin{multline}
  a + b + c \\
  \shoveleft{+ d + e + f} \\
  \shoveright{+ g + h + i} \\
  = j + k + l + m + n
\end{multline}
```

$$\begin{multline} a + b + c \\ \shoveleft{+ d + e + f} \\ \shoveright{+ g + h + i} \\ = j + k + l + m + n \end{multline}$$

As with `equation*`, there also exists a starred `multline*` version that suppresses the equation number.

3.4.2 Multiple Unaligned Equations

When typsetting multiple equations within several `equation` environments, unneeded spacing appears between them.

```
\begin{equation}
  2 + 2 = 4
\end{equation}
\begin{equation}
  2 \times 2 = 4
\end{equation}
\begin{equation}
  2 + 2 \times 2 = 6
\end{equation}
```

$$2 + 2 = 4 \tag{1}$$
$$2 \times 2 = 4 \tag{2}$$
$$2 + 2 \times 2 = 6 \tag{3}$$

Use the `gather` environment to eliminate the extra space. It allows you to break lines using the `\\` command, and centres each equation.

```
\begin{gather}
  2 + 2 = 4 \\
  2 \times 2 = 4 \\
  2 + 2 \times 2 = 6
\end{gather}
```

$$\begin{gather}
2 + 2 = 4 \\
2 \times 2 = 4 \\
2 + 2 \times 2 = 6
\end{gather}$$

The equation numbering commands, such as `\eqref`, `\label` and `\tag`, now apply to the line they are present in. Additionally the command `\notag` allows you to suppress equation numbering for a particular line.

```
\begin{gather}
  2 + 2 = 4
    \tag{Easy}\label{easy} \\
  2 \times 2 = 4 \notag \\
  2 + 2 \times 2 = 6
    \label{hard}
\end{gather}
Note:~\eqref{easy} is easier
than~\eqref{hard}.
```

$$\begin{gather}
2 + 2 = 4 \tag{Easy}\label{easy} \\
2 \times 2 = 4 \notag \\
2 + 2 \times 2 = 6 \tag{1}\label{hard}
\end{gather}$$

Note: (Easy) is easier than (1).

The above example also illustrates that the centring of the equations inside the environment may depend upon the length of the tag. Keep this in mind if your tags and/or equations get long.

As always, the starred version of the environment, `gather*`, suppresses all equation numbers.

3.4.3 Multiple Aligned Equations

If the gathered equations have a natural midpoint, it may be the case that the `align` environment is much better suited to typeset them. It allows you to align the equations by inserting the `&` character, the same as in the `tabular` environment. Note that the `&` should come *before* any binary operator, in order to produce the correct spacing.

```
\begin{align}
  2 + 2 & = 2 \times 2 \\
  3 + 3 & \neq 3 \times 3 \\
  2 + 2 \times 2 & < 8
\end{align}
```

$$
\begin{aligned}
2 + 2 &= 2 \times 2 && (1)\\
3 + 3 &\neq 3 \times 3 && (2)\\
2 + 2 \times 2 &< 8 && (3)
\end{aligned}
$$

The `align` environment allows placing multiple bits of math on a single line by using `&` as a separator. The bits are considered to be paired, so every second alignment character will produce a bigger space to accommodate for the spacing between columns.

```
\begin{align}
  a & \succeq b & c & \leq d \\
  a & \geq d & d & \prec c
\end{align}
```

$$
\begin{aligned}
a &\succeq b & c &\leq d && (1)\\
a &\geq d & d &\prec c && (2)
\end{aligned}
$$

The `align` environment is also really useful when writing transformations of a single equation over multiple lines.

```
\begin{align}
  \sum_{k=0}^n k
    &= \sum_{k=0}^{n-1} k + n \\
    &= \frac{n(n-1)}{2} + n \\
    &= \frac{n(n+1)}{2}
\end{align}
```

$$
\begin{aligned}
\sum_{k=0}^{n} k &= \sum_{k=0}^{n-1} k + n && (1)\\
&= \frac{n(n-1)}{2} + n && (2)\\
&= \frac{n(n+1)}{2} && (3)
\end{aligned}
$$

We can even use additional columns to add line-by-line comments with the \text command.

```
\begin{align*}
  \sum_{k=0}^n k
    &= \sum_{k=0}^{n-1} k + n
      && \text{definition} \\
    &= \frac{n(n - 1)}{2} + n
      && \text{induction} \\
    &= \frac{n(n + 1)}{2}
      && \text{trivial}
\end{align*}
```

$$\begin{aligned}
\sum_{k=0}^{n} k &= \sum_{k=0}^{n-1} k + n && \text{definition} \\
&= \frac{n(n-1)}{2} + n && \text{induction} \\
&= \frac{n(n+1)}{2} && \text{trivial}
\end{aligned}$$

The \phantom command from subsection 3.7.3 allows us to correctly align equation systems.

```
\begin{align}
  a + b + c
    &= d + e + f \nonumber \\
    & \phantom{=}\ + g + h
                \nonumber \\
    & \phantom{=}\ + i \\
    &= j + k + l
\end{align}
```

$$\begin{aligned}
a + b + c &= d + e + f \\
&\ + g + h \\
&\ + i && (1) \\
&= j + k + l && (2)
\end{aligned}$$

LaTeX encloses relation symbols in thick spaces \; and binary symbols in medium spaces \:. The symbols = and + in our example are each of this type. The \phantom command strips = of its relation status, leaving it short of two thick spaces. On the other hand, LaTeX interprets + following \phantom as a binary symbol, creating a spurious medium space. So we need an extra normal space: \␣ = \; + \; - \:.

If this aproach seems a bit complicated to you, have a look at subsection 3.4.5.

3.4.4 Equations as Building Blocks

All the environments presented above can be used inside other equations by appending "-ed" to their name. They no longer number the equations inside them, and accept an optional vertical positioning argument, with

the same semantics as those accepted by tabular (see Section 2.18 on page 63).

```
\[
  \begin{multlined}[t]
    1 + 2 \\
    + 3 + 4
  \end{multlined} =
  \begin{gathered}[c]
    1 + 2 \\
    + 3 + 4
  \end{gathered} =
  \begin{aligned}[b]
    1 & + 2 \\
      & + 3 + 4
  \end{aligned}
\]
```

$$\begin{multlined}[t] 1 + 2 \\ + 3 + 4 \end{multlined} = \begin{gathered}[c] 1 + 2 \\ + 3 + 4 \end{gathered} = \begin{aligned}[b] 1 & + 2 \\ & + 3 + 4 \end{aligned}$$

These are useful if you want to, for example, break an equation into multiple lines with alignment, but still retain only a single equation number.

```
\begin{equation}
  \begin{aligned}
    2 + 2 \times 2
      &= 2 \times 2 + 2 \\
      &\neq 2 \times (2 + 2) \\
  \end{aligned}
\end{equation}
```

$$\begin{aligned} 2 + 2 \times 2 &= 2 \times 2 + 2 \\ &\neq 2 \times (2 + 2) \end{aligned} \tag{1}$$

They can be also freely nested, if the need arises.

```
\begin{align}
  a + b + c
    &= \begin{multlined}[t]
      d + e \\
      + e + f \\
      + h
    \end{multlined} \\
    &= \begin{aligned}[t]
      i &+ j \\
        &- k \\
        &\div m \\
        &\times n
    \end{aligned} \\
    &= o + p + q
\end{align}
```

$$
\begin{aligned}
a + b + c &= \begin{multlined}[t] d + e \\ + e + f \\ + h \end{multlined} && (1) \\
&= \begin{aligned}[t] i &+ j \\ &- k \\ &\div m \\ &\times n \end{aligned} && (2) \\
&= o + p + q && (3)
\end{aligned}
$$

3.4.5 IEEEeqnarray Environment

If the environments defined in `amsmath` do not meet your needs, you can try the `IEEEeqnarray` from the IEEEtrantools [69] package. It allows you to specify the alignment of each math column separately, like in the `tabular` environment. Here we will present some of its basic functionalities. More information about `IEEEeqnarray` can be found in Appendix F of [68].

To specify column alignments, use `l`, `c` and `r`. Their uppercase version also adds a small space around the column, which is useful in case of binary operators. For example, to emulate the `align` environment we could use `rCl`, i.e., three columns: the first column right-justified, the middle one centred with a little more space around it, and the third column left-justified.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c \\
  & = & d + e + f + g + h \\
  & = & i + j + k
\end{IEEEeqnarray}
```

$$
\begin{IEEEeqnarray}{rCl}
a & = & b + c && (1) \\
& = & d + e + f + g + h && (2) \\
& = & i + j + k && (3)
\end{IEEEeqnarray}
$$

In contrast to the amsmath environments, `IEEEeqnarray` does not

try to avoid collisions between the equation and its number.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c \\
  & = & d + e + f
    + g + h + i + j \\
  & = & k + l + m + n.
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c && (1)\\ &= d + e + f + g + h + i + j && (2)\\ &= k + l + m + n. && (3) \end{aligned}$$

To avoid this, use the `\IEEEeqnarraynumspace` on the offending line. It will shift the whole equation left to accommodate the number present on the given line.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c  \\
  & = & d + e + f
    + g + h + i + j
    \IEEEeqnarraynumspace \\
  & = & k + l + m + n.
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c && (1)\\ &= d + e + f + g + h + i + j && (2)\\ &= k + l + m + n. && (3) \end{aligned}$$

You can also use `\IEEEeqnarraymulticol` to adjust only a single line within the environment, with its usage being similar to that of the `\multicol` environment.

```
\begin{IEEEeqnarray}{rCl}
  \IEEEeqnarraymulticol{3}{l}{
    a + b + c + d = e + f
  } \\
  e + f & = & g + h \\
  & = & i + j + k + l + m
\end{IEEEeqnarray}
```

$$\begin{aligned} a + b + c + d &= e + f && (1)\\ e + f &= g + h && (2)\\ &= i + j + k + l + m && (3) \end{aligned}$$

If a particular line should not have an equation number, the number can be suppressed using `\notag`, as discussed before. Additionally, IEEEtrantools defines `\IEEEyesnumber` and `\IEEEyessubnumber`. The

former command can be used to turn on numbering within the starred version of an environment.

```
\begin{IEEEeqnarray*}{rCl}
  a & = & b + c \\
  & = & d + e \IEEEyesnumber\\
  & = & f + g
\end{IEEEeqnarray*}
```

$$\begin{aligned} a &= b + c \\ &= d + e \qquad (1) \\ &= f + g \end{aligned}$$

The latter changes the numbering to indicate sub expressions.

```
\begin{IEEEeqnarray}{rCl}
  a & = & b + c \\
  & = & d + e \notag \\
  & = & f + g
    \IEEEyessubnumber
\end{IEEEeqnarray}
```

$$\begin{aligned} a &= b + c && (1) \\ &= d + e \\ &= f + g && (1a) \end{aligned}$$

Note that `\label` commands should eventually follow after these.

Here is an alternative approach to the `\phantom` example above. Notice the extra `\negmedspace` required to properly separate the plus sign from the variable name. The `\negmedspace{}` command adds a medium negative space, which cancels the space that LaTeX inserts between variable and operator in a formula like $a + b$, while playing the role of a variable when LaTeX decides whether the next symbol is a relation symbol or a unary operator.

```
\begin{IEEEeqnarray*}{rCl}
a & = & b + c + d \\
& & + e \quad \text{too narrow!}\\
& & \negmedspace{} + e \\
& = & f + 2e
\end{IEEEeqnarray*}
```

$$\begin{aligned} a &= b + c + d \\ &\quad + e \quad \text{too narrow!} \\ &\quad + e \\ &= f + 2e \end{aligned}$$

3.5 Units

When dealing with real world data, you will often find yourself writing units such as 10 kg or 25 C mol^{-1}. In technical writing, it is crucial to accurately convey values to avoid errors and ambiguity. The *International System of Units* (*SI*) [54], comes with a detailed set of typesetting rules. For example, you may have noticed that in the first sentence the space between 10 and kg is smaller than between the words.

Fortunately, thanks to the logical nature of LaTeX markup and the excellent siunitx [86] package, you do not need to know most of the rules. Just use the special unit-commands provided by the package, and all the units in your document will be typeset correctly.

3.5.1 Pitfalls of Naïvely Entered Units

Naïvely, one might write units like this:

```
The speed of light
is exactly 299792458 m/s.
```

The speed of light is exactly 299792458 m/s.

This may look fine initially, but if your document gets longer you will probably get into a situation where an unfortunate line break occurs.

```
\ldots{} The speed of light
is 299792458 m/s. Thus, we
can calculate \ldots
```

… The speed of light is 299792458 m/s. Thus, we can calculate …

A non-breaking space will fix this problem, but the next one is already waiting—the space between numbers. Digits of long numbers should be grouped to make them easier to read. The above example should then be written like this:

```
The speed of light is
299~792~458~m/s.
```

The speed of light is 299 792 458 m/s.

Yet another challenge arises when trying to write the units inside math mode

```
\(10~\frac{m}{s}
  \times 12~s = 120~m\)
```

$10\ \frac{m}{s} \times 12\ s = 120\ m$

Here we need to make sure that the units are written in roman font. The above example should be then written as

```
\(10~\frac{\textrm{m}}
  {\textrm{s}}
  \times 12~\textrm{s}
  = 120~\textrm{m}\)
```

$10\ \frac{\mathrm{m}}{\mathrm{s}} \times 12\ \mathrm{s} = 120\ \mathrm{m}$

Hopefully, these examples illustrate why a better method is needed.

3.5.2 Basic Commands of the siunitx Package

The basic commands of siunitx package are

```
\num[⟨options⟩]{⟨number⟩}
\unit[⟨options⟩]{⟨unit⟩}
\qty[⟨options⟩]{⟨number⟩}{⟨unit⟩}
```

The `\num` command typesets a number, `\unit` typesets a unit, and `\qty` typesets a quantity, i.e. a number followed by a unit. All the commands feature an optional ⟨*options*⟩ argument, which is a comma delimited key-value pair list that can influence the output of the command. The ⟨*unit*⟩ argument can be provided in one of two styles, either "literal" or "interpreted".

In literal mode units are entered as strings of letters

```
\unit{kg} is a unit,
\qty{10}{m/s} is a quantity
```

kg is a unit, 10 m/s is a quantity

Spaces in literal mode are ignored. If you want to insert a product of units, use either dot `.` or tilde `~`.

```
\unit{N} is
not \unit{kg m / s^2},
but \unit{kg.m / s^2}
or \unit{kg~m / s^2}
```

N is not kgm/s^2, but kg m/s^2 or kg m/s^2

In interpreted mode, on the other hand, units are entered using predefined macros:

```
\unit{\kilo\gram} is a unit,
\qty{10}{\metre\per\second}
is a quantity
```

kg is a unit, $10\,\mathrm{m\,s^{-1}}$ is a quantity

At first glance this may seem less convenient, but this enables the package to recognize the logical structure of the units, which means that the formatting can be changed on the fly.

```
Which one do you prefer:
\(\unit{\newton} = \unit{
  \kilo\gram\metre\per\square\second}\),
\(\unit{\newton} = \unit[per-mode = fraction]{
  \kilo\gram\metre\per\second\squared}\) or
\(\unit{\N} = \unit[per-mode = symbol]{
  \kg\m\per\s\tothe{2}}\)?
```

Which one do you prefer: $\mathrm{N} = \mathrm{kg\,m\,s^{-2}}$, $\mathrm{N} = \frac{\mathrm{kg\,m}}{\mathrm{s^2}}$ or $\mathrm{N} = \mathrm{kg\,m/s^2}$?

The commands `\gram`, `\kilo`, `\per` are rather self-explanatory. As you can see, we have also used the `per-mode` option to influence the unit style. While the command `\tothe` typesets arbitrary superscripts, the command `\of` typesets arbitrary subscripts, known as qualifiers.

```
\unit{\kg\of{mol}},
\unit{\bel\of{i}}
```

$\mathrm{kg_{mol}}$, $\mathrm{B_i}$

Note that the `\per` command only works on the next unit by default. If we want it to apply to all following units, we may use `sticky-per` option.

```
The unit of thermal conductivity is
\unit{\watt\per\metre\per\kelvin}
or equivalently
\unit[sticky-per]{\m\kg\per\s\cubed\K}.
```

The unit of thermal conductivity is $\mathrm{W\,m^{-1}\,K^{-1}}$ or equivalently $\mathrm{m\,kg\,s^{-3}\,K^{-1}}$.

Sometimes, you may want to make the units use a different colour, so they figure more prominently. This can be done using `unit-color`. Options `number-color` and `color` may be used to colour only the numeric values or whole quantities. If you want to highlight a single unit in a compound unit, you may use the `\highlight` command.

```
\qty[unit-color = red   ]{9.81}{\m\per\s\squared},
\qty[number-color = blue]{9.81}{\m\per\s\squared},
\qty[color = green      ]{9.81}{\m\per\s\squared},
\qty{9.81}{\m\per\highlight{orange}\s\squared}
```

$9.81\,\mathrm{m\,s^{-2}}$, $9.81\,\mathrm{m\,s^{-2}}$, $9.81\,\mathrm{m\,s^{-2}}$, $9.81\,\mathrm{m\,s^{-2}}$

Now let's turn our attention to numbers. Numeric values will always use dot as a decimal separator, regardless of the symbol entered. If you want to use comma as the decimal separator, change the `output-decimal-marker` option.

```
\(\cpi \approx \num{3.14159}\)
\(\cpi \approx \num{3,14159}\)
\(\cpi \approx \num[
  output-decimal-marker={,}
]{3.14159}\)
```

$\pi \approx 3.141\,59$
$\pi \approx 3.141\,59$
$\pi \approx 3{,}141\,59$

Numbers may also be entered using exponent notation. The symbols before the exponent and base may be controlled using `exponent-product` and `exponent-base` options

```
\(\qty{1}{\tonne}
  = \qty{1e6}{\g}\) \\
\(\qty{1}{\gibi\byte} = \qty[
  exponent-base = 2,
]{1e30}{\byte}\) \\
\(\qty{1}{\L} = \qty[
  exponent-product = \cdot,
]{1e-3}{\cubic\m}\)
```

$1\,\mathrm{t} = 1 \times 10^{6}\,\mathrm{g}$
$1\,\mathrm{GiB} = 1 \times 2^{30}\,\mathrm{B}$
$1\,\mathrm{L} = 1 \cdot 10^{-3}\,\mathrm{m^3}$

If we want all numbers to be formatted in exponent notation, regardless of their entered form, we may use the `exponent-mode` option.

```
\(2^{16} = \num{65536}\) \\
\(2^{16} = \num[
  exponent-mode = scientific,
]{65536}\) \\
\(2^{16} = \num[
  exponent-mode = engineering,
]{65536}\) \\
\(2^{16} = \num[
  exponent-mode = fixed,
]{65.536e3}\) \\
\(2^{16} = \num[
  exponent-mode = fixed,
  fixed-exponent = 5,
]{65.536e3}\)
```

$2^{16} = 65\,536$
$2^{16} = 6.5536 \times 10^{4}$
$2^{16} = 65.536 \times 10^{3}$
$2^{16} = 65\,536$
$2^{16} = 0.655\,36 \times 10^{5}$

In some fields, it may be common to write an uncertainty next to the numbers. This may be done in several ways. In order to customize the appearance of the uncertainties, use the `uncertainty-mode` option.

```
\(M_{\oplus} = \qty{
  5.9722(6) e24}{\kg}\)
\(M_{\oplus} = \qty{
  5.9722 +- 0.0006 e24}{\kg}\)
\(M_{\oplus} = \qty{
  5.9722 \pm 0.0006 e24}{\kg}\)
\(M_{\oplus} = \qty[
    uncertainty-mode = separate
  ]{5.9722(6) e24}{\kg}\)
\(M_{\oplus} = \qty[
  uncertainty-mode = full
]{5.9722(6) e24}{\kg}\)
```

$M_{\oplus} = 5.9722(6) \times 10^{24}\,\mathrm{kg}$
$M_{\oplus} = 5.9722(6) \times 10^{24}\,\mathrm{kg}$
$M_{\oplus} = 5.9722(6) \times 10^{24}\,\mathrm{kg}$
$M_{\oplus} = (5.9722 \pm 0.0006) \times 10^{24}\,\mathrm{kg}$
$M_{\oplus} = 5.9722(0.0006) \times 10^{24}\,\mathrm{kg}$

By default, siunitx typesets the quantities in math mode. This may be especially jarring if you are using a different font for the document

text. One simple solution is to set the `mode` option to `text`.

```
\qty{1}{\degreeCelsius} is
equal to \qty{1}{\kelvin},
but \qty[
  mode = text,
]{0}{\degreeCelsius} is \qty[
  mode = text,
]{-273.15}{\kelvin}.
Bizarre, isn't it?
```

1 °C is equal to 1 K, but 0 °C is −273.15 K. Bizarre, isn't it?

If your document contains both math and normal text, it is probably better to use a matching math font. You will learn more about this in subsection 7.3.4.

The behaviour of the commands when it comes to typesetting units and quantities may be customized even further. For a full list of options check out the siunitx package documentation.

To avoid repeating the options with every command, use the

`\sisetup{⟨options⟩}`

command. It can be used in the preamble to ensure consistent style throughout the document, or inside the document when the style needs to be changed just in a given element, such as a table. These options may be also passed as package options. Listing 3.1 shows an example usage of the package in a full document.

3.5.3 Other siunitx Commands

Besides the basic commands described above, siunitx also defines several other useful commands.

Angles are special in that they use base 60 instead of base 10 when subdividing. The command

`\ang[⟨options⟩]{⟨angle⟩}`

exists in order to simplify entering them. If they are entered using a semicolon as a separator, then the following numbers are treated as

```latex
\documentclass{article}

\usepackage{booktabs}
\usepackage[per-mode = fraction, unit-color = red]{siunitx}

\begin{document}
Units are typeset like this: \qty{9.81}{\m\per\s\squared}.
\begin{table}[h]
  \centering
  \sisetup{exponent-mode = fixed, fixed-exponent = 1}
  \begin{tabular}{@{}lr@{}}
    \toprule
    No. & Result (\unit{\kg}) \\
    \midrule
    1   & \num{12.3}          \\
    2   & \num{0.3}           \\
    3   & \num{1e2}           \\
    \bottomrule
  \end{tabular}
  \caption{Measurement results.
    Here custom options are used.}\label{table}
\end{table}

Now we are back to normal: \qty{3.637e-4}{\m\squared\per\s}.
\end{document}
```

Units are typeset like this: $9.81\ \frac{\mathrm{m}}{\mathrm{s}^2}$.

No.	Result (kg)
1	1.23×10^1
2	0.03×10^1
3	10×10^1

Table 1: Measurement results. Here custom options are used.

Now we are back to normal: $3.637 \times 10^{-4}\ \frac{\mathrm{m}^2}{\mathrm{s}}$.

Listing 3.1: An example of using siunitx in a document.

minutes and seconds. If they are given as a decimal number, they will be displayed as such.

```
\ang{10}
\ang{1;2;3}
\ang{;;20} \\
\ang{32.5}
\qty{60}{\degree}
```

10° 1°2′3″ 20″
32.5° 60°

When listing several numbers, we may use the

`\numlist[⟨options⟩]{⟨numbers⟩}`

command, where ⟨*numbers*⟩ are delimited by semicolons. This command is context-aware, which means it will produce correct results when used with polyglossia.

```
\numlist{1;2;3;4}

\begin{german}
  \numlist{1;2;3;4}
\end{german}
```

1, 2, 3 and 4
1, 2, 3 und 4

The package also contains dedicated commands for dealing with products and ranges of numbers. The numbers in a product should be delimited with the `x` letter.

```
Pick a number from
range \numrange{1}{10}. \\
\numproduct{2x5} is \(10\). \\
\begin{german}
  \numrange{1}{10}
\end{german}
```

Pick a number from range 1 to 10.
2 × 5 is 10.
1 bis 10

All these commands also have quantity versions.

```
Obtained results:
\qtylist{2;5;7}{\L}.\\
Acceptable range is
\qtyrange{1}{10}{\kg}. \\
This area is
\qtyproduct{2x5}{\metre}.
```

Obtained results: 2 L, 5 L and 7 L.
Acceptable range is 1 kg to 10 kg.
This area is 2 m × 5 m.

If we are dealing with quantum mechanics, then complex numbers may be useful. To typeset them, use `\complexnum` and `\complexqty`. They can be entered in both Cartesian form and polar coordinates.

```
The conjugate of
\complexnum{2+3i} is
\complexnum{2-3i}
which is approximately
\complexnum{3.6056:-56.310}. \\
Why is my scale showing my
weight is
\complexqty{65+i21}{\kg}?
```

The conjugate of $2 + 3\mathrm{i}$ is $2 - 3\mathrm{i}$ which is approximately $3.6056\angle{-56.310}°$.
Why is my scale showing my weight is $(65 + 21\mathrm{i})\,\mathrm{kg}$?

Sometimes the unit we want to use is missing. For example, throughout this book, the unit of typographical point is used when talking about font sizes. To define it we may use the

`\DeclareSIUnit[`⟨*options*⟩`]{`⟨*unit*⟩`}{`⟨*symbol*⟩`}`

command. The ⟨*unit*⟩ is the macro we will use in the interpreted mode, while the ⟨*symbol*⟩ is the typeset symbol. The ⟨*options*⟩ allow us to define defaults concerning, for example, the spacing of the unit in quantities.

```
\DeclareSIUnit{\pt}{pt}
The default font size
in \LaTeX{} is \qty{10}{\pt}.
```

The default font size in LaTeX is 10 pt.

This command may also be used when we want to adjust the appearance of a predefined unit. For example, the default litre is typeset using uppercase ‘L’, but you may want to change to lowercase ‘l’, instead.

```
Litre before redefining:
\unit{\L}. \\
\DeclareSIUnit{\litre}{l}
Litre after redefining:
\unit{\L}.
```

Litre before redefining: L.
Litre after redefining: l.

Similarly, the commands

```
\DeclareSIPrefix
\DeclareSIPower
\DeclareSIQualifier
```

allow definition of unit-related macros in case additional are needed.

```
\DeclareSIUnit{\pt}{pt}
\DeclareSIPower{\quartic}{\tothefourth}{4}
\DeclareSIPrefix{\decakilo}{dk}{4}
\DeclareSIQualifier{\polymer}{pol}

It's over \qty{9000}{\quartic\decakilo\pt\polymer}!
```

It's over $9000\,\mathrm{dkpt}^{4}_{\mathrm{pol}}$!

3.5.4 Table Columns with Numbers

When typesetting tables that contain numeric data, it is often useful to align them along the decimal point, so they can be compared easily. The siunitx package adds a special column specifier, `S`, to the `tabular` environment for this purpose. Non-numeric data must be surrounded by curly brackets in such columns.

When presenting numeric data, such as above, it is good to remember the following two rules:

- Never drop the leading zero before the decimal point.
- If the unit is the same in all cells, put it in the heading.

All numbers within `S` columns are automatically parsed by siunitx, which means that, if we use it, the first rule will be enforced for us.

In order to influence parsing/displaying options of a single column, pass them in square brackets to the column specification. See Listing 3.3 for an example.

Besides the standard options, there are also table-specific options that can be passed to the columns. An important one is `table-format`.

```
\begin{tabular}{@{}lS@{}}
  \toprule
  Day       & {Candy eaten (\unit{\g})} \\
  \midrule
  Monday    & .3011                     \\
  Tuesday   & 54.86                     \\
  Wednesday & 1000.9722                 \\
  Thursday  & -1000.9722                \\
  \bottomrule
\end{tabular}
```

Day	Candy eaten (g)
Monday	0.3011
Tuesday	54.86
Wednesday	1000.9722
Thursday	−1000.9722

Listing 3.2: A simple example of using siunitx's S column specification.

```latex
\DeclareSIUnit{\eur}{\euro}
\begin{tabular} {
    @{}l
    S[output-decimal-marker={,},
      per-mode=symbol]@{}
  }
  \toprule
  Candy        & {Price (\unit{\eur\per\kg})} \\
  \midrule
  Chocolate    & 11.30                        \\
  Lollipops    & 15.86                        \\
  Marshmallows & 5.97                         \\
  Golden taffy & 1125.12                      \\
  \bottomrule
\end{tabular}
```

Candy	Price (€/kg)
Chocolate	11,30
Lollipops	15,86
Marshmallows	5,97
Golden taffy	1125,12

Listing 3.3: An example of using optional parameters in a single `S` column specification.

By default, the `S` specifier centres the decimal point leaving equal space to the left and to the right of it. This leads to a lot of empty space if lengths of integer and fraction parts are widely different as can be seen in Listing 3.4. We can then use `table-format` to tell LaTeX how much space should be reserved before and after dot. For example, `6.2` means that it should reserve space for 6 digits before the decimal point, and for two digits after. The format may also contain information about signs, exponents, or text around the numbers. See Listing 3.5 for an example.

By default, numbers and text within the `S` columns and centred. If you want to change the alignment, use the `table-number-alignment` and `table-text-alignment` options. See Listing 3.6 for an example. The numbers may also be aligned inside a `\multicol` or `\multirow`, using the `\tablenum` command as seen in Listing 3.7.

If you don't want the decimal alignment but would still like to have numbers in columns formatted by siunitx, change the `table-alignment-mode` to `none`. See Listing 3.8 for an example.

3.6 Matrices and the like

Typesetting matrices in LaTeX is possible with the `matrix` environment from the amsmath [55] packages. It simply aligns the lines and columns as in the `tabular` environment and does not place any delimiters around its contents. In contrast to tabular, it does not accept column specifications, and each of its entries is typeset in math mode. The maximum number of columns a `matrix` can have is 10.

```
\( \left[
  \begin{matrix}
  1 & 2 \\
  3 & 4 \\
  \end{matrix}
  \right] \)
```

$$\left[\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}\right]$$

Because matrices are usually delimited, there are five additional versions that simply surround the matrix contents within delimiters. These are

`pmatrix` that uses parentheses as delimiters: $\left(\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}\right)$.

```
\sisetup{exponent-mode = fixed}
\begin{tabular} {@{}SS@{}}
  \toprule
  {\unit{\ug}} & {\unit{\kg}} \\
  \midrule
  1 & 1e-9 \\
  0.2 & 2e-10 \\
  35 & 35e-8 \\
  -100 & -1e-7 \\
  \bottomrule
\end{tabular}
```

µg	kg
1	0.000 000 001
0.2	0.000 000 000 2
35	0.000 000 35
−100	−0.000 000 1

Listing 3.4: An anti-example of using siunitx's `S` column specification without setting the `table-format`.

```
\sisetup{exponent-mode = fixed}
\begin{tabular} {
    @{}
    S[table-format = +3.1]
    S[table-format = +1.10]
    @{}
  }
  \toprule
  \multicolumn{2}{c}{Weight} \\
  \midrule
  {\unit{\ug}} & {\unit{\kg}} \\
  \midrule
  1 & 1e-9 \\
  0.2 & 2e-10 \\
  35 & 35e-8 \\
  -100 & -1e-7 \\
  \bottomrule
\end{tabular}
```

Weight	
µg	kg
1	0.000 000 001
0.2	0.000 000 000 2
35	0.000 000 35
−100	−0.000 000 1

Listing 3.5: An example of using siunitx's `S` column specification with the `table-format` key.

```
\begin{tabular} {
    @{}
    S[
      table-format = 2.1(1.1),
      table-number-alignment = left,
      uncertainty-mode = separate,
    ]
    S[
      table-format = 2.2e2{ big!},
      table-text-alignment = right,
    ]
    @{}
  }
  \toprule
  {Confidence (\unit{\percent})}
    & {Value} \\
  \midrule
  57.1(2) & 10e2 \\
  25.3(0) & 2e-1 \\
  0(0.1)  & 99.99e99{ big!} \\
  92(1)      & 7.2e10 \\
  \bottomrule
\end{tabular}
```

Confidence (%)	Value
57.1 ± 0.2	10×10^{2}
25.3	2×10^{-1}
0.0 ± 0.1	99.99×10^{99} big!
92 ± 1	7.2×10^{10}

Listing 3.6: An example of aligning text and numbers inside the siunitx's `S` column.

```
\sisetup{table-format = 4.4e1}
\begin{tabular}{@{}lr@{}}
  \toprule
  Candy     & Friend                          \\
  \midrule
  Chocolate & Peter                           \\
  Lollipop  & Jane                            \\
  \multicolumn{2}{c}{\tablenum{12,34 e0}}     \\
  \multicolumn{2}{c}{\tablenum{333.5567 e1}} \\
  \multicolumn{2}{c}{\tablenum{4563.21 e2}}  \\
  \bottomrule
\end{tabular}
```

Candy	Friend
Chocolate	Peter
Lollipop	Jane
12.34	
333.5567×10^{1}	
4563.21×10^{2}	

Listing 3.7: An example of using the `\tablenum` command.

```
\sisetup{
  table-alignment-mode = none,
  output-decimal-marker = {,},
  unit-color = red,
}
\begin{tabular}{@{}cS@{}}
  \toprule
  Variable & {Value} \\
  \midrule
  foo & 101.892 \\
  bar & 2e-1 \\
  baz & \qty{2.1}{\kg}  \\
  \bottomrule
\end{tabular}
```

Variable	Value
foo	101,892
bar	2×10^{-1}
baz	2,1 kg

Listing 3.8: An example of using the `S` column specifier without aligning numbers.

`bmatrix` that uses brackets as delimiters: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

`Bmatrix` that uses braces as delimiters: $\begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}$.

`vmatrix` that uses vertical lines as delimiters: $\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}$.

`Vmatrix` that uses double vertical lines as delimiters: $\begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix}$.

Matrices created using these environments are rather large, because they are meant to be typeset in display math. If for some reason you want to typeset a matrix within text, you may get better results with the `smallmatrix` environment. Note that there are no delimited variants, so you must provide any delimiters yourself.

```
Compare \(
  \begin{matrix}
    1 & 2 \\
    3 & 4 \\
  \end{matrix}
\) with \(
  \begin{smallmatrix}
    1 & 2 \\
    3 & 4 \\
  \end{smallmatrix}
\)
```

$$\text{Compare } \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \text{ with } \begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}$$

The `matrix` environment could also be used to typeset piecewise functions, by using the `.` character as an invisible `\right` delimiter in conjunction with a left brace. However, a much better alternative is the `cases` environment. Its contents are left aligned, and the delimiters are already provided.

```
\[
  \lvert x \rvert =
  \begin{cases}
    -x & \text{if } x < 0, \\
    0  & \text{if } x = 0, \\
    x  & \text{if } x > 0.
  \end{cases}
\]
```

$$|x| = \begin{cases} -x & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ x & \text{if } x > 0. \end{cases}$$

If you need more control over the alignment of the individual columns, you may also use the `array` environment. It accepts the column specifier, just like `tabular`, but typesets its contents in math mode. Note that it introduces a bit more spacing compared to the `matrix` environments.

```
\[
  \begin{array}{rcl}
    12 & 1 & 21 \\
    1 & 22 & 101 \\
  \end{array}
\]
```

$$\begin{array}{rcl} 12 & 1 & 21 \\ 1 & 22 & 101 \end{array}$$

3.7 Spacing in Math Mode

3.7.1 Mathematical Object Classes

LaTeX decides how much space to surround the symbols in math mode based on their *class*. For example, $\leq$ belongs to class of binary relations, so it is typeset with surrounding spaces. The class of a symbol can be changed using the commands listed in Table 3.4. For example, the symbol R is normally treated as alphabetical character, but we may force it to treat it as relation symbol using the `\mathrel` command.

```
Let \(R\) be any relation.
We should now write it like
this \(a \mathrel{R} b\),
and not like this \(a R b\).
```

Let R be any relation. We should now write it like this $a\ R\ b$, and not like this aRb.

It is important to use symbols with proper classes while typesetting mathematics. Some symbols may look the same as a single glyph but are different in terms of their classes. For example, `\bot` and `\perp` look very similar, but the former is meant to typeset a symbol, while the latter represents the relation of perpendicularity.

```
Let \(x \in \{\top, \bot\}\).
Given two lines \(k \perp l\).
```

Let $x \in \{\top, \bot\}$.
Given two lines $k \perp l$.

Another good example is the difference between the vertical bar symbol, `|`, as compared to the `\lvert` and `\rvert` commands. The former is a symbol, while the latter are commands suitable for delimiting expressions.

Class commands should not be used throughout the body of the document. If you want to use a new symbol (for example, an image) or repurpose an old one (such as the R relation), define a logical wrapper for it that uses the appropriate class inside.

```
\NewDocumentCommand{\modulo}{}{%
  \mathbin{\%}%
}
\( 17 \modulo 3 = 2 \)
```

$17 \mathbin{\%} 3 = 2$

Table 3.4: Commands to influence mathematical object classes

Command	Explanation	Examples
`\mathord`	Ordinary symbols, that do not require any special handling	`x` x `\bot` $\bot$
`\mathop`	Large operators, that are vertically centred within expression	`\sum` $\sum$ `\bigcap` $\bigcap$
`\mathbin`	Binary operators, with extra spacing around them	`+` $+$ `\times` $\times$
`\mathrel`	Binary relations, with extra spacing around them	`=` $=$ `\succeq` $\succeq$
`\mathopen`	Symbols that open groups, with space before	`(` $($ `\lBrack` $[\![$
`\mathclose`	Symbols that close groups, with space after	`)` $)$ `\rBrack` $]\!]$
`\mathpunct`	Punctuation symbols, with space after	`,` $,$ `\colon` $:$

3.7.2 Manual Spacing

If LaTeX's choice of spacing within formulae is not satisfactory, it can be adjusted manually. The most basic command to do so is

```
\mspace{⟨width⟩}
```

The ⟨*width*⟩ is specified using a special unit "mu", which is roughly equal to 1/18 of the width of letter "M" in the current math font.[5]

```
\( 1 \mspace{18mu} 2 \)
\textinterrobang{} vs.\
\( ? \mspace{-7mu} ! \)
```

$1 \quad 2$
‽ vs. ‽

There also exist predefined synonyms for the values most commonly used. These are presented in Table 3.5 and are usually sufficient when creating a document. For example, if you wanted to define the "d" in differentials, you may want to insert a small space before it, to separate it from the contents of the integral.

```
\NewDocumentCommand{\ud}{}{%
  \,\symrm{d}%
}
\[
  \int_a^b f(x)\ud x
\]
```

$$\int_a^b f(x)\,\mathrm{d}x$$

Negative spaces can also be used if you want to create your own mathematical symbols by overlaying existing ones.

```
\NewDocumentCommand{\myrel}{}{
  \mathrel{
    -\mspace{-11mu}
    \infty{}
    \mspace{-11mu}-
  }
}
\( x \myrel y \)
```

$x \mathrel{-\mspace{-11mu}\infty\mspace{-11mu}-} y$

[5] Units are explained in more detail in subsection 7.5.1.

Table 3.5: Commands for manual math spacing

Command	Alias	Equivalent to	Effect
none			$\rightarrow\!\!\leftarrow$
`\thinspace`	`\,`	`\mspace{3mu}`	$\rightarrow\,\leftarrow$
`\medspace`	`\:`	`\mspace{4mu}`	$\rightarrow\:\leftarrow$
`\thickspace`	`\;`	`\mspace{5mu}`	$\rightarrow\;\leftarrow$
`\quad`		`\mspace{18mu}`	$\rightarrow\quad\leftarrow$
`\qquad`		`\mspace{36mu}`	$\rightarrow\qquad\leftarrow$
`\negthinspace`	`\!`	`\mspace{-3mu}`	$\rightarrow\!\!\!\leftarrow$
`\negmedspace`		`\mspace{-4mu}`	$\rightarrow\!\!\!\leftarrow$
`\negthickspace`		`\mspace{-5mu}`	$\rightarrow\!\!\!\leftarrow$

3.7.3 Phantoms

In some situations, it may be useful to insert the space that would normally be occupied by some existing symbol. While you could eyeball it and insert appropriate `\mspace`, there is a better way to do so—phantoms. There are three commands for inserting phantoms:

```
\phantom{⟨text⟩}
\vphantom{⟨text⟩}
\hphantom{⟨text⟩}
```

Each of these typesets an invisible box that has dimensions equal to the text that it received as an argument. The `\vphantom` has zero width, while the `\hphantom` has zero height.

For example, when creating left superscripts and subscripts, they will be aligned to the left, since they actually apply to the previous (empty) symbol. A simple fix would be to pad the space using appropriate phantom. In the real world, it would be better to use the `\prescript` command from the mathtools package.

```
\( {}^{14}_{6}A \) vs.\
\( \prescript{14}{6}A \) vs.\
\( {}^{14}_{\phantom{1}6}A \)
```

$\boxed{{}^{14}_{6}A \text{ vs. } {}^{14}_{6}A \text{ vs. } {}^{14}_{6}A}$

Left superscripts and subscripts are often found in chemistry, when typesetting isotopes. If you intend to use LaTeX for that, it is better to use a dedicated package, such as chemformula [49] or mhchem [24].

The \vphantom command is useful if you want to typeset a formula having delimiters split over multiple lines (since delimiters only work with a single-line formula inside of them). Using that command, we can artificially increase the height of the sub-formula, so that the height of the delimiters will be calculated correctly.

```
\begin{multline*}
  f(a, b) = \left(
    \int_a^b f(x)
    looooooong
  \right. \\
  \left.
    \vphantom{
      \int_a^b f(x)
      looooooong
    }
    short \ud x
  \right)
\end{multline*}
```

$$f(a,b) = \left(\int_a^b f(x) looooooong \\ short\,\mathrm{d}x\right)$$

3.8 Theorems and Proofs

When writing mathematical documents, you probably need a way to typeset "lemmas", "definitions", "axioms", and similar structures. These are known as *theorems* in LaTeX, and can be created using the

\newtheorem{⟨*name*⟩}[⟨*counter*⟩]{⟨*caption*⟩}[⟨*section*⟩]

command. The ⟨*name*⟩ argument is name of the newly created environment used to typeset the theorem. The ⟨*caption*⟩ argument defines the actual name, which will be printed in the final document.

The two optional arguments are both used to specify the numbering used on the "theorem". Only one of them may be present. Use the ⟨*counter*⟩ argument to specify the ⟨*name*⟩ of a previously declared "theorem". This will make the new theorem be numbered the same way as the old one. The ⟨*section*⟩ argument allows you to specify the sectional unit

within which the theorem should get its numbers. The starred version of the command defines a theorem without a counter.

After executing the `\newtheorem` command in the preamble of your document, you can use the following environment within the document.

```
\begin{⟨name⟩}[⟨title⟩]
  This is my interesting theorem
\end{⟨name⟩}
```

The optional ⟨*title*⟩ argument can be used to locally typeset the theorem name next to the caption.

```
\newtheorem{proposition}{Proposition}
% ...
\begin{proposition}
  I propose that this should
  be enough of dummy text.
\end{proposition}
\begin{proposition}[More]
  It is not enough for more
  than one proposition.
\end{proposition}
```

Proposition 1 *I propose that this should be enough of dummy text.*

Proposition 2 (More) *It is not enough for more than one proposition.*

The amsthm [3] package provides the

```
\theoremstyle{⟨style⟩}
```

command, which lets you further customise a theorem by picking from three predefined styles: `definition` (bold title, roman body), `plain` (bold title, italic body), or `remark` (italic title, roman body). An example of using these commands is presented in Listing 3.9

If you want to customise your theorems down to the last dot, the ntheorem [45] package offers a plethora of options.

The amsthm package also provides the `proof` environment to use with theorems.

```
\begin{proof}
  Trivial.
\end{proof}
```

Proof. Trivial. □

```latex
\usepackage{amsthm}

\theoremstyle{definition}
\newtheorem{axiom}{Axiom}[section]
\theoremstyle{plain}
\newtheorem{theorem}[axiom]{Theorem}
\theoremstyle{remark}
\newtheorem*{remark}{Remark}
% ...
\section{First}
\begin{axiom}[Uncertainty]
  Nothing is certain.
\end{axiom}
\begin{theorem}
  It is uncertain whether
  this theorem is true.
\end{theorem}
\begin{remark}[Other things]
  Other things are
  probably also uncertain.
\end{remark}
```

1 First

Axiom 1.1 (Uncertainty)**.** Nothing is certain.

Theorem 1.2. *It is uncertain whether this theorem is true.*

Remark (Other things)*.* Other things are probably also uncertain.

Listing 3.9: An example of creating several theorem environments with different styles.

The QED symbol (□) is put at the end of the last line in the environment. This may create unnecessary lines if the proof does not end with a paragraph.

```
\begin{proof}
  Trivial, use
  \[ E=mc^2. \]
\end{proof}
```

Proof. Trivial, use

$$E = mc^2.$$

□

You can adjust its placement using the `\qedhere` command.

```
\begin{proof}
 Trivial, use
 \[ E=mc^2. \qedhere \]
\end{proof}
```

Proof. Trivial, use

$$E = mc^2. \qedhere$$

□

If you do not like the default symbol, you can redefine the `\qedsymbol` macro to your liking.

```
\RenewDocumentCommand{%
  \qedsymbol}{}{QED}
\begin{proof}
  Trivial.
\end{proof}
\RenewDocumentCommand{%
  \qedsymbol}{}{\(\QED\)}
\begin{proof}
  Trivial.
\end{proof}
```

Proof. Trivial. QED

Proof. Trivial. ■

3.9 Fiddling with Math Styles

You have already seen that LaTeX typesets mathematics differently, according to whether it is in inline or display style. There are actually four styles, the additional two being typically used for super- and subscripts. If you want to change the style chosen by LaTeX, you can do so by using the commands presented in Table 3.6.

Table 3.6: Math style commands available in LaTeX

Command	Example
`\displaystyle`	$\displaystyle\sum_{k=0}^{\infty} ABCabc123$
`\textstyle`	$\textstyle\sum_{k=0}^{\infty} ABCabc123$
`\scriptstyle`	$\scriptstyle\sum_{k=0}^{\infty} ABCabc123$
`\scriptscriptstyle`	$\scriptscriptstyle\sum_{k=0}^{\infty} ABCabc123$

By default, the `\frac` command decreases the style of its contents by one level when possible.

```
\[
  \frac
    {\frac
      {\frac{1}{2}}
      {2}}
    {2}
\]
```

$$\frac{\frac{\frac{1}{2}}{2}}{2}$$

If you want to prevent this, use the appropriate command in its arguments.

```
\[
\frac
  {\displaystyle
   \sum_{k=1}^n (x_k - x)^2}
  {\displaystyle
   \left(
    \sum_{k=1}^n (x_k - x)
   \right)^2}
\]
```

$$\frac{\displaystyle\sum_{k=1}^n (x_k - x)^2}{\displaystyle\left(\sum_{k=1}^n (x_k - x)\right)^2}$$

When typesetting formulae within text, you may enclose tall or deep math expressions, or sub-expressions, within `\smash`. This makes LaTeX ignore the height of these expressions and keeps the line spacing even,

but risks overlap with the surrounding text.

```
A \(d_{e_{e_p}}\) mathematical
expression followed by a
\(h^{i^{g^h}}\) expression.
As opposed to a smashed
\smash{\(d_{e_{e_p}}\)}
expression followed by a
\smash{\(h^{i^{g^h}}\)}
expression.
```

A $d_{e_{e_p}}$ mathematical expression followed by a $h^{i^{g^h}}$ expression. As opposed to a smashed $d_{e_{e_p}}$ expression followed by a $h^{i^{g^h}}$ expression.

The `\smash` command accepts an optional argument—either `t` or `b`—that changes the height of top or bottom part of the symbol to zero. This may be useful in some rare circumstances, for example, when adjacent radical symbols don't line up due to differing content heights.

```
\(  \sqrt{x}
  + \sqrt{y}
  + \sqrt{z}\)
vs.\
\(  \sqrt{x}
  + \sqrt{\smash[b]{y}}
  + \sqrt{z} \)
```

$\sqrt{x} + \sqrt{y} + \sqrt{z}$ vs. $\sqrt{x} + \sqrt{y} + \sqrt{z}$

3.10 Dots

Use the `\dots` command to omit an easily deduced part of a mathematical expression, like $0, 1, 2, \dots$ to indicate the sequence of all natural numbers. It attempts to adjust its output based on the surrounding symbols.

```
\(1, 2, 3, \dots, 100\) vs.\
\(1 + 2 + 3 + \dots + 100\)
```

$1, 2, 3, \dots, 100$ vs. $1+2+3+\cdots+100$

If the spacing chosen by LaTeX is not appropriate, you can specify it explicitly by using:

`\dotsc` for dots between commas ($1, 2, \dots, 100$)

`\dotsb` for dots between binary operators or relations ($1 < 2 < \cdots < 100$)

`\dotsm` for dots indicating multiplication ($X_1 X_2 \cdots X_{100}$)

`\dotsi` for dots between integrals ($\int_{X_1} \int_{X_2} \cdots \int_{X_{100}}$)

`\dotso` for dots in situations not matched by any of the above ($1 \ldots +$)

```
\(1 + 2 + 3 + \dots\) vs.\
\(1 + 2 + 3 + \dotsb\)
```

$1 + 2 + 3 + \ldots$ vs. $1 + 2 + 3 + \cdots$

You can also redefine the above wrappers if the version chosen by LaTeX is not to your liking. By default, all of these commands choose between `\ldots`, for dots on baseline, and `\cdots`, for centred dots.

```
\RenewDocumentCommand{%
  \dotsb}{}{\ldots}
\( 1 + 2 + 3 + \dotsb + 100 \)
```

$1 + 2 + 3 + \ldots + 100$

In the case of matrices, you may also need non-horizontal dots. These are accessible from the commands `\vdots` (vertical dots $\vdots$), `\ddots` (descending dots $\ddots$), and `\adots` (ascending dots $\iddots$).

```
\[ \begin{bmatrix}
  a_{1,1} & \cdots  & a_{1,n} \\
  \vdots  & \ddots  & \vdots  \\
  a_{m,1} & \cdots  & a_{m,n} \\
\end{bmatrix} \]
```

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{bmatrix}$$

3.11 More About Fractions

You have already learned about the `\frac` command for typesetting built-up fractions. Its output depends on whether it is typeset in display or text style.

```
\( \frac{1}{2} \) vs.\
\[ \frac{1}{2} \]
```

$\frac{1}{2}$ vs.

$$\frac{1}{2}$$

If you don't want to type \displaystyle or \textstyle every time to correct the style of fractions, you can use \tfrac and \dfrac. These always typeset the fractions in text or display style, respectively.

```
\( \dfrac{1}{2} \) vs.\
\( \tfrac{1}{2} \)
\[ \dfrac{1}{2}\text{ vs.\ }
  \tfrac{1}{2} \]
```

$\dfrac{1}{2}$ vs. $\tfrac{1}{2}$

$$\dfrac{1}{2} \text{ vs. } \tfrac{1}{2}$$

When writing the fractions in-line, often the slashed form 1/2 is preferable for small fractions. You may also want to use the xfrac [26] package and the \sfrac command it provides to typeset them in a slashed form with smaller numbers.

```
\( \frac{1}{2} \) vs.\
\sfrac{1}{2} vs.\
\( 1/2 \) vs.\ \textonehalf{}
```

$\frac{1}{2}$ vs. ½ vs. 1/2 vs. ½

Note that \sfrac essentially 'fakes' the fraction by manually placing relevant symbols in the given positions. If the faked version is not to your liking, the package provides more customization options. These are described in the package documentation. Some fonts support arbitrary fractions as symbols, which may produce better results; you can find more information about it in the Section 7.3.

If you have tried writing continued fractions, you may have noticed that the spacing is not ideal. The amsmath package defines a special \cfrac command to fix this issue.

```
\[
  \dfrac{1}{
    1 + \dfrac{1}{
      1 + \dfrac{1}{
        1 + \dotsb}}}
\]
vs.\
\[
  \cfrac{1}{
    1 + \cfrac{1}{
      1 + \cfrac{1}{
        1 + \dotsb}}}
\]
```

$$\dfrac{1}{1+\dfrac{1}{1+\dfrac{1}{1+\dotsb}}}$$

vs.

$$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\dotsb}}}$$

Binomial coefficients can be typeset using the \binom, \tbinom and \dbinom commands. These work the same as the \frac, \tfrac and \dfrac commands.

```
\[
  \sum_{k=0}^{n} \binom{n}{k}
    = 2^{n}
\]
```

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Built-up fractions and binomial coefficients are specialized examples of a more general command called \genfrac. It allows you to typeset one expression over another, with an optional line between them and optional delimiters. Its full syntax is

\genfrac{⟨*left*⟩}{⟨*right*⟩}{⟨*thickness*⟩}{⟨*style*⟩}{⟨*num*⟩}{⟨*den*⟩}

The ⟨*left*⟩ and ⟨*right*⟩ arguments are the left and right delimiters, such as parentheses in the case of \binom. The ⟨*thickness*⟩ argument determines the thickness of the line between the expressions. If you leave it empty, it defaults to the same as that used by \frac. The ⟨*style*⟩ argument is a number from 0 to 3, and overrides the default math style used to typeset the symbol. For example, \tfrac sets it to 1, to always typeset the symbol in text style.

As an example, to create a wrapper for the unsigned Stirling numbers of the first kind you could write

```
\NewDocumentCommand{\stirfst}{}{
  \genfrac{[}{]}{0cm}{}
}
\[
  \sum_{k=0}^{n} \stirfst{n}{k}
    = n!
\]
```

$$\sum_{k=0}^{n} \left[{n \atop k}\right] = n!$$

You should avoid using \genfrac inside the body of the document, and always define logical wrappers if you intend to use it in multiple places.

Chapter 4

Bibliographies

When writing articles or books concerned with some topic you will often need to reference other books and articles to point out where some information might be found. You have already seen this done multiple times throughout this book. Doing this by hand would be tedious, so LaTeX comes with an option to manage the bibliographic data of our document.

This chapter will describe two approaches to bibliography management in LaTeX:

`thebibliography` environment which is suitable for rather small bibliographies.

biblatex with biber which is an advanced bibliography management system that is suitable for books and publications with extensive bibliographies. These systems also make it easy to format all bibliography entries in exactly the style required by the publisher.

4.1 `thebibliography` environment

Produce a bibliography with the `thebibliography` environment. Each entry starts with

`\bibitem[`⟨*label*⟩`]{`⟨*marker*⟩`}`

The ⟨*marker*⟩ is then used to cite the book, article, or paper within the

document.

`\cite{⟨marker⟩}`

If you do not use the ⟨*label*⟩ option, the entries will get enumerated automatically. The parameter after the `\begin{thebibliography}` command defines how much space to reserve for the number of labels. In the example below, `{99}` tells LaTeX to expect that none of the bibliography item numbers will be wider than the number 99.

```
Partl~\cite{pa} has
proposed that \ldots
\begin{thebibliography}{99}
\bibitem{pa} H.~Partl:
\emph{German \TeX},
TUGboat Volume~9, Issue~1 (1988)
\end{thebibliography}
```

Partl [1] has proposed that …

References

[1] H. Partl: *German TEX*, TUGboat Volume 9, Issue 1 (1988)

In order to get the citations typeset properly, two passes of LaTeX compiler are needed similar to how tables of contents are done.

This approach has the advantage that it is simple and entirely self-contained—no external packages and programs are needed since it is supported out of the box by LaTeX. There are however many problems:

- Each entry has to be formatted manually, which may lead to inconsistent styling. Moreover, simple changes in format will require editing each entry by hand.
- There is no logical/structural markup to imply which part of bibliographic entry is author, title, journal, etc.
- No sorting is performed. If you want the entries to be processed in citation order, you have to ensure this manually.
- If you want to use different citation style than numeric, you have to manually set all the labels.

For all projects that contain more than few citations biblatex is strongly recommended.

4.2 biblatex with biber

The biblatex [44] package uses a separate program, biber [14], for bibliography management. biber is a successor of the popular BibTeX program. Its main advantage is UTF-8 support, which was lacking in the original BibTeX. The database format is largely backward compatible, so on many sites you can find "BibTeX entry" or "Export Bibtex Citation", which will be useful when compiling your own citation database.

4.2.1 Database files

The bibliographic database is stored in special `.bib` files with their own syntax. A single bibliographic entry is of the form

```
@⟨entry type⟩{⟨marker⟩,
  ⟨field1⟩ = {⟨value1⟩},
  ⟨field2⟩ = {⟨value2⟩},
  ⟨field3⟩ = {⟨value3⟩},
      ⋮
}
```

The fields required depend on the bibliography style you choose, the default style of biblatex will accept the following ⟨*entry type*⟩s:

`article` for articles from journals or other periodicals. Important fields are `author`, `title`, `journaltitle`, `date`, `url`, `doi`.

`book` for single-volume book. With the fields `author`, `title`, `date`, `publisher`, `volume`.

`online` for accessed online resources. With `author`, `title`, `date`, `url`, `urldate`.

`manual` for technical documentation. With `author`, `title`, `date`, `url`, `version`.

`misc` for entries that do not fit any of the predefined categories. With `author`, `title`, `date`, `howpublished`, `note`.

Note that the above list is far from exhaustive both in terms of ⟨*entry type*⟩s and the field names mentioned. For a full list check the biblatex manual or the style you are using.

Each `.bib` file will typically contain multiple entries. See Listing 4.1 for an example.

4.2.2 Using biblatex

In order to use biblatex, the following three commands are necessary.

The biblatex package must be loaded with the

```
\usepackage[⟨options⟩]{biblatex}
```

command. The ⟨*options*⟩ are comma delimited key value pairs that allow to customise the behaviour of biblatex package. Some of them will be explained later.

The command

```
\addbibresource[⟨options⟩]{⟨file⟩}
```

must be put in the preamble. The ⟨*file*⟩ is the name of the `.bib` file containing bibliographic entries. You may also specify a remote location here, but then you must also put `location=remote` in the ⟨*options*⟩.

Finally the

```
\printbibliography[⟨options⟩]
```

typesets the loaded bibliography. The ⟨*options*⟩ may be used to alter the title or filter the entries included.

Processing files requires three LaTeX passes in addition to the biber command. A typical command line run may look like this:

```
$ xelatex document.tex
$ biber document
$ xelatex document.tex
$ xelatex document.tex
```

The first LaTeX pass extracts the citation data from the document which are then read by biber. In the second pass, the bibliography is placed in

```bibtex
@book{lshort,
  title   = {The Not So Short Introduction to \LaTeXe},
  author  = {Tobias Oetiker and Hubert Partl
             and Irene Hyna and Elisabeth Schlegl},
  date    = {2021-03-09},
  version = {6.4},
  url     = {https://www.ctan.org/pkg/lshort-english}
}

@article{curie,
  title        = {Les nouvelles substances radioactives},
  author       = {Curie, Marie},
  year         = {1900},
  journaltitle = {Revue scientifique},
  series       = {4},
  volume       = {14},
  number       = {3}
}

@misc{dream,
  title        = {Yesterday's dream},
  howpublished = {The vision came to me while I was sleeping},
  note         = {It was very vivid},
  author       = {Jane Diviner},
  date         = {3012-07-19},
  keywords     = {dreamy, unreliable}
}
```

Listing 4.1: An example bibliography database for biber (`.bib` file)

the document and the third pass finally typesets the document with all the correct citations.

Listing 4.2 is a complete example that shows how the bits work together. In this and all the following examples, the `example.bib` file is assumed to be the same as in the Listing 4.1.

4.2.3 Controlling the bibliography

The default sorting order is *N*ame (author/editor), *T*itle, *Y*ear or `nty` in short. This order can be changed by passing `sorting=`⟨*order*⟩ as a package option. For example if you want to sort entries in chronological order simply pass the `ynt` option. Other options allow sorting by citation order (`none`) and by the number of citations (`count`).

```
% In preamble
\usepackage[
  sorting=ynt
]{biblatex}

% ...
```


[1] Marie Curie. "Les nouvelles substances radioactives". In: *Revue scientifique*. 4th ser. 14.3 (1900).

[2] Tobias Oetiker et al. *The Not So Short Introduction to LaTeX 2ε*. Mar. 9, 2021. URL: `https://www.ctan.org/pkg/lshort-english`.

[3] Jane Diviner. *Yesterday's dream*. The vision came to me while I was sleeping. It was very vivid. July 19, 3012.

While the default citation style is `numeric`, this can be easily changed by setting the preferred style via the ⟨*style*⟩ option. Choose from `alphabetic`, `authoryear`, `authortitle` or `verbose`. Some styles come in several variations.

```latex
\documentclass{article}

\usepackage{biblatex}
\addbibresource{example.bib}

\begin{document}
\ldots{} Recently I was learning to use Bib\LaTeX{}
from~\cite{lshort}. It seems very useful. \ldots

\ldots{} which was already shown by Mrs.~Curie
in~\cite{curie}. \ldots

\ldots{} this can be easily explained by
the fact that Einstein was a time
traveller~\cite{dream}. \ldots

\printbibliography
\end{document}
```

… Recently I was learning to use BibLaTeX from [3]. It seems very useful. …
… which was already shown by Mrs. Curie in [1]. …
… this can be easily explained by the fact that Einstein was a time traveller [2]. …

References

[1] Marie Curie. "Les nouvelles substances radioactives". In: *Revue scientifique*. 4th ser. 14.3 (1900).

[2] Jane Diviner. *Yesterday's dream*. The vision came to me while I was sleeping. It was very vivid. July 19, 3012.

[3] Tobias Oetiker et al. *The Not So Short Introduction to LaTeX 2ε*. Mar. 9, 2021. URL: https://www.ctan.org/pkg/lshort-english.

Listing 4.2: An example of using biblatex to manage references in an article

```
% In preamble
\usepackage[
  style=alphabetic
]{biblatex}

% ...

The validity of~\cite{dream}
as a scientific source was
recently called into question
though it correctly claims
that polonium was first
described in~\cite{curie}.
```

The validity of [Div12] as a scientific source was recently called into question though it correctly claims that polonium was first described in [Cur00].

References

[Cur00] Marie Curie. "Les nouvelles substances radioactives". In: *Revue scientifique*. 4th ser. 14.3 (1900).

[Div12] Jane Diviner. *Yesterday's dream*. The vision came to me while I was sleeping. It was very vivid. July 19, 3012.

In contrast to the `thebibliography` environment, the biblatex's `\printbibliography` command only prints entries that were referenced in the document. If you want to print entries not mentioned in the document, you may use the `\nocite{`⟨*marker*⟩`}` command. It will insert invisible citations, thus instructing biblatex to put them in the bibliography. The special value `*` can be passed as a ⟨*marker*⟩ if you want to print all entries in the database.

```
I've only
cited~\cite{lshort}.

\nocite{*}
\printbibliography
```

I've only cited [3].

References

[1] Marie Curie. "Les nouvelles substances radioactives". In: *Revue scientifique*. 4th ser. 14.3 (1900).

[2] Jane Diviner. *Yesterday's dream*. The vision came to me while I was sleeping. It was very vivid. July 19, 3012.

[3] Tobias Oetiker et al. *The Not So Short Introduction to LaTeX 2ε*. Mar. 9, 2021. URL: https://www.ctan.org/pkg/lshort-english.

If your bibliography gets really large it may make sense to split it into several parts. This can be done by passing filter options to the `\printbibliography` command. Available filters include `type`, `category`, `keyword`. It may be useful to also change the titles of different categories using `title` option.

```
\printbibliography[
  type=book,
  title=Books I've referenced
]

\printbibliography[
  keyword=unreliable,
  title=Don't trust those
]
```

Books I've referenced

[3] Tobias Oetiker et al. *The Not So Short Introduction to LaTeX 2ε*. Mar. 9, 2021. URL: https://www.ctan.org/pkg/lshort-english.

Don't trust those

[2] Jane Diviner. *Yesterday's dream*. The vision came to me while I was sleeping. It was very vivid. July 19, 3012.

By default, the bibliography does not appear in the table of contents. This is because it is using starred version of `\chapter`/`\section` (depending on the class) to generate its heading. In order to change the default we may pass the option `heading` to `\printbibliography`. Available options include

`bibliography` the default, starred version of heading

`bibintoc` starred version of heading, will appear in table of contents

`subbibliography` will drop the bibliography one level in hierarchy (`\section*` instead of `\chapter*` and so on)

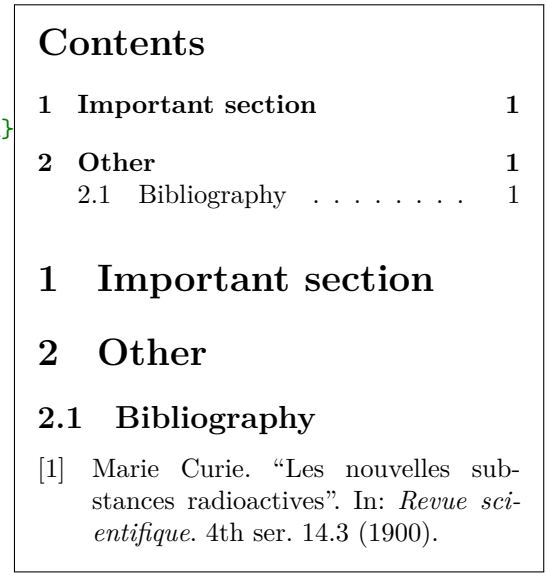
`bibnumbered` will use non starred version for heading, thus numbering it and appearing in table of contents

`subbibintoc` will drop the bibliography one level and put it in table of contents

`subbibnumbered` will drop the bibliography one level and use non starred version of heading

`none` will not print the heading

```
\tableofcontents
\section{Important section}

% ...

\section{Other}

\printbibliography[
  heading=subbibnumbered,
  title=Bibliography,
]

% ...
```



Contents

1 Important section 1

2 Other 1

2.1 Bibliography 1

1 Important section

2 Other

2.1 Bibliography

[1] Marie Curie. “Les nouvelles substances radioactives”. In: *Revue scientifique*. 4th ser. 14.3 (1900).

As you may have noticed, when entries have many authors, then not all of them will be printed; instead "et al." shows up at the end of the author list. This behaviour may be controlled via `maxnames` (default 3) and `minnames` (default 1) options. If the number of names is greater than ⟨*maxnames*⟩ then it will be shortened to ⟨*minnames*⟩ and "et al." will be added.

```
\usepackage[
  maxnames=4,
]{biblatex}
```

[1] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The Not So Short Introduction to LaTeX 2ε*. Mar. 9, 2021. URL: https://www.ctan.org/pkg/lshort-english.

```
\usepackage[
  minnames=2,
]{biblatex}
```

[1] Tobias Oetiker, Hubert Partl, et al. *The Not So Short Introduction to LaTeX 2ε*. Mar. 9, 2021. URL: https://www.ctan.org/pkg/lshort-english.

4.2.4 Citing commands

Until now we have only used the basic `\cite` command. biblatex extends the command set to allow extra control with citations.

Most of the citing commands (`\cite` included) allow inserting notes around citations:

`\cite[⟨pre⟩][⟨post⟩]{⟨marker⟩}`

Other standard citation commands include `\parencite` (citation in parentheses), `\footcite` (in footnotes), `\textcite` (citation intended to

be subject in a sentence), \smartcite (context dependent).

```
Indicate page in
citation~\cite[25]{lshort}.
This citation is in
parentheses~\parencite{curie}.
Footnote cite~\footcite{dream}.
Smart \smartcite[See][78]{lshort}.
Again\footnote{Smart
\smartcite[12--56]{dream}.}.
\enquote{\Textcite{curie}
  was an important paper.
}
```

Indicate page in citation [Oet+21, p. 25]. This citation is in parentheses [Cur00]. Footnote cite[1]. Smart[2]. Again[3]. "Curie [Cur00] was an important paper. "

[1] Div12.
[2] See Oet+21, p. 78.
[3] Smart [Div12, pp. 12–56].

Another set of commands allows us to extract specific information from the bibliographic entry, and use the information directly in the text. This includes commands such as \citeauthor, \citetitle, \citeyear, \citedate, \citeurl.

```
\Citetitle{lshort} is a book by
\citeauthor{lshort}. Latest version
was released in \citeyear{lshort},
or to be more precise on
\citedate{lshort}. It is available
at \citeurl{lshort}.
```

The Not So Short Introduction to LaTeX 2ε is a book by Oetiker et al. Latest version was released in 2021, or to be more precise on Mar. 9, 2021. It is available at `https://www.ctan.org/pkg/lshort-english`.

If you want to be less tied to a specific style, the \autocite command

follows the citation style specified in the package options.

```
\usepackage[
  style=verbose,
  autocite=footnote,
]{biblatex}

% ...

This is auto citation
\autocite{curie}.
```

This is auto citation.[1]

[1] Marie Curie. "Les nouvelles substances radioactives". In: *Revue scientifique*. 4th ser. 14.3 (1900).

```
\usepackage[
  style=authoryear,
  autocite=inline,
]{biblatex}

% ...
```

This is auto citation (Curie 1900).

4.2.5 More about entries

biber uses "and" as a separator in certain entries. To prevent this behaviour, enclose "and" in curly brackets

```
@book{kru,
  publisher = {Kruger {and} sons}
}
```

The same trick may be useful when biber changes capitalisation, even though it shouldn't. On top of this biber splits author entries into smaller bits which are then used elsewhere. So in Listing 4.1 the name Jane Diviner is split into the first name Jane and the surname Diviner. When you don't want this, enclose the entire name in braces

```
@book{kru,
  author = {{Kruger brothers}}
}
```

When writing about certain subject it often happens that the same author or publishing company released several books. In order to reuse the information in several entries in the `.bib` file, a special entry `xdata` is available. It may be used like this

```
@xdata{krugers,
  author    = {{Kruger brothers}},
  publisher = {Kruger {and} sons},
  location  = {Paris}
}

@book{kru21,
  title = {Why is \LaTeX{} so hard?},
  year  = {2021},
  xdata = {krugers}
}

@book{kru22,
  title = {\LaTeX{} is awesome!},
  year  = {2022},
  xdata = {krugers}
}
```

References

[1] Kruger brothers. *LaTeX is awesome!* Paris: Kruger and sons, 2022.

[2] Kruger brothers. *Why is LaTeX so hard?* Paris: Kruger and sons, 2021.

Chapter 5

Specialities

When putting together a large document, LaTeX will help with some special features like index generation, automatic linking to relevant pages and other things. A much more complete description of specialities and enhancements possible with LaTeX can be found in the *LaTeX Manual* [42] and *The LaTeX Companion* [48].

5.1 Indexing

A very useful feature of many books is their index. With LaTeX and the support program `makeindex`,[1] an index can be generated quite easily. This introduction will only explain the basic index generation commands. For a more in-depth view, please refer to *The LaTeX Companion* [48].

To enable the indexing feature of LaTeX, the makeidx package must be loaded in the preamble with

```
\usepackage{makeidx}
```

and the special indexing commands must be enabled by putting the

```
\makeindex
```

command in the preamble.

[1] On systems not necessarily supporting filenames longer than 8 characters, the program may be called `makeidx`.

The content of the index is specified with

`\index{`⟨*key*⟩`@`⟨*formatted entry*⟩`}`

commands, where ⟨*formatted entry*⟩ will appear in the index and ⟨*key*⟩ will be used for sorting. The ⟨*formatted entry*⟩ is optional. If it is missing the ⟨*key*⟩ will be used. You enter the index commands at the points in the text that you want the final index entries to point to. Table 5.1 explains the syntax with several examples.

When the input file is processed with LaTeX, each `\index` command writes an appropriate index entry, together with the current page number, to a special file. The file has the same name as the LaTeX input file, but a different extension (`.idx`). This `.idx` file can then be processed with the `makeindex` program:

`makeindex` *filename*

The `makeindex` program generates a sorted index with the same base file name, but this time with the extension `.ind`. If now the LaTeX input file is processed again, this sorted index gets included into the document at the point where LaTeX finds

`\printindex`

The showidx package that comes with LaTeX prints out all index entries in the left margin of the text. This is quite useful for proofreading a document and verifying the index. Make sure to load the package afer the hyperref package.

Note that the `\index` command can affect your layout if not used carefully.

```
My Word \index{Word}. As opposed
to Word\index{Word}. Note the
position of the full stop.
```

My Word . As opposed to Word. Note the position of the full stop.

Note that the textttmakeindex has no clue about characters outside the ASCII range. To get the sorting correct, use the `@` character as shown in the Käse and école examples above.

Table 5.1: Index Key Syntax Examples.

Code	Entry	Explanation
`\index{hello}`	hello, 1	Plain entry
`\index{hello!Peter}`	Peter, 3	Subentry under 'hello'
`\index{Sam@\emph{Sam}}`	*Sam*, 2	Formatted entry
`\index{Kaese@\emph{K\"ase}}`	*Käse*, 33	Formatted entry
`\index{ecole@\'ecole}`	école, 4	Formatted entry
`\index{Jenny|emph}`	Jenny, *3*	Formatted page number
`\index{Joe@\emph{Joe}|emph}`	*Joe*, *5*	Formatted page number

5.2 Installing Extra Packages

Most LaTeX installations come with a large set of pre-installed style packages, but many more are available on the net. The main place to look for style packages on the Internet is CTAN (http://www.ctan.org/).

Packages such as geometry, hyphenat, and many others are typically made up of two files: a file with the extension `.ins` and another with the extension `.dtx`. There will often be a `readme.txt` with a brief description of the package. You should of course read this file first.

In any event, once you have copied the package files onto your machine, you still have to process them in a way that (a) tells your TeX distribution about the new style package, and (b) gives you the documentation. Here's how you do the first part:

1. Run LaTeX on the `.ins` file. This will extract a `.sty` file.

2. Move the `.sty` file to a place where your distribution can find it. Usually this is in your *…/localtexmf*/tex/latex subdirectory (Windows or OS/2 users should feel free to change the direction of the slashes).

3. Refresh your distribution's file-name database. The command depends on the LaTeX distribution you use: TeXLive — `texhash`; web2c — `maktexlsr`; MiKTeX — `initexmf --update-fndb` or use the GUI.

Now extract the documentation from the `.dtx` file:

1. Run XeLaTeX on the `.dtx` file. This will generate a `.pdf` file. Note that you may have to run XeLaTeX several times before it gets the cross-references right.

2. Check to see if LaTeX has produced a `.idx` file among the various files you now have. If you do not see this file, then the documentation has no index. Continue with step 5.

3. In order to generate the index, type the following:
   `makeindex -s gind.ist` *`name`*
   (where *name* stands for the main-file name without any extension).

4. Run LaTeX on the `.dtx` file once again.

5. Last but not least, make a `.ps` or `.pdf` file to increase your reading pleasure.

Sometimes you will see that a `.glo` (glossary) file has been produced. Run the following command between step 4 and 5:
`makeindex -s gglo.ist -o` *`name`*`.gls` *`name`*`.glo`
Be sure to run LaTeX on the `.dtx` one last time before moving on to step 5.

5.3 LaTeX and PDF

The initial release of TeX predated the PDF format by nearly 16 years. The original output files it produced—`.dvis`—were meant to be only printed. Today many documents are never or seldom printed, we read them directly on a screen. PDF format contains many improvements for viewing documents like this but they are not implemented in core LaTeX. These are accessible via the hyperref [58] package.

5.3.1 Hypertext Links

Hyperlinks are used to quickly jump around the document. The prime example of using them is the table of contents, you don't have to manually

scroll to a given page—just click on a given chapter and you will be immediately transported there. You already know that table of contents can be typeset using the `\tableofcontents` command, but it doesn't contain any hyperlinks.

Luckily the process of updating your document is extremely easy: just add `\usepackage{hyperref}` as the *last* package loaded in your preamble. Doing so redefines internal LaTeX commands to produce hyperlinks.

```
% In preamble
\usepackage{hyperref}
% ...
The reference to this section
now looks like that:\footnote{
  Footnotes are also
  made into hyperlinks.}
Section~\ref{hyperlinks} is
on page~\pageref{hyperlinks}.
The hyperref bibliographic
entry is~\cite{pack:hyperref}.
```

> The reference to this section now looks like that:[1] Section 5.3.1 is on page 170. The hyperref bibliographic entry is [58].
>
> [1] Footnotes are also made into hyperlinks.

By default links are marked by a red box around them. This box is only visible when viewing the document on screen and will not be printed. The boxes are, however, rather ugly, so you may want to add the `colorlinks` option while loading the package.

```
% In preamble
\usepackage[
  colorlinks
]{hyperref}
% ...
The reference to this section
now looks like that:
Section~\ref{hyperlinks} is
on page~\pageref{hyperlinks}.
```

> The reference to this section now looks like that: Section 5.3.1 is on page 170.

This is the option used throughout this booklet and assumed in further examples. While this makes the links more visually appealing, it has the disadvantage that the coloured links will be printed. To marry the best of both worlds you can use the ocgx2 [19] package with the option `ocgcolorlinks` like this

```
\usepackage{hyperref}
\usepackage[ocglinks]{ocgx2}
```

Be warned though that it is not well supported by popular PDF viewers. Another option is to use option `hidelinks` that makes the links clickable but does not distinguish them visually.

In addition to redefining internal LaTeX command, hyperref defines some additional ones. To typeset URLs you can now use the `\url` command.

```
\url{https://www.ctan.org/}
```

https://www.ctan.org/

If you want to display different text for the clickable link you can use the

`\href{⟨URL⟩}{⟨text⟩}`

command. Note that if you intend for the document to be useful when printed, you have to provide the full URLs anyway.

```
Packages can be found on
\href{https://www.ctan.org/}{
  CTAN}.
```

Packages can be found on CTAN.

A similar command is

`\hyperref[⟨marker⟩]{⟨text⟩}`

that allows to create a hyperlink within the document with a different text.

```
\hyperref[hyperlinks]{
  Hyperlinks section}
describes the usage of
hyperlinks.
```

Hyperlinks section describes the usage of hyperlinks.

To avoid nesting hyperlinks inside one another, hyperref provides starred versions of `\ref` and `\pageref` commands that produce text without

hyperlinking them.

```
\hyperref[hyperlinks]{
  Section~\ref*{hyperlinks}}
describes the usage of
hyperlinks.

This ref~\ref*{hyperlinks}
is not hyperlinked.
```

Section 5.3.1 describes the usage of hyperlinks.
This ref 5.3.1 is not hyperlinked.

Throughout this booklet you have seen references such as 'subsection 5.3.1 on page 170'. While this could be achieved using the aforementioned \hyperref command, this usecase is so common that hyperref provides two commands that make it much easier:

\autoref{⟨*marker*⟩}
\autopageref{⟨*marker*⟩}

Their usage is identical to the \ref and \pageref commands, but they produce additional text based on the counter the ⟨*marker*⟩ refers to.

```
\autoref{hyperlinks} in
\autoref{specialities} on
\autopageref{hyperlinks}.
```

subsection 5.3.1 in chapter 5 on page 170.

The names are controlled by commands such as \autorefchaptername or \autorefsectionname. See the hyperref [58] documentation for a full list.

So far we have used the default colours for URLs, hyperlinks. These can be changed using the \hypersetup command. It accepts a key value list customising the appearance of links. Colours may be specified using `linkcolor`, `citecolor` and `urlcolor`. If you have the xcolor package loaded, you may specify colours the same way as described in Section 7.4.

```
\hypersetup{
  urlcolor = pink,
  citecolor = purple,
  linkcolor = teal!50!yellow,
}
\url{https://www.ctan.org/} \\
\cite{pack:hyperref} \\
\autoref{hyperlinks}
```

https://www.ctan.org/
[58]
subsection 5.3.1

You can also adjust the borders around the links. The basic key is `pdfborder`. It accepts three numbers: horizontal corner radius, vertical corner radius and border width.

```
\hypersetup{pdfborder = 0 0 1}
\url{https://www.ctan.org/} \\
\hypersetup{pdfborder = 10 10 3}
\url{https://www.ctan.org/} \\
\hypersetup{pdfborder = 10 5 2}
\url{https://www.ctan.org/} \\
\hypersetup{pdfborder = 2 7 5}
\url{https://www.ctan.org/}
```

https://www.ctan.org/
https://www.ctan.org/
https://www.ctan.org/
https://www.ctan.org/

The colour of the boxes may be adjusted with the `linkbordercolor`, `citebordercolor` and `urlbordercolor` keys, assuming the xcolor package is loaded.

```
\hypersetup{
  pdfborder = 0 0 2,
  urlbordercolor = violet,
  citebordercolor = pink,
  linkbordercolor = teal,
}
\url{https://www.ctan.org/} \\
\cite{pack:hyperref} \\
\autoref{hyperlinks}
```

https://www.ctan.org/
[58]
subsection 5.3.1

5.3.2 Document Metadata

Another thing that may be adjusted with the hyperref package is document metadata. These are information about your document that are not visible in the document itself, but may be used by your PDF viewer in various ways. For example, the title of the document may be shown in its top window bar.

Additional information about your document may be set using `pdfinfo` key. This key itself accepts a key value list of document properties.

```
\hypersetup{
  pdfinfo = {
```

```
    Title = Title of the Book,
    Author = {Us, Ourselves and We},
    Subject = Book creation with LaTeX,
    Creator = Our House,
    Keywords = {LaTeX, typesetting},
    Producer = LuaTeX,
  }
}
```

You can also control the way your document presents itself when opening. For example you can choose whether the bookmarks should be shown (`bookmarksopen`), whether external links should be opened in new windows (`pdfnewwindow`) whether the pages should initially fit the window (`pdffitwindow`).

If you want to set the metadata without redefining internal LaTeX commands to produce links, you may pass `implicit=false` to hyperref package options.

5.3.3 Problems with Outline

The hyperref package automatically uses table of contents generated by the document as a document outline for easier navigation. This may lead to some problems if your section titles contain some non-text content (for example, "LaTeX"). If this is the case, then it will be ignored by the hyperref package and the following warning will be reported

```
Package hyperref Warning:
Token not allowed in a PDFDocEncoded string:
```

To get around this problem you can use the

`\texorpdfstring{`⟨*TeX text*⟩`}{`⟨*outline text*⟩`}`

command. Its first argument, ⟨*TeX text*⟩, is the text to be displayed inside the document, while the second is the fallback for hyperref to use. An example would be to change

```
\section{\LaTeX{} is awesome!}
```

to

```
\section{\texorpdfstring{\LaTeX}{LaTeX} is awesome!}
```

While the above method may be necessary for some complicated math formulae that need to be nicely printed in the outline, usually the replacement texts are rather obvious for a given command. In this case you can use the

`\pdfstringdefDisableCommands{⟨commands⟩}`

command, which allows you to define general fallbacks for some commands. The ⟨*commands*⟩ argument is a list of redefinitions to be done when evaluating outline titles. Commands may be redefined using the `\RenewExpandableDocumentCommand` command. For a description of how to redefine the commands see subsection 7.1.1. An example of using this command would be

```
\pdfstringdefDisableCommands {
  \RenewExpandableDocumentCommand{\ldots}{}{...}
  \RenewExpandableDocumentCommand{\LaTeX}{}{LaTeX}
  \RenewExpandableDocumentCommand{\emph}{m}{*#1*}
}
```

5.4 Creating Presentations

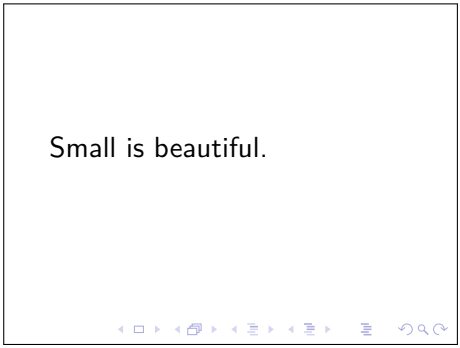
TEX and LATEX were primarily designed for creating text documents, and that is where they shine. Still, it is also possible to use them for creating presentations. In presentations created with LATEX you will be able to use all the LATEX features such as logical markup, mathematical typesetting and all the typesetting magic you are acustomed too. While these presentations are just PDF files, you may be amazed what is possible with the PDF in this respect.

Historically, there have been several ways to create presentations; for example, the standard LATEX slides class, or the powerdot package. While these are still supported, this booklet will focus on the beamer [74] package, which is the most popular option these days. This section only scratches the surface of beamer's capabilities. For a more in-depth tutorial please see the User Guide [74] distributed with beamer.

5.4.1 Basic Usage

The beamer package provides the beamer class, which loads all the necessary packages. The fundamental environment is the `frame` which adds a single page to your presentation.[2]

```
\documentclass{beamer}

\begin{document}
\begin{frame}
  Small is beautiful.
\end{frame}
\end{document}
```

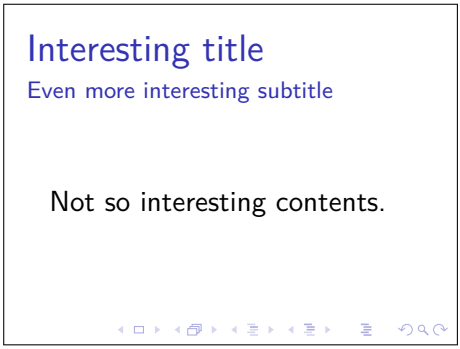


The full syntax of the `frame` is

```
\begin{frame}[⟨options⟩]{⟨title⟩}{⟨subtitle⟩}
```

where ⟨*options*⟩ is a key-value list applied to the frame, while ⟨*title*⟩ and ⟨*subtitle*⟩ are displayed at the top of the slide. Despite the curly brackets, these are optional arguments and have been omitted in the previous example.

```
\begin{frame}{%
  Interesting title}{%
  Even more interesting
  subtitle}

  Not so interesting contents.
\end{frame}
```

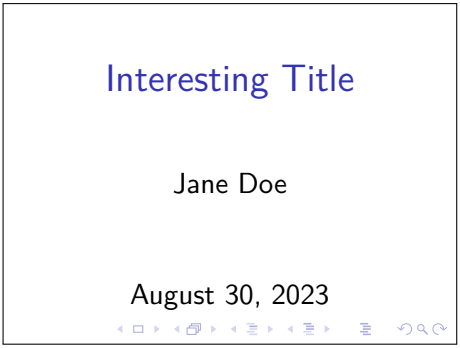


[2] Other presentation software often uses the term "slide" instead of frame, but slides are a slightly different concept in beamer.

The title and subtitle can also be specified using the `\frametitle` and `\framesubtitle` commands.

As in the standard classes, you can specify a title, author and date in the preamble by using the `\title`, `\author` and `\date` commands. The `\maketitle` command uses these settings to create a title frame.

```
\author{Jane Doe}
\title{Interesting Title}
\date{\today}
% ...

\maketitle
```

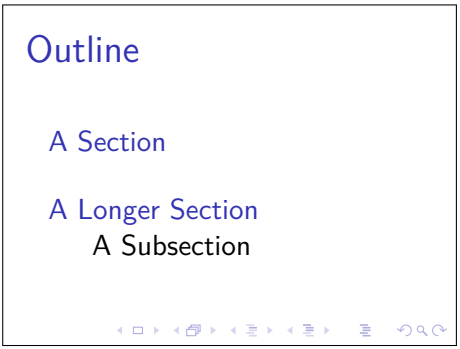


Additionally, `\subtitle`, `\institute` and `\titlegraphic` are available to add extra information to the title frame. The `\date` is often repurposed to hold the conference name, since it is more informative than the exact date. Some themes (discussed later) display these fields on each slide. If you find that they are too long for such use, you may specify shorter versions via an optional argument, like this:

```
\title[Short Title]{A Title That Is Too Long}
```

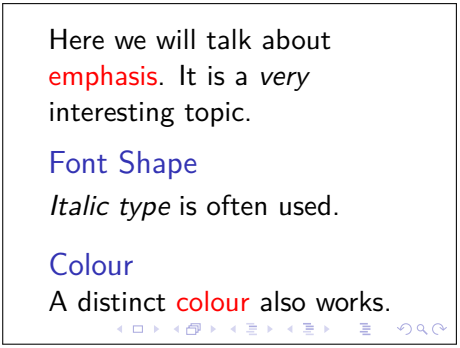
Sectioning commands, such as `\section`, are meant to be used between frames and define the structure of the presentation. Using them has no visible impact on the frames themselves, but they allow for a nice `\tableofcontents` with properly hyperlinked entries.

```
\begin{frame}{Outline}
  \tableofcontents
\end{frame}

\section{A Section}
\begin{frame}
  % ...
\end{frame}
\section{A Longer Section}
% ...
\subsection{A Subsection}
% ...
```



beamer provides special commands for emphasising information on the slide. The `\alert` command typesets its argument using a bright red colour. The `block` environment displays its contents with a title and separation from the rest of the text.

```
Here we will talk about
\alert{emphasis}. It is a
\emph{very} interesting topic.
\begin{block}{Font Shape}
  \emph{Italic type} is often
  used.
\end{block}
\begin{block}{Colour}
  A distinct \alert{colour}
  also works.
\end{block}
```



beamer redefines many amsmath[3] environments, including `theorem` and `proof`, to also produce blocks.

You will find that most of the LaTeX commands and environments, such as `\ref`s or `itemize`, work just as expected with beamer. A notable exception is the verbatim input and similar constructs described in Section 2.17. Using them requires you to pass the `fragile` option to the

[3] amsmath is loaded automatically by beamer.

frame options.

```
\begin{frame}[fragile]
  Inside \verb|fragile| frames
  you can use the verbatim
  freely.
  \begin{verbatim}
Hello!
  \end{verbatim}
\end{frame}
```

Inside fragile frames you can use the verbatim freely.

```
Hello!
```

5.4.2 Overlay Specification

So far, our frames have been static—they show all of their contents at once. It is, however, possible to show a frame piece-wise, in order to not overwhelm an audience with too much information.

The simplest way to do this is the \pause command. It splits the frame so that the content before the pause is presented on the current page, while the rest of the frame is presented on the next page. These partial frames are called slides.

```
\begin{frame}
  This will be shown first. \pause Then this. \pause
  And finally this.
\end{frame}
```

This will be shown first.	This will be shown first. Then this.	This will be shown first. Then this. And finally this.

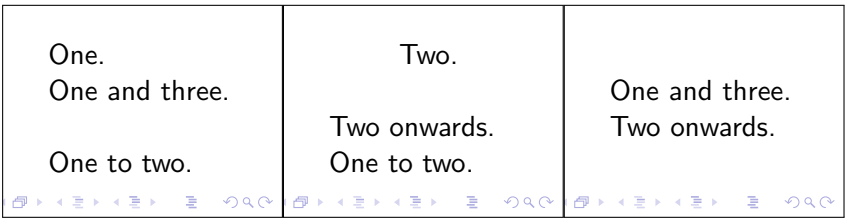
While presenting, this has the effect of revealing statements one by one.

A more powerful frame splitting command is exists:

`\onslide<`*⟨overlay specification⟩*`>`

The *⟨overlay specification⟩* argument specifies which slides the contents should appear on. It may be a single number (slides start at 1), a range of numbers (separated by `-`) or a comma delimited list of the two.

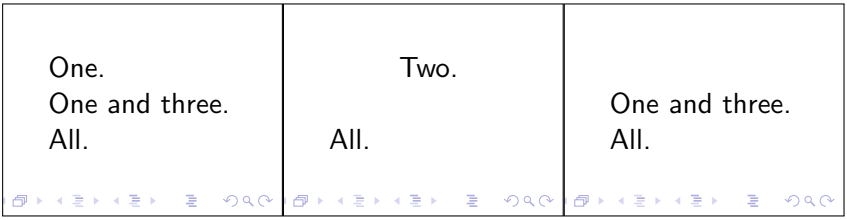
```
\begin{frame}
  \onslide<1> One. \onslide<2> Two. \onslide<1,3> One and
  three. \onslide<2-> Two onwards. \onslide<1-2> One to two.
\end{frame}
```



Internally, the `\pause` command uses `\onslide` with a counter, so these can be mixed together as desired (although care is required to set the slide numbers correctly).

A similar command is `\uncover`. It accepts the same optional *⟨overlay specification⟩* argument, but only applies it to its mandatory argument.

```
\begin{frame}
  \uncover<1>{One.} \uncover<2>{Two.} \uncover<1,3>{One
    and three.} All.
\end{frame}
```

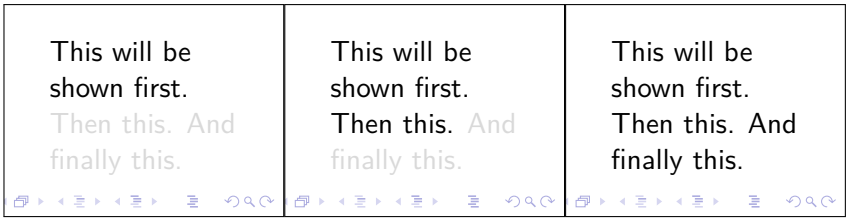


There are more commands that accept *⟨overlay specification⟩* argument, and many more preexisting commands and environments that are ex-

tended by beamer to support them. Refer to its User Guide [74] for more examples.

By default, content remains invisible until uncovered. If you don't want a lot of empty space on your slides, or if you prefer not to surprise your audience, you may adjust this behaviour to typeset the covered text as transparent. To do so, use the `\setbeamercovered` with the `transparent` option.

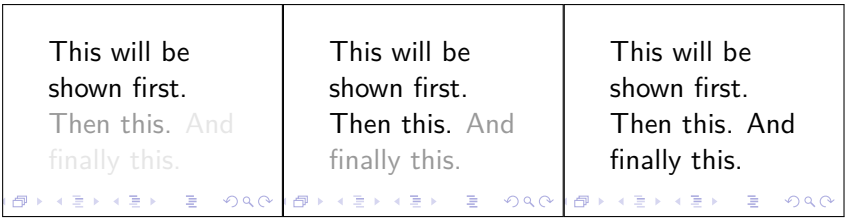
```
\setbeamercovered{transparent}
% ...

\begin{frame}
  This will be shown first. \pause Then this. \pause
  And finally this.
\end{frame}
```



Among the other possible arguments to these commands, `dynamic` makes the text increasingly transparent the later it is shown.[4]

[4]The effect is amplified a bit in the below example, so it is more visible. The default variant works better when there are more `\pause`s.

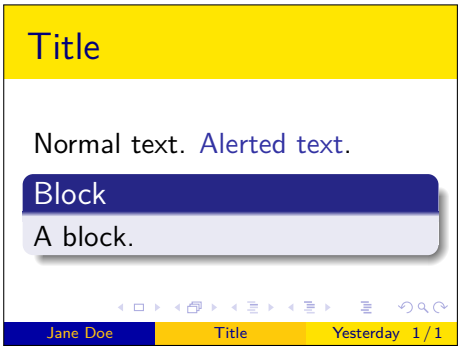
```
\setbeamercovered{dynamic}
% ...

\begin{frame}
  This will be shown first. \pause Then this. \pause
  And finally this.
\end{frame}
```



5.4.3 Customisation

If the default appearance of the presentation is not to your liking, beamer provides a lot of options to customise it. The most basic commands are \usetheme and \usecolortheme. These allow you to choose from a predefined set of themes that alter the style and colours of the presentation.

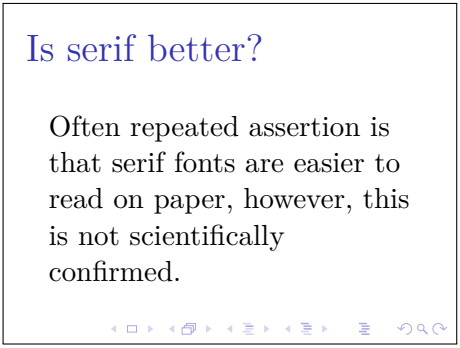
```
\usetheme{Madrid}
\usecolortheme{wolverine}
\author{Jane Doe}
\title{Title}
\date{Yesterday}
% ...

\begin{frame}{Title}
  Normal text.
  \alert{Alerted text}.
  \begin{block}{Block}
    A block.
  \end{block}
\end{frame}
```



The full list of available themes can be viewed at reference *Another Beamer Theme Matrix*.

The `\usefonttheme` command can be used in a similar fashion. By default the text in frames is typeset using a sans serif font. This choice makes sense, because it is easier to read on lower resolution projectors, but is less relevant when using a high-resolution display. You can pass the `serif` option to `\usefonttheme` to switch to the default LaTeX font.

```
\usefonttheme{serif}
% ...

\begin{frame}{Is serif better?}
  Often repeated assertion is
  that serif fonts are easier
  to read on paper, however,
  this is not scientifically
  confirmed.
\end{frame}
```

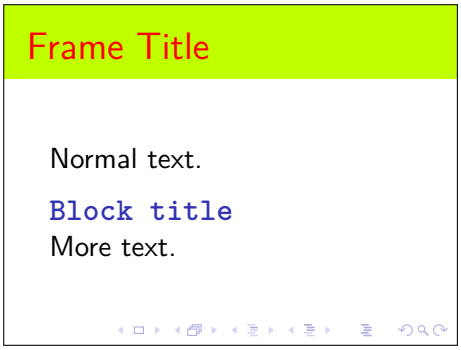


Greater customisation is possible using the

```
\setbeamerfont{⟨element⟩}{⟨attributes⟩}
\setbeamercolor{⟨element⟩}{⟨attributes⟩}
```

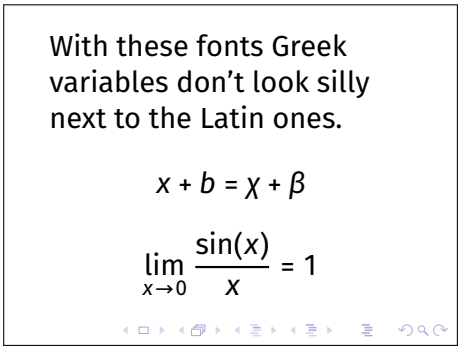
commands. The ⟨*element*⟩ argument is the element to which the customised font or colour should be applied, while the ⟨*attributes*⟩ specifies font attributes, such as `size`, `shape` and `family`, and the foreground and background colours, `fg` and `bg`. Fonts are further discussed in Section 7.3, while colours are covered in Section 7.4.

```
\setbeamercolor{frametitle}{
  fg=red,
  bg=lime,
}
\setbeamerfont{block title}{
  series=\bfseries,
  family=\ttfamily,
}
% ...

\begin{frame}{Frame Title}
  Normal text.
  \begin{block}{Block title}
    More text.
  \end{block}
\end{frame}
```



By default, LaTeX typesets mathematics in serif fonts. When beamer attempts the same typesetting in a sans serif font, not all symbols may exist (Greek letters, for example). If your presentation is mathematically heavy, it may be best to switch to a sans serif math font and matching text font. Switching math fonts is described further in subsection 7.3.4.

```
\usepackage{unicode-math}
\setsansfont{Fira Sans}
\setmathfont{Fira Math}
\setoperatorfont{\mathsf}
% ...

With these fonts Greek
variables don't look silly
next to the Latin ones.
\[ x + b = \chi + \beta \]
\[ \lim_{x \to 0}
   \frac{\sin(x)}{x} = 1 \]
```



By default, beamer typesets onto 4 : 3 aspect-ratio pages. If the projector uses some other aspect ratio, you may prefer to match it. You may do so, by passing an `aspectratio` to the `\documentclass` options. It accepts a single integer that encodes the desired ratio. For example, `169` is interpreted as 16 : 9, while `54` is interpreted as 5 : 4. Wider ratios

are especially useful if you want to incorporate a sidebar with the table of contents into your slides.

```
\documentclass[
  aspectratio=169
]{beamer}

\begin{document}
\begin{frame}
  Wide is beautiful.
\end{frame}
\end{document}
```

Wide is beautiful.

5.4.4 Handouts

A feature of beamer is the ability to easily create handouts. The simplest way to do this is to add the `handout` option to the `\documentclass` command. Doing so will make the beamer class ignore all of the `\pause`s, and similar commands, to produce documents with fewer pages.

```
\documentclass[handout]{beamer}

\begin{document}
\begin{frame}
  All of the \pause text is on
  \pause a single slide, \pause
  even though \pause pauses
  are present.
\end{frame}
\end{document}
```

All of the text is on a single slide, even though pauses are present.

This mode is also useful while creating a presentation, as it previews the frames as a whole without uncovering effects.

Internally, handout generation is accomplished by using overlay specifications with modes. The default mode is `beamer`, while the `handout` option switches to `handout` mode. You can specify modes explicitly within an overlay specification by passing ⟨*mode*⟩:⟨*spec*⟩. Several mode

specifications can be specified by separating them with | (vertical bar). If no mode is specified, beamer mode is assumed.

```
\documentclass[handout]{beamer}

\begin{document}
\begin{frame}
  Some text \pause with pauses.
  \onslide<beamer: 3-| handout: 2-> This text
  will be on a separate slide, even in handout.
\end{frame}
\end{document}
```

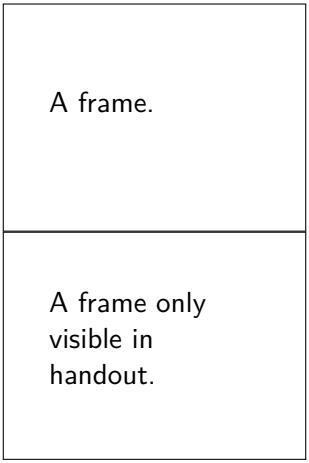
Some text with pauses.	Some text with pauses. This text will be on a separate slide, even in handout.

Often you will want to show some parts only in one mode—to provide additional commentary in handouts, or to hide things that only make sense during the presentation. This is easily achieved by using a special zeroth slide that is not included in the rendered document. An example of doing so is presented in Listing 5.1.

Handouts created in this way still use the slide structure—titles, lots of empty space, and bright colours. As an alternative, the article mode will typeset the document in a fashion similar to a regular article. Due to the more pronounced differences between layouts, this may require more tweaking than the handout mode, but the result will often be much more suitable for a printed handout.

Switching to the article mode differs from switching to other beamer modes. Rather than passing options to the class, the class is replaced by article and the beamerarticle is included in the preamble. See Listing 5.2 for an example. The beamerarticle package defines all of the beamer commands and environments, so that they will produce sensible output in the article.

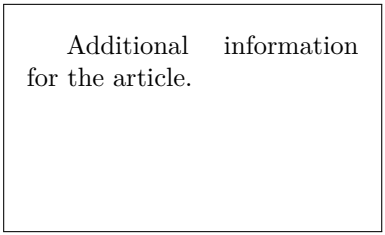
```latex
\documentclass[handout]{beamer}

\begin{document}
\begin{frame}
  A frame.
\end{frame}
\begin{frame}<handout: 0>
  A frame only visible in presentation.
\end{frame}
\begin{frame}<0>
  A frame only visible in handout.
\end{frame}
\end{document}
```



Listing 5.1: An example of using zeroth slides to hide content in presentation and in handout.

Because the concept of slides is not present in an article,[5] the overlay specifications will be useless. However the special zeroth slide works normally and can be used to hide content.

```latex
\begin{frame}<0>
  Additional information for
  the article.
  \uncover<article: 2->{This
    won't print.}
\end{frame}

\begin{frame}<article: 0>
  Additional information for
  the presentation.
\end{frame}
```



Because the overlay specification is often used to provide contents for a single mode, there exists a short cut: f you provide only the mode name, this will be the same as setting all other modes to zeroth slide

[5] To be more precise, only contents of the first slide are typeset.

```latex
% \documentclass{beamer}
\documentclass{article}

\usepackage{beamerarticle}

\title{Handouts}
\author{Me and Myself}

\begin{document}
\maketitle

\begin{frame}
  Handouts are \alert{very}
  important.
\end{frame}

\begin{frame}
  Almost as important as the
  presentations.
\end{frame}
\end{document}
```

Handouts

Me and Myself

August 30, 2023

Handouts are *very* important.

Almost as important as the presentations.

Listing 5.2: An example of using beamer in article mode.

and leaving the first slide for the specified mode. This usually results in a more readable code.

```
\begin{frame}<article>
  Content for the article.
\end{frame}

\begin{frame}<beamer>
  Content for the presentation.
\end{frame}
```

Content for the article.

Chapter 6

Graphics in Your Document

Most documents these days contain some graphics alongside the text. While photos and drawings can be easily added, integrating diagrams and schematics seamlessly with your document might prove difficult. Fonts, colours, and lines must be adjusted so that they do not look out of place. Furthermore, later changes to your document layout may force you to redo this work. Fortunately it is possible to create your graphics directly in LaTeX, which will automatically take care of adjusting the aforementioned settings and keep them in sync with the document itself.

6.1 Overview

First, some background on how to think about graphics. Roughly, there are three types of pictures that you may find yourself dealing with:

Photos are pictures that contain realistic shading and lots of detail. These include actual photos, photo-realistic renders, and screenshots from video games. In photos, exact pixel colour is not really important, and lossy compression can be used without visual degradation. It is best to store photos in the JPEG format.

Drawings are pictures with flat colours and relatively few details. This category includes pixel art and screenshots of program interfaces. Here, lossy compression would result in visible degradation, and

would not be very efficient anyway. It is best to use a lossless compression format such as PNG. This will ensure that each picture is reproduced with pixel-perfect fidelity, while keeping the file sizes small.

Diagrams or Charts are simple graphics that contain text, lines, and other geometric objects. Logos and schematics are prime examples of these. Ideally, they should be stored as drawing instructions in a vector based graphics in format, such as EPS, SVG or PDF. Vector based graphics can be scaled to any size without loss of quality.

You have already learned about including photos and drawings, in Section 2.19. You can use the same techniques to include diagrams, and this is totally fine. Some programs, such as Inkscape [29], even support ways to easily include produced graphics in LaTeX documents. However, this chapter will focus on drawing diagrams directly in LaTeX. This has many advantages, such as logical structuring, plain text formatting, and seamless integration with the document layout.

Creating graphical output with LaTeX has a long tradition. It started out with the `picture` environment, which allows you to create graphics by cleverly placing predefined elements onto a canvas. Unfortunately, this environment is not very robust, and so many alternatives have been developed over the years, such as metapost [80], asymptote [22] or xypic [62]. Today, the most widely used package is pgf [73] (short for "Portable Graphics Format") and its user interface Ti*k*Z (a recursive acronym—"Ti*k*Z ist *kein* Zeichenprogramm"—German for "Ti*k*Z is *not* a drawing program"). This chapter will introduce the basic concepts of writing graphics in Ti*k*Z. More detailed tutorials can be found in the pgf documentation.

While Ti*k*Z is a powerful and versatile language, using it for complicated graphics is not always easy. Many packages and libraries build upon the pgf foundation to simplify the creation of specialized diagrams. These include pgfplots [18], for plotting functions and presenting data, and commutative-diagrams [8], for creating commutative diagrams. It's usually a good idea to search CTAN for a package which already solves your problem before writing new Ti*k*Z code from the ground up, yourself.

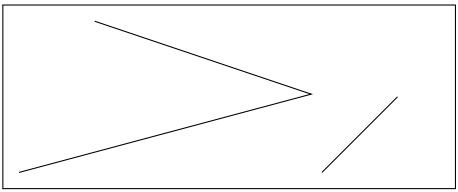
6.2 Basic Usage

To use pgf and Ti*k*Z, simply put the following line in the preamble:

```
\usepackage{tikz}
```

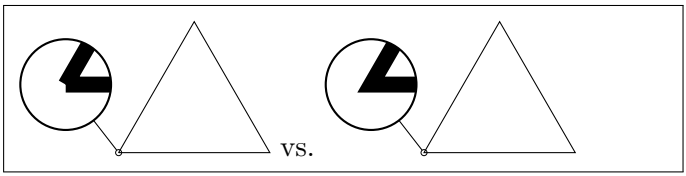
Doing so provides the `tikzpicture` environment and the `\tikz` command, inside of which you can execute Ti*k*Z commands. Each Ti*k*Z command is terminated with a semicolon (`;`). The simplest command is the `\draw` command, which draws points connected by a path. Points may be given in either Cartesian coordinates (x, y) or polar coordinates $(\theta{:}\, r)$, where θ is specified in degrees, while distances are specified in centimetres by default. The path between points may be specified in many ways, the simplest is `--`, which draws a straight line.

```
\begin{tikzpicture}
  \draw (0, 0)
    -- (15:4)
    -- (1, 2);
\end{tikzpicture}
\tikz{\draw (0, 0) -- (1,1);}
```



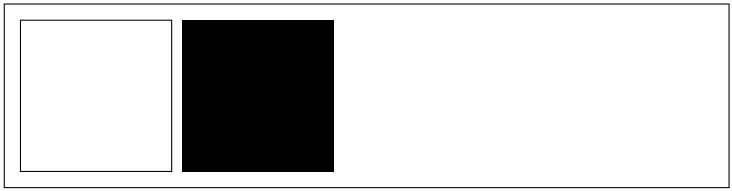
If you want to close the shape you are drawing, use `cycle` instead of repeating the first point. In addition to being clearer about the intention, it also produces a better looking line connection (a properly mitred join, as they say).

```
\begin{tikzpicture}
  \draw (0, 0) -- (2, 0) -- (60: 2) -- (0, 0);
\end{tikzpicture}
vs.\
\begin{tikzpicture}
  \draw (0, 0) -- (2, 0) -- (60: 2) -- cycle;
\end{tikzpicture}
```



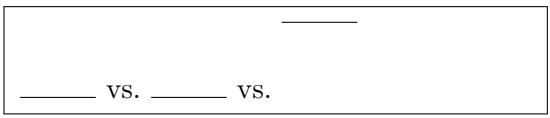
The `\draw` command is just a shortcut to the more general `\path` command. The latter accepts a path in the same way, but an action must be specified in square brackets. The `\draw` command simply supplies the `draw` action. The `fill` action fills the area under the specified path. Multiple actions may be supplied.

```
\begin{tikzpicture}
  \path[draw]
    (0, 0) -- (0, 2) -- (2, 2) -- (2, 0) -- cycle;
\end{tikzpicture}
\begin{tikzpicture}
  \path[fill]
    (0, 0) -- (0, 2) -- (2, 2) -- (2, 0) -- cycle;
\end{tikzpicture}
```



By default, Ti*k*Z pictures adjust their size to the minimal surrounding rectangle. If you want to specify the bounding box yourself, use the `use as bounding box` action, or the equivalent `\useasboundingbox` command.

```
\tikz{\draw (0, 0) -- (1, 0);}
vs.\
\tikz{\draw (0, 1) -- (1, 1);}
vs.\
\tikz{
  \useasboundingbox (0, 0) -- (0, 1)
    -- (1, 1) -- (1, 0) -- cycle;
  \draw (0, 1) -- (1, 1);
}
```

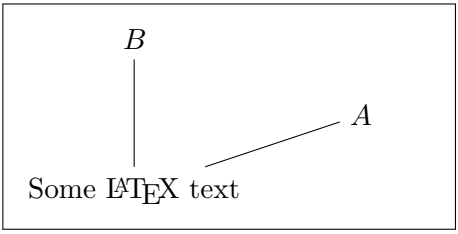


Normal LaTeX input may be displayed inside so-called nodes. To create a node, use the

```
\node (⟨name⟩) at ⟨coordinate⟩ {⟨input⟩};
```

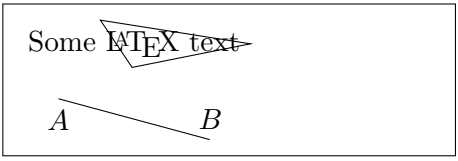
command. The ⟨*name*⟩ argument is optional and enables you to use the ⟨*name*⟩ as a shorthand for a node's coordinate.

```
\begin{tikzpicture}
  \node (text) at (0,0) {%
    Some \LaTeX{} text};
  \node (A) at (3, 1) {\(A\)};
  \node (B) at (0, 2) {\(B\)};
  \draw (A) -- (text) -- (B);
\end{tikzpicture}
```



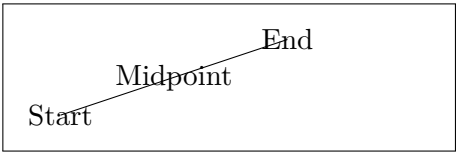
TikZ attempts to be smart about the way it positions lines between nodes. If you want to influence this, you can specify the exact connection-point on a node after a dot. The available points are, for example, `north`, `west`, `south east` and so on. You can also put a number, which will be interpreted as an angle (in degrees).

```
\begin{tikzpicture}
  \node (T) at (1,1) {%
    Some \LaTeX{} text};
  \node (A) at (0, 0) {\(A\)};
  \node (B) at (2, 0) {\(B\)};
  \draw (A.north) -- (B.south);
  \draw (T.0) -- (T.145)
    -- (T.265) -- cycle;
\end{tikzpicture}
```



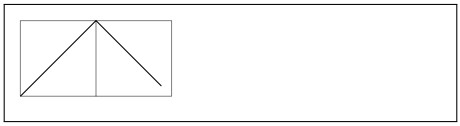
Nodes can be created within paths by typing `node` after a given coordinate or line. A node added after a line is placed at its midpoint.

```
\begin{tikzpicture}
  \draw (0, 0) node {Start}
    -- node {Midpoint}
    (3, 1) node {End};
\end{tikzpicture}
```



If you want to create an empty node for the purpose of naming a specific point, it is better to use the `\coordinate` command. This ensures that the node is actually empty, whereas nodes created by `\node` take up some space by default, even when they have no content.

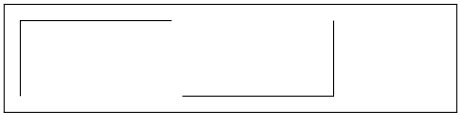
```
\begin{tikzpicture}
  \coordinate (A) at (0, 0);
  \coordinate (B) at (1, 1);
  \node (C) at (2, 0) {};
  \draw (A) -- (B) -- (C);
\end{tikzpicture}
```



6.3 Curves and Shapes

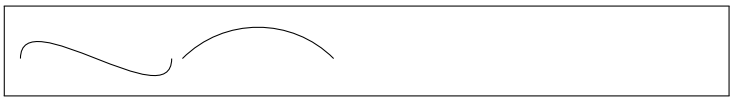
So far we have always used `--` to connect points. However, this is not the only way. For example, if you wanted to only use horizontal and vertical lines, you could specify `-|` or `|-` (according to preferred order) as the connection between points.

```
\tikz{\draw (0, 0) |- (2, 1);}
\tikz{\draw (0, 0) -| (2, 1);}
```



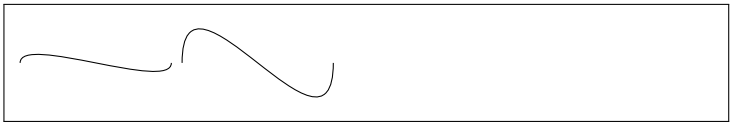
If you want to create curved lines, the simplest way is to use `to` between points. It works the same as `--`, but you can provide an optional argument, with `in` and `out` keys, to define the terminal angles of the connection.

```
\tikz{\draw (0, 0) to[out=90, in=-90] (2, 0);}
\tikz{\draw (0, 0) to[out=45] (2, 0);}
```



The `looseness` key may be used to define how much the curve is outstretched.

```
\tikz{\draw (0, 0)
  to[out=90, in=-90, looseness=0.5] (2, 0);}
\tikz{\draw (0, 0)
  to[out=90, in=-90, looseness=2] (2, 0);}
```

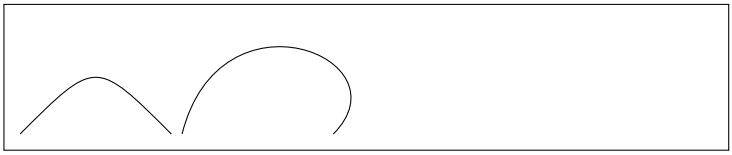


Often it might be easier to specify relative angles, using the `bend left` and `bend right` keys. If no angle is provided, a default value is used.

```
\tikz{\draw (0, 0) to[bend left] (2, 0);}
\tikz{\draw (0, 0) to[bend right=90] (2, 0);}
```

If you need even finer control over the curves you, can use a `.. controls ..` connection to specify a Bézier curve with one or two control points.

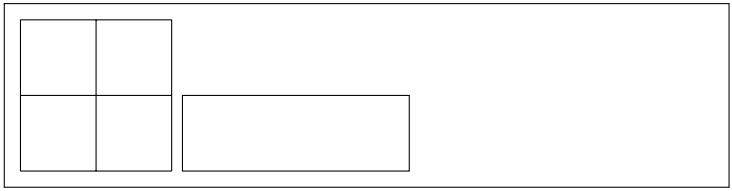
```
\tikz{\draw (0, 0) .. controls (1, 1) .. (2, 0);}
\tikz{\draw (0, 0) .. controls (.5, 2) and (3, 1)
  .. (2, 0);}
```



In addition to curves, the points may also be connected using various shapes. For example, the `grid` draws a grid between the points, while

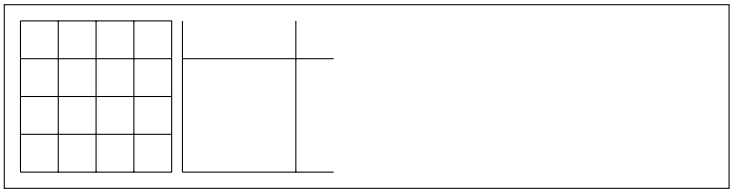
`rectangle` draws a rectangle.

```
\tikz{\draw (0, 0) grid (2, 2);}
\tikz{\draw (0, 0) rectangle (3, 1);}
```



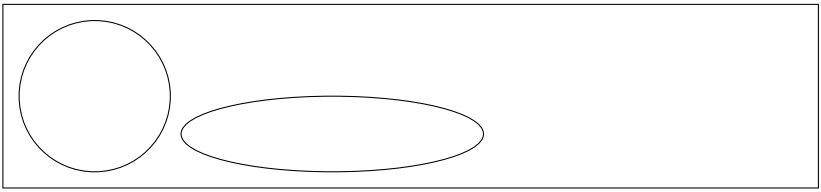
To specify how fine the grid is, use the `step` key.

```
\tikz{\draw (0, 0) grid[step=0.5] (2, 2);}
\tikz{\draw (0, 0) grid[step=1.5] (2, 2);}
```



Other shapes may also be drawn this way, sometimes requiring a specific syntax. For example the `circle` interprets the left coordinate as its centre and receives its radius via the `radius` key. You can also specify `x radius` and `y radius` separately, thus drawing an ellipse.[1]

```
\tikz{\draw (0, 0) circle[radius=1];}
\tikz{\draw (0, 0) circle[x radius=2, y radius=0.5];}
```



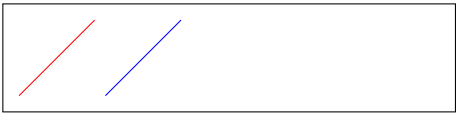
Many more shapes are available, such as `arc`, `parabola` or `sin`. Be sure to check out the documentation[73] for the specifics of their use.

[1] The `ellipse` shape can also be used in the same way if the naming irks you `;-)`.

6.4 Customizing Paths and Nodes

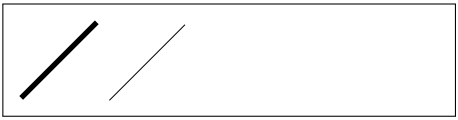
By default, all paths are drawn with a continuous black line. You can modify them by passing options to the `\draw` command. For example, passing a colour name will change the colour of the line.

```
\tikz{\draw[red]
  (0, 0) -- (1, 1);}
\tikz{\draw[blue]
  (0, 0) -- (1, 1);}
```



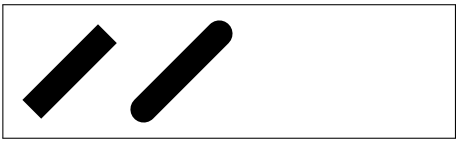
The thickness of a line can be controlled by passing the `line width` key (specified in points by default), or by using one of the predefined values, such as `semithick`, `very thin` or `ultra thick`.

```
\tikz{\draw[line width=2]
  (0, 0) -- (1, 1);}
\tikz{\draw[very thin]
  (0, 0) -- (1, 1);}
```



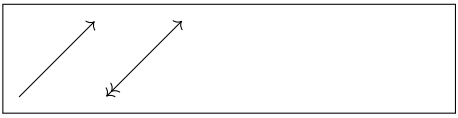
The endings of lines can also be customized. For example, `line cap` allows you to specify rounded end-caps.

```
\tikz{\draw[line width=10]
  (0, 0) -- (1, 1);}
\tikz{\draw[line width=10,
  line cap=round]
  (0, 0) -- (1, 1);}
```



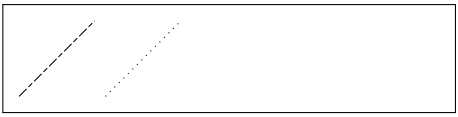
You can turn a line into an arrow by specifying the `arrows` key.

```
\tikz{\draw[arrows=->]
  (0, 0) -- (1, 1);}
\tikz{\draw[arrows=<<->]
  (0, 0) -- (1, 1);}
```



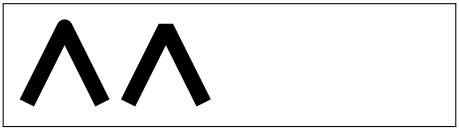
Lines do not need to be continuous, either. You can specify a `dash pattern` or use one of the predefined patterns.

```
\tikz{\draw[dash pattern=
  on 4 off 1 on 2 off 1]
  (0, 0) -- (1, 1);}
\tikz{\draw[dotted]
  (0, 0) -- (1, 1);}
```



Adjust how lines are connected by using `line join`. Set it to `round`, `bevel`, or `miter`.

```
\tikz{\draw[line join=round]
  (0, 0) -- (0.5, 1) -- (1, 0);}
\tikz{\draw[line join=bevel]
  (0, 0) -- (0.5, 1) -- (1, 0);}
```



If you want the line joins to be rounded, you can also pass the `rounded corners` key with the value set to the radius of the arc.

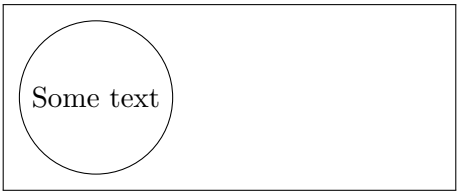
```
\tikz{\draw[rounded corners]
  (0, 0) -- (0.5, 1) -- (1, 0);}
\tikz{\draw[rounded corners=25]
  (0, 0) -- (0.5, 1) -- (1, 0);}
```

Now, let's turn our attention to nodes. By default, a node's boundary is not drawn, but you can pass `draw` and `fill`, optionally set to a colour, to reveal it.

```
\tikz{\node[draw] (0, 0)
  {Some Text};}
\tikz{\node[fill=red] (0, 0)
  {Some Text};}
```

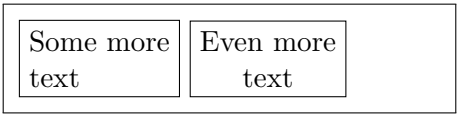
By default, all nodes are rectangles, but they can be changed to circles by passing the `circle` key to their options.

```
\tikz{\node[draw, circle]
  (0, 0) {Some text};}
```



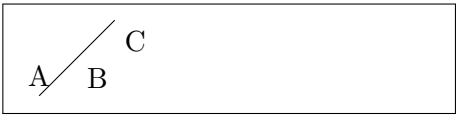
If you want to place multi-line text inside a node, you have to specify the `align` key; without it, new lines are ignored. Possible values are `left`, `center` and `right`.

```
\tikz{\node[draw, align=left]
  (0, 0) {Some more\\ text};}
\tikz{\node[draw, align=center]
  (0, 0) {Even more \\ text};}
```



When nodes are placed along a path, their positions may be adjusted using the `anchor` key. Its value is the point on the node boundary that should be anchored on the given point in path.

```
\tikz{\draw
  (0, 0) node[anchor=south] {A}
  -- node[anchor=north west] {B}
  (1, 1) node[anchor=135] {C};
}
```

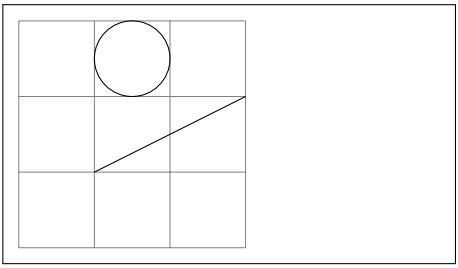


Using relative commands, such as `left` or `above right`, for this purpose usually leads to more readable code, though they are not so powerful.

```
\tikz{\draw
  (0, 0) node[above] {A}
  -- node[below right] {B}
  (1, 1) node[right] {C};
}
```

All these options can be freely combined, but the resulting style specification may turn out to be lengthy. To avoid retyping it for many nodes or paths, you can define new styles. Some predefined styles already exist. For example, the `help lines` style sets the colour of the lines to grey and makes them a bit thinner, which is useful for drawing alignment grids when constructing your own pictures.

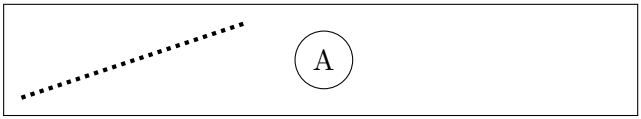
```
\begin{tikzpicture}
  \draw[help lines]
    (0, 0) grid (3,3);
  \draw (1, 1) -- (3,2);
  \draw (1.5, 2.5)
    circle[radius=0.5];
\end{tikzpicture}
```



To define your own style, pass a key of the form ⟨*name*⟩`/.style=`⟨*options*⟩

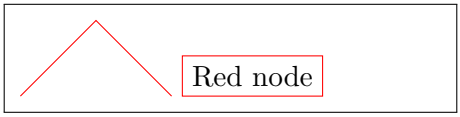
to the Ti*k*Z environment or command options.

```
\begin{tikzpicture}[
  my line/.style={dotted, ultra thick},
  my node/.style={draw, circle},
]
  \draw[my line] (0, 0) -- (3, 1);
  \node[my node] at (4, 0.5) {A};
\end{tikzpicture}
```



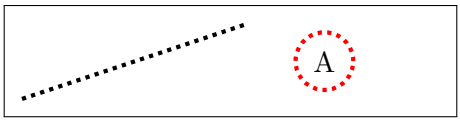
If you want to set up some styles globally, you can also use the `\tikzset` command.

```
\tikzset{
  red style/.style={draw=red},
}
\tikz{\draw[red style]
  (0, 0) -- (1, 1) -- (2, 0);}
\tikz{\node[red style]
  at (0, 0) {Red node};}
```



If you want to avoid specifying the style for every node or path within a Ti*k*Z picture, you can set the special styles `every node` and `every path` to change them all at once.

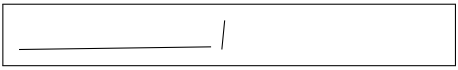
```
\begin{tikzpicture}[
  every path/.style={
    ultra thick, dotted},
  every node/.style={
    circle, draw=red},
]
  \draw (0, 0) -- (3, 1);
  \node at (4, 0.5) {A};
\end{tikzpicture}
```



6.5 Coordinates

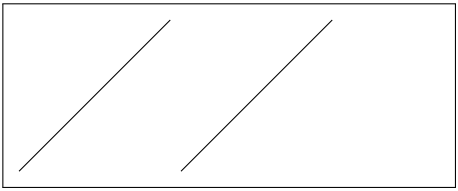
So far, we have always used the default unit of centimetres for specifying coordinates. However, it is possible to use any LaTeX dimension (see subsection 7.5.1 for details).

```
\tikz{\draw (0, 0)
  -- (1in, 1pt);}
\tikz{\draw (0, 0)
  -- (1dd, 1em);}
```



It is also possible to rescale how the distances are measured by using the `scale` key. This is useful if you find that your picture is too big or too small after drawing it, or if there exists some intuitive coordinate system (for example, when drawing function plots).

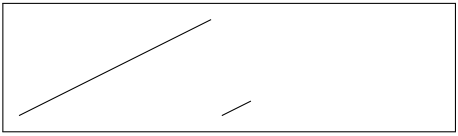
```
\tikz[scale=2]{\draw
  (0, 0) -- (1, 1);}
\tikz{\draw (0, 0) -- (2, 2);}
```



You can also specify `xscale` and `yscale` separately.

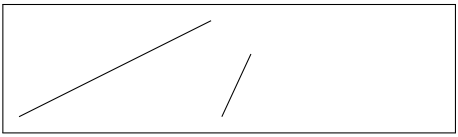
When specifying scale, you can use simple arithmetic operations. For example, to change the dimensionless values to inches, you can pass `scale=1in/1cm`.

```
\tikz[scale=1in/1cm]{\draw
  (0, 0) -- (1, 0.5);}
\tikz[scale=1em/1cm]{\draw
  (0, 0) -- (1, 0.5);}
```



Keep in mind that coordinates with dimensions will also be scaled, which may have unintended consequences. A more robust way of changing dimensionless values is the `x` and `y` keys.

```
\tikz[x=1in, y=1in]{\draw
  (0, 0) -- (1, 0.5);}
\tikz[x=1em, y=10ex]{\draw
  (0, 0) -- (1, 0.5);}
```



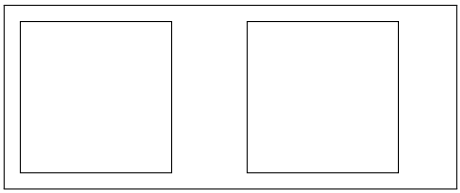
You can actually specify three-dimensional coordinates when drawing pictures. By default, the third coordinate is interpreted as a vector pointing 45 degrees to the bottom left and is a bit shorter.[2] This gives the effect of a parallel (or axonometric) projection.

```
\begin{tikzpicture}[scale=2]
  \draw (0, 0, 0) -- (0, 0, 1) -- (0, 1, 1) --
    (0, 1, 0) -- cycle;
  \draw (1, 0, 0) -- (1, 0, 1) -- (1, 1, 1) --
    (1, 1, 0) -- cycle;
  \draw (0, 0, 0) -- (1, 0, 0) (0, 0, 1) -- (1, 0, 1)
    (0, 1, 1) -- (1, 1, 1) (1, 1, 0) -- (0, 1, 0);
\end{tikzpicture}
```

Length and direction can be changed by using the `z` key.

Sometimes, it may be easier to specify a path by using coordinates relative to the previous point instead of absolute ones. To do so, prepend `++` to the coordinate, which will be interpreted as the previous specified point plus this vector.

```
\begin{tikzpicture}[scale=2]
  \draw (0, 0) -- ++(0, 1) --
    ++(1, 0) -- ++(0, -1) --
    cycle;
  \draw (1.5, 0) -- ++(0, 1) --
    ++(1, 0) -- ++(0, -1) --
    cycle;
\end{tikzpicture}
```



Scaling is not the only transformation that can be applied to points. It is also possible to rotate, shift, or apply an arbitrary linear transformation

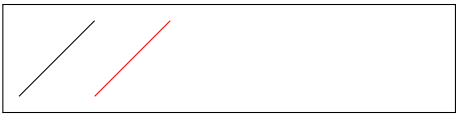
[2] Formally, $(0, 0, 1)$ is the same as $(-0.385, -0.385)$.

by specifying its matrix. Check out the documentation for a detailed description.

```
\begin{tikzpicture}[rotate=45]
  \draw (0, 0) -- (0, 1) --
    (1, 1) -- (1, 0) -- cycle;
\end{tikzpicture}
```

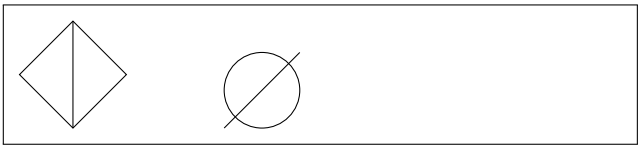
It is also possible to apply transformations to a single command.

```
\begin{tikzpicture}
  \draw (0, 0) -- (1, 1);
  \draw[red, xshift=1cm]
    (0, 0) -- (1, 1);
\end{tikzpicture}
```



If you want to apply the same transformation to more than one command within the same picture, you can use the `scope` environment. It is especially useful when your picture consists of more than one sub-picture, each of which is more or less independent.

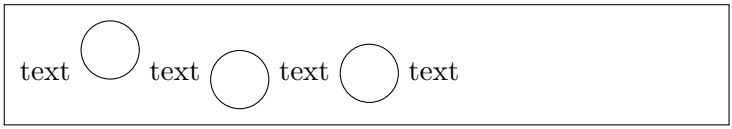
```
\begin{tikzpicture}
  \begin{scope}[rotate=45]
    \draw (0, 0) rectangle (1, 1);
    \draw (0, 0) -- (1, 1);
  \end{scope}
  \begin{scope}[xshift=2cm]
    \draw (0.5, 0.5) circle [radius=0.5];
    \draw (0, 0) -- (1, 1);
  \end{scope}
\end{tikzpicture}
```



When Ti*k*Z pictures are positioned within text, their bottom end sits on the baseline of the text. If you want to modify this, you can use the

`baseline` key to set the y coordinate at which the baseline should be. If no position is specified, it defaults to 0.

```
text \tikz{\draw (0, 0) circle [radius=1em];}
text \tikz[baseline]{\draw
  (0, 0) circle [radius=1em];}
text \tikz[baseline=-0.5ex]{\draw
  (0, 0) circle [radius=1em];} text
```

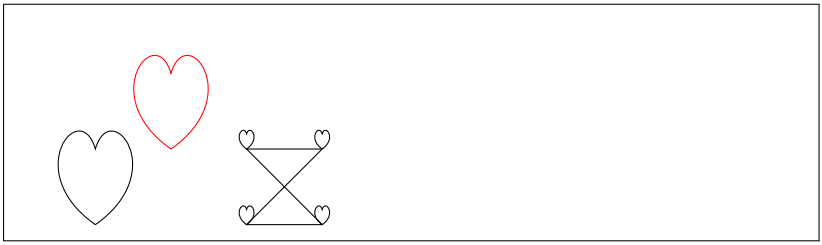


6.6 Reusing Pictures

Sometimes, you may wish to draw the same picture at a few places within a bigger one. While you could use the commands described in Section 2.9, this would be problematic, since modifying their placement or style is non-trivial. Ti*k*Z comes with its own method of defining smaller pictures, called 'pics'. They can be created by passing ⟨*name*⟩`/.pic=`⟨*commands*⟩ to the Ti*k*Z environment or command, and are then used by invoking the `\pic` command. An example is presented in Listing 6.1.

If you find yourself repeating a lot of simple commands (for example, drawing ticks on an axis), you may simplify your code by using the `\foreach` command. It repeats the drawing command for each value in a list, so that you can write repeatable code once, and need only modify the important parts.

```
\begin{tikzpicture}[
  heart/.pic={
    \draw (0,0) .. controls (-1, 0.7) and (-0.2, 1.7)
    .. (0, 1) .. controls  (0.2, 1.7) and (1, 0.7)
    .. (0, 0);
  },
]
  \pic at (0, 0) {heart};
  \pic[red] at (1, 1) {heart};

  \begin{scope}[xshift=2cm, every pic/.style={scale=0.2}]
    \draw (0, 0) pic {heart} -- (1, 1) pic {heart} --
      (0, 1) pic {heart} -- (1, 0) pic {heart} -- cycle;
  \end{scope}
\end{tikzpicture}
```



Listing 6.1: An example of using pics in Ti*k*Z.

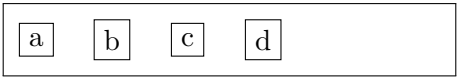
```
\begin{tikzpicture}
  \draw (0, 0) circle[radius=1cm];
  \foreach \i in {0, 60, 120, 180, 240, 300} {
    \draw (0, 0) -- (\i: 1);
    \fill (\i: 1) circle[radius=0.1cm];
  }
\end{tikzpicture}
```

If your values are a simple arithmetic sequence, you need only provide the first two values and the last, replacing the rest with triple dots.

```
\begin{tikzpicture}
  \draw (0, 0) circle[radius=1cm];
  \foreach \i in {0, 20, ..., 340} {
    \draw (0, 0) -- (\i: 1);
    \fill (\i: 1) circle[radius=0.1cm];
  }
\end{tikzpicture}
```

You can also iterate over pairs by separating respective parts with /.

```
\begin{tikzpicture}
  \foreach \i/\j in {0/a,
    1/b, 2/c, 3/d} {
    \node[draw] at (\i, 0) {\j};
  }
\end{tikzpicture}
```



6.7 Libraries

pgf and Ti*k*Z do not rely on LaTeX. In fact, they can be used with any TeX based system or even TeX itself. For this reason, Ti*k*Z provides its own system of extensions that does not use LaTeX's `\usepackage` command. Extensions of Ti*k*Z are called libraries, and can be loaded using the `\usetikzlibrary` command, which receives a list of comma separated libraries.

For example, if you intend to draw some geometry problems, you will often find yourself looking for intersections of objects. While you could calculate their exact coordinates by hand, the `intersections` library will do it for you. An example is presented in Listing 6.2.

Other libraries extend the number of available shapes. For example, the `arrows.meta` library defines numerous additional arrow tips, if you do not like the classical one. Some are presented in Listing 6.3.

You can perform additional calculations on existing coordinates. The `calc` library allows you to do so by enclosing them within `$` symbols. An example is presented in Listing 6.4.

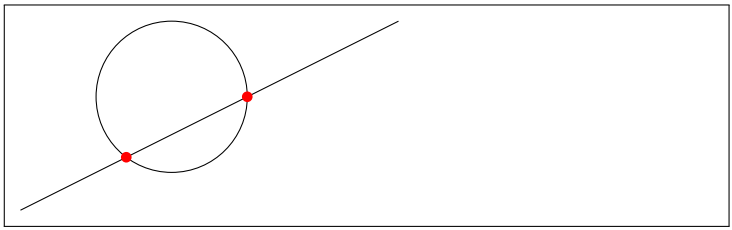
If you have a lot of Ti*k*Z code in your document, you may notice that compiling it takes much longer. This is because Ti*k*Z redraws each picture with every LaTeX pass. If this becomes annoying, you can cache your images to external files and reuse them on subsequent runs. To do this, simply put the following code in your preamble.

```
\usetikzlibrary{external}
\tikzexternalize
```

This method has some limitations, but should be sufficient for most uses. Read up on it in the documentation if problems occur.

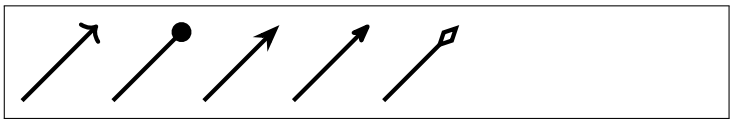
These are not the only libraries—additional node shapes, real 3D-perspective, matrices, mind maps and many more are covered. Most of them are described in the pgf [73] package documentation. Check it out if you haven't found solution to your problem here.

```
% In preamble
\usetikzlibrary{intersections}
% ...

\begin{tikzpicture}
  \draw[name path=O] (0, 0) circle [radius=1];
  \draw[name path=L] (-2, -1.5) -- (3, 1);

  \fill[red, name intersections={of=O and L}]
    (intersection-1) circle[radius=2pt]
    (intersection-2) circle[radius=2pt];
\end{tikzpicture}
```



Listing 6.2: An example of using `intersections` library.

```
% In preamble
\usetikzlibrary{arrows.meta}
% ...

\tikz{\draw[->] (0, 0) -- (1, 1);}
\tikz{\draw[-{Circle}] (0, 0) -- (1, 1);}
\tikz{\draw[-{Stealth}] (0, 0) -- (1, 1);}
\tikz{\draw[-{Stealth[round]}] (0, 0) -- (1, 1);}
\tikz{\draw[-{Diamond[open]}] (0, 0) -- (1, 1);}
```



Listing 6.3: Some of the arrow tips defined by `arrows.meta` library.

```
% In preamble
\usetikzlibrary{calc}
% ...

\begin{tikzpicture}
  \coordinate (A) at (1, -1);
  \coordinate (B) at (0, 1);
  \draw[red, ->] (0, 0) -- node[above] {\(A\)} (A);
  \draw[blue, ->] (0, 0) -- node[left] {\(B\)} (B);
  \draw[green, ->] (0, 0) --
      node[below right] {\(A+2B\)} ($(A)+2*(B)$);
\end{tikzpicture}
```

Listing 6.4: An example of using the `calc` library.

Chapter 7

Customising LaTeX

Documents produced with the commands you have learned up to this point will look acceptable to a large audience. While they are not fancy-looking, they obey all the established rules of good typesetting, which will make them easy to read and pleasant to look at.

However, there are situations where LaTeX does not provide a command or environment that matches your needs, or the output produced by some existing command may not meet your requirements.

In this chapter, I will try to give some hints on how to teach LaTeX new tricks and how to make it produce output that looks different from what is provided by default.

7.1 New Commands, Environments and Packages

At the beginning of this book we have mentioned that LaTeX allows us to write documents using logical markup, with commands like `\emph` or `\section`. There may be however situations where LaTeX does not provide an appropriate command for the content you want to write about. You may have noticed that all the commands I introduce in this book are typeset in a box, and that they show up in the index at the end of the book. There is no LaTeX markup to format example code or commands,

but LaTeX allows me to define my own commands for this purpose.

```
\begin{lscommand}
  \csi{dum}
\end{lscommand}
```

$\boxed{\texttt{\textbackslash dum}}$

In this example, a new environment called `lscommand` draws a box around the command, and a new command named `\csi`, typesets the command name and makes a corresponding entry in the index. Check this out by looking up the `\dum` command in the index at the back of this book, where you'll find an entry for `\dum`, pointing to every page where we mention the `\dum` command.

If I ever decide that I do not like having the commands typeset in a box any more, I can simply change the definition of the `lscommand` environment to create a new look. This is much easier than going through the whole document to hunt down all the places where I have used some generic LaTeX commands to draw a box around some word.

7.1.1 New Commands

You have already learned some basic command creation in Section 2.9. The main command is the

`\NewDocumentCommand{\`⟨*name*⟩`}{`⟨*argspec*⟩`}{`⟨*definition*⟩`}`

It requires three arguments: the ⟨*name*⟩ of the command you want to create, the ⟨*argspec*⟩ (argument specification) and the *definition* of the command.

The ⟨*argspec*⟩ argument specifies the number and types of arguments the command receives. The two most important types are `m`, for *mandatory* and `o` for *optional*. To create a command that takes two optional arguments, then two mandatory, then again one optional and finally three mandatory you would write `oommommm`. If the ⟨*argspec*⟩ argument is empty then the command will take no arguments, as you have already seen.

This example defines a new command called `\tnss`. This is short for "The Not So Short Introduction to LaTeX". Such a command could come

in handy if you had to write the title of this book over and over again.

```
\NewDocumentCommand{\tnss}{}{%
  The not so Short Introduction
    to \LaTeX}
This is \enquote{\tnss} \ldots{}
\enquote{\tnss}
```

This is “The not so Short Introduction to LaTeX” … “The not so Short Introduction to LaTeX”

The next example illustrates how to define a new command that takes two arguments. In order to refer to the received arguments you use `#1` for the first argument, `#2` for the second, and so on.

```
\NewDocumentCommand{\txsit}{mm}
 {This is the \emph{#1}
  #2 Introduction to \LaTeX}

% in the document body:
\txsit{not so}{short}

\txsit{very}{long}
```

This is the *not so* short Introduction to LaTeX
This is the *very* long Introduction to LaTeX

If your command accepts an optional argument, but the user does not supply one, a special marker `-NoValue-` will be inserted instead.

```
\NewDocumentCommand{\txsit}{om}
{This is the \emph{#1}
  #2 Introduction to \LaTeX}

% in the document body:
\txsit{definitive}

\txsit[very]{long}
```

This is the *-NoValue-* definitive Introduction to LaTeX
This is the *very* long Introduction to LaTeX

In order to test whether the user supplied a value, use

\IfValueTF{⟨*argument*⟩}{⟨*value version*⟩}{⟨*no value version*⟩}

macro.

```
\NewDocumentCommand{\MyCommand}{o}{
  \IfValueTF {#1} {
    Optional argument: #1.
  } {
    No optional argument given.
  }%
}

\MyCommand\\
\MyCommand[hello]
```

No optional argument given.
Optional argument: hello.

There are two variations of it: \IfValueT and \IfValueF which may be used if you only need output for one of the branches. The example with -NoValue- in the output could be fixed by writing

```
\NewDocumentCommand{\txsit}{om}
{This is the
  \IfValueT{#1}{\emph{#1} }%
  #2 Introduction to \LaTeX}

% in the document body:
\txsit{definitive}

\txsit[very]{long}
```

This is the definitive Introduction to LaTeX
This is the *very* long Introduction to LaTeX

The commands \IfNoValueTF, \IfNoValueT and \IfNoValueF work exactly the same, but the value/no-value branches are swapped.

Often you will want to use optional arguments when present but use some default values when the user does not provide them. This could be achieved by \IfValueTF, but with the O argument a default value can be set directly. It works like o but allows setting a default if no value is

supplied. Write

```
\NewDocumentCommand{\txsit}{O{not so}m}
{This is the \emph{#1}
  #2 Introduction to \LaTeX}

% in the document body:
\txsit{definitive}

\txsit[very]{long}
```

This is the *not so* definitive Introduction to LaTeX
This is the *very* long Introduction to LaTeX

Another useful argument specification is `s`, short for star. This argument allows providing different definitions based on whether the starred or non-starred version of command was issued by the user. It uses `\IfBooleanTF` command (and its variations) that works like the `\IfValueTF` command.

```
\NewDocumentCommand{\txsit}{sO{not so}m}
{This is the \emph{#2}
  #3 Introduction to \LaTeX%
  \IfBooleanT{#1}{%
    : Superstar Edition%
  }%
}

% in the document body:
\txsit{long}\\
\txsit*{long}
```

This is the *not so* long Introduction to LaTeX
This is the *not so* long Introduction to LaTeX: Superstar Edition

These are just the most common argument specifications. For a full description, take a look at the [57].

As we have discussed in Section 2.9, LaTeX will not allow you to create a new command that would overwrite an existing one, you can do so using `\RenewDocumentCommand`. It uses the same argument specification syntax as the `\NewDocumentCommand` command.

In some cases you might want to use the `\ProvideDocumentCommand` command. It works like `\NewDocumentCommand`, but if the command is already defined, LaTeX will silently ignore the new definition. Yet another variant is the `\DeclareDocumentCommand`. It always creates the given command, overwriting old definition if it exists.

7.1.2 New Environments

The \NewDocumentEnvironment command lets you create your own environments. It has the following syntax:

```
\NewDocumentEnvironment{⟨name⟩}{⟨argspec⟩}{⟨at begin⟩}{⟨at end⟩}
```

The ⟨*argspec*⟩ argument is the same as in the \NewDocumentCommand command. The contents of ⟨*at begin*⟩ and ⟨*at end*⟩ arguments will be inserted respectively when the commands \begin{⟨*name*⟩} and \end{⟨*name*⟩} is encountered. The example presented in Listing 7.1 illustrates the usage of this command.

Note that when environment argument are read, they are read *after* the \begin{⟨*name*⟩} command. This may be especially counterintuitive when we consider the `s` specification. Listing 7.2 illustrates this. If you want to create a starred version of an environment (similar to the $\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX environments) you have to define it separately.

```
\NewDocumentEnvironment{king}{moo} { ... } { ... }
\NewDocumentEnvironment{king*}{moo} { ... } { ... }
```

Obviously you can use the same internal commands to define them, for example renaming the previous implementation to `kinginternal` making the `king` and `king*` a thin wrapper around it.

The \NewDocumentEnvironment also introduces a special argument specification: `+b`, short for body.[1] It is only allowed as the last argument in the ⟨*argspec*⟩. It allows you to receive the body of the environment as an argument.

```
\NewDocumentEnvironment{twice}{+b} {%
  First time:\\ #1

  Second time:\\ #1
} {}

\begin{twice}
This will be printed twice!
\end{twice}
```

First time:
This will be printed twice!
Second time:
This will be printed twice!

[1] The `+` indicates that it may contain multiple paragraphs.

```
\NewDocumentEnvironment{king}{}{%
  \emph{Listen! For the
    king made a statement:}%
  \\[1em]%
} {%
  \\[1em]%
  \emph{This concludes
    the king's statement.}%
}

\begin{king}
My humble subjects \ldots
\end{king}
```

Listen! For the king made a statement:

My humble subjects …

This concludes the king's statement.

Listing 7.1: An example of using `\NewDocumentEnvironment` command.

```
\NewDocumentEnvironment{king}{s}{%
  \IfBooleanTF{#1}{
    \begin{center}
      \emph{Thus spoke Charles I:}
      \\[1em]%
    \end{center}%
  } {
    \emph{Listen! For the
      king made a statement:}%
      \\[1em]%
  }%
} {%
  \\[1em]%
  \emph{This concludes
    the king's statement.}%
}

\begin{king}*
My humble subjects \ldots
\end{king}
```

Thus spoke Charles I:

My humble subjects …

This concludes the king's statement.

Listing 7.2: An example of using the `s` specifier when defining a new environment.

While this makes one of the ⟨*at begin*⟩, ⟨*at end*⟩ arguments redundant, they are still required. (In the example above we provided an empty ⟨*at end*⟩.)

Do not overuse `+b` as this will both add limitations to the environments like `\verb` not being allowed inside and it will slow down the typesetting.

Similar to the `\NewDocumentCommand`, LaTeX makes sure that you do not define an environment that already exists. If you ever want to change an existing environment, use the `\RenewDocumentEnvironment` command. Its arguments are the same as the `\NewDocumentEnvironment` command.

7.1.3 Copying commands

When redefining commands you may want to use the original version of the command. Your initial code may look like this

```
\RenewDocumentCommand{\emph}{m}{%
  \emph{#1}~(\enquote{#1} is emphasised)%
}
```

but when you try to compile the document you will get the error message

```
! TeX capacity exceeded, sorry [input stack size=5000].
```

To understand why this happens it is instructive to consider how TeX expands the defined commands. The above `\RenewDocumentCommand` tells the TeX engine that whenever `\emph{foo}` is seen it must replace it with `\emph{foo}~(\enquote{foo} is emphasised)`. You may already see the problem here. In the next stage it will again replace the `\emph{foo}` yielding

```
\emph{foo}~(\enquote{foo} is emphasised)~(\enquote{foo} is
↪  emphasised)
```

This process will never end and at some point TeX simply gives up.

Note that

```
\NewDocumentCommand{\oldemph}{m}{\emph{#1}}
\RenewDocumentCommand{\emph}{m}{%
  \oldemph{#1}~(\enquote{#1} is emphasised)%
}
```

will suffer the same fate since `\oldemph{...}` will be replaced by TEX with `\emph{...}` and the cycle repeats.

In order to avoid this problem a special command exists

`\NewCommandCopy{\⟨name⟩}{\⟨command⟩}`

It makes the ⟨*name*⟩ the exact copy of the ⟨*command*⟩. The following example shows how this works

```
\NewDocumentCommand{\foo}{}{Batman!}

\NewDocumentCommand{\newfoo}{}{\foo}
\NewCommandCopy{\copiedfoo}{\foo}

\RenewDocumentCommand{\foo}{}{Na Na Na}

\foo{} \newfoo{} \copiedfoo{}
```

Na Na Na Na Na Na Batman!

This is precisely the behaviour we need in order to redefine the `\emph` command as we have tried to do earlier.

```
\NewCommandCopy{\oldemph}{\emph}
\RenewDocumentCommand{\emph}{m}{%
  \oldemph{#1}~(\enquote{#1}
  is emphasised)%
}

And here it \emph{comes}.
```

And here it *comes* ("comes" is emphasised).

7.1.4 Command-line LATEX

If you work on a Unix-like OS, you might be using Makefiles to build your LATEX projects. In that connection it might be interesting to produce different versions of the same document by calling LATEX with command-line parameters. If you add the following structure to your document:

```
\IfBooleanTF{\blackandwhite} {
  % "black and white" mode; do something...
} {
  % "color" mode; do something different...
}
```

Now compile document like this:

```
xelatex '\NewCommandCopy{\blackandwhite}{\BooleanTrue}
    \input{test.tex}'
```

First the command `\blackandwhite` is defined as the `\BooleanTrue` macro which holds a special value used in `\IfBooleanTF` checks. Then the actual file is read with input. By setting `\blackandwhite` to `\BooleanFalse` the colour version of the document would be produced.

7.1.5 Your Own Package

If you define a lot of new environments and commands, the preamble of your document will get quite long. In this situation, it is a good idea to create a LaTeX package containing all your command and environment definitions. Use the `\usepackage` command to make the package available in your document.

Writing a package basically consists of copying the contents of your document preamble (with minor adjustments) into a separate file with a name ending in `.sty`. There is one special command,

`\ProvidesExplPackage{`⟨*name*⟩`}{`⟨*date*⟩`}{`⟨*version*⟩`}{`⟨*description*⟩`}`

for use at the very beginning of your package file. This command tells the LaTeX to process the file in *expl mode*. The most visible effect of this is that all whitespace is ignored. You may have noticed that in many of the examples above we had to end most lines with `%` to get correct spacing in the output. In *expl mode*, spaces have to be added explicitly if needed at all. They are usually quite rare when writing a package. To insert spaces, use the `~` character, which normally denotes non-breaking space.[2] Paragraphs can be started with the `\par` command.

The arguments are used to provide information about package in the log file. If you use this package and look at the log file you will find

```
Package: demopack 2022-05-05 v0.1 Package by Tobias Oetiker
```

in the `.log` file.

`\ProvidesExplPackage` will also issue a sensible error message when you try to include a package twice. Listing 7.3 shows a small example package that contains the commands defined in the examples above.

[2] If you want to insert non-breaking space in *expl mode*, use `\nobreakspace`.

```
\ProvidesExplPackage{demopack}{2022-05-05}{0.1}{%
  Package by Tobias Oetiker
}

\NewDocumentCommand{\tnss}{} {
  The~not~so~Short~Introduction~to~\LaTeX
}
\NewDocumentCommand{\txsit}{O{not~so}} {
  The~\emph{#1}~Short~Introduction~to~\LaTeX
}

\NewDocumentEnvironment{king}{} {
  \begin{quote}
} {
  \end{quote}
}
```

Listing 7.3: Example Package.

7.2 Fonts and Sizes

7.2.1 Font Changing Commands

LaTeX fonts are influenced by four parameters

family The collection of fonts. For example, 'Latin Modern Roman' or 'Source Code Pro'.

series The weight of the font. For example, 'bold' or 'medium'.

shape The shape of glyphs within a font family. For example, 'small caps' or 'italics'.

size The size of the glyphs. For example, '10 pt' or '12 pt'.

LaTeX automatically chooses the appropriate font family, series, shape and size based on the logical structure of the document (sections, footnotes, emphasis, ...). It is possible however to instruct LaTeX manually which font to use. It is important to note that not every combination of family/series/shape exists as an actual font. LaTeX will complain if you try for something that does not exist.

LaTeX predefines three font families to use throughout the document: the upright or roman family accessible via `\textrm`, the sans serif family accessible via `\textsf` and monospace or typewriter family accessible via `\texttt`.

```
\textrm{Roman is the default
  in articles.} \\
\textsf{Sans serif is used in
  presentations.} \\
\texttt{Monospace is used in
  verbatim code blocks.}
```

Roman is the default in articles.
Sans serif is used in presentations.
`Monospace is used in verbatim code blocks.`

There are only two predefined LaTeX series: medium (`\textmd`) and bold (`\textbf`).

```
\textmd{The default.} \\
\textbf{Bold font.}
```

The default.
Bold font.

Shapes are a bit more complicated. The three basic shapes are: italics (\textit), oblique or slanted[3] (\textsl) and small capitals (\textsc).

```
\textit{Italic shape.} \\
\textsl{Slanted shape.} \\
\textsc{Small Capitals.}
```

Italic shape.
Slanted shape.
SMALL CAPITALS.

However there are two additional shapes that are not provided by default LaTeX fonts: swash (\textsw), for decorative fonts and spaced caps and small caps (\textssc). These are rarely used but may come in handy when using custom fonts as described in Section 7.3.

```
Question vs. \textsw{Question}
```

Question vs. Question

In addition two virtual shapes are provided: upright (\textup) and upper-lowercase (\textulc). These are not actually shapes but utility commands. The former one switches back to upright font while the latter disables small capitals. The command \textnormal is just the combination of the two.

```
\textsl{\textsc{Back to
  \textup{upright.}}} \\
\textsl{\textsc{Back to
  \textulc{lowercase.}}} \\
\textsl{\textsc{Back to
  \textnormal{normal.}}}
```

BACK TO UPRIGHT.
BACK TO lowercase.
BACK TO normal.

All of the commands described above also exist in their switch version. Instead of receiving the text via argument, they change the font permanently until it is changed again. For example, the switch version of \textit and \textrm are \itshape and \rmfamily, respectively. While

[3] Oblique shape differs from the italics in that italic shape uses different glyphs while oblique shape uses the same glyphs but slanted. The difference is really obvious when you look at the unslanted italic font.

the argument versions are useful for defining commands, switch versions are especially useful when defining your own environments.

```
Only \textit{argument} is
affected. After \itshape
everything is in italics
until \upshape is encountered.
```

Only *argument* is affected. After *everything is in italics until* is encountered.

Both argument and switch versions of the described commands are presented in Table 7.1.

When working with switch versions of fonts that are slanted right it is important to remember about italic correction. This is a small space after the end of right slanting text that is sometimes necessary to avoid overlapping letters. It is inserted using `\/` command.

```
Without: {\itshape oof}bar \\
With: {\itshape oof\/}bar
```

Without: *oof*bar
With: *oof*bar

The italic correction is handled automatically by the argument versions of the commands.

In contrast to the previous font changing commands, the size of font can only be controlled via switch versions. LaTeX predefines some switches for changing font size, see Table 7.2 and Table 7.3 for their description.

When using these commands it is important to remember that the line spacing is only updated after the paragraph ends. To avoid putting empty lines before the closing curly brace you may use the `\par` command.

```
{\Large Here the line spacing
  is not updated. A bit tight!}

{\Large Much better! I can
  breathe freely again!\par}
```

Here the line spacing is not updated. A bit tight!

Much better! I can breathe freely again!

Table 7.1: Default font changing commands of LaTeX.

Argument Command	Switch	Example
`\textrm{`⟨*text*⟩`}`	`\rmfamily`	roman
`\textsf{`⟨*text*⟩`}`	`\sffamily`	sans serif
`\texttt{`⟨*text*⟩`}`	`\ttfamily`	`typewriter`
`\textmd{`⟨*text*⟩`}`	`\mdseries`	medium
`\textbf{`⟨*text*⟩`}`	`\bfseries`	**bold face**
`\textup{`⟨*text*⟩`}`	`\upshape`	upright
`\textit{`⟨*text*⟩`}`	`\itshape`	*italic*
`\textsl{`⟨*text*⟩`}`	`\slshape`	*slanted*
`\textsc{`⟨*text*⟩`}`	`\scshape`	Small Caps
`\textsw{`⟨*text*⟩`}`	`\swshape`	Queen of Swash
`\textnormal{`⟨*text*⟩`}`	`\normalfont`	document font

Table 7.2: Commands changing font size.

Command	Size	Command	Size
`\tiny`	tiny	`\Large`	larger
`\scriptsize`	very small	`\LARGE`	very large
`\footnotesize`	quite small	`\huge`	huge
`\small`	small		
`\normalsize`	normal	`\Huge`	largest
`\large`	large		

Table 7.3: Absolute point sizes in standard classes depending on the class option. The default class option is `10pt`.

	Size (pt)		
Command	`10pt`	`11pt`	`12pt`
`\tiny`	5	6	6
`\scriptsize`	7	8	8
`\footnotesize`	8	9	10
`\small`	9	10	10.95
`\normalsize`	10	10.95	12
`\large`	12	12	14.4
`\Large`	14.4	14.4	17.28
`\LARGE`	17.28	17.28	20.74
`\huge`	20.74	20.74	24.88
`\Huge`	24.88	24.88	24.88

An arbitrary font size can be specified using the

`\fontsize{`⟨*size*⟩`}{`⟨*line skip*⟩`}\selectfont`

command combo. The ⟨*line skip*⟩ determines the height of the text line and should be usually around 1.2 times larger than the ⟨*size*⟩.

```
\fontsize{2cm}{2.4cm}\selectfont A big one!
```

A big one!

Fun fact: LaTeX default font is a bit unusual in that it looks slightly different depending on its size. The difference is presented in the table below where the text written using different sizes was rescaled to the same height.

\tiny	\normalsize	\Huge
Text	Text	Text

If you need to access even more font variants and shapes[4] check out *LaTeX 2ε font selection* [75].

7.2.2 Danger, Will Robinson, Danger

Note! Using explicit font setting commands defies the basic idea of LaTeX described in Section 1.6, which is to separate the logical and visual markup. The fonts should get switched automatically according to the requirements of the context. A simple rule of thumb: If you use the same font changing command in several places in order to typeset a special kind of information, you should use `\NewDocumentCommand` to define a "logical wrapper command" for the font changing command.

```
\NewDocumentCommand{\oops}{m}{%
 \textbf{#1}}
Do not \oops{enter} this room,
it's occupied by \oops{machines}
of unknown origin and purpose.
```

Do not **enter** this room, it's occupied by **machines** of unknown origin and purpose.

This approach has the advantage that you can decide at some later stage that you want to use a visual representation of danger other than `\textbf`, without having to wade through your document, identifying all the occurrences of `\textbf` and then figuring out for each one whether it was used for pointing out danger or for some other reason.

7.2.3 Advice

To conclude this journey into the land of fonts and font sizes, here is a little word of advice:

> **Remember!** *The* MO *RE* fonts YOU use **in** a document, *the* more READABLE and *beautiful it* becomes.

[4] For example the aforementioned upright italic shape.

7.3 Custom Fonts with fontspec

In the following examples we use Adobe Source fonts [20, 28, 27]. These fonts are included with TeXLive LaTeX distributions and should be available in the directory

```
.../texmf-dist/fonts/opentype/adobe
```

where the ... denotes the install-path of TeXLive. LuaTeX checks this directory automatically, so it should work fine, but if you are using XeTeX you must first install these fonts in your system. You can also download and install them manually from the links provided in the bibliography. Alternatively swap out their respective names with some other fonts installed in your system.

Many free OpenType fonts are available at https://fontlibrary.org/.

7.3.1 Main Document Fonts

If you are not pleased with the default Latin Modern font, you can change it to any font installed in your system using the fontspec [60] package. It provides three main commands for changing document fonts:

```
\setmainfont[⟨options⟩]{⟨font⟩}
\setsansfont[⟨options⟩]{⟨font⟩}
\setmonofont[⟨options⟩]{⟨font⟩}
```

This commands change, respectively, the main font of the document, the sans serif font used in the document and the monospace font in the document.

```
Normal text.
  \emph{Emphasised.} \\
\textsf{Sans serif text.
  \emph{Emphasised}.} \\
\texttt{Monospace text.
  \emph{Emphasised}.} \\

\setmainfont{Source Serif Pro}
\setsansfont{Source Sans Pro}
\setmonofont{Source Code Pro}
Normal text.
  \emph{Emphasised.} \\
\textsf{Sans serif text.
  \emph{Emphasised}.} \\
\texttt{Monospace text.
  \emph{Emphasised}.}
```

Normal text. *Emphasised.*
Sans serif text. *Emphasised.*
`Monospace text.` *`Emphasised.`*

Normal text. *Emphasised.*
Sans serif text. *Emphasised.*
`Monospace text.`
`Emphasised.`

Note that it is best to put these commands in the preamble of your document, because some fonts are frozen when the body starts.

The optional ⟨*options*⟩ argument accepts key value lists that allow to customise the font features. For example, many fonts contain, old style numerals that are not used by default. You can pass `Number=OldStyle` if you want to use them in your document.

```
\setmainfont{Source Serif Pro}
0123456789

\setmainfont[
  Numbers=OldStyle,
]{Source Serif Pro}
0123456789
```

0123456789
0123456789

Some fonts also provide special glyphs for a given language. For example the Latin Modern Font provides a special "fk" ligature for the Polish language. You can set the `Language` key to a given language to enable these features.

```
agrafka

\setmainfont[
  Language=Polish,
]{Latin Modern Roman}
agrafka
```

agrafka
agrafka

The polyglossia package activates these features automatically so you don't have to worry about them if you use it.

If your font supports it you may wish to enable automatic fractions insertion with `Fractions=On` key.

```
1/2 3/4 123/456

\setmainfont[
  Fractions=On,
]{Latin Modern Roman}
1/2 3/4 123/456

\setmainfont[
  Fractions=On,
]{Source Serif Pro}
1/2 3/4 123/456
```

1/2 3/4 123/456
½ ¾ 12¾56
½ ¾ 123⁄456

The OpenType font format defines a lot of more font features, that may or may not be supported by your font of choice. Consult with the fontspec [60] package documentation for a comprehensive description and examples.

7.3.2 Specifying Fonts via Filenames

If you do not want to install fonts in your system or you are working on a collaborative project where not everybody has the necessary fonts installed on their system, you can add font files to your project and specify the fonts directly via their filenames. In this case you must specify font variations manually. Because the filenames are usually very similar, it is possible to enter them using `*` patterns, where `*` is replaced by the main name defined. Extension may also be passed via the `Extension` key to avoid repetition. See Listing 7.4 for a comparison of font loading techniques. If the font files are not present in the same directory as the document you may have to specify it directly using the `Path` key.

The default LaTeX fonts are rather atypical in that they distinguish between *italics* and *slanted* font. Most fonts do not do this, so fontspec defines slanted font to be the same as italics. This may be fixed by

```latex
\setmainfont{Source Serif Pro}
Normal text. \textit{Italics.} \textbf{Bold.}
\textit{\textbf{Bold italics.}} \\

\setmainfont{SourceSerifPro-Regular.otf}
Normal text. \textit{Italics.} \textbf{Bold.}
\textit{\textbf{Bold italics.}} \\

\setmainfont[
  ItalicFont=SourceSerifPro-RegularIt.otf,
  BoldFont=SourceSerifPro-Bold.otf,
  BoldItalicFont=SourceSerifPro-BoldIt.otf,
]{SourceSerifPro-Regular.otf}
Normal text. \textit{Italics.} \textbf{Bold.}
\textit{\textbf{Bold italics.}} \\

\setmainfont[
  Extension=.otf,
  UprightFont=*-Regular,
  ItalicFont=*-RegularIt,
  BoldFont=*-Bold,
  BoldItalicFont=*-BoldIt,
]{SourceSerifPro}
Normal text. \textit{Italics.} \textbf{Bold.}
\textit{\textbf{Bold italics.}}
```

Normal text. *Italics.* **Bold.** ***Bold italics.***

Normal text. Italics. Bold. Bold italics.

Normal text. *Italics.* **Bold.** ***Bold italics.***

Normal text. *Italics.* **Bold.** ***Bold italics.***

Listing 7.4: Comparison of font loading with the fontspec package.

setting the `SlantedFont` key explicitly.

```
\setmainfont{Latin Modern Roman}
\textit{italics} vs. \textsl{slanted}

\setmainfont[
  SlantedFont=Latin Modern Roman Slanted,
]{Latin Modern Roman}
\textit{italics} vs. \textsl{slanted}
```

italics vs. *slanted*
italics vs. *slanted*

7.3.3 Defining New Fonts

So far we have only talked about changing the fonts for the whole document. It is possible however to define new fonts that are used only sporadically throughout the document, for emphasis or decorative purposes. It is possible to do so using the

`\newfontfamily{\`⟨*command*⟩`}[`⟨*options*⟩`]{`⟨*font*⟩`}`

It defines new ⟨*command*⟩ that works like to the `\rmfamily` or `\sffamily` commands.

```
\newfontfamily{\sourcefamily}[
  Numbers=OldStyle,
]{Source Serif Pro}

Normal text when suddenly
\ldots{} \sourcefamily
a different font! 0123456789
```

Normal text when suddenly … a different font! 0123456789

This is especially useful when working with multiple languages as you have already seen in Section 2.8.

The `\newfontfamily` checks whether the font family is already defined and raises an error if it is. As in subsection 7.1.1 the `\renewfontfamily` and `\providefontfamily` are available if you want to redefine existing font families.

7.3.4 Math Fonts

The package unicode-math, introduced in Chapter 3, uses fontspec under the hood and already enables you to use any OpenType math font within your document. The main command to do so is called `\setmathfont`. It accepts either a font name or a filename. In contrast to the text fonts that often consist of multiple files, math fonts typically consist of a single file, thus specifying it via a filename is not as complicated as presented in subsection 7.3.2.

```
\setmathfont{STIX Two Math}
```

is equivalent to

```
\setmathfont{STIXTwoMath-Regular.otf}
```

and the latter works in both LuaLaTeX and XeLaTeX. While changing math fonts throughout the document is possible, it may lead to some problems; prefer to set them in the preamble for the whole document.

```
\setmainfont{EB Garamond}
\setmathfont{Garamond Math}
% ...

Now we are using Garamond fonts.
\[
  \symrm{e}^{\symrm{\pi}
    \symrm{i}} + 1 = 0 \quad
  \sum_{i=0}^\infty \iint_a^b
  \lim_{h\to0}\frac{\sqrt[3]{
    \symbb{A}}}{2^h}\,\symrm{d}x
\]
```

Now we are using Garamond fonts.

$$\mathrm{e}^{\mathrm{\pi}\mathrm{i}} + 1 = 0 \quad \sum_{i=0}^{\infty} \iint_a^b \lim_{h\to0}\frac{\sqrt[3]{\mathbb{A}}}{2^h}\,\mathrm{d}x$$

Not all math fonts have the same character coverage. For example, the default font doesn't have lowercase script letters. If you don't want to switch the fonts entirely but just use some characters from a different font, you can use the `range` key in the options to the `\setmathfont`

command.

```
\(xyz = \symscr{Hello}\) vs.\
\setmathfont[
  range=scr,
]{STIX Two Math}
\(xyz = \symscr{Hello}\)
```

$xyz = \mathscr{H}ello$ vs. $xyz = \mathscr{H}\mathscr{e}\mathscr{l}\mathscr{l}\mathscr{o}$

You can also set it to exact Unicode ranges if you need more control over replaced symbols.

Some fonts define two types of script font *roundhand* and *chancery*. These are normally available as the first stylistic set feature of the font. You can map `\symcal` which is normally a synonym for `\symscr`, to produce the alternative script letters.

```
% TODO: Waiting for unicode-math fix
\setmathfont{STIX Two Math}
\setmathfont[
  range={cal, bfcal},
  StylisticSet=1,
]{STIX Two Math}

\(\symscr{ABCDabcd}\) vs.\
\(\symcal{ABCDabcd}\)
```

$\mathscr{A}\mathscr{B}\mathscr{C}\mathscr{D}\mathscr{a}\mathscr{b}\mathscr{c}\mathscr{d}$ vs. $\mathcal{A}\mathcal{B}\mathcal{C}\mathcal{D}\mathcal{a}\mathcal{b}\mathcal{c}\mathcal{d}$

The default behaviour of the `\not` command, is to combine the negating glyph with the following symbol. This usually produces satisfactory results. If the font defines a dedicated negated symbol it is probably better to use it in such situations. The `\not` command is able to use a predefined mapping to use such glyphs based on the negated symbol. If the default mapping does not contain the combination, or if you prefer to use a different negation you can create a new mapping by using `\NewNegationCommand`.

```
\(\not\cong\) vs.\
\NewNegationCommand{%
    \cong}{\simneqq}%
\(\not\cong\)
```

$\not\cong$ vs. $\ncong$

7.4 Colours

7.4.1 Coloured Text

In the Section 1.6 we have used different text colours to illustrate an example. These can be obtained with the xcolor [31] package. It provides three commands to change the colour of text:

```
\color[⟨model⟩]{⟨color⟩}
\textcolor[⟨model⟩]{⟨color⟩}{⟨text⟩} \mathcolor[⟨model⟩]{⟨color⟩}{⟨text⟩}
```

The `\color` is a switch version while `\textcolor` and `\mathcolor` only apply to their argument. If no ⟨*model*⟩ is specified, then ⟨*color*⟩ is specified as colour expression. The simplest colour expression is just the name of the colour, for example `yellow` or `red`.

```
\textcolor{yellow}{foo} \\
\color{red} baz
\[
  \mathcolor{blue}{
    \sum_{k=0}
  }^{10} i
\]
```

foo
baz

$$\mathcolor{blue}{\sum_{k=0}}^{10} i$$

The list of predefined colours can be found in Table 7.4. You can also pass `dvipsnames`, `svgnames` or `x11names` as a package options to extend the predefined colours. Consult the package documentation for a full list.

Another type of colour expression is a mix of two colours. The syntax is

⟨*first color*⟩`!`⟨*percentage*⟩`!`⟨*second color*⟩.

The resulting colour will be the result of mixing ⟨*percentage*⟩ % of the ⟨*first color*⟩ and 100 − ⟨*percentage*⟩ % of the ⟨*second color*⟩. If you omit

Table 7.4: Basic colours predefined by the xcolor package.

Name	Demo	Name	Demo	Name	Demo
black		lightgray		purple	
blue		lime		red	
brown		magenta		teal	
cyan		olive		violet	
darkgray		orange		white	
gray		pink		yellow	
green					

the ⟨*second color*⟩ it defaults to white.

```
\textcolor{green!100!red}{C}%
\textcolor{green!80!red}{o}%
\textcolor{green!60!red}{l}%
\textcolor{green!40!red}{o}%
\textcolor{green!20!red}{r}%
\textcolor{green!0!red}{s} \\
\textcolor{blue!100}{B}%
\textcolor{blue!75}{l}%
\textcolor{blue!50}{u}%
\textcolor{blue!25}{e}
```

Colors
Blue

Colour mixing is left associative so

$$\langle A\rangle\,!\,\langle n\rangle\,!\,\langle B\rangle\,!\,\langle m\rangle\,!\,\langle C\rangle$$

means calculate the mixture of ⟨A⟩ and ⟨B⟩ and then mixture of the result and ⟨C⟩. You can also use the minus sign before the expression to get the complementary colour.

```
\color{green!20!red!60!blue}
\LaTeX{} \\
\color{-green!20!red!60!blue}
\LaTeX{}
```

LaTeX
LaTeX

7.4.2 Models

While colour mixing via expression is useful for simple colour specification, it is often the case that we want to use colour that is defined in terms of its RGB or HSB values. Different input method colours can be specified using the optional ⟨*model*⟩ argument. Note that it is case-sensitive.

The simplest model is `Gray`. It accepts a single number from 0to 15 and produces a grey colour with the given brightness.

```
\textcolor[Gray]{0}{Zero}     \\
\textcolor[Gray]{3}{Three}    \\
\textcolor[Gray]{7}{Seven}    \\
\textcolor[Gray]{11}{Eleven} \\
\textcolor[Gray]{15}{Fifteen}
```

Zero
Three
Seven
Eleven

You can input RGB values in three ways: `rgb`, `RGB` and `HTML` models. The `HTML` model accepts a hexadecimal colour code. The code may be either upper or lowercase.

```
\textcolor[HTML]{e63946}{e63946}
\textcolor[HTML]{06D6A0}{06D6A0}
```

e63946 06D6A0

The `rgb` model accepts three decimal numbers, each between 0 and 1, while the `RGB` model accepts three integers from 0 to 255.

```
\textcolor[RGB]{255, 204, 102}{
  255, 204, 102
} \\
\textcolor[rgb]{0.4, 0.4, 1.0}{
  0.4, 0.4, 1.0
}
```

255, 204, 102
0.4, 0.4, 1.0

If you prefer the subtractive colour model, both `cmy` and `cmyk` are available. They accept decimal numbers between 0 and 1 to specify the

amount of each colour.

```
\textcolor[cmy]{0.7, 0.4, 0.3}{
  0.7, 0.4, 0.3
} \\
\textcolor[cmyk]{
  0.7, 0.4, 0.3, 0.5
}{
  0.7, 0.4, 0.3, 0.5
}
```

0.7, 0.4, 0.3
0.7, 0.4, 0.3, 0.5

There are three models that enable defining colours by HSB: `hsb`, `Hsb` and `HSB`. The first two accept three decimal numbers for each value, the difference being that the `Hsb` accepts hue as an angle in degrees, that is a number between 0 and 360. The `hsb` accepts it as a number between 0 and 1, while saturation and brightness are passed the same way in both model—as a number between 0 and 1.

```
\textcolor[hsb]{
  0.4, 0.8, 0.75
}{
  0.4, 0.8, 0.75
}\\
\textcolor[Hsb]{
  144, 0.8, 0.75
}{
  144, 0.8, 0.75
}
```

0.4, 0.8, 0.75
144, 0.8, 0.75

The `HSB` in turn accepts all three as integers—each between 0 and 240.

```
\textcolor[HSB]{
  144, 200, 120
}{
  144, 200, 120
}
```

144, 200, 120

If you are writing a paper about light you may also find that the `wave` model comes in handy. It allows you to specify a colour by its wavelength.

It accepts a single decimal number that represents a wavelength in visible spectrum in nanometres.

```
\textcolor[wave]{452}{
  If a light has wavelength
  \qty{452}{\nm} it looks
  like this.
} \\
\textcolor[wave]{700}{
  Light with wavelength above
  \qty{814}{\nm} is called
  infrared.
}
```

If a light has wavelength 452 nm it looks like this.
Light with wavelength above 814 nm is called infrared.

7.4.3 Defining Your Own Colours

If you want to use a given colour more than once it makes sense to define it as a macro. While you could use the `\NewDocumentCommand` to define it, the xcolor package provides a better way via the

`\definecolor{`⟨*name*⟩`}{`⟨*model*⟩`}{`⟨*value*⟩`}`.

command. Using it makes it possible to use the newly defined colour in colour mixing and such.

```
\definecolor{MyRed}{wave}{712}
\textcolor{MyRed}{MyRed is
  the perfect colour for you!}
\textcolor{MyRed!60}{Tints
  are also available!}
```

MyRed is the perfect colour for you! Tints are also available!

Be careful though, since it doesn't guard against redefinition. If you want to check whether you haven't redefined some colour put `\tracingcolors` in your preamble. This will produce warnings when redefinition happens.

If you want to make sure a colour is present but don't want to redefine it if it already exists then `\providecolor` does exactly that. There is also `\colorlet` that simply creates a copy of a given colour similar to the `\NewCommandCopy` command.

Colours defined in different models may need to be converted when mixing them. This may lead to a situation where `\color{a!75!b}` will result in different colour than `\color{b!25!a}`. Keep that in mind when mixing your own colours.

7.4.4 Colourful Pages and Boxes

So far we have only considered changing the text colour. It is however possible to also change the background colour of the document page. To do this use the

```
\pagecolor[⟨model⟩]{⟨color⟩}
```

command, which accepts the same arguments as the `\color` command. If you want to revert to the default transparent background you may do so with the `\nopagecolor` command.

```
\pagecolor{orange} \color{-orange}
Small is colourful \ldots?
```

Small is colourful …?

If you only want to specify a background of some text instead of the whole page you can use the

```
\colorbox[⟨model⟩]{⟨color⟩}{⟨text⟩}
\fcolorbox[⟨model⟩]{⟨color⟩}[⟨model⟩]{⟨color⟩}{⟨text⟩}
```

commands. The first one only colours the background, while the second one allows also drawing a frame (the 'f' stands for "framed").

```
It is \colorbox{gray}{curious}
how much a document can be
enhanced or ruined by
\fcolorbox{blue}{red}{
  colours.
}
```

It is curious how much a document can be enhanced or ruined by colours.

Boxes are explored further in Section 7.8.

7.5 Lengths and Spacing

7.5.1 LaTeX Units

Throughout this booklet we have often presented commands that accept length as one of its parameters such as `\\` or `\fontsize`. When introducing them we have used cm and pt which stand for centimetre and point, but these are not the only units available in LaTeX.

The most fundamental unit in LaTeX is sp which stands for *scaled point*. Its width is equal to $^1/_{65\,536}$ pt, where 1 pt is equal to $^1/_{72.27}$ of an international inch which in turn is defined as exactly 25.4 mm. All units in TeX are ultimately represented as a whole numbers of sp. See Table 7.5 for the exact values.

The last three units mentioned in the table are relative to the current font used. Historically they were related to the 'M' and 'x' glyphs in a given font but today they are arbitrarily set by fonts. These units are useful if we want the length to scale proportionally when used with different font sizes. The `em` unit is usually used for horizontal lengths, while the `ex` is used for vertical lengths. The `mu` unit can only be used in math mode for math spacing (see Section 3.7).

```
foo\\[1ex] bar

\tiny foo\\[1ex] bar
```

foo

bar

foo

bar

The desktop publishing point (DTP point) is the *de facto* standard point as used in most programs, and it is defined as $^1/_{72}$ in. For historical reasons the default TeX points are a bit smaller, while the DTP points are called "big points". While this shouldn't be noticeable in normal circumstances, remember to use `bp` if exact point values are required of you.

```
\fontsize{12pt}{15pt}\selectfont
Text in 12 \TeX{} points.

\fontsize{12bp}{15bp}\selectfont
Text in 12 DTP points.
```

Text in 12 TeX points.
Text in 12 DTP points.

7.5.2 Horizontal Space

LaTeX determines the spaces between words and sentences automatically. However, similarly to commands described in subsection 3.7.2, there are ways to influence the spaces in normal text. For example, to add horizontal space, you can use:

`\hspace{`⟨*length*⟩`}`

The ⟨*length*⟩ argument can be specified using the units described in

Table 7.5: LaTeX Units.

Unit	Meaning	Definition	Value (sp)	Demo
`cm`	centimetre	0.01 m	1 864 679	
`mm`	millimetre	0.001 m	186 467	
`in`	inch	25.4 mm	4 736 286	
`pt`	point	$^1/_{72.27}$ in	65 536	
`sp`	scaled point	$^1/_{65\,536}$ pt	1	
`pc`	pica	12 pt	786 432	
`dd`	didot	0.376 065 mm	70 124	
`cc`	cicero	12 dd	841 489	
`nd`	new didot	0.375 mm	69 925	
`bp`	big point	$^1/_{72}$ in	65 781	
`em`	roughly width of an 'M' in the current font			
`ex`	roughly height of an 'x' in the current font			
`mu`	equal to $^1/_{18}$ em, where em is taken from the current math font			

previous section in the usual way. You can use decimal and even negative numbers as values.

```
This\hspace{1.5cm}is a space
of \qty{1.5}{\cm}. A bit too
cramped\hspace{-5pt}here.
```

This is a space of 1.5 cm.
A bit too crampedhere.

The space added that way will disappear if it lands on the end of a line, similarly to an interword spacing. If you want to retain the extra space, use the starred version of the command.

```
The gap here is\hspace{1cm}%
\linebreak missing.

Here the gap is\hspace*{1cm}%
\linebreak not missing.
```

The gap here is
missing.
Here the gap is
not missing.

So far all the lengths we have seen have been *rigid*, that is the length is exactly as specified. But you probably noticed that the spaces between

words are not rigid—they can stretch and shrink, so that TEX can make the right margin equal. Lengths that can do that are called *rubber* lengths.

Such lengths can be specified using a special `plus` and `minus` syntax. For example to specify that a space can stretch if a need arises you can specify it by writing `plus` followed by the maximum allowed stretch.

```
This small\hspace{1em plus 2cm}%
space may grow if a need arises.

Here\hspace{1em plus 2cm}the
need\linebreak very much arises.
```

This small space may grow if a need arises.
Here the need very much arises.

The additional space can extend even beyond the specified maximum, but in such cases LATEX will print a warning.

The shrinking can be specified in a similar way using the `minus` syntax.

```
This\hspace{1em minus 2em}%
space may disappear if it gets
too crampy.
```

Thisspace may disappear if it gets too crampy.

When there are multiple rubber spaces in text TEX calculates the amount of 'stretch' proportionally to the specified maximum. Thus if TEX needs additional 2 cm of whitespace and one length has `plus 1cm` while the other has `plus 3cm` modifier, it will result in the first one being enlarged by $\left(\frac{1}{1+3}\right) \times 2\,\text{cm}$ while the second one by $\left(\frac{3}{1+3}\right) \times 2\,\text{cm}$.

```
This\hspace{0pt plus 3cm}is
stretched\hspace{0pt plus 1cm}%
three\linebreak times as much.
```

This is stretched three times as much.

With these informations you can use the spaces to automatically centre the text in a page by setting the allowed stretching to a high number.

```
\hspace*{0pt plus 100cm}Hello
\hspace*{0pt plus 100cm}
\linebreak
```

Hello

This approach will, however, interfere with the spacing inside the centred expression, since the spaces are still distributed proportionally. The effect will be getting smaller if you set the allowed stretching to a larger number but it will still be present. For situations like these, TeX actually supports a concept of infinitely stretchable space—by using the special `fill` unit, allowed only as stretching and shrinking value, we can ensure that all the other rubber lengths will not stretch.[5]

```
\hspace*{0pt plus 1fill}%
No\hspace{0pt plus 100cm}
stretching allowed.%
\hspace*{0pt plus 1fill}%
\linebreak
```

No stretching allowed.

Because the `fill` value is often used with zero width space LaTeX defines a macro that simplifies entering it—the

`\stretch{`$\langle n \rangle$`}`

command.

```
\hspace*{\stretch{1}}
is equivalent to
\hspace*{0pt plus 1fill}
\linebreak
```

is equivalent to

The $\langle n \rangle$ argument is the coefficient by which the `fill` is multiplied. Recall that the spaces are distributed proportionally and this is still the case when infinities are involved.

```
x\hspace{\stretch{1}}%
x\hspace{\stretch{3}}x
```

x x x

[5] TeX actually recognises three orders of infinity: `fil`, `fill` and `filll`, but as a document author you should stick to using only the second one. The first order infinity—`fil`—is used by some of the internal LaTeX such as `\\` or `\newpage`. The third one can be used to disallow stretching of the second order infinity when it's needed in some very rare circumstances.

Still, the most common value to use is `\hspace{0pt plus 1fill}` and so LaTeX defines `\hfill` that is equivalent to it. It's often used when you want to flush the rest of the line right.

```
Peter Pan\hfill Neverland

Dear Wendy, \ldots
```

Peter Pan Neverland
Dear Wendy, …

7.5.3 Vertical Space

You have already seen that vertical space between lines can be inserted using `\\` command. However, it does not work well when used for spacing between paragraphs—the reason being that it always starts a new line, so if it's used at the end of paragraph, it will end in an empty line.

```
Paragraph.\\

There's an empty line above.
```

Paragraph.

There's an empty line above.

LaTeX has a dedicated command for setting the space between paragraphs, the

`\vspace{⟨length⟩}`

command. It works similarly to the `\hspace` command, however when used inside a line it will only produce the space after the line is ended.

```
Some\vspace{1em} text that
spans multiple lines.
```

Some text that spans multiple

lines.

Since it does not produce empty lines when used, it is perfect for inserting a space between paragraphs. Similarly to the `\hspace` command, the space will be discarded if it lands at the end of a page—use the starred version if this is not desirable.

The rubber lengths, \stretch and \vfill work for vertical space too. However, since manual vertical spacing is much more common compared to manual horizontal spacing, LaTeX also declares three semantic commands for inserting them:

```
\bigskip
\medskip
\smallskip
```

The exact sizes of these skips is dependent on the class used. You can access them as length by appending "amount" to their name, for example, \medskipamount. These are useful when creating your own environments or to indicate a thought break between paragraphs.

There also exist the \addvspace command, which works similarly to the \vspace command, however when multiple such commands are entered one after another, only the one with the biggest length will be used. Note that using it will lead to an error if it is not used between paragraphs.

```
Hello.

\addvspace{1pt}
\addvspace{1em}
\addvspace{1cm}
There is \qty{1}{\cm} space
above me.
```

Hello.

There is 1 cm space above me.

This command is useful if you want to ensure that a vertical space is present but avoid entering several of them accidentally.

7.5.4 Length Variables

Like many things in LaTeX, lengths can be stored inside commands to allow reuse.

```
\NewDocumentCommand{%
  \mylength}{}{2em}
foo\hspace{\mylength}bar
```

foo bar

However, LaTeX provides dedicated length variables, which are much better suited for the purpose. These are created using

```
\newlength{\⟨variable⟩}
```

and set using

```
\setlength{\⟨variable⟩}{⟨length⟩}
```

It's important to note that `\newlength` declares the length globally, but `\setlength` only affects the current group. If you declare a length variable without setting it, it will have a default value of zero.

In contrast to the command approach, length variables are stored as numbers and not as text. This allows you to to do simple arithmetic operations, for example, to scale them by prepending them with a number.

```
\newlength{\mylength}
\setlength{\mylength}{2em}
foo\hspace{0.5\mylength}bar%
\hspace{2\mylength}baz
```

foo bar baz

In order to increase an existing length variable you can use `\addtolength`.

```
foo\hspace{\mylength}bar\\
\addtolength{\mylength}{2em}
foo\hspace{\mylength}bar
```

foo bar
foo bar

Since length variables are not stored as text macros, but as TeX internal numbers, trying to typeset them in a document results in an error. To translate them back into their textual representation use the `\the` command. Note that their value will always be printed using points as units.

```
\setlength{\mylength}{1cm}
\the\mylength
```

28.45274pt

Lengths can be also determined dynamically from the content. The following commands allow you to determine the width, height and depth

of a LaTeX text and assign it to length variable.

```
\settoheight{\⟨variable⟩}{⟨element⟩}
\settodepth{\⟨variable⟩}{⟨element⟩}
\settowidth{\⟨variable⟩}{⟨element⟩}
```

The element height is calculated by taking into account the part of the element that extends above baseline, while the depth takes into account the part that extends below baseline.

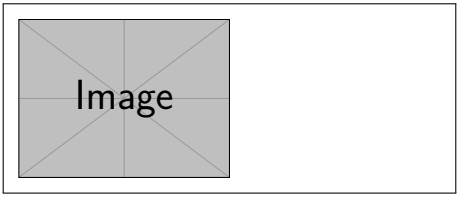
```
\newlength{\myheight} \settoheight{\myheight}{Major}
\newlength{\mydepth} \settodepth{\mydepth}{Major}
\newlength{\mywidth} \settowidth{\mywidth}{Major}
The word Major has width of \the\mywidth, height of
\the\myheight{} and depth of \the\mydepth.
```

The word Major has width of 28.94086pt, height of 7.47885pt and depth of 2.24475pt.

These commands are especially useful when creating your own environments requiring complicated alignment or spacing. An example of using them is presented in Listing 7.5.

As you will see LaTeX uses length variables for many things. When typesetting a document you can use some of them when setting lengths to make them relative. For example, the `\linewidth` length holds the length of the current line. You can use it when inserting a picture to make it fill the page or scale it to take up half the space available.

```
\includegraphics[
  width=0.5\linewidth,
]{example-image}
```



```
\newlength{\vardescindent}
\NewDocumentEnvironment{vardesc}{m}{%
  \settowidth{\vardescindent}{#1:\ }%
  \RenewDocumentCommand{\item}{so}{%
    \IfBooleanTF{##1}{#1: }{\\\hspace*{\vardescindent}}%
    \IfValueT{##2}{##2 ---}%
  }%
  \ignorespaces
}{}

\[  a^2+b^2=c^2 \]
\begin{vardesc}{Where}
  \item*[\(a\), \(b\)] are adjacent to the right angle
    of a right-angled triangle.
  \item[\(c\)] is the hypotenuse of the triangle and
    feels lonely.
  \item[\(d\)] finally does not show up here at all.
    Isn't that puzzling?
\end{vardesc}
```

$$a^2 + b^2 = c^2$$

Where: a, b — are adjacent to the right angle of a right-angled triangle.

c — is the hypotenuse of the triangle and feels lonely.

d — finally does not show up here at all. Isn't that puzzling?

Listing 7.5: An example of using `\settowidth` to align all of the definitions to a preceding phrase.

7.6 The Layout of the Document

7.6.1 Document Class Options

The easiest way to influence the layout of the document is to pass options to the

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

command at the beginning of your file. The available classes where already described in Table 1.1 on page 10. The ⟨*options*⟩ have to be separated by commas. The most common options for the standard document classes are listed in Table 7.6.

For example, if an input file for a LaTeX document starts with the line

```
\documentclass[11pt, twoside, a4paper]{article}
```

then it instructs LaTeX to typeset the document as an article with a base font size of eleven points, and to produce a layout suitable for double sided printing[6] on A4 paper.

If you do not like the appearance of the standard LaTeX classes, the easiest way to change it is to use some alternatives. For example, the koma-script [40] package provides alternatives that produce documents with European typography traditions in mind. Another popular package is memoir [85], which provides a single memoir class with extensive customization options. On CTAN you can find many more specialized classes that will let you produce documents as prescribed by various universities or typographical traditions.

7.6.2 Page Styles

LaTeX supports three predefined header/footer combinations—so-called page styles. The ⟨*style*⟩ parameter of the

```
\pagestyle{⟨style⟩}
```

command defines which one to use. Table 7.7 lists the predefined page styles.

[6] Note that this only influences the appearance of the document to be adequate for

Table 7.6: Document Class Options.

Options	Description
`10pt`, `11pt`, `12pt`	Sets the size of the main font in the document. If no option is specified, `10pt` is assumed.
`a4paper`, `letterpaper`	Defines the paper size. The default size can be configured when installing LaTeX. Besides these, `a5paper`, `b5paper`, `executivepaper`, and `legalpaper` can be specified. Note that it only affects the layout of margins, not the PDF paper size itself, see subsection 7.6.5 for more details.
`fleqn`	Typesets displayed formulae left-aligned instead of centred.
`leqno`	Places the numbering of formulae on the left hand side instead of the right.
`titlepage`, `notitlepage`	Specifies whether a new page should be started after the document title or not. The article class does not start a new page by default, while the report and book classes do.
`onecolumn`, `twocolumn`	Instructs LaTeX to typeset the document in one column or two columns.
`twoside`, `oneside`	Specifies whether double or single sided output should be generated. The classes article and report are single sided, while the book class is double sided by default. Note that this option concerns the style of the document only. `twoside` does *not* tell the printer to make a two-sided printout.
`landscape`	Changes the layout of the document to print in landscape mode.
`openright`, `openany`	Makes chapters begin either only on right hand pages o,r on the next page available. This does not work with the article class, as it does not know about chapters. By default, the report class by default starts chapters on the next page available, while the book class starts them on right hand pages.

Table 7.7: The Predefined Page Styles of LaTeX.

Style	Description
`plain`	Prints the page numbers on the bottom of the page, in the middle of the footer. This is the default page style.
`headings`	Prints the current chapter heading and the page number in the header on each page, while the footer remains empty. (This is the style used in this document.)
`empty`	Sets both the header and footer to be empty.
`myheadings`	Similar to the `headings` style but leaves the headers and footers empty, allowing them to be defined by the author. A description of how to do this is in Section 7.7.

It is possible to change the page style of the current page with the command

```
\thispagestyle{⟨style⟩}
```

You may also control the style of the displayed page numbers. To change it, use the

```
\pagenumbering{⟨style⟩}
```

command, where ⟨*style*⟩ is one of the styles presented in Table 7.8.

If would like more control over the apparance of your headers and footers, refere to Section 7.7 on page 260.

7.6.3 Line Spacing

When using custom fonts, you may find that the default line spacing does not match your taste. You could redefine it using the `\fontsize`

double sided printing—you still have to pass proper instructions to your printer to print it on both sides.

Table 7.8: Possible argument of the `\pagenumbering` command.

Style	Description
`arabic`	Arabic numerals (1, 2, 3, …)
`roman`	Lowercase Roman numerals (i, ii, iii, …)
`Roman`	Uppercase Roman numerals (I, II, III, …)
`alph`	Lowercase Latin letters (a, b, c, …)
`Alph`	Uppercase Latin letters (A, B, C, …)

commands, however, this will be overwritten once you use size changing commands described in Table 7.2.

To make such adjustments easier LaTeX defines the

`\linespread{`⟨*factor*⟩`}`

command. Once used, each line skip will be multiplied by the ⟨*factor*⟩. Note, that outside the preamble, you must follow it by `\selectfont` for the changes to be visible.

```
\linespread{0.9}\selectfont
If you want to save paper, set
the linespread to a value
below 1 to sacrifice a bit of
space between lines.

\linespread{1.1}\selectfont
On the other hand, if your
assignment is short a few
pages, setting it to a value
above 1 might just save you
some typing.
```

If you want to save paper, set the linespread to a value below 1 to sacrifice a bit of space between lines.

On the other hand, if your assignment is short a few pages, setting it to a value above 1 might just save you some typing.

Using the `\linespread` command it's also possible to create an effect of "one and a half" or "double" line spacing. Recall that the default line spacing is around 1.2 em. Thus if you want to make the document use double line spacing you have to set the ⟨*factor*⟩ to $^{2}/_{1.2} \approx 1.667$.

Note that setting the `\linespread` will change the line spacing *everywhere*—including footnotes, table of contents and floats. Doing so is

not always desirable, so another approach is to set the \baselineskip length. This will only affect the line spacing of the main document text. Similarly to the size changing commands it affects the whole paragraph.

```
{\setlength{\baselineskip}{%
  1.5\baselineskip}
This paragraph is typeset with
the baseline skip set to 1.5 of
what it was before. Note the
par command at the end of the
paragraph.\par}

Here the line spacing returns
to normal, because line skip
changes are local to a group.
```

This paragraph is typeset with the baseline skip set to 1.5 of what it was before. Note the par command at the end of the paragraph. Here the line spacing returns to normal, because line skip changes are local to a group.

Yet another approach is to use a dedicated package, like setspace [12] that defines \doublespacing and similar commands. Note that in general you should avoid using excessive line spacing, unless you are emulating an old document look.

7.6.4 Paragraph Formatting

In LaTeX, there are two lengths influencing paragraph layout: \parindent defines how much a paragraph is indented, while \parskip is the amount of space inserted between paragraphs.

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{%
  \medskipamount}

On the web it is common to
separate paragraphs by some
space instead of indenting.

Like this.
```

On the web it is common to separate paragraphs by some space instead of indenting.

Like this.

Beware, that these lengths also affect the table of contents—its lines get spaced more loosely. To avoid this, you might want to put the two commands after the \tableofcontents command or to not use them at

all, because you'll find that most professional books use indenting and not spacing to separate paragraphs.

If you want to indent a paragraph that is not indented, use \indent at the beginning of the paragraph. Obviously, this will only have an effect when \parindent is not set to zero. In continental Europe it is sometimes the case that every paragraph should be indented, even after sections. To avoid using ths \indent command everywhere, simply use the indentfirst [9] package in your preamble.

```
\usepackage{indentfirst}
% ...

\section{Title}

The first paragraph is now
indented.
```

> **1 Title**
>
> The first paragraph is now indented.

To create a non-indented paragraph, use \noindent as the first command of the paragraph. This might come in handy when you start a document with body text and not with a sectioning command.

7.6.5 Page Layout

As you have seen, LaTeX allows you to specify the paper size via options in the \documentclass command. It then automatically picks the right text margins, but sometimes you may not be happy with the predefined values. Naturally, you can customize them to your liking. But before you start making the margins as narrow as possible to cram the text in, take a few seconds to think. As with most things in LaTeX, there are good reasons for the page layout to be as it is.

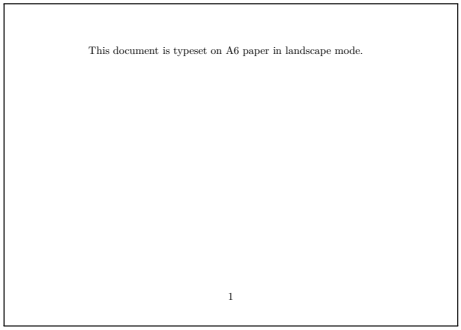
Sure, compared to your off-the-shelf page of your favourite WYSIWYG editor, the margins look awfully wide. But take a look at your favourite, professionally printed book and count the number of characters on a standard text line. You will find that most lines contain between 45 and 80 characters. Now do the same on your LaTeX page. You will find that the same relationship holds. Empirical studies suggest that averaging around 66 characters per line creates the optimal reading

experience for readers. If you want to save space when printing your document consider using `twocolumn` option or smaller page sizes.

With that warning in place let us proceed to the proper introduction of geometry [83] package that allows you to easily customize the page dimensions of your document. It is worth noting that simply including the package in your preamble will result in considerably narrower margins, (the very thing we warned you to avoid) so only use the package if you intend to set them manually or use the `pass` option that disables most of the package functions but retains the paper size adjustments.

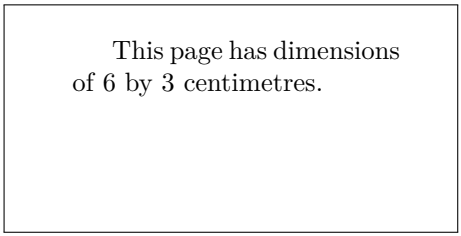
As we have mentioned in Table 7.6, simply setting `a5paper` (or similar) option will only adjust margins of the document without changing the paper dimensions themselves. The simplest way to fix that is to add the geometry package to your preamble. It will read the page size option and adjust it accordingly. The package itself also supports many more page sizes, such as `a0paper` or `a6paper`.

```
\documentclass{article}

\usepackage[
  a6paper,
  landscape,
]{geometry}

\begin{document}
This document is typeset on
A6 paper in landscape mode.
\end{document}
```



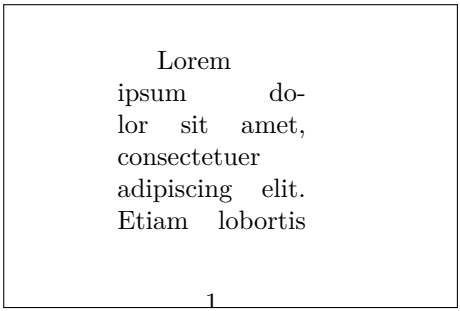
If the predefined dimensions are not enough you can always set it directly using `paperheight` and `paperwidth` keys.

```
\usepackage[
  paperwidth=6cm,
  paperheight=3cm
]{geometry}
% ...

This page has dimensions of
6 by 3 centimetres.
```



As we have indicated, the package is also capable of adjusting the margins. The simplest way to do so is to use the `left`, `right`, `top`, and `bottom` options, each of which sets the size of the respective margin.

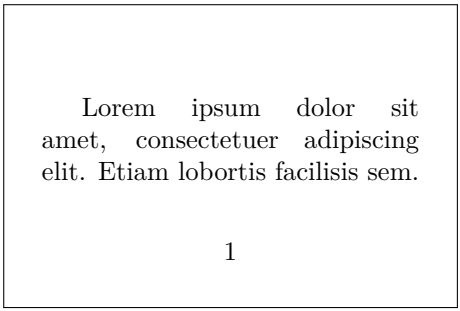
```
\usepackage[
  top=0.5cm,
  bottom=1cm,
  left=1.5cm,
  right=2cm,
]{geometry}
% ...
```



Note that by default the header and footer are not considered part of the page body, so they may not fit on page if the margins are too narrow (like in the example above). If you want to include them when considering margins use the `includefoot` and/or `includehead` options.

One of the killer features of the geometry package is its ability to calculate the correct margin sizes based on other metrics. For example, we can declare that we want the text to have a given width via `textwidth` option and that each page should have a given number of lines via `lines` option and the appropriate sizes of margins will be calculated automatically.

```
\usepackage[
  textwidth=5cm,
  lines=3
]{geometry}
% ...
```



There are many more options to influence the margins: specifying their ratios (`ratio`), defining them to take up certain percent of available

paper size (`scale`), or including binding offset (`bindingoffset`). Check out the package documentation [83] for a full list with examples.

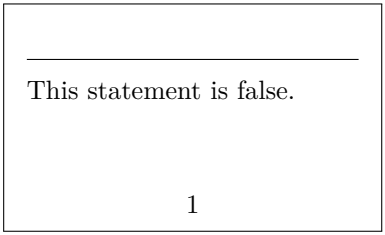
A useful option for prototyping your layout is the `showframe` option that draws frames around document body and margins to easier evaluate the chosen layout.

7.7 Fancy Headers

7.7.1 Basic commands

The fancyhdr [50] package provides a few simple commands that allow you to customise the header and footer lines of your document. It defines an additional page style `fancy` and a set of commands to customise it to your liking. By default, it only adds a line separating the header from the page body.

```
\documentclass{article}

\usepackage{fancyhdr}
\pagestyle{fancy}

\begin{document}
This statement is false.
\end{document}
```



The primary command of the package is

`\fancyhf[⟨places⟩]{⟨field⟩}`

The ⟨*places*⟩ is a comma separated list of places where the ⟨*field*⟩ should be displayed. There are total of 12 different places and each is identified by a combination of three letters:

- The first specifies whether the location is in the header (`H`) or in the footer (`F`).
- The second letter specifies the location within the header or footer: `L` for left, `C` for centre, and `R` for right.

- The third letter defines whether the field should be printed on even (E) or odd (O) pages. If the document is not two sided, then all pages are treated as odd.

For example, the combination FCE identifies the centre part of the footer on even pages. If any of the letters is omitted, then the identifier points toward all positions specifiable by the omitted letter. For example, HR is right side of the header on both odd and even pages.

```
\documentclass[twoside]{article}

\usepackage{fancyhdr}
\pagestyle{fancy}

\fancyhf[HCE]{A}
\fancyhf[L]{\emph{B}}
\fancyhf[FR]{\textbf{C}}
\fancyhf[HCO, HRE]{\textsl{D}}

\begin{document}
The next statement is false.
The previous statement is true.
\end{document}
```

B *D*

The next statement is false. The

B 1 **C**

B A *D*

previous statement is true.

B 2 **C**

There are two additional commands, \fancyhead and \fancyfoot, that work in the same way, except that they respectively assume H and F in their ⟨*places*⟩ argument, unless otherwise specified.

```
\fancyhead[L]{A}
\fancyfoot[R]{B}

\enquote{Yields a falsehood when
appended to its own quotation}
yields a falsehood when appended
to its own quotation.
```

A

"Yields a falsehood when appended to its own quotation" yields a falsehood when appended to its own quotation.

1 B

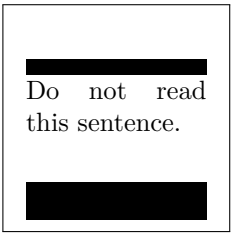
The lines drawn by the fancyhdr package may also be customised. To

change their thickness, redefine the

```
\headrulewidth
\footrulewidth
```

macros to the desired size.

```
\RenewDocumentCommand{\headrulewidth}{}{.2cm}
\RenewDocumentCommand{\footrulewidth}{}{.5cm}

Do not read this sentence.
```

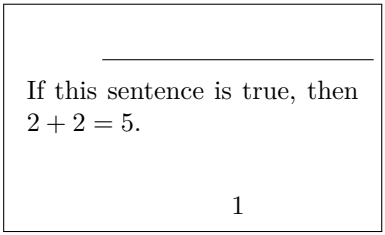


By default, headers and footers are as long as the text on the page. If you want to extend or shorten them, use the

```
\fancyhfoffset[⟨places⟩]{⟨offset⟩}
```

Added comma after `\fancyhf`. command. The ⟨*places*⟩ argument is the same as in `\fancyhf`, except that it cannot contain `C`.

```
\fancyhfoffset[L]{-1cm}
\fancyhfoffset[R]{.2cm}

If this sentence is true,
then \(2 + 2 = 5\).
```

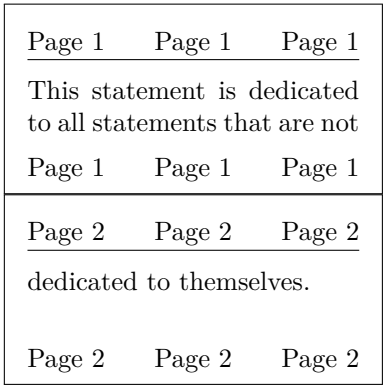


The command `\fancyheadoffset` and `\fancyfootoffset` are used the same way as `\fancyf`, but they only modify header or footer respectively.

7.7.2 Contents of the headers

The default footer of the article class contains the current page number. To use it inside the fancy header, simply use the command `\thepage`.

```
\fancyhf{Page~\thepage}

This statement is dedicated to
all statements that are not
dedicated to themselves.
```



It is often useful to have the header and footer contain information based on the content of the page. These are called "marks" in LaTeX terminology. Before we talk about the default ones, let's consider how you can define your own using the extramarks[7] package.

```
\extramarks{⟨left⟩}{⟨right⟩}
\firstleftxmark
\firstrightxmark
\lastleftxmark
\lastrightxmark
```

The `\extramarks` command sets the contents of ⟨*left*⟩ and ⟨*right*⟩ marks. Then you can access these marks inside the headers by using the appropriate command. The `first-` commands refer to the first mark occurring

[7] extramarks is part of fancyhdr [50].

on the page, while the `last-` refer to the last one.

```
\usepackage{extramarks}

\fancyhead[L]{\firstleftxmark}
\fancyhead[R]{\lastleftxmark}
\fancyfoot[L]{\firstrightxmark}
\fancyfoot[R]{\lastrightxmark}

The second statement is false.
\extramarks{One}{2 is false}
The third statement is false.
\extramarks{Two}{3 is false}
The first statement is false.
\extramarks{Three}{1 is false}
```

One Three

The second statement is false. The third statement is false. The first statement is false.

2 is false 1 1 is false

Let us now look at the default marks defined by LaTeX. After loading extramarks, these may be set and accessed similarly:

```
\markboth{⟨left⟩}{⟨right⟩}
\firstleftmark
\firstrightmark
\lastleftmark
\lastrightmark
```

The only difference between these and the extra marks is that LaTeX classes automatically fill them. Hence, if you are not careful, you may lose your content.[8] For example, in the article class, ⟨*left*⟩ is set by the `\section` command, while ⟨*right*⟩ is set by the `\subsection` command.

[8] You may prevent this by setting the pagestyle to `myheadings` and then redefining it to use it with fancyhdr as described later.

```
\fancyhead[L]{\firstleftmark}
\fancyhead[R]{\lastleftmark}
\fancyfoot[L]{\firstrightmark}
\fancyfoot[R]{\lastrightmark}

\section{First}
\subsection{Sub}
\section{Second}
\subsection{Sub}
```

1 FIRST 2 SECOND

1 First

1.1 Sub

2 Second

2.1 Sub

1 2.1 Sub

Note that `\firstrightmark` is empty in the above example. This is caused by the fact that `\section` commands set both marks, leaving the right one empty.

You may have noticed that section titles are typeset in uppercase when provided by `-mark` commands. To disable this use the `\nouppercase` command inside the `\fancyhf` command.

```
\fancyhead[R]{%
  \nouppercase{\firstleftmark}%
}

\section{Today}
is opposite day.
```

1 Today

1 Today

is opposite day.

1

7.7.3 Advanced commands

If you want even more control over section titles, you can redefine the `\sectionmark`.[9] The command receives the section title as its first

[9] Or `\chaptermark`, `\subsectionmark` ...

argument, while the section number is available as \thesection.

```
\fancyhead[R]{\firstleftmark}
\RenewDocumentCommand{\sectionmark}{m}{%
  \markboth{%
    Section no.\,\thesection: #1}{}%
}

\section{Person}
There exists a person such that
if they are reading this then
everybody is reading this.
```

Section no. 1: Person

1 Person

There exists a person such that if they are reading this then everybody is reading this.

1

If you only want to change the right mark, use the \markright command.

By default, the field in the centre of the header or footer will expand equally to the left and right. This is usually the desired behaviour, but in some cases it may overlap with either left or right text, while still having some space on the other side.

```
\fancyhead[L]{\thepage}
\fancyhead[C]{\firstleftmark}
\fancyhead[R]{Jane Doe}

\section{Section}
Section by Jane Doe
```

1 1 SECTION Jane Doe

1 Section

Section by Jane Doe

1

In this situation you may want to use the

\fancycenter[⟨*distance*⟩][⟨*stretch*⟩]{⟨*left*⟩}{⟨*centre*⟩}{⟨*right*⟩}

command, which automatically shifts the centre toward the shorter text. The ⟨*distance*⟩ is the minimum distance by which the elements are always surrounded (1em, by default). The ⟨*stretch*⟩ controls the preference for shifting the ⟨*centre*⟩; 1 means shift only when and as much as necessary. Higher numbers will start shifting sooner and more aggressively. The default is 3. This command writes over the whole header/footer space

so it should only be put in one place (typically `C`) and other places (`L,R`) should be empty.

```
\fancyhead[L,R]{}
\fancyhead[C]{%
  \fancycenter%
    {\thepage}%
    {\firstleftmark}%
    {Jane Doe}%
}

\section{Section}
Section by Jane Doe
```

1 1 SECTION Jane Doe

1 Section

Section by Jane Doe

1

You may want to present different headers or footers when the corresponding page starts with a float or ends with a footnote. The fancyhdr package defines four commands that let you achieve this.

```
\iftopfloat{⟨true branch⟩}{⟨false branch⟩}
\ifbotfloat{⟨true branch⟩}{⟨false branch⟩}
\iffloatpage{⟨true branch⟩}{⟨false branch⟩}
\iffootnote{⟨true branch⟩}{⟨false branch⟩}
```

Commands `\iftopfloat` and `\ifbotfloat` execute their ⟨*true branch*⟩ if a float sits at the top or bottom of the page (respectively).

Similarly, `\iffloatpage` checks whether the page is a special float-only page, while `\iffootnote` checks for a footnote at the bottom of the page.

```latex
\fancyhead[C]{%
  \iftopfloat{%
    A float is below me%
  } {}
}
\fancyfoot[C]{%
  \iffootnote{%
    A footnote is above me%
  } {%
    \thepage%
  }%
}

Ignore this footnote.%
\footnote{Read this.}
\begin{figure}[t]
  \centering
  A floating float.
  \caption{Hmm}
\end{figure}
```

Ignore this footnote.[1]

[1] Read this.

A footnote is above me

A float is below me

A floating float.

Figure 1: Hmm

2

The ruled lines in headers and footers are created by invoking `\headrule` and `\footrule` commands. If you want finer control over the lines, consider redefining these macros. Use `\headruleskip` and `\footruleskip` to raise or lower them, if necessary.

```latex
\RenewDocumentCommand{\headrule}{}{
  \rule{0.05\headwidth}{0.2cm}%
  \rule[0.1cm]{0.9\headwidth}{\headrulewidth}%
  \rule{0.05\headwidth}{0.2cm}%
}

\RenewDocumentCommand{\headruleskip}{}{-0.2cm}

My age is the first number not
nameable in under twenty words.
```

My age is the first number not nameable in under twenty words.

1

While working on a longer document, you may want to have several page styles for different occasions. You may also notice that some commands (`\chapter` and `\maketitle`, for example) change the page

style to `plain`. The solution to both of these problems is the

`\fancypagestyle{⟨name⟩}{⟨code⟩}`

command. The ⟨*name*⟩ is the name of page style to (re)define, while the ⟨*code*⟩ is the code to set the page style. All page styles declared this way use the `fancy` page style as their basis, so if empty ⟨*code*⟩ is given they will match `fancy` exactly.

```
\fancypagestyle{mine}{
  \fancyhead[L]{My style}
}
\pagestyle{mine}

Were you expecting a paradox here?
```

My style

You may be disappointed.

1

7.8 Boxes

LaTeX builds up its pages by pushing around boxes. At first, each letter is a little box, which is then glued to other letters to form words. These are again glued to other words, but with special glue, which is elastic so that a series of words can be squeezed or stretched as to exactly fill a line on the page.

I admit, this is a very simplistic version of what really happens, but the point is that TeX operates on glue and boxes. Letters are not the only things that can be boxes. You can put virtually everything into a box, including other boxes. Each box will then be handled by LaTeX as if it were a single letter.

In earlier chapters you encountered some boxes, although I did not tell you. The `tabular` environment and the `\includegraphics`, for example, both produce a box. This means that you can easily arrange two tables or images side by side. You just have to make sure that their combined width is not larger than the text width.

You can also pack a paragraph of your choice into a box with either

the

```
\parbox[pos]{width}{text}
```

command or the

```
\begin{minipage}[pos]{width} text \end{minipage}
```

environment. The `pos` parameter can take one of the letters `c`, `t` or `b` to control the vertical alignment of the box, relative to the baseline of the surrounding text. `width` takes a length argument specifying the width of the box. The main difference between a `minipage` and a `\parbox` is that you cannot use all commands and environments inside a `parbox`, while almost anything is possible in a `minipage`.

While `\parbox` packs up a whole paragraph doing line breaking and everything, there is also a class of boxing commands that operates only on horizontally aligned material. We already know one of them; it's called `\mbox`. It simply packs up a series of boxes into another one, and can be used to prevent LaTeX from breaking two words. As boxes can be put inside boxes, these horizontal box packers give you ultimate flexibility.

```
\makebox[width][pos]{text}
```

`width` defines the width of the resulting box as seen from the outside.[10] Besides the length expressions, you can also use `\width`, `\height`, `\depth`, and `\totalheight` in the width parameter. They are set from values obtained by measuring the typeset *text*. The *pos* parameter takes a one letter value: **c**enter, flush**l**eft, flush**r**ight, or **s**pread the text to fill the box.

The command `\framebox` works exactly the same as `\makebox`, but it draws a box around the text.

[10] This means it can be smaller than the material inside the box. You can even set the width to 0pt so that the text inside the box will be typeset without influencing the surrounding boxes.

The following example shows you some things you could do with the \makebox and \framebox commands.

```
\makebox[\textwidth]{%
    c e n t r a l}\par
\makebox[\textwidth][s]{%
    s p r e a d}\par
\framebox[1.1\width]{Guess I'm
    framed now!} \par
\framebox[0.8\width][r]{Bummer,
    I am too wide} \par
\framebox[1cm][l]{never
    mind, so am I}
Can you read this?
```

c e n t r a l
s p r e a d
Guess I'm framed now!
Bummer, I am too wide
never mind, so am I
Can you read this?

Now that we control the horizontal, the obvious next step is to go for the vertical.[11] No problem for LaTeX. The

\raisebox{*lift*}[*extend-above-baseline*][*extend-below-baseline*]{*text*}

command lets you define the vertical properties of a box. You can use \width, \height, \depth, and \totalheight in the first three parameters, in order to act upon the size of the box inside the *text* argument.

```
\raisebox{0pt}[0pt][0pt]{\Large%
\textbf{Aaaa\raisebox{-0.3ex}{a}%
\raisebox{-0.7ex}{aa}%
\raisebox{-1.2ex}{r}%
\raisebox{-2.2ex}{g}%
\raisebox{-4.5ex}{h}}}
she shouted, but not even the next
one in line noticed that something
terrible had happened to her.
```

Aaaaaaaargh she shouted, but not even the next one in line noticed that something terrible had happened to her.

[11] Total control is only to be obtained by controlling both the horizontal and the vertical …

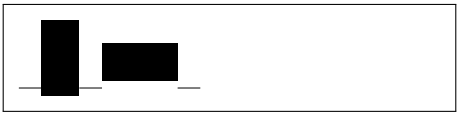
7.9 Rules

A few pages back you may have noticed the command

```
\rule[lift]{width}{height}
```

In normal use it produces a simple black box.

```
\rule{3mm}{.1pt}%
\rule[-1mm]{5mm}{1cm}%
\rule{3mm}{.1pt}%
\rule[1mm]{1cm}{5mm}%
\rule{3mm}{.1pt}
```



This is useful for drawing vertical and horizontal lines. The line on the title page, for example, has been created with a `\rule` command.

The End.

Appendix A

Installing LaTeX

Knuth published the source to TeX back in a time when nobody knew about Open Source and/or Free Software. The license that comes with TeX lets you do whatever you want with the source, but you can only call the result of your work TeX if the program passes a set of tests Knuth has also provided. This has led to a situation where we have free TeX implementations for almost every Operating System under the sun. This chapter will give some hints on what to install on Linux, macOS and Windows, to get a working TeX setup.

A.1 What to Install

To use LaTeX on any computer system, you need multiple programs.

1. The TeX/LaTeX program for processing your LaTeX source files into typeset PDF documents.

2. A text editor for editing your LaTeX source files. Some products even let you start the LaTeX program from within the editor.

3. A PDF viewer program for previewing and printing your documents.

4. A program to handle PostScript files and images for inclusion into your documents.

For every platform there are several programs that fit the requirements above. Here we just tell about the ones we know, like and have some experience with.

A.2 Cross Platform Editor

While TeX is available on many computing platforms, LaTeX editors have long been highly platform specific.

Over the past few years I have come to like Texmaker quite a lot. Apart from being very a useful editor with integrated pdf-preview and syntax high-lighting, it has the advantage of running on Windows, Mac, and Unix/Linux equally well. See [6] for further information. There is also a forked version of Texmaker called TeXstudio [87]. It also seems well maintained and is also available for all three major platforms.

You will find some platform specific editor suggestions in the OS sections below.

A.3 TeX on macOS

A.3.1 TeX Distribution

Just download MacTeX [21]. It is a pre-compiled LaTeX distribution for macOS. MacTeX provides a full LaTeX installation plus a number of additional tools.

A.3.2 macOS TeX Editor

If you are not happy with our cross-platform suggestion Texmaker (Section A.2).

The most popular open source editor for LaTeX on the mac seems to be TeXshop [2]. It is also contained in the MacTeX distribution.

Recent TeXLive distributions contain the TeXworks editor [32] which is a multi-platform editor based on the TeXShop design. Since TeXworks uses the Qt toolkit, it is available on any platform supported by this toolkit (macOS, Windows, Linux).

A.3.3 Treat yourself to PDFView

Use PDFView [4] for viewing PDF files generated by LATEX, it integrates tightly with your LATEX text editor. After installing, open PDFViews preferences dialog and make sure that the *automatically reload documents* option is enabled and that PDFSync support is set appropriately.

A.4 TEX on Windows

A.4.1 Getting TEX

First, get a copy of the excellent MiKTEX [65] distribution. It contains all the basic programs and files required to compile LATEX documents. The coolest feature in my eyes, is that MiKTEX will download missing LATEX packages on the fly and install them magically while compiling a document. Alternatively you can also use the TEXLive [5] distribution which exists for Windows, Unix, and Mac OS to get your base setup going.

A.4.2 A LATEX editor

If you are not happy with our cross-platform suggestion Texmaker (Section A.2).

TeXnicCenter [77] uses many concepts from the programming-world to provide a nice and efficient LATEX writing environment in Windows. TeXnicCenter integrates nicely with MiKTeX.

Recent TEXLive distributions contain the TEXworks Editor [32]. It supports Unicode and requires at least Windows XP.

A.4.3 Document Preview

You will most likely be using Yap for DVI preview as it gets installed with MikTeX. For PDF you may want to look at Sumatra PDF [41]. I mention Sumatra PDF because it lets you jump from any position in the PDF document back into corresponding position in your source document.

A.4.4 Working with graphics

Working with high quality graphics in LaTeX means that you have to use Encapsulated PostScript (eps) or PDF as your picture format. The program that helps you deal with this is called GhostScript [16]. It comes with its own front-end GhostView.

If you deal with bitmap graphics (photos and scanned material), you may want to have a look at the open source Photoshop alternative Gimp [36].

A.5 TeX on Linux

If you work with Linux, chances are high that LaTeX is already installed on your system, or at least available on the installation source you used to setup. Use your package manager to install the following packages:

- texlive — the base TeX/LaTeX setup.
- emacs (with AUCTeX) — an editor that integrates tightly with LaTeX through the add-on AUCTeX package.
- ghostscript — a PostScript preview program.
- xpdf and acrobat — a PDF preview program.
- imagemagick — a free program for converting bitmap images.
- gimp — a free Photoshop look-a-like.
- inkscape — a free illustrator/corel draw look-a-like.

If you are looking for a more windows like graphical editing environment, check out Texmaker. See Section A.2.

Most Linux distros insist on splitting up their TeX environments into many optional packages, so if something is missing after your first install, go check again.

Appendix B

Things You Shouldn't Use

LaTeX has been around for well over 30 years now. And it has remained largely backward compatible. This means you can easily steal snippets from a friends document written ten years ago and use them in your document today. In many instances this work without any problem. But even though LaTeX seems happy with the old code, you still should not do it. Have a look!

B.1 ... for Display Math

Instead of	Use

`$$`	`\[`, `\]`
	`\begin{equation*}`
	`\begin{displaymath}`
	(All with the amsmath package.)

`$$` is plain TeX syntax and it cannot be modified. The `displaymath` and `\[` are LaTeX commands that are a bit better in terms of spacing and features, but most importantly they can be redefined. With amsmath, both of them are redefined to be synonyms for the `equation*` which produces optimal and consistent spacing.

```
$$ 2 + 2 = 4 $$        ⟶        \[ 2 + 2 = 4 \]
```

B.2 ... for Inline Math

Instead of	Use
`$`	`\(`, `\)`

Like Section B.1, `$` is also plain TEX syntax and it cannot be redefined. Currently, there are no visual differences between `$` and the new version, but it may become an issue if you want to modify the command yourself. Also, the `\(`, `\)` commands can detect math mode nesting and produce an error if it happens.

```
$ 2 + 2 = 4 $          ⟶      \( 2 + 2 = 4 \)
```

B.3 ... for Typesetting Math

Instead of	Use
`\over`	`\frac`
`\choose`	`\binom` (amsmath)
`\overwithdelims`	`\genfrac` (amsmath)
`\atop`	
`\atopwithdelims`	
`\above`	
`\abovewithdelims`	

These are plain TEX commands for producing fractions, binomial coefficients and related structures. Due to their unusual syntax, their use may lead to ambiguous code. It's better to use the commands described in Section 3.11.

```
{a \over b}            ⟶      \frac{a}{b}

                               \usepackage{amsmath}
{a \choose b}          ⟶      % ...
                               \binom{a}{b}
```

B.4 ... for Defining New Commands

Instead of	Use
`\newcommand`	`\NewDocumentCommand`
`\renewcommand`	`\RenewDocumentCommand`
`\def`	`\DeclareDocumentCommand`

Both `\newcommand` and `\renewcommand` are LaTeX 2_ε macros. They are not as expressive as `...DocumentCommand` family described in subsection 7.1.1. Their syntax only allows a single optional argument with default value (specified as second optional argument) followed by few mandatory ones (specified as number in first optional argument).

```
\newcommand{\foo}[4][bar]{ ... }
```

↓

```
\NewDocumentCommand{\foo}{O{bar}mmmm}{ ... }
```

The `\def` command is a plain TeX primitive. It *always* defines the command, even if it was already defined. This is usually not desirable, but if it's needed you can use `\DeclareDocumentCommand`.

```
\def\foo#1#2#3{ ... }
```

↓

```
\NewDocumentCommand{\foo}{mmm}{ ... }
```

B.5 ... for Copying Commands

Instead of	Use
`\let`	`\NewCommandCopy`
	`\RenewCommandCopy`
	`\DeclareCommandCopy`

Like Section B.4, `\let` is also a plain TeX primitive that does not guard against accidental redefinition. Moreover it does not work correctly with some LaTeX commands. Use `\NewCommandCopy` as described in subsection 7.1.3.

`\let\foo\bar`	⟶	`\NewCommandCopy\foo\bar`

B.6 ... for Aligning Equations

Instead of	Use
`\begin{eqnarray}`	`\begin{align}`
`\begin{eqnarray*}`	`\begin{align*}`
	(Both in amsmath.)

`eqnarray` and `eqnarray*` are LaTeX environments that allow aligning equations. However spacing around the binary operators in these environments is far from ideal. Therefore, it is recommended to always use the `align` environments from amsmath or `IEEEeqnarray` as described in subsection 3.4.3.

```
\begin{eqnarray}
  f(x) & = &  1 + 2 \\
  g(x) & > & 52
\end{eqnarray}
```

⟶

```
\usepackage{amsmath}
% ...
\begin{align}
  f(x) & = 1 + 2 \\
  g(x) & > 52
\end{align}
```

B.7 ... for Changing Fonts

Instead of	Use
`\bf`	`\bfseries`
`\rm`	`\rmfamily`
`\sf`	`\sffamily`
`\tt`	`\ttfamily`
`\it`	`\itshape`
`\sc`	`\scshape`
`\sl`	`\slshape`

These are plain TeX commands. Each of them resets the font to normal before changing it, so for example bold italics cannot be achieved using them. Use newer commands as described in Section 7.2.

```
{\bf foo}          ⟶    \textbf{foo}

{\bf foo}          ⟶    {\bfseries foo}
```

B.8 ... for Changing Text Alignment

Instead of	Use
`\begin{flushleft}`	`\begin{FlushLeft}`
`\begin{flushright}`	`\begin{FlushRight}`
`\begin{center}`	`\begin{Center}`
	(ragged2e)

The default LaTeX 2_ε environments for changing text alignment make it nearly impossible to hyphenate words inside them. Therefore, it is recommended to use the ragged2e equivalents, as described in Section 2.15, that make the text less 'ragged' than it should be.

```
                                  \usepackage{ragged2e}
\begin{center}                    % ...
  text                      ⟶    \begin{Center}
\end{center}                        text
                                  \end{Center}
```

B.9 ... for Typesetting Quotations

Instead of	Use
``	`\enquote`
,,	`\enquote*`
<<	
>>	
''	

The TeX version of entering quotes was to rely on ligatures for the given quotation mark. This method is not context aware and cannot be customized. Using csquotes package as described in Section 2.16 allows greater control over the typesetting of the quotations.

```
``quote''
```
⟶
```
\usepackage{csquotes}
% ...
\enquote{quote}
```

B.10 ... for Printing Verbatim

Instead of	Use
`verbatim`	`verbatim` (from the verbatim package)

LaTeX comes with a `verbatim` environment built into the core, however there are problems with it. Firstly, due to the way delimiting is handled you cannot put spaces between `\end` and `{verbatim}`. The second and more important problem is that it reads the whole input at once which risks overflowing TeX's memory. The verbatim package described in subsection 2.17.1 solves both of these problems.

```
\begin{verbatim}
  code
\end{verbatim}
```
⟶
```
\usepackage{verbatim}
% ...
\begin{verbatim}
  code
\end{verbatim}
```

B.11 ... for Adding a Bibliography

Instead of	Use
`natbib` and `bibtex`	`biblatex` and `biber`

As was already mentioned in the Chapter 4 before `biber` there was `bibtex`. It used the `natbib` package for typesetting the bibliography itself. Both are no longer maintained and don't handle UTF-8 input, so they should be avoided. However, the `natbib` styles (`.bst` files) don't work with `biblatex`, so if you are forced to use such a bibliograph style, you don't have a choice.

```
\usepackage{natbib}
```
⟶
```
\usepackage{biblatex}
```

B.12 ... for Creating Relations and Operators

Instead of	Use
`\stackrel`	`\overset`
`\stackbin`	

The commands are primitives that always define the underlying symbol to be either a relation or binary operator. The `\overset` detects this based on the symbol that is overset. See subsection 3.3.6 for an example of usage.

`\stackrel{f}{=}` ⟶ `\overset{f}{=}`

`\stackbin{f}{+}` ⟶ `\overset{f}{+}`

B.13 ... for Changing Math Font

Instead of	Use
`\mathrm`	`\symrm`
`\mathit`	`\symit`
`\mathbf`	`\symbf`
`\mathcal`	`\symcal`
`\mathscr`	`\symscr`
`\mathbb`	`\symbb`
`\mathfrak`	`\symfrak`
	(unicode-math package)

When LaTeX typesets operators it uses `\math...` commands to typeset their name, usually `\mathrm`. This uses a different font compared to the symbols used in mathematics.

```
\(\mathit{ffi}\) vs.\
  \(\symit{ffi}\)
```

ffi vs. $\symit{ffi}$

These commands should not be used for changing the font for symbols. For historical reasons, some of the `\math...` commands are actually just synonyms for the symbol versions. For example there is no `\mathscr`

version for math operators, so it is just a synonym for \symscr. For operators it is better to use the \DeclareMathOperator command.

```
\(\mathrm{foo}(\mathcal{F})\)
```

↓

```
\DeclareMathOperator{\foo}{foo}
% ...
\(\foo(\symcal{F})\)
```

B.14 ... for Spacing

Instead of	Use
\vskip	\vspace
\hskip	\hspace
\mskip	\mspace (amsmath package)
\kern	
\mkern	

These commands are TEX primitives that use non-standard syntax, which may lead to weird errors. It is preferable to use the LATEX equivalents, which use the standard syntax. The \kern primitive acts as a vertical space when used between two paragraphs, and as a horizontal space when used inside a paragraph.

```
\vskip 1cm        ⟶    \vspace{1cm}

\mskip 1mu        ⟶    \mspace{1mu}
```

Appendix C

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. https://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces

of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

   The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License

with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless

of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified

versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the

violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through

a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PRO-

GRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program 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.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see https://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read https://www.gnu.org/licenses/why-not-lgpl.html.

Bibliography

[1] Each LaTeX installation should provide a so-called *LaTeX Local Guide*, which explains the things that are special to the local system. It should be contained in a file called `local.tex`. Unfortunately, some lazy sysops do not provide such a document. In this case, go and ask your local LaTeX guru for help.

[2] Richard Koch et al. *TeXShop*. Version 4.70. 2022. URL: https://pages.uoregon.edu/koch/texshop/.

[3] American Mathematical Society. *amsthm – Typesetting theorems (AMS style)*. Version 2.20.6. May 29, 2020. URL: https://www.ctan.org/pkg/amsthm.

[4] Andrea Bergia. *PDFView*. Version 0.14.3. 2007. URL: http://pdfview.sourceforge.net/.

[5] Karl Berry. *TeX Live*. Version 2022. 2022. URL: https://www.tug.org/texlive/.

[6] Pascal Brachet. *Texmaker*. Version 5.1.3. 2022. URL: https://www.xm1math.net/texmaker/.

[7] Georg Brandl and Matthäus Chajdas. *Pygments – Python syntax highlighter*. Version 2.6. Dec. 24, 2021. URL: https://pygments.org.

[8] Paolo Brasolin. *commutative-diagrams – CoDi: Commutative Diagrams for TeX*. Version 1.0.1. June 11, 2020. URL: https://ctan.org/pkg/commutative-diagrams.

[9] David Carlisle and The LaTeX3 Project et al. *indentfirst – Indent first paragraph after section header*. Version 1.03. Nov. 23, 1995. URL: https://ctan.org/pkg/indentfirst.

[10] David Carlisle and LaTeX3 Project. *graphicx – Enhanced support for graphics*. Version 1.2d. Sept. 16, 2021. URL: https://www.ctan.org/pkg/graphicx.

[11] David Carlisle and The LaTeX Team. *longtable – Allow tables to flow over page boundaries*. Version 4.17. Sept. 1, 2021. URL: https://www.ctan.org/pkg/longtable.

[12] David Carlisle et al. *setspace – Set space between lines*. Version 6.7a. Dec. 19, 2011. URL: https://ctan.org/pkg/setspace.

[13] François Charette. *arabxetex – An ArabTeX-like interface for XeTeX*. Version 1.2.1. Sept. 4, 2015. URL: https://www.ctan.org/pkg/arabxetex.

[14] François Charette and Philip Kime. *biber*. Version 2.17. 2022. URL: https://www.ctan.org/pkg/biber.

[15] François Charette et al. *polyglossia – An alternative to babel for XeLaTeX and LuaLaTeX*. Version 1.57. July 18, 2022. URL: https://www.ctan.org/pkg/polyglossia.

[16] L. Peter Deutsch. *Ghostscript*. Version 9.56.1. 2022. URL: https://www.ghostscript.com/.

[17] Simon Fear and Danie Els. *booktabs – Publication quality tables in LaTeX*. Version 1.61803398. Jan. 14, 2020. URL: https://www.ctan.org/pkg/booktabs.

[18] Christian Feuersänger. *pgfplots – Create normal/logarithmic plots in two and three dimensions*. Version 1.18.1. May 17, 2021. URL: https://www.ctan.org/pkg/pgfplots.

[19] Alexander Grahn. *ocgx2 – Drop-in replacement for 'ocgx' and 'ocg-p'*. Version 0.54. Apr. 1, 2022. URL: https://www.ctan.org/pkg/ocgx2.

[20] Frank Grießhammer and Adobe. *Source Serif*. Version 4.004. Jan. 25, 2021. URL: https://github.com/adobe-fonts/source-serif.

[21] MacTeX TeXnical working group. *MacTeX*. Version MacTeX-2022. 2022. URL: https://tug.org/mactex/.

[22] Andy Hammerlindl, John Bowman, and Tom Prince. *asymptote – 2D and 3D TEX-Aware Vector Graphics Language*. Version 2.80. Apr. 5, 2022. URL: https://www.ctan.org/pkg/asymptote.

[23] Carsten Heinz, Brooks Moses, and Jobst Hoffmann. *listings – Typeset source code listings using LATEX*. Version 1.8d. Mar. 24, 2020. URL: https://www.ctan.org/pkg/listings.

[24] Martin Hensel. *mhchem – Typeset chemical formulae/equations and H and P statements*. Version 2021-12-31. Dec. 31, 2021. URL: https://www.ctan.org/pkg/mhchem.

[25] Morten Høgholm, Lars Madsen, and The LATEX3 Project. *mathtools – Mathematical tools to use with amsmath*. Version 1.28a. Feb. 7, 2022. URL: https://www.ctan.org/pkg/mathtools.

[26] Morten Høgholm and The LATEX3 Project. *xfrac – Split-level fractions in LATEX 2ε **. Version 2022-06-22. June 22, 2022. URL: https://www.ctan.org/pkg/xfrac.

[27] Paul D. Hunt and Adobe. *Source Code Pro*. Version 2.038. Jan. 15, 2021. URL: https://github.com/adobe-fonts/source-code-pro.

[28] Paul D. Hunt and Adobe. *Source Sans 3*. Version 3.046. July 14, 2021. URL: https://github.com/adobe-fonts/source-sans.

[29] *Inkscape*. Version 1.2.1. July 14, 2022. URL: https://inkscape.org/.

[30] International Organization for Standardization. *ISO 80000-2:2019*. 2nd ed. Nov. 2021. URL: https://www.iso.org/obp/ui/#iso:std:iso:80000:-2:ed-2:v2:en.

[31] Dr. Uwe Kern. *xcolor – Driver-independent color extensions for LATEX and pdfLATEX*. Version 2.14. June 12, 2022. URL: https://www.ctan.org/pkg/xcolor.

[32] Jonathan Kew, Stefan Löffler, and Charlie Sharpsteen. *TeXworks*. Version 0.6.7. 2022. URL: https://www.tug.org/texworks/.

[33] Vafa Khalighi and bidi-tex GitHub Organisation. *bidi – Bidirectional typesetting in plain TEX and LATEX, using XƎTEX*. Version 36.7. Feb. 18, 2022. URL: https://www.ctan.org/pkg/bidi.

[34] Vafa Khalighi and bidi-tex GitHub Organisation. *XePersian – Persian for LATEX, using XƎTEX*. Version 23.7. Feb. 20, 2022. URL: https://www.ctan.org/pkg/xepersian.

[35] Vafa Khalighi, Arthur Reutenauer, and Jürgen Spitzmüller. *luabidi – Bidi functions for LuaTEX*. Version 0.5. Oct. 27, 2019. URL: https://www.ctan.org/pkg/luabidi.

[36] Spencer Kimball and Peter Mattis. *GIMP*. Version 2.10.30. 2021. URL: https://www.gimp.org/.

[37] Daniel Kirsch. *Detexify*. URL: https://detexify.kirelabs.org/classify.html.

[38] Donald E. Knuth. *Knuth message on 'Diskussion:TeX — Aussprache'*. Posted by user S (https://de.wikipedia.org/wiki/Benutzer:S). Apr. 5, 2004. URL: https://de.wikipedia.org/w/index.php?title=Diskussion:TeX&oldid=217574275#Aussprache.

[39] Donald E. Knuth. *The TEXbook*. 2nd ed. Vol. A. Computers and Typesetting. Reading, Massachusetts: Addison-Wesley, 1984. ISBN: 0-201-13448-9.

[40] Markus Kohm. *koma-script – A bundle of versatile classes and packages*. Version 3.37. July 4, 2022. URL: https://ctan.org/pkg/koma-script.

[41] Krzysztof Kowalczyk. *Sumatra PDF*. Version 3.3.3. 2021. URL: https://www.sumatrapdfreader.org/.

[42] Leslie Lamport. *LATEX: A Document Preparation System*. 2nd ed. Reading, Massachusetts: Addison-Wesley, 1994. ISBN: 0-201-52983-1.

[43] Philipp Lehman and Joseph Wright. *csquotes – Context sensitive quotation facilities*. Version 5.2l. Feb. 22, 2021. URL: https://www.ctan.org/pkg/csquotes.

[44] Philipp Lehman et al. *BibLATEX*. Version 3.17. 2022. URL: https://www.ctan.org/pkg/biblatex.

[45] Wolfgang May and Andreas Schedler. *ntheorem – Enhanced theorem environment*. Version 1.33. URL: https://www.ctan.org/pkg/ntheorem.

[46] Frank Mittelbach and The LaTeX Team. *array – Extending the array and tabular environments*. Version 2.5f. Oct. 4, 2021. URL: https://www.ctan.org/pkg/array.

[47] Frank Mittelbach and the LaTeX Project Team. "Quo vadis LaTeX(3) Team — A look back and at the upcoming years." In: *TUGboat* 41.2 (2020). URL: https://www.latex-project.org/publications/2020-FMi-TUB-tb128mitt-quovadis.pdf.

[48] Frank Mittelbach et al. *The LaTeX Companion*. 2nd ed. Reading, Massachusetts: Addison-Wesley, 2004. ISBN: 0-201-36299-6.

[49] Clemens Niederberger. *chemformula – Command for typesetting chemical formulas and reactions*. Version 4.17. Jan. 23, 2022. URL: https://www.ctan.org/pkg/chemformula.

[50] Pieter van Oostrum. *fancyhdr – Extensive control of page headers and footers in LaTeX2e*. Version 4.0.2. May 10, 2022. URL: https://www.ctan.org/pkg/fancyhdr.

[51] Pieter van Oostrum, Øystein Bache, and Jerry Leichter. *multirow – Create tabular cells spanning multiple rows*. Version 2.8. Mar. 15, 2021. URL: https://www.ctan.org/pkg/multirow.

[52] Scott Pakin. *The Comprehensive LaTeX Symbol List – Symbols accessible from LaTeX*. Version 14.0. May 5, 2021. URL: https://www.ctan.org/pkg/comprehensive.

[53] Matthew Petroff. *Another Beamer Theme Matrix*. URL: https://mpetroff.net/files/beamer-theme-matrix/.

[54] Bureau International des Poids et Mesures. *The International System of Units (SI)*. May 20, 2019. URL: https://www.bipm.org/en/measurement-units/.

[55] LaTeX3 Project and American Mathematical Society. *amsmath*. Version 2021-11-15. 2021. URL: https://www.ctan.org/pkg/latex-amsmath.

[56] The LaTeX Project. *Future releases of LaTeX will require an eTeX-enabled engine*. Nov. 4, 2016. URL: https://www.latex-project.org/news/2016/11/04/eTeX-kernel/.

[57] LaTeX Project Team. *New LaTeX methods for authors (starting 2020)*. Feb. 19, 2022. URL: http://mirrors.ctan.org/macros/latex/base/usrguide3.pdf.

[58] Sebastian Rahtz et al. *hyperref – Extensive support for hypertext in LaTeX*. Version 7.00q. May 17, 2022. URL: https://www.ctan.org/pkg/hyperref.

[59] Will Robertson. *Symbols defined by unicode-math*. Version 0.8q. Jan. 31, 2020. URL: http://mirrors.ctan.org/macros/unicodetex/latex/unicode-math/unimath-symbols.pdf.

[60] Will Robertson and Khaled Hosny. *fontspec – Advanced font selection in XeLaTeX and LuaLaTeX*. Version 2.8a. Jan. 16, 2022. URL: https://www.ctan.org/pkg/fontspec.

[61] Will Robertson et al. *unicode-math – Unicode mathematics support for XeTeX and LuaTeX*. Version 0.8q. Jan. 31, 2020. URL: https://www.ctan.org/pkg/unicode-math.

[62] Kristoffer H. Rose, Ross Moore, et al. *xypic – Flexible diagramming macros*. Version 3.8.9. Oct. 6, 2013. URL: https://www.ctan.org/pkg/xypic.

[63] Konrad Rudolph and Geoffrey M. Poore. *minted – Highlighted source code for LaTeX*. Version 2.6. Dec. 24, 2021. URL: https://www.ctan.org/pkg/minted.

[64] Martin Scharrer. *mwe – Packages and image files for MWEs*. Version 0.5. Mar. 30, 2018. URL: https://www.ctan.org/pkg/mwe.

[65] Christian Schenk. *MiKTeX*. Version 22.3. 2022. URL: https://miktex.org/.

[66] Rainer Schöpf and The LaTeX3 Project. *verbatim – Reimplementation of and extensions to LaTeX verbatim*. Version 1.5u. July 7, 2020. URL: https://www.ctan.org/pkg/verbatim.

[67] Martin Schröder and Marei Peischl. *ragged2e – Alternative versions of "ragged"-type commands*. Version 3.1. Dec. 15, 2021. URL: https://www.ctan.org/pkg/ragged2e.

[68] Michael Shell. "How to Use the IEEEtran LaTeX Class." In: *Journal of LaTeX Class Files* 14.8 (Aug. 2015). URL: http://www.ctan.org/tex-archive/macros/latex/contrib/IEEEtran/IEEEtran_HOWTO.pdf.

[69] Michael Shell. *ieeetrantools*. Version 1.4. 2014. URL: https://www.ctan.org/pkg/ieeetrantools.

[70] Axel Sommerfeldt. *newfloat – Define new floating environments*. Version 1.1l. Sept. 2, 2019. URL: https://www.ctan.org/pkg/newfloat.

[71] Wenchang Sun, Leo Liu, and Qing Lee. *xecjk – Support for CJK documents in XeLaTeX*. Version 3.8.8. Dec. 14, 2021. URL: https://github.com/CTeX-org/ctex-kit.

[72] Supreme Council of Information and Communication Technology and Mohammad Saleh Souzanchi. *Iran Nastaliq font*. 2020. URL: https://github.com/font-store/font-IranNastaliq.

[73] Till Tantau and Henri Menke. *pgf – Create PostScript and PDF graphics in TeX*. Version 3.1.9a. May 15, 2021. URL: https://www.ctan.org/pkg/pgf.

[74] Till Tantau, Vedran Miletić, and Joseph Wright. *beamer – A LaTeX class for producing presentations and slides*. Version 3.67. May 17, 2022. URL: https://www.ctan.org/pkg/beamer.

[75] The LaTeX Team. *LaTeX 2ε font selection*. Mar. 2021. URL: https://www.ctan.org/pkg/fntguide.

[76] The LaTeX Team. *makeidx – Standard LaTeX package for creating indexes*. Version 1.0m. Sept. 29, 2014. URL: https://www.ctan.org/pkg/makeidx.

[77] The TeXnicCenter Team. *TeXnicCenter*. Version 2.02. 2013. URL: https://www.texniccenter.org/.

[78] SIL Language Technology. *Ezra SIL*. Version 2.51. Oct. 4, 2007. URL: https://software.sil.org/ezra/.

[79] The TEX Live Team and Peter Breitenlohner. *ε-TEX – An extended version of TEX, from the NTS project*. Version 2.0. Feb. 1998. URL: https://www.ctan.org/pkg/etex.

[80] John Hobby and The METAPOST Team. *metapost – A development of METAFONT for creating graphics*. Version 2.00. Dec. 31, 2020. URL: https://www.ctan.org/pkg/metapost.

[81] The SageMath Developers. *SageMath*. Version 9.6. 2022. DOI: 10.5281/zenodo.593563. URL: https://www.sagemath.org/.

[82] Antonis Tsolomitis. *New Computer Modern*. Version 4.1. Dec. 15, 2021. URL: https://www.ctan.org/pkg/newcomputermodern.

[83] Hideo Umeki and David Carlisle. *geometry – Flexible and complete interface to document dimensions*. Version 5.9. Jan. 2, 2020. URL: https://ctan.org/pkg/geometry.

[84] Wikipedia contributors. *Help:Displaying a formula — Wikipedia, The Free Encyclopedia, year = 2022*. [Online; accessed 10-June-2022]. URL: https://en.wikipedia.org/w/index.php?title=Help:Displaying_a_formula&oldid=1091087776.

[85] Peter R. Wilson and Lars Madsen. *memoir – Typeset fiction, non-fiction and mathematical books*. Version 3.7r. July 29, 2022. URL: https://ctan.org/pkg/memoir.

[86] Joseph Wright. *siunitx – A comprehensive (SI) units package*. Version 3.1.1. May 3, 2022. URL: https://www.ctan.org/pkg/siunitx.

[87] Benito van der Zander. *TeXstudio*. Version 4.2.3. 2022. URL: https://www.texstudio.org/.

[88] Timothy Van Zandt and Herbert Voss. *fancyvrb – Sophisticated verbatim text*. Version 4.2. Apr. 12, 2022. URL: https://www.ctan.org/pkg/fancyvrb.

Index

Symbols

"', 45
\(, 95, 278
\), 95, 278
\,, 142
-, 27
\-, 26
–, 27
—, 27
., 109, 121
., space after, 40
…, 28
\/, 226
\:, 115, 142
\;, 115, 142
\[, 95, 277
$, 278
$$, 277
%, 7, 38, 222
&, 64, 114
\, 24
^, 96, 97
_, 96
\\, 75
articleclass, 9, 10, 187, 253
beamerclass, 10, 177, 179, 186
bookclass, 10, 253
letterclass, 10
memoirclass, 252
procclass, 10
reportclass, 10, 253
slidesclass, 176
{, 8
}, 8
~, 39, 222
~, 121
\], 95, 277

A

A4 paper, 253
A5 paper, 253
å, 31
\above, 278
\abovewithdelims, 278
abstract, 28
accent, 29
acute, 31
\acute, 107
\addbibresource, 156
\addtolength, 249
\addvspace command, 248
\adots, 149
advantages of LaTeX, 5
æ, 31
\alert, 179
align, 114, 117, 280
align*, 280

\Alpha, 101
\alpha, 100, 101
American Mathematical Society, 93
amsmath, 18, 93, 111, 117, 132, 150, 179, 277, 278, 280, 284
amsmath, 117
amsthm, 144
\and, 42
\ang, 125
\appendix, 42
Arabic, 36
arabxetex, 36
\arccos, 104
\arcsin, 104
\arctan, 104
\arg, 104
array, 64, 75, 77
array, 138
asymptote, 192
\atop, 278
\atopwithdelims, 278
\author, 42, 178
\autocite, 164
\autopageref, 173
\autoref, 173

B

B5 paper, 253
babel, 29
\backmatter, 42
backslash, 8
\bar, 107
base font size, 253
\baselineskip, 256
beamer, 18, 176, 177, 179, 182, 183, 185–187, 189
beamerarticle, 187
\begin, 15, 218
\Beta, 101
\beta, 100, 101
\bf, 280
\bfseries, 227, 280
biber, 155
\bibitem, 153
biblatex, xii, 18, 153–156, 159, 160, 163
biblatex, 282
bibliography, 153
bidi, 35
\Big, 110
\big, 110
\bigcap, 140
\Bigg, 110
\bigg, 110
\bigskip, 248
\bigwedge, 105
binary relations, 108
\binom, 151, 278
block, 179
\blockquote, 46, 47
Bmatrix, 137
bmatrix, 137
bold face, 227
booktabs, 18, 63–65
booktabs, 66
\BooleanFalse, 222
\BooleanTrue, 222
\bot, 139, 140
\bottomrule, 65

C

\cap, 100
\caption, 85, 87, 89
cases, 138

\cdot, 97
\cdots, 149
Center, 44, 281
center, 281
\cfrac, 150
\chapter, 40, 162, 268
\chapter*, 162
\chaptermark, 265
chemformula, 143
\Chi, 101
\chi, 101
Chinese, 37
\choose, 278
\cite, 154, 163
\citeauthor, 164
\citedate, 164
\citetitle, 164
\citeurl, 164
\citeyear, 164
\cleardoublepage, 87
\clearpage, 87
\cline, 78
\cmidrule, 68, 70
\colon, 100, 140
\coloneq, 94
\Coloneqq, 94
\coloneqq, 94
\color, 237, 242
\colorbox, 242
\colorlet, 241
coloured text, 16
colours, 237
comma, 27
commands, 8
\(, 95, 278
\), 95, 278
\,, 142
\-, 26
\/, 226
\:, 115, 142
\;, 115, 142
\[, 95, 277
\, 24
\\, 75
\], 95, 277
\above, 278
\abovewithdelims, 278
\acute, 107
\addbibresource, 156
\addtolength, 249
\addvspace command, 248
\adots, 149
\alert, 179
\Alpha, 101
\alpha, 100, 101
\and, 42
\ang, 125
\appendix, 42
\arccos, 104
\arcsin, 104
\arctan, 104
\arg, 104
\atop, 278
\atopwithdelims, 278
\author, 42, 178
\autocite, 164
\autopageref, 173
\autoref, 173
\backmatter, 42
\bar, 107
\baselineskip, 256
\begin, 15, 218

\Beta, 101
\beta, 100, 101
\bf, 280
\bfseries, 227, 280
\bibitem, 153
\Big, 110
\big, 110
\bigcap, 140
\Bigg, 110
\bigg, 110
\bigskip, 248
\bigwedge, 105
\binom, 151, 278
\blockquote, 46, 47
\BooleanFalse, 222
\BooleanTrue, 222
\bot, 139, 140
\bottomrule, 65
\cap, 100
\caption, 85, 87, 89
\cdot, 97
\cdots, 149
\cfrac, 150
\chapter, 40, 162, 268
\chapter*, 162
\chaptermark, 265
\Chi, 101
\chi, 101
\choose, 278
\cite, 154, 163
\citeauthor, 164
\citedate, 164
\citetitle, 164
\citeurl, 164
\citeyear, 164
\cleardoublepage, 87
\clearpage, 87
\cline, 78
\cmidrule, 68, 70
\colon, 100, 140
\coloneq, 94
\Coloneqq, 94
\coloneqq, 94
\color, 237, 242
\colorbox, 242
\colorlet, 241
\complexnum, 128
\complexqty, 128
\coordinate, 196
\cos, 104
\cosh, 104
\cot, 104
\coth, 104
\csc, 104
\csi, 214
\cup, 100
\date, 42, 178
\dbinom, 151
\dblcolon, 94
\ddot, 107
\ddots, 149
\DeclareCommandCopy, 279
\DeclareDocumentCommand, 217, 279
\DeclareFloatingEnvironment, 87
\DeclareMathOperator, 104, 105, 284
\DeclarePairedDelimiter, 110, 111
\DeclarePairedDelimiterX, 111
\DeclareSIPower, 129

\DeclareSIPrefix, 129
\DeclareSIQualifier, 129
\DeclareSIUnit, 128
\def, 279
\definecolor, 241
\deg, 104
\Delta, 101
\delta, 101
\depth, 270, 271
\det, 104
\dfrac, 150, 151
\dim, 104
\displaystyle, 147, 150
\div, 97
\divslash, 97
\documentclass, 9, 21, 25, 82, 252
\dots, 148
\dotsb, 148
\dotsc, 148
\dotsi, 149
\dotsm, 149
\dotso, 149
\doublespacing, 256
\draw, 193, 194, 199
\dum, 214
\emph, 14–16, 213, 220, 221
\emptyset, 100
\end, 15, 218
\endfirsthead, 75
\endfoot, 75
\endhead, 75
\endlastfoot, 75
\enquote, 45, 281
\enquote*, 281
\Epsilon, 101
\epsilon, 101
\eqqcolon, 94
\eqref, 95, 113
\Eta, 101
\eta, 101
\exists, 99
\exp, 104
\extramarks, 263
\fancycenter, 266
\fancyf, 262
\fancyfoot, 261
\fancyfootoffset, 262
\fancyhead, 261
\fancyheadoffset, 262
\fancyhf, 260, 262, 265
\fancyhfoffset, 262
\fancypagestyle, 269
\fcolorbox, 242
\firstleftmark, 264
\firstleftxmark, 263
\firstrightmark, 264, 265
\firstrightxmark, 263
\fontsize, 228, 242, 254
\footcite, 163
\footnote, 43
\footnotesize, 227, 228
\footrule, 268
\footruleskip, 268
\footrulewidth, 262
\forall, 99
\foreach, 206
\foreignblockquote, 48
\foreignquote, 48
\foreigntextquote, 48
\frac, 99, 147, 149, 151, 278
\framebox, 270, 271

\framesubtitle, 178
\frametitle, 178
\frenchspacing, 40
\frontmatter, 42
\fussy, 25
\Gamma, 101
\gamma, 100, 101
\gcd, 104
\genfrac, 151, 278
\geq, 97
\geqslant, 97
\gram, 122
\grave, 107
\hat, 107
\headrule, 268
\headruleskip, 268
\headrulewidth, 262
\height, 270, 271
\hfill, 247
\highlight, 123
\hline, 78
\hom, 104
\hphantom, 142
\href, 172
\hskip, 284
\hspace, 66, 243, 284
\Huge, 227–229
\huge, 227, 228
\hyperref, 172, 173
\hypersetup, 173
\hyphenation, 25, 26
\IEEEeqnarraymulticol, 118
\IEEEeqnarraynumspace, 118
\IEEEyesnumber, 118
\IEEEyessubnumber, 118
\IfBooleanF, 217
\IfBooleanT, 217
\IfBooleanTF, 217, 222
\ifbotfloat, 267
\iff, 99
\iffloatpage, 267
\iffootnote, 267
\IfNoValueF, 216
\IfNoValueT, 216
\IfNoValueTF, 216
\iftopfloat, 267
\IfValueF, 216
\IfValueT, 216
\IfValueTF, 216, 217
\iiiint, 106
\iiint, 106
\iint, 106
\impliedBy, 99
\implies, 99
\in, 100
\include, 89
\includegraphics, 79, 81, 269
\includeonly, 89
\indent, 257
\index, 168
\inf, 104
\input, 89
\inputminted, 60
\institute, 178
\Iota, 101
\iota, 101
\it, 280
\item, 44
\itshape, 225, 227, 280
\Kappa, 101
\kappa, 101
\ker, 104

\kern, 284
\kilo, 122
\L, 37, 38
\label, 43, 58, 87, 113
\Lambda, 101
\lambda, 101
\land, 99
\LARGE, 227, 228
\Large, 227, 228
\large, 227, 228
\lastleftmark, 264
\lastleftxmark, 263
\lastrightmark, 264
\lastrightxmark, 263
\LaTeX, 26, 37
\LaTeXe, 26
\lBrack, 140
\ldots, 28, 149
\left, 109, 110
\leq, 97
\leqslant, 97
\let, 279
\lg, 104
\lim, 104
\liminf, 104
\limsup, 104
\linebreak, 24
\linespread, 255
\linewidth, 250
\listoffigures, 85
\listoftables, 85, 87
\ln, 104
\lnot, 99
\log, 104
\lor, 99
\lstinline, 52
\lstinputlisting, 52
\lstset, 53
\lvert, 139
\mainmatter, 42
\makebox, 270, 271
\makeindex, 167
\maketitle, 42, 178, 268
\markboth, 264
\markright, 266
\mathbb, 283
\mathbf, 283
\mathbin, 140
\mathcal, 283
\mathclose, 140
\mathcolor, 237
\mathfrak, 283
\mathit, 283
\mathop, 140
\mathopen, 140
\mathord, 140
\mathpunct, 140
\mathrel, 139, 140
\mathrm, 283
\mathscr, 283
\max, 104
\mbox, 26, 270
\mdseries, 227
\medskip, 248
\medskipamount, 248
\medspace, 142
\midrule, 65
\min, 104
\mintinline, 60
\mkern, 284
\mskip, 284
\mspace, 141, 142, 284

\Mu, 101
\mu, 101
\multicol, 132
\multicolumn, 66, 69, 71, 75
\multirow, 69, 71, 72, 75, 132
\negmedspace, 119, 142
\negmedspace, 119
\negthickspace, 142
\negthinspace, 142
\neq, 97
\newcolumntype, 77
\newcommand, 279
\NewCommandCopy, 221, 241, 279
\NewDocumentCommand, 37, 38, 214, 217, 218, 220, 229, 241, 279
\NewDocumentEnvironment, 218–220
\newfontfamily, 33, 234
\newlength, 249
\newline, 23
\NewNegationCommand, 236
\newpage, 24
\newtheorem, 143, 144
\nobreakspace, 222
\nocite, 160
\node, 195, 196
\noindent, 257
\nolinebreak, 24
\nopagebreak, 24
\nopagecolor, 242
\normalfont, 227
\normalsize, 227–229
\not, 97, 236
\notag, 113, 118
\nouppercase, 265
\Nu, 101
\nu, 101
\num, 121
\numlist, 127
\of, 122
\Omega, 101
\omega, 101
\Omicron, 101
\omicron, 101
\onslide, 181
\over, 278
\overbrace, 107
\overbracket, 94
\overline, 107
\overset, 108, 283
\overwithdelims, 278
\pagebreak, 24
\pagecolor, 242
\pagenumbering, xvii, 254, 255
\pageref, 43, 172, 173
\pagestyle, 252
\par, 222, 226
\paragraph, 40
\parbox, 270
\parencite, 163
\parindent, 256, 257
\parskip, 256
\part, 40
\path, 194
\pause, 180–182, 186
\pdfstringdefDisableCommands, 176
\per, 122
\perp, 139
\phantom, 115, 119, 142
\Phi, 101
\phi, 101

\Pi, 101
\pi, 101
\pic, 206
\Pr, 104
\prescript, 142
\printbibliography, 156, 160–162
\printindex, 168
\prod, 105
\providecolor, 241
\ProvideDocumentCommand, 217
\providefontfamily, 234
\ProvidesExplPackage, 222
\Psi, 101
\psi, 101
\qedhere, 146
\qedsymbol, 146
\qquad, 142
\qty, 121
\quad, 142
\raisebox, 271
\rBrack, 140
\ref, 43, 87, 172, 173
\renewcommand, 279
\RenewCommandCopy, 279
\RenewDocumentCommand, 39, 217, 220, 279
\RenewDocumentEnvironment, 220
\RenewExpandableDocumentCommand, 176
\renewfontfamily, 234
\Rho, 101
\rho, 101
\right, 109, 110, 138
\rm, 280
\rmfamily, 225, 227, 234, 280
\rule, 272
\rvert, 139
\sc, 280
\scriptscriptstyle, 147
\scriptsize, 227, 228
\scriptstyle, 147
\scshape, 227, 280
\sec, 104
\section, 36, 40, 162, 178, 213, 264, 265
\section*, 162
\sectionmark, 265
\selectfont, 228, 255
\setbeamercolor, 184
\setbeamercovered, 182
\setbeamerfont, 184
\setdefaultlanguage, 30
\setlength, 249
\setmainfont, 35, 230
\setmathfont, 235
\setminted, 63
\setminus, 100
\setmonofont, 35, 230
\setotherlanguage, 30
\setsansfont, 35, 230
\settodepth, 250
\settoheight, 250
\settowidth, 250
\sf, 280
\sffamily, 227, 234, 280
\sfrac, 150
\shoveleft, 112
\shoveright, 112
\sideset, 106
\Sigma, 101
\sigma, 101
\sin, 104

\sinh, 104
\sisetup, 125
\sl, 280
\slash, 27
\sloppy, 25
\slshape, 227, 280
\small, 227, 228
\smallskip, 248
\smartcite, 164
\smash, 147, 148
\sqrt, 98
\stackbin, 283
\stackrel, 283
\stretch, 246, 248
\subparagraph, 40
\subsection, 40, 264
\subsectionmark, 265
\subset, 100
\subseteq, 100
\subsetneq, 100
\substack, 105
\subsubsection, 40
\subtitle, 178
\succeq, 140
\sum, 104, 140
\sup, 104
\supset, 100
\supseteq, 100
\supsetneq, 100
\swshape, 227
\symbb, 101, 103, 283
\symbbit, 103
\symbf, 283
\symbffrak, 103
\symbfit, 103
\symbfscr, 103
\symbfsfit, 103
\symbfsfup, 103
\symbfup, 103
\symcal, 236, 283
\symfrak, 103, 283
\symit, 103, 283
\symrm, 102, 283
\symscr, 103, 236, 283, 284
\symsfit, 103
\symsfup, 103
\symtt, 103
\symup, 103
\tablenum, 132, 136
\tableofcontents, 41, 85, 171, 178
\tag, 95, 113
\tan, 104
\tanh, 104
\Tau, 101
\tau, 101
\tbinom, 151
\TeX, 26
\texorpdfstring, 175
\text, 96, 115
\textbackslash, 8
\textbackslash*, 71
\textbf, 224, 227, 229
\textcite, 163
\textcolor, 237
\textit, 225, 227
\textlang, 32
\textmd, 224, 227
\textnormal, 225, 227
\textquote, 46, 48
\textrm, 224, 225, 227
\textsc, 225, 227

\textsf, 224, 227
\textsl, 225, 227
\textssc, 225
\textstyle, 147, 150
\textsw, 225, 227
\texttt, 224, 227
\textulc, 225
\textup, 225, 227
\text«*language*», 32
\tfrac, 150, 151
\the, 249
\theoremstyle, 144
\thepage, 263
\thesection, 266
\Theta, 101
\theta, 101
\thickspace, 142
\thinspace, 142
\thispagestyle, 254
\tikz, 193
\tikzset, 202
\tilde, 107
\times, 97, 140
\tiny, 227–229
\title, 42, 178
\titlegraphic, 178
\tnss, 214
\today, 26, 32
\toprule, 65
\totalheight, 270, 271
\tothe, 122
\tracingcolors, 241
\tt, 280
\ttfamily, 227, 280
\uncover, 181
\underbrace, 107
\underbracket, 94
\underset, 108
\unit, 121
\upshape, 227
\Upsilon, 101
\upsilon, 101
\url, 172
\useasboundingbox, 194
\usecolortheme, 183
\usefonttheme, 183, 184
\usepackage, 10, 16, 17, 21, 156, 222
\usetheme, 183
\usetikzlibrary, 209
\varepsilon, 101, 102
\varkappa, 102
\varphi, 102
\varpi, 102
\varrho, 102
\varsigma, 102
\varTheta, 102
\vartheta, 102
\vdots, 149
\veebar, 99
\verb, 50, 52, 220
\verbatiminput, 51, 52
\verbatiminput*, 52
\vfill, 248
\vphantom, 142, 143
\vskip, 284
\vspace, 247, 284
\widehat, 107
\width, 270, 271
\Xi, 101
\xi, 101
\Zeta, 101

\zeta, 101
comments, 7
commutative-diagrams, 192
\complexnum, 128
\complexqty, 128
\coordinate, 196
\cos, 104
\cosh, 104
\cot, 104
\coth, 104
cross-references, 42
\csc, 104
\csi, 214
csquotes, 45, 47–49, 281
\cup, 100

D

dash, 27
\date, 42, 178
\dbinom, 151
\dblcolon, 94
\ddot, 107
\ddots, 149
decimal alignment, 129
\DeclareCommandCopy, 279
\DeclareDocumentCommand, 217, 279
\DeclareFloatingEnvironment, 87
\DeclareMathOperator, 104, 105, 284
\DeclarePairedDelimiter, 110, 111
\DeclarePairedDelimiterX, 111
\DeclareSIPower, 129
\DeclareSIPrefix, 129
\DeclareSIQualifier, 129
\DeclareSIUnit, 128
\def, 279
\definecolor, 241
\deg, 104
delimiters, 108
\Delta, 101
\delta, 101
\depth, 270, 271
description, 44
\det, 104
\dfrac, 150, 151
\dim, 104
dimensions, 242
display style, 94
displaymath, 277
displayquote, 46
\displaystyle, 147, 150
\div, 97
\divslash, 97
document font size, 253
document title, 253
\documentclass, 9, 21, 25, 82, 252
dotless ı and ȷ, 31
\dots, 148
\dotsb, 148
\dotsc, 148
\dotsi, 149
\dotsm, 149
\dotso, 149
double sided, 253
\doublespacing, 256
\draw, 193, 194, 199
\dum, 214

E

ellipsis, 28
em, 16
em-dash, 27
\emph, 14–16, 213, 220, 221
empty, 252
\emptyset, 100

en-dash, 27
Encapsulated PostScript, 276
\end, 15, 218
\endfirsthead, 75
\endfoot, 75
\endhead, 75
\endlastfoot, 75
\enquote, 45, 281
\enquote*, 281
enumerate, 44
environments
abstract, 28
align, 114, 117, 280
align*, 280
amsmath, 117
array, 138
biblatex, 282
block, 179
Bmatrix, 137
bmatrix, 137
booktabs, 66
cases, 138
Center, 44, 281
center, 281
description, 44
displaymath, 277
displayquote, 46
em, 16
enumerate, 44
eqnarray, 280
eqnarray*, 280
equation, 95
equation*, 95, 112, 277
figure, 84–88
FlushLeft, 44, 281
flushleft, 281
FlushRight, 44, 281
flushright, 281
foreigndisplayquote, 48
frame, 177
gather, 113
gather*, 114
german, 31
IEEEeqnarray, 117, 280
itemize, 44
lang, 31
listing, 57
listings, x, 52
longtable, xi, 71, 74–76, 87, 90
lscommand, 214
lstlisting, 52
matrix, 132, 138
minipage, 270
minted, 60
multline, 112
multline*, 112
natbib, 282
parbox, 270
picture, 192
pmatrix, 132
proof, 144, 179
quotation, 49
scope, 205
smallmatrix, 137
table, 84, 85, 87
tabular, 63, 64, 68, 71, 129, 269
thebibliography, xii, 153, 160
theorem, 179
tikzpicture, 193
verbatim, 51, 52, 282

verbatim environment, 53
verse, 49
Vmatrix, 137
vmatrix, 137
\Epsilon, 101
\epsilon, 101
eqnarray, 280
eqnarray*, 280
\eqqcolon, 94
\eqref, 95, 113
equation
multiple, 113
equation, 95
equation*, 95, 112, 277
\Eta, 101
\eta, 101
executive paper, 253
\exists, 99
\exp, 104
exponent, 97
extension, 20
.aux, 21
.bib, 155–157, 166
.bst, 282
.cls, 21
.dtx, 21
.dvi, 2, 170
.fd, 21
.idx, 21, 168
.ilg, 21
.ind, 21, 168
.ins, 21
.jb2, 79
.jbig2, 79
.jpeg, 79
.jpg, 79
.lof, 21
.log, 17, 21, 222
.lot, 21
.mps, 79
.pdf, 13, 79
.png, 79
.sty, 21, 169
.tex, 11, 20
.toc, 21
extramarks, 263, 264
\extramarks, 263

F

\fancycenter, 266
\fancyf, 262
\fancyfoot, 261
\fancyfootoffset, 262
fancyhdr, 18, 260, 261, 263, 264, 267
\fancyhead, 261
\fancyheadoffset, 262
\fancyhf, 260, 262, 265
\fancyhfoffset, 262
\fancypagestyle, 269
fancyvrb, 52
\fcolorbox, 242
figure, 84–88
file types, 20
\firstleftmark, 264
\firstleftxmark, 263
\firstrightmark, 264, 265
\firstrightxmark, 263
floating bodies, 82
FlushLeft, 44, 281
flushleft, 281
FlushRight, 44, 281
flushright, 281
font, 224

font size, 224, 227
\fontsize, 228, 242, 254
fontspec, xiii, 33, 230, 232, 233, 235
\footcite, 163
footer, 252
\footnote, 43
\footnotesize, 227, 228
\footrule, 268
\footruleskip, 268
\footrulewidth, 262
\forall, 99
\foreach, 206
\foreignblockquote, 48
foreigndisplayquote, 48
\foreignquote, 48
\foreigntextquote, 48
\frac, 99, 147, 149, 151, 278
frame, 177
\framebox, 270, 271
\framesubtitle, 178
\frametitle, 178
\frenchspacing, 40
\frontmatter, 42
\fussy, 25

G

\Gamma, 101
\gamma, 100, 101
gather, 113
gather*, 114
\gcd, 104
\genfrac, 151, 278
geometry, 169, 258, 259
\geq, 97
\geqslant, 97
german, 31
GhostScript, 276
GhostView, 276
Gimp, 276
\gram, 122
graphics, 16
graphicx, 79
grave, 31
\grave, 107

H

\hat, 107
header, 252
headings, 252
\headrule, 268
\headruleskip, 268
\headrulewidth, 262
Hebrew, 36
\height, 270, 271
\hfill, 247
\highlight, 123
\hline, 78
\hom, 104
horizontal
space, 243
\hphantom, 142
\href, 172
\hskip, 284
\hspace, 66, 243, 284
\Huge, 227–229
\huge, 227, 228
hyperref, 35, 168, 170, 172–175
\hyperref, 172, 173
\hypersetup, 173
hyphen, 27
hyphenat, 169
\hyphenation, 25, 26

I

IEEEeqnarray, 117, 280
\IEEEeqnarraymulticol, 118
\IEEEeqnarraynumspace, 118
IEEEtrantools, 117, 118
\IEEEyesnumber, 118
\IEEEyessubnumber, 118
\IfBooleanF, 217
\IfBooleanT, 217
\IfBooleanTF, 217, 222
\ifbotfloat, 267
\iff, 99
\iffloatpage, 267
\iffootnote, 267
\IfNoValueF, 216
\IfNoValueT, 216
\IfNoValueTF, 216
\iftopfloat, 267
\IfValueF, 216
\IfValueT, 216
\IfValueTF, 216, 217
\iiiint, 106
\iiint, 106
\iint, 106
\impliedBy, 99
\implies, 99
\in, 100
\include, 89
\includegraphics, 79, 81, 269
\includeonly, 89
\indent, 257
indentfirst, 257
index, 167
\index, 168
\inf, 104
\input, 89
input file, 10
\inputminted, 60
\institute, 178
international, 29
\Iota, 101
\iota, 101
\it, 280
italic, 227
italic correction, 226
\item, 44
itemize, 44
\itshape, 225, 227, 280

J

Japanese, 37
Jawi, 36

K

\Kappa, 101
\kappa, 101
kashida, 36
Kashmiri, 36
\ker, 104
\kern, 284
\kilo, 122
Knuth, Donald E., 1
koma-script, 252
Korean, 37
Kurdish, 36

L

\L, 37, 38
\label, 43, 58, 87, 113
\Lambda, 101
\lambda, 101
Lamport, Leslie, 3
\land, 99

lang, 31
language, 29
\LARGE, 227, 228
\Large, 227, 228
\large, 227, 228
\lastleftmark, 264
\lastleftxmark, 263
\lastrightmark, 264
\lastrightxmark, 263
\LaTeX, 26, 37
\LaTeXe, 26
\lBrack, 140
\ldots, 28, 149
\left, 109, 110
left-aligned, 44
legal paper, 253
length
rigid, 244
rubber, 245
\leq, 97
\leqslant, 97
\let, 279
letter paper, 253
\lg, 104
ligature, 28
\lim, 104
\liminf, 104
\limsup, 104
line break, 23
line spacing, 254
\linebreak, 24
\linespread, 255
\linewidth, 250
listing, 57
listings, 52, 56, 59–62
listings, x, 52
\listoffigures, 85
\listoftables, 85, 87
\ln, 104
\lnot, 99
\log, 104
long equations, 111
longtable, 64, 71, 82, 87
longtable, xi, 71, 74–76, 87, 90
\lor, 99
lscommand, 214
\lstinline, 52
\lstinputlisting, 52
lstlisting, 52
\lstset, 53
luabidi, 35
\lvert, 139

M

MacTeX, 274
\mainmatter, 42
\makebox, 270, 271
makeidx, 18, 167
makeidx package, 167
\makeindex, 167
makeindex program, 167
\maketitle, 42, 178, 268
Malay, 36
margins, 257
\markboth, 264
\markright, 266
math mode, 96
math spacing, 139
\mathbb, 283
\mathbf, 283
\mathbin, 140
\mathcal, 283
\mathclose, 140

\mathcolor, 237
mathematical
accents, 107
\mathfrak, 283
\mathit, 283
\mathop, 140
\mathopen, 140
\mathord, 140
\mathpunct, 140
\mathrel, 139, 140
\mathrm, 283
\mathscr, 283
mathtools, 93, 94, 111, 142
matrix, 132
matrix, 132, 138
\max, 104
\mbox, 26, 270
\mdseries, 227
\medskip, 248
\medskipamount, 248
\medspace, 142
memoir, 252
metapost, 192
mhchem, 143
\midrule, 65
MiKTEX, 275
\min, 104
minipage, 270
minted, x, 59, 60
minted, 60
\mintinline, 60
\mkern, 284
\mskip, 284
\mspace, 141, 142, 284
\Mu, 101
\mu, 101
\multicol, 132
\multicolumn, 66, 69, 71, 75
multirow, 64, 69
\multirow, 69, 71, 72, 75, 132
multline, 112
multline*, 112
mwe, 79

N

natbib, 282
\negmedspace, 119, 142
\negmedspace, 119
\negthickspace, 142
\negthinspace, 142
\neq, 97
\newcolumntype, 77
\newcommand, 279
\NewCommandCopy, 221, 241, 279
\NewDocumentCommand, 37, 38, 214, 217, 218, 220, 229, 241, 279
\NewDocumentEnvironment, 218–220
newfloat, 87
\newfontfamily, 33, 234
\newlength, 249
\newline, 23
\NewNegationCommand, 236
\newpage, 24
\newtheorem, 143, 144
\nobreakspace, 222
\nocite, 160
\node, 195, 196
\noindent, 257
\nolinebreak, 24
\nopagebreak, 24
\nopagecolor, 242
\normalfont, 227
\normalsize, 227–229

\not, 97, 236
\notag, 113, 118
\nouppercase, 265
ntheorem, 144
\Nu, 101
\nu, 101
\num, 121
\numlist, 127

O

ocgx2, 171
œ, 31
\of, 122
\Omega, 101
\omega, 101
\Omicron, 101
\omicron, 101
one column, 253
\onslide, 181
Ottoman, 36
\over, 278
\overbrace, 107
\overbracket, 94
overfull hbox, 25
\overline, 107
\overset, 108, 283
\overwithdelims, 278

P

package, 9, 16
packages
amsmath, 18, 93, 111, 117, 132, 150, 179, 277, 278, 280, 284
amsthm, 144
arabxetex, 36
array, 64, 75, 77
asymptote, 192
babel, 29
beamer, 18, 176, 177, 179, 182, 183, 185–187, 189
beamerarticle, 187
biber, 155
biblatex, xii, 18, 153–156, 159, 160, 163
bidi, 35
booktabs, 18, 63–65
chemformula, 143
commutative-diagrams, 192
csquotes, 45, 47–49, 281
extramarks, 263, 264
fancyhdr, 18, 260, 261, 263, 264, 267
fancyvrb, 52
fontspec, xiii, 33, 230, 232, 233, 235
geometry, 169, 258, 259
graphicx, 79
hyperref, 35, 168, 170, 172–175
hyphenat, 169
IEEEtrantools, 117, 118
indentfirst, 257
koma-script, 252
listings, 52, 56, 59–62
longtable, 64, 71, 82, 87
luabidi, 35
makeidx, 18, 167
mathtools, 93, 94, 111, 142
memoir, 252
metapost, 192
mhchem, 143
minted, x, 59, 60
multirow, 64, 69
mwe, 79

newfloat, 87
ntheorem, 144
ocgx2, 171
pgf, 192, 193, 209
pgfplots, 192
polyglossia, 18, 25, 29–31, 33, 35, 36, 40, 47, 127, 232
powerdot, 176
ragged2e, 44, 281
setspace, 256
showidx, 168
siunitx, xi, 120, 121, 124–126, 129, 130, 132–135
syntonly, 89
unicode-math, 94, 235, 283
verbatim, 51, 282
xcolor, xvii, 173, 174, 237, 238, 241
xeCJK, 37
xepersian, 36
xfrac, 150
xypic, 192
page style
empty, 252
headings, 252
page style, 252
plain, 252
\pagebreak, 24
\pagecolor, 242
\pagenumbering, xvii, 254, 255
\pageref, 43, 172, 173
\pagestyle, 252
paper size, 253
paper size, 257
\par, 222, 226
paragraph, 17
\paragraph, 40
\parbox, 270
parbox, 270
\parencite, 163
\parindent, 256, 257
\parskip, 256
\part, 40
Pashto, 36
\path, 194
\pause, 180–182, 186
PDF, 170
\pdfstringdefDisableCommands, 176
PDFView, 275
\per, 122
period, 27
\perp, 139
Persian, 36
pgf, 192, 193, 209
pgfplots, 192
\phantom, 115, 119, 142
\Phi, 101
\phi, 101
\Pi, 101
\pi, 101
\pic, 206
picture, 192
piecewise function, 138
placement specifier, 84
plain, 252
pmatrix, 132
polyglossia, 18, 25, 29–31, 33, 35, 36, 40, 47, 127, 232
PostScript, 5, 273, 276
Encapsulated, 276
powerdot, 176
\Pr, 104

preamble, 9
\prescript, 142
\printbibliography, 156, 160–162
\printindex, 168
\prod, 105
proof, 144, 179
\providecolor, 241
\ProvideDocumentCommand, 217
\providefontfamily, 234
\ProvidesExplPackage, 222
\Psi, 101
\psi, 101

Q

\qedhere, 146
\qedsymbol, 146
\qquad, 142
\qty, 121
\quad, 142
Queen of Swash, 227
quotation, 49
quotation marks, 45

R

ragged2e, 44, 281
\raisebox, 271
\rBrack, 140
\ref, 43, 87, 172, 173
\renewcommand, 279
\RenewCommandCopy, 279
\RenewDocumentCommand, 39, 217, 220, 279
\RenewDocumentEnvironment, 220
\RenewExpandableDocumentCommand, 176
\renewfontfamily, 234
reserved characters, 7
\Rho, 101
\rho, 101
\right, 109, 110, 138
right-aligned, 44
rigid length, 244
\rm, 280
\rmfamily, 225, 227, 234, 280
roman, 227
rubber length, 245
\rule, 272
\rvert, 139

S

sans serif, 227
\sc, 280
Scandinavian letters, 31
scope, 205
\scriptscriptstyle, 147
\scriptsize, 227, 228
\scriptstyle, 147
\scshape, 227, 280
\sec, 104
\section, 36, 40, 162, 178, 213, 264, 265
\section*, 162
\sectionmark, 265
\selectfont, 228, 255
\setbeamercolor, 184
\setbeamercovered, 182
\setbeamerfont, 184
\setdefaultlanguage, 30
\setlength, 249
\setmainfont, 35, 230
\setmathfont, 235
\setminted, 63
\setminus, 100
\setmonofont, 35, 230

\setotherlanguage, 30
\setsansfont, 35, 230
setspace, 256
\settodepth, 250
\settoheight, 250
\settowidth, 250
\sf, 280
\sffamily, 227, 234, 280
\sfrac, 150
\shoveleft, 112
\shoveright, 112
showidx, 168
SI, 120
\sideset, 106
\Sigma, 101
\sigma, 101
\sin, 104
Sindhi, 36
single sided, 253
\sinh, 104
\sisetup, 125
siunitx, xi, 120, 121, 124–126, 129, 130, 132–135
\sl, 280
slanted, 227
Slash, 27
\slash, 27
\sloppy, 25
\slshape, 227, 280
\small, 227, 228
Small Caps, 227
smallmatrix, 137
\smallskip, 248
\smartcite, 164
\smash, 147, 148
space, 6
special character, 29
\sqrt, 98
square root, 98
\stackbin, 283
\stackrel, 283
\stretch, 246, 248
structure, 9
\subparagraph, 40
subscript, 96
\subsection, 40, 264
\subsectionmark, 265
\subset, 100
\subseteq, 100
\subsetneq, 100
\substack, 105
\subsubsection, 40
\subtitle, 178
\succeq, 140
\sum, 104, 140
\sup, 104
superscript, 96
\supset, 100
\supseteq, 100
\supsetneq, 100
\swshape, 227
\symbb, 101, 103, 283
\symbbit, 103
\symbf, 283
\symbffrak, 103
\symbfit, 103
\symbfscr, 103
\symbfsfit, 103
\symbfsfup, 103
\symbfup, 103
\symcal, 236, 283
\symfrak, 103, 283

\symit, 103, 283
\symrm, 102, 283
\symscr, 103, 236, 283, 284
\symsfit, 103
\symsfup, 103
\symtt, 103
\symup, 103
syntonly, 89

T

table, 84, 85, 87
table of contents, 41
\tablenum, 132, 136
\tableofcontents, 41, 85, 171, 178
tabular, 63, 64, 68, 71, 129, 269
\tag, 95, 113
\tan, 104
\tanh, 104
\Tau, 101
\tau, 101
\tbinom, 151
\TeX, 26
TeXnicCenter, 275
\texorpdfstring, 175
\text, 96, 115
text style, 94
\textbackslash, 8
\textbackslash*, 71
\textbf, 224, 227, 229
\textcite, 163
\textcolor, 237
\textit, 225, 227
\textlang, 32
\textmd, 224, 227
\textnormal, 225, 227
\textquote, 46, 48
\textrm, 224, 225, 227
\textsc, 225, 227
\textsf, 224, 227
\textsl, 225, 227
\textssc, 225
\textstyle, 147, 150
\textsw, 225, 227
\texttt, 224, 227
\textulc, 225
\textup, 225, 227
\text«*language*», 32
\tfrac, 150, 151
\the, 249
thebibliography, xii, 153, 160
theorem, 179
\theoremstyle, 144
\thepage, 263
\thesection, 266
\Theta, 101
\theta, 101
\thickspace, 142
\thinspace, 142
\thispagestyle, 254
\tikz, 193
tikzpicture, 193
\tikzset, 202
\tilde, 107
tilde (~), 40
\times, 97, 140
\tiny, 227–229
title, 42, 253
\title, 42, 178
\titlegraphic, 178
\tnss, 214
\today, 26, 32
\toprule, 65
\totalheight, 270, 271

\tothe, 122
\tracingcolors, 241
\tt, 280
\ttfamily, 227, 280
Turkish, 36
two column, 253

U

Uighur, 36
umlaut, 31
\uncover, 181
\underbrace, 107
\underbracket, 94
underfull hbox, 25
\underset, 108
unicode-math, 94, 235, 283
\unit, 121
units, 244
units (TEX), 242
units (SI), 120
upright, 227
\upshape, 227
\Upsilon, 101
\upsilon, 101
Urdu, 36
\url, 172
\useasboundingbox, 194
\usecolortheme, 183
\usefonttheme, 183, 184
\usepackage, 10, 16, 17, 21, 156, 222
\usetheme, 183
\usetikzlibrary, 209

V

\varepsilon, 101, 102
\varkappa, 102
\varphi, 102
\varpi, 102
\varrho, 102
\varsigma, 102
\varTheta, 102
\vartheta, 102
\vdots, 149
\veebar, 99
\verb, 50, 52, 220
verbatim, 51, 282
verbatim, 51, 52, 282
verbatim environment, 53
\verbatiminput, 51, 52
\verbatiminput*, 52
verse, 49
vertical space, 248
\vfill, 248
Vmatrix, 137
vmatrix, 137
\vphantom, 142, 143
\vskip, 284
\vspace, 247, 284

W

whitespace, 6
at the start of a line, 6
\widehat, 107
\width, 270, 271
Word, 168
WYSIWYG, 4, 5

X

xcolor, xvii, 173, 174, 237, 238, 241
xeCJK, 37
xepersian, 36
xfrac, 150
\Xi, 101

\xi, 101
xypic, 192

Z

\Zeta, 101
\zeta, 101