现代LaTeX

第二版

马特·克莱因

印刷版书籍可通过TheBookPatch和Amazon按成本价购买。

The author apologizes for any typos, formatting mistakes, inaccuracies, and other flubs. He welcomes fixes and improvements in this book's Git repository at `https://github.com/mrkline/latex-book`

问题、评论、关切以及抨击性长文也可以通过电子邮件发送至 `matt <at> bitbashing.io`

作者在圣塞里夫银行没有支票账户，但等我们见面时，他会很乐意请你喝一杯，以感谢你的帮助。

Second edition (online PDF), typeset October 25, 2022.

To Max, who once told me about a cool program he used to type up his college papers.

内容

1. 排版与你

生活是一场由书面语言组成的游行。广告、应用、文章、电子邮件、论文、菜单、消息等等，不断把文字推到你眼前。而当你阅读这些文字时，你看到的远不止作者的措辞。有意识或无意识地，你会注意到字母的形状和大小。你会注意到这些字母如何组合成单词，这些单词如何排列成段落，这些段落又如何布局在页面和屏幕上。你注意到的是排版。

> 排版正是为什么这两行会让你想起你在学校写过的糟糕作文。很多书看起来都是这样的吗？为什么不是呢？

街道标志不会长成这样是有原因的：

> EGorhamSt.

以及为什么宇宙飞船上一个非常重要的开关会被这样标注：

> CM/SM SEP

不是这样：

> *CM/SM Sep*

有效的写作不仅仅在于你选择的词语，也在于它们的外观和版式。良好的排版不仅是艺术——它还是一种工具，帮助人们更好、更快地理解你。如果你想利用这一工具，你应该试试 LaTeX！

LATEX 是什么?

LATEX（发音为“lay-tech”或“lah-tech”）是一个替代像Microsoft Word、Apple Pages、Google Docs和LibreOffice等文字处理器的工具。这些应用程序遵循“所见即所得”（wysiwyg）原则，屏幕上的内容与打印出来的内容相同。而LATEX则不同。在这里，文档以“纯”文本文件形式编写，使用标记指定最终结果的显示方式。如果你做过网页设计，这个过程类似——就像html和css描述你希望浏览器绘制的页面一样，标记描述LATEX文档的外观。

```
\LaTeX{} (pronounced ``lay-tech'' or ``lah-tech'') is an
alternative to word processors like Microsoft Word,
Apple Pages, Google Docs, and LibreOffice.
These other applications follow the principle of
\introduce{What You See Is What You Get}
\acronym{(wysiwyg)}, where what is on screen is the same
as what comes out of your printer.
\LaTeX{} is different. Here, documents are written as
``plain'' text files, using \introduce{markup}
to specify how the final result should look.
If you've done any web design, this is a similar
process---just as \acronym{html} and \acronym{css}
describe the page you want browsers to draw,
markup describes the appearance of your document to \LaTeX.
```

LATEX 标记语言用于上面的段落

这可能看起来很奇怪，如果你以前没有使用过标记语言，但它有一些优势:

1. 你可以将写作的内容和表现分开处理。在每个文档的开始，你描述你想要的设计。LATEX 从那里开始，始终如你所要求的那样格式化整个文本。与 WYSIWYG 系统相比，你在写作时需要不断处理外观。如果你改变了标题的外观，你是否确保找到了所有其他标题并做了相同的更改？如果程序以你不喜欢的方式排列内容，修正起来是否困难?

2. 你可以定义自己的命令，然后通过调整它们来立即修改所有使用到它们的地方。例如，上面示例中的 `\introduce` 和 `\acronym` 命令就是我自己创建的。一个用于将文本设为斜体，另一个则把单词设置为小型大写，并稍微增加字母间距，这样字符看起来就不会太拥挤。如果我决定希望新术语采用这种外观，或者希望首字母缩略词被格式化成这种样式，我只需要修改定义这两个命令的那两行代码，本书中所有对应的实例就会立刻呈现出新的外观。

3. 能够将文档保存为纯文本有其自身的好处：

   - 你可以使用任何基本的文本编辑器来编辑它。
   - 结构一目了然，且易于复现。*
   - 你可以使用脚本和程序来自动化内容创作。
   - 你可以使用版本控制软件（如 Git 或 Mercurial）来跟踪你的更改。

又一份指南?

你也许会疑惑，为什么世界还需要另一本关于 LaTeX 的指南。毕竟，它已经存在了几十年。简单一搜就能找到将近十来本相关书籍。网上也有大量资源。

不幸的是，大多数 LaTeX 指南都有两个致命缺陷：它们又长，又旧。初学者并不想——也不需要——为了学习基础而阅读数百页，而较老的指南会用过时的信息浪费你的时间。当 LaTeX 于 1986 年首次发布时，我们今天使用的出版技术一项也不存在。Adobe 还要再过七年才会推出其便携式文档格式（Portable Document Format），而桌面出版仍只是一个刚起步的新奇事物。这一点在许多 LaTeX 指南中体现得——非常糟糕。如果你寻找更改文档字体的说明，你会被定制化的胡言乱语淹没。†

*Compare this to WYSIWYG systems, where it's not always obvious how certain formatting was produced or how to match it.

†Take these criticisms with a grain of salt. The fact that LaTeX is still here after all of the technology around it became obsolete—multiple times—is a testament to its staying power.

好消息是，LaTeX近年来有了飞跃式的进展。现在是时候推出一本不会让你陷入几十年历史负担的指南，也不会徒劳地尝试成为一本全面的参考书。毕竟，你是一个聪明、机智的人，知道如何使用搜索引擎。本书将：

1. 教授你LaTeX的基础知识。
2. 指引你到可以进一步学习的地方。
3. 展示如何利用现代排版技术。*
4. 随后立即结束。

让我们开始。

*By modern, I mean “from the mid-1990s”, but most web browsers and desktop publishing software are only just starting to catch up.

2. 安装

当你在计算机上安装 LaTeX 时，它是以一个发行版的形式打包的，其中包含：

1. LaTeX，这个程序——把标记转换为排版文档的东西。* 2. 一组常用的 LaTeX 宏包。宏包是代码的集合，能做各种事情，比如提供新命令或改变文档的样式。我们将在本书中看到它们大量的实际应用。 3. 编辑器和其他有用的工具。

每个主要的操作系统都有其自己的 LaTeX 发行版：

Mac OS 有 MacTeX。从 `http://www.tug.org/mactex` 获取它，并按照那里的说明进行安装。

Windows有MikTEX。可以从`https://miktex.org/download`安装它。MikTEX具有一个有用的功能，即在文档第一次使用某些包时自动下载它们。

Linux 和 BSD 使用 TEX Live。像大多数软件一样，它通过您的操作系统的包管理器提供。Linux 发行版通常提供 `texlive-full` 或 `texlive-most` 包，安装所需的一切。†

*Well, actually, multiple LaTeX programs, but we're getting to that.

†If you would prefer a smaller install, Linux distributions usually break TeX Live into multiple packages. Look for ones with names like `texlive-core`, `texlive-luatex` and `texlive-xetex`. As you use LaTeX more, you may need less-common packages, which usually have names like `texlive-latexextra`, `texlive-science`, and so on. Of course, all of this may vary from one Linux distribution to another.

编辑

由于 LaTeX 源文件是普通的文本文件，你可以用常见的工具来编写：Vim、Emacs、VS Code 等等。* 还有一些专门为 LaTeX 设计的编辑器，通常自带内置的 PDF 查看器。（你可以在 LaTeX Wikibook 的安装章节中找到一份不错的列表。参见附录 B。）

在线选项

如果你不想在电脑上安装 LaTeX，可以尝试像 ShareLaTeX 或 Overleaf 这样的在线编辑器。本书并不专注于这些基于网页的工具，但相同的基础原理适用。当然，你对某些方面的控制较少，比如可用的字体、使用的 LaTeX 版本等等。

*If you've never used any of these, try a few. They're popular with programmers and other folks who shuffle text around screens all day. Just don't use Notepad. Life is too short.

3. 你好，LaTeX!

既然你已经安装了 LaTeX，让我们来试一试。打开你最喜欢的文本编辑器，并将以下内容保存为 `hello.tex`：

```
\documentclass{article}
% Say hello
\begin{document}
Hello, World!
\end{document}
```

接下来，我们通过 LaTeX（该程序）运行此文件以生成文档。安装过程中将多个不同版本——或称引擎——安装到您的计算机上，但在本书中，我们使用最新的版本：LuaLaTeX 和 XeLaTeX.[†]

如果你正在使用一个 LaTeX 专用编辑器，它应该有一个菜单用于选择你想使用的引擎，以及一个用于生成文档的按钮。否则，请在你的终端中运行以下命令：[‡]

```
$ xelatex hello.tex
```

也可以随意改用 `lualatex`——两者之间有一些差异，我们稍后会讨论，但目前用哪个都可以。运气好的话，你应该会看到一些输出，最后以类似这样的消息结尾：

```
Output written on hello.pdf (1 page).
Transcript written on hello.log.
```

*Not to be confused with LaTeX the lunchbox, LaTeX the breakfast cereal, or LaTeX the flamethrower. The kids love this stuff!

[†]See Appendix A for a comparison of the various LaTeX engines.

[‡]How to work a terminal, make sure the newly-installed LaTeX programs are in your `PATH`, and so on are all outside the scope of this book. As is tradition, the leading dollar sign in this example just denotes a console prompt, and shouldn't actually be typed.

在你当前的目录中，你应该能找到一个新生成的 `hello.pdf`。打开它，你应该会看到页面顶部有如下内容：

> 你好，世界！

恭喜你，创建了你的第一个文档！让我们来解读一下我们刚刚做了什么。

所有 LaTeX 文档都以 `\documentclass` 命令开始，该命令选择一个基础“样式”供使用。有许多类可供选择——你甚至可以创建自己的——但常见的包括 `article`、`report`、`book` 和 `beamer`。* 对于一般文档，`article` 可能是一个不错的选择。下一行，`% Say hello`，是一个注释。LaTeX 在看到百分号符号后会忽略该行的其余部分，因此我们用它来为任何阅读文档源代码的人留下注释。† 最后，`\begin{document}` 告诉 LaTeX 后续内容是实际的文档内容，`\end{document}` 则表示我们已经完成。

让我们再讲一些基础知识。

间距

LATEX 通常会为你处理单词之间的间距，无论你按下空格键或制表键多少次。例如，将以下内容输入到你的编辑器中

```
The number  of   spaces    between words doesn't   matter.
The same is true for space between sentences.

An empty line ends the previous paragraph and
starts the next.
```

产量

> 词与词之间的空格数量没有关系。句子之间的空格也是如此。
>
> 一个空行结束前一段并开始下一段。

*This last one is for slideshows, named after a German term for a projector.

†Including, perhaps most importantly, a confused version of your future self!

请注意 LaTeX 如何自动遵循排版规范，例如缩进新段落，以及在句子之间留出比词与词之间稍大的空白。需要注意的一个小怪癖是，注释会“吃掉”下一行开头的任何空格，因此

```
This% weird, right?
  is strange.
```

给出

这很奇怪。

命令

LATEX 提供了许多命令来格式化你的文本，而且你也可以定义你自己的！命令总是以反斜杠（ \ ）开始，只包含字母，并且区分大小写。* 有些命令需要更多信息，或参数：`\documentclass`，例如，需要知道我们想要哪个类。参数被包含在连续成对的花括号中，所以如果某个命令需要两个参数，我们会输入：

```
\somecommand{argument1}{argument2}
```

许多命令也接受可选参数。它们位于必选参数之前，用方括号括起，并以逗号分隔。假设你想将文档以双面页面†、11 点字号打印。我们将这些要求作为可选`\documentclass`参数：

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

*`\foo` is different from `\Foo`, for example.

†`twoside` introduces commands that only make sense for double-sided printing, like one that skips to the start of the next odd page. It also lets you have different margins for even and odd pages, which is useful for texts like this book.

其他命令完全不需要参数——`\LaTeX`，它打印LaTeX标志，就是一个例子。这些命令会消耗它们后面跟随的所有空格。例如，

```
\LaTeX is great, but it can be a bit odd sometimes.
```

将给你

> LaTeX 很棒，但有时也会有点奇怪。

你可以通过在命令后面加上一对空的大括号来修复这个问题。当然，如果没有需要保留的空格，就不需要使用大括号：

```
Let's learn \LaTeX! \LaTeX{} is a powerful tool,
but a few of its rules are a little weird.
```

让我们

> 让我们学习 LaTeX！LaTeX 是一个强大的工具，但它的一些规则有点奇怪。

特殊字符和换行符

在 LaTeX 中，一些字符具有特殊含义。例如，我们刚刚看到，`%` 用于开始一个注释，而 `\` 用于开始一个命令。特殊字符的完整列表是：

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

每个都有一个对应的命令，用于在你的文档中实际打印它。分别是：

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

无论它们后面跟着什么，插入符（^）和波浪号（~）始终需要用花括号括起来。这是它们用于生成变音符号的年代留下的遗物：从前，LaTeX 用户会用 `jalape\~no` 来排版“jalapeño”。如今我们只需在源文件中直接输入 ñ。*

如果你想知道为什么我们使用 `\textbackslash` 来打印 \ 而不是 `\\`，那是因为后者是用于强制换行的命令。

```
Give me \\
a brand new line!
```

服从：

给我一条全新的
线！

谨慎使用这个功能——优雅地将段落分割成行是LaTeX最伟大的技巧之一。

环境

我们经常通过将文本放入环境中来格式化LaTeX。这些环境总是以`\begin{name}`开头，以`\end{name}`结尾，其中`name`是所需环境的名称。以`quote`环境为例，它在块引用的两侧添加额外的空白：

```
Donald Knuth once wrote,
\begin{quote}
We should forget about small efficiencies,
say about 97\% of the time:
premature optimization is the root of all evil.
Yet we should not pass up our opportunities in
that critical 3\%.
\end{quote}
```

*Of course, this depends on your keyboard, your editor, and your language settings in your OS. We will talk more about languages and Unicode fun in chapter 11.

产生

唐纳德·克努特曾经写道，

> 我们应该忘掉微小的效率，大约在 97% 的情况下如此：过早的优化是一切罪恶的根源。然而，在那关键的 3% 中，我们不应放弃机会。

组和作用域

某些命令会改变 L{ LaTeX 对其后文本的设置方式。例如，`\itshape` 会将其后所有内容设为斜体。要将命令的作用范围限制在特定区域内，可用花括号将其括起来。

```
{\itshape Sometimes we want italics}, but only sometimes.
```

变成

Sometimes we want italics，但只是有时候。

用花括号括起来的区域称为一个组，在组内发出的命令在组结束后就会失效。环境也会创建它们自己的组：

```
\begin{quote}
\itshape If I italicize a quote, the following text will
use upright type again.
\end{quote}
See? Back to normal.
```

排版

> *If I italicize a quote, the following text will use upright type again.*
>
> 看到了吗？恢复正常了。

你也可以使用分组来防止零参数命令出现奇怪的间距问题：有些人更偏好 `{\LaTeX}` 而不是 `\LaTeX{}`。

4. 文档结构

每个 LaTeX 文档都各不相同，但都具有一些共同的元素。

前言和宏包

在上一章中，你使用以下内容构建了你的第一个文档：

```
\documentclass{article}

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

`\documentclass` 和 `document` 环境开始之间的空白称为前言。在这里，我们处理任何需要的设置，包括导入包。这些包添加新的命令，或以有趣的方式修改文档。您在 LaTeX 分发版中的包来自于综合 TeX 存档网络——或称 ctan——位于 `https://ctan.org`。* 您还可以在那里找到包的手册，因此在学习如何使用某个包时，将其作为您的首选资源。

要导入一个宏包，添加一个 `\usepackage` 命令，并将其名称作为参数。作为一个简单的示例，让我们使用 `metalogo` 宏包来编写一个文档，它会添加 `\LuaLaTeX` 和 `\XeLaTeX`：

```
\documentclass{article}
\usepackage{metalogo}
```

*Curious readers might wonder what TeX is, and how it differs from LaTeX. The short answer is that TeX is the original program, and LaTeX is a set of common commands that were later built on top of it. A longer answer is at the end of this guide under Appendix A. We won't discuss how to use plain TeX here. That is for another book—The TeXbook.

```
\begin{document}
\XeLaTeX{} and \LuaLaTeX{} are neat.
\end{document}
```

应该为您生成一个可以阅读的PDF

XELATEX 和 LuaLATEX 很整洁。

`\usepackage` 接受可选参数并将其传递给您导入的任何代码。例如，`geometry` 包接受您想要的纸张大小和边距。对于美国信纸大小并设置一英寸边距，输入：

```
\usepackage[
    letterpaper,
    left=1in, right=1in, top=1in, bottom=1in
]{geometry}
```

参数可以按任何方式间隔，只要它们之间没有空行。

层次结构

作者常常将写作划分为若干部分，以帮助读者更好地浏览内容。LATEX 提供了七种不同的命令来拆分文档：`\part`、`\chapter`、`\section`、`\subsection`、`\subsubsection`、`\paragraph` 和 `\subparagraph`。在希望某个区域开始的位置发出该命令，并将其名称作为参数提供。例如，

```
\documentclass{book}

\begin{document}

\chapter{The Start}
This is a very short chapter in a very short book.

\chapter{The End}
```

```
Is the book over yet?

\section{No!}
There's some more we must do before we go.

\section{Yes!}
Goodbye!
\end{document}
```

某些级别仅在特定的文档类别中可用——例如，章节仅出现在书籍中。而且不要过度使用这些命令。大多数作品只需要几个级别来组织它们。

这些结构片段会自动编号。本章的标题是用 `\chapter{Document Structure}` 生成的，而 LaTeX 判断出它是第4章。

靠你自己

正如承诺的那样，本书不会试图成为一部全面的参考手册，但会指引你前往可以进一步学习的地方。我们将在大多数章节的结尾附上一些相关主题，供你自行探索。

考虑学习如何：

- 使用 `\maketitle` 自动以文档标题、你的姓名和日期开始你的文档。
- 用`\tableofcontents`构建目录。
- 使用 `\setcounter{secnumdepth}` 或类似 `\subsection*{foo}` 的带星号命令来控制章节编号。
- 使用 `\label` 和 `\ref` 创建交叉引用。
- 使用KOMA Script，一套可以让你自定义文档几乎所有方面的宏包，从标题字体到脚注。
- 包含图像在`graphicx`包中。

- 使用 `hyperref` 包添加超链接。
- 使用 `\input` 将大型文档拆分为多个源文件。

5. 格式化文本

强调

有时你需要一些额外的力度来把你的观点表达清楚。在 LaTeX 中强调文本的最简单方法是使用 `\emph` 命令，它会将其参数设置为斜体：

```
\emph{Oh my!}
```

给我们

Oh my!

我们还有其他工具可供使用：

```
We can also use \textbf{boldface} or \textsc{small caps}.
```

生产

We can also use **boldface** or SMALL CAPS.

在使用强调时要谨慎，尤其是加粗，它能将读者的注意力从周围的内容中吸引开。过多的使用会让人分心。

认识整个（类型）家族

粗体和斜体只是你可以使用的众多样式中的一小部分。下面列出了一份（基本）完整的清单：

Command	Alternative	Style
`\textnormal{...}`	`{\normalfont ...}`	the default
`\emph{...}`	`{\em ...}`	*emphasis, typically italics*
`\textrm{...}`	`{\rmfamily ...}`	roman (serif) type
`\textsf{...}`	`{\sffamily ...}`	sans serif type
`\texttt{...}`	`{\ttfamily ...}`	`teletype (monospaced)`
`\textit{...}`	`{\itshape ...}`	*italics*
`\textsl{...}`	`{\slshape ...}`	*slanted, or oblique type*
`\textsc{...}`	`{\scshape ...}`	Small Capitals
`\textbf{...}`	`{\bfseries ...}`	**boldface**

偏好第一种形式，它将要格式化的文本作为参数，而不是第二种形式，它影响所发出的组。第一种形式会自动改善格式化文本周围的间距。例如，斜体字与直立字体之间应该有少量额外的间距，称为“斜体修正”。当格式化多个段落或定义其他命令的样式时，后一种形式是唯一的选择。

尺寸

正文文本的字号——也就是你的主要内容——通常为十点大小，[†] 但可以通过向 `\documentclass` 传递参数来调整。[‡] 若要相对于此默认大小缩放文本，请使用以下命令：

`\tiny` 示例文本
`\scriptsize` 示例文本
`\footnotesize` 示例文本 `\small` 示例文本 `\normalsize` 示例文本 `\large` 示例文本

*For instance, this book's section headers are styled with `\Large\itshape`.

[†] The standard digital publishing point, sometimes called the PostScript point, is $1/72$ of an inch. LaTeX, for historical reasons, defines its point (`pt`) as $100/7227$ of an inch and the former as "big points", or `bp`. Use whichever you would like.

[‡] Stock LaTeX classes accept `10pt`, `11pt`, or `12pt` as optional arguments. KOMA Script classes accept arbitrary sizes with `fontsize=<size>`.

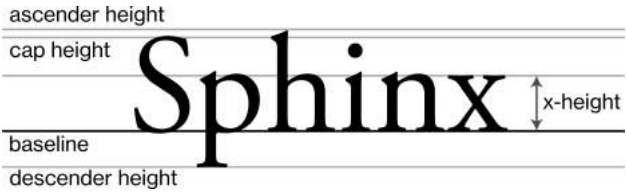
\Large 示例文本 \LARGE 示例文本 \huge 示例文本 \Huge 示例文本

如果你仔细观察，你会发现这里有一些微妙的变化。LaTeX的默认字体系列，Latin Modern，有几个不同的光学大小。较小的字体不仅仅是它们大号兄弟的缩小版——它们有更粗的笔画、更夸张的特征以及更宽松的间距，以提高在其大小下的可读性。

> 如果我将5磅字号设为与11磅字号相同的高度，你可以很容易地发现其中的差异。

当字体是由金属制作时，多种光学大小是标准配置。但许多数字字体只有一种，因为每种光学大小都需要大量精心设计。

点数和光学大小并不能完全说明问题。每种字体的比例不同，这会影响其感知大小。（比较 Garamond、Latin Modern、Futura 和 Helvetica，均为 11 点。）下面列出了一些常见术语：



字体位于基线，升至大写字母的高度，并下降到降部高度。大写字母高度指的是大写字母的大小，x高度指的是小写字母的大小。

如果之前的命令无法提供你需要的字号，你可以使用 `\fontsize` 创建自定义的字号，它同时接受一个文本大小和一个基线之间的距离。必须随后使用 `\selectfont` 才能生效。例如，`\fontsize{30pt}{30pt}\selectfont` 会产生

*If you have typefaces with multiple optical sizes, LuaLaTeX and XeLaTeX can make good use of them! See chapter 9 for more on font selection.

大字号，行间没有额外空白

注意如果没有一些额外的空间或行距,* 一行的下伸部几乎会与下一行的上伸部和大写字母相互碰撞。行距很重要——没有它，成段文本会变得不舒适，尤其是在常规正文字号下。让你的文字呼吸起来！†

靠你自己

- 学习如何使用 `ulem` 包为文本添加下划线。‡
- 使用 KOMA Script 修改章节标题的大小和样式。
- 了解斜体与倾斜体字体的区别。
- 通过重新定义 `\familydefault` 来更改默认文本样式（由 `\textnormal` 和 `\normalfont` 使用）。

*This term comes from the days of metal type, when strips of lead or brass were inserted between lines to space them out.[1]

†For a discussion of how much leading to use, see *Practical Typography*, as mentioned in Appendix B.

‡Other typographical tools—like italics, boldface, and small caps—are generally preferable to underlining, but it has its uses.

6. 标点符号

你宁愿遇到一只吃竹笋和树叶的熊猫，而不是一只“吃了、开枪、然后离开”的熊猫。2 标点是写作中至关重要的一部分，而且它的学问远不止你的键盘所暗示的那样。

引号

LATEX 并不会自动将 "直"引号转换为正确方向的“花括”引号：

```
"This isn't right."
```

会得到你

“这不对。”

相反，使用 ‘作为开引号，使用 ’ 作为闭引号。

```
``It depends on what the meaning of the word `is' is.''
```

引用一位前美国总统的话，

“这取决于‘is’这个词的含义是什么。”

Sure! Please provide the

*If your keyboard happens to have keys for “curly” quotes (“ ”), feel free to use those instead! And don’t use " for closing double quotes. Not only does ``example" look a bit unbalanced, but " is used as a formatting command for some languages, like German. (See chapter 11 for more on international typesetting.)

连字符和破折号

尽管它们看起来相似，连字符（ - ）、短破折号（ – ）、长破折号（ — ）和负号（ – ）有着不同的用途。

连字符有一些应用：3

- 它们允许单词跨越一行的末尾和下一行的开头。LaTeX 会自动执行此操作。
- 复合词如 long-range 和 field-effect 使用连字符。
- 它们用于短语形容词。如果我要求“five dollar bills”，我想要的是五张 $1 钞票，还是几张 $5 钞票？“five-dollar bills”更清楚地表明我想要后者。毫不意外，你可以通过输入连字符（ - ）来得到它。

En 破折号用于表示范围，例如“第4至12页”，以及连接词，如“美加边界”。LaTeX 会在你输入两个相邻的连字符（ -- ）时自动插入一个破折号。

破折号用来分隔句子的从句。其他标点符号——如括号和逗号——起着类似的作用。破折号的排版使用三个连字符（ --- ）。

减号用于表示负数和数学表达式。它们的长度与短破折号相似，但位置不同。减号通过 `\textminus` 设置，或者在数学环境中使用连字符（见第8章）。

椭圆

一组表示停顿或省略的三个点称为省略号。它设置为 `\dots`。

```
I'm\dots{} not sure.
```

变成

我……不太确定。

省略号的间距不同于连续的句点。不要用后者作为前者的拙劣替代。

间距

正如我们在第 3 章中发现的那样，LaTeX 会在句点与其后内容——很可能是下一句的开头——之间插入额外的空白。这并不总是我们想要的！例如，像 Mr. 和 Ms. 这样的敬称。在这些情况下，我们还需要防止 LaTeX 在句点之后另起一行。这就需要使用不换行空格，我们用波浪号来设置。

```
Please call Ms.~Shrdlu.
```

生成适当的间距：

请给 Shrdlu 女士打电话。

在其他情况下，比如当我们缩写计量单位时*，我们希望使用比通常的词间空格更窄的空格。对此，我们使用 `\,`：

```
Launch in 2\,h 10\,m.
```

宣布

2 小时 10 分钟后发射。

在 你自己的

- 了解更多用于间距的命令，例如 `\:`、`\;`、`\enspace` 和 `\quad`。
- 使用 `csquotes` 宏包的 `\enquote` 命令，可以更轻松地嵌套引号，例如：“她惊呼，‘我简直不敢相信！’”。

*There are also dedicated packages for doing so, like `siunitx`.

- 探索诸如 en、em 和 quad 等术语的排版起源。
- 熟悉 `/` 与 `\slash` 之间的差异。
- 使用 `\hyphenate` 或 `\-` 为不常见的词添加断字 .*

*LaTeX usually does a great job of automatically hyphenating words, based on a dictionary of patterns stored for each language. You should rarely need these commands.

7. 布局

理由和对齐

LATEX 令人印象深刻地排版文本。它不是像大多数文字处理器、网页浏览器和电子阅读器那样一行一行地安排文本，而是考虑段落中每一个可能的换行点，然后选择那些能提供最佳整体间距的换行[4]。结合自动断字功能，它允许在单词中间进行换行 ，能够比几乎任何其他软件生成更好的段落布局。

但是有时我们不希望文本对齐。如果你希望它左对齐，将其放入`flushleft`环境中或在当前组中添加`\raggedright`。要居中它，将其放入`center`环境中或在当前组中添加`\centering`。要右对齐，请使用`flushright`环境或`\raggedleft`。

```
\begin{flushleft}
This text is flush left, with a ragged right edge.
Some prefer this layout since the space between words
is more consistent than it is in justified text.
\end{flushleft}

\begin{center}
This text is centered.
\end{center}

\begin{flushright}
And this text is flush right.
\end{flushright}
```

sets

这段文字左对齐，右边缘参差不齐。有些人更喜欢这种布局，因为词与词之间的间距比两端对齐的文本更一致。

这段文本是居中的。

这段文本是右对齐的。

列表

LaTeX 提供了若干用于创建列表的环境：`itemize`、`enumerate` 和 `description`。在这三种环境中，每个条目都以 `\item` 命令开始。要创建项目符号列表，请使用 `itemize` 环境。使用

```
\begin{itemize}
\item 5.56 millimeter
\item 9 millimeter
\item 7.62 millimeter
\end{itemize}
```

你得到

- 5.56 毫米
- 9 毫米
- 7.62 毫米

`enumerate` 为其列表编号：

```
\begin{enumerate}
\item Collect underpants
\item ?
\item Profit
\end{enumerate}
```

从而得到

1. 收集内裤2. ？ 3. 获
利

`description` 环境以某个强调的标签开始每个条目，然后为该条目的后续行进行缩进：

```
\begin{description}
\item[Alan Turing] was a British mathematician who
    laid much of the groundwork for computer science.
    He is perhaps most remembered for his model of
    computation, the Turing machine.
\item[Edsger Dijkstra] was a Dutch computer scientist.
    His work in many domains---such as concurrency and
    graph theory---are still in wide use.
\item[Leslie Lamport] is an American computer scientist.
    He defined the concept of sequential consistency,
    which is used to safely communicate between tasks
    running in parallel.
\end{description}
```

给我们

艾伦·图灵是一位英国数学家，为计算机科学奠定了大量基础。他也许最为人所知的是他的计算模型——图灵机。艾兹赫尔·戴克斯特拉是一位荷兰计算机科学家。他在并发、图论等多个领域的工作至今仍被广泛使用。莱斯利·兰波特是一位美国计算机科学家。他提出了顺序一致性的概念，用于在并行运行的任务之间进行安全通信。

列

我们经常将页面分成多个栏，尤其是在使用 A4 或美国信纸打印时，因为对于标准 8–12 pt 的字号，这样能够提供更舒适的行宽。* 你可以在文档类中加入 `twocolumn` 选项，使所有内容分为两栏，或者使用 `multicol` 宏包中的 `multicols` 环境：

```
One nice feature of the \texttt{multicol} package
is that you can combine arbitrary layouts.
\begin{multicols}{2}
This example starts with one column,
then sets the following section as two.
The \texttt{multicols} environment splits the text
inside it so that each column is about the same height.
\end{multicols}
```

安排

`multicol` 软件包的一个优点是您可以组合任意布局。

这个示例从单栏开始，然后将后续部分设置为两栏。
环境会将其中的文本拆分，使每一栏的高度大致相同
。
`multicols`

分页符

一些命令，如 `\chapter`，插入分页符。您可以使用 `\clearpage` 添加自己的分页符。当使用 `twoside` 文档类选项进行双面打印时，您可以使用 `\cleardoublepage` 跳到下一页的前面。

*You will find different advice depending on where you look, but as a rule of thumb, aim for 45 to 80 characters (including spaces) per line. If a line is too long, readers have an uncomfortable time scanning for the start of the next one. If a line is too short, it doesn't have much inter-word spacing to adjust, which can lead to odd gaps or excessive hyphenation.

脚注

脚注对于参考文献，或读者可能觉得有帮助但并非正文关键的说明非常有用。`\footnote` 命令会在正文中的相应位置放置一个标记，然后将其参数设置在当前页面的底部：

```
I love footnotes!\footnote{Perhaps a bit too much\dots}
```

宣告

我喜欢脚注！*

靠你自己

- 使用 KOMA Script 或 `parskip` 宏包控制段落间距。
- 使用 `geometry` 宏包设置页面大小和页边距。
- 使用 `enumitem` 包自定义列表格式。
- 使用 `tabular` 环境创建表格。
- 使用 `tabbing` 环境通过制表位对齐文本。
- 使用 `footmisc` 宏包和 KOMA Script 自定义脚注符号和版式。

- 使用诸如 `\vspace`、`\hspace`、`\vfill` 和 `\hfill` 等命令插入水平和垂直间距。
- 了解 LaTeX 提供了哪些用于测量空间的单位。（我们已经在这里提到了一些，比如 `pt`、`bp` 和 `in`。）

*Perhaps a bit too much...

8. 数学

LaTeX 在数学排版方面表现出色，无论是在正文中（$x_n^2 + y_n^2 = r^2$）还是作为单独成行的公式：

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

前者在 `$...$` 或 `\(...\)` 中输入，后者在 `\[...\]` 中。在这些数学环境中，LaTeX 的规则发生变化：

- 大多数空格和换行符都会被忽略。间距的决定将根据数学排版的惯例为你自动完成。`$x+y+z$` 和 `$x + y + z$` 都会给你 $x + y + z$。
- 不允许空行——每个公式占据一个单独的“段落”。
- 字母会自动设置为斜体，因为它们被假定为变量。

要在公式中返回到正常的“文本模式”，请使用 `\text` 命令及其相关命令。标准的格式化命令在这些块中都可以使用。从

```
\[ \text{fake formulas} = \textbf{annoyed mathematicians} \]
```

我们得到

$$\text{虚假的公式} = \textbf{惹恼了数学家们}$$

示例

数学排版是 LaTeX,* 的存在理由，但仅仅涵盖基础内容我们就可能需要写上几十页。鉴于现代数学的广度，存在许多

好吧，TEX

不同的命令和环境。你应该为自己找到一些真实的参考资料，了解LaTeX的能力。但在继续之前，让我们看看它能做些什么。

1.
```
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2.
```
e^{j \theta} = \cos(\theta) + j \sin(\theta)
```

$$e^{j\theta} = \cos(\theta) + j\sin(\theta)$$

3.
```
\begin{bmatrix}
x' \\
y'
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & -\sin\theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
```

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

4.
```
\oint_{\partial \Sigma} \mathbf{E} \cdot
\mathrm{d}\boldsymbol{\ell}
= - \frac{\mathrm{d}}{\mathrm{d}t}
    \iint_{\Sigma} \mathbf{B} \cdot \mathrm{d}\mathbf{S}
```

$$\oint_{\partial\Sigma} \mathbf{E} \cdot \mathrm{d}\boldsymbol{\ell} = -\frac{\mathrm{d}}{\mathrm{d}t} \iint_{\Sigma} \mathbf{B} \cdot \mathrm{d}\mathbf{S}$$

靠你自己

- 稍后参考的数字方程使用`equaiton`环境。
- 自动调整括号和大括号的大小以适应其内容，使用`\left`和`\right`。
- 了解美国数学学会的 `amsmath` 宏包的许多有用功能，例如用于对齐等价方程的 `align` 环境。

9. 字体

自从几十年前创建了 LaTeX 以来，数字字体已经发生了翻天覆地的变化。LaTeX 最初使用 METAFONT，这是一种由 Donald Knuth 专门为 TEX 设计的格式。随着时间的推移，增加了对 PostScript* 字体的支持。如今，LuaLaTeX 和 XELaTeX 支持你在计算机上可以找到的现代字体格式：TrueType 和 OpenType。†

TrueType 由苹果和微软在 20 世纪 80 年代末开发。系统预装的大多数字体可能都是这种格式。TrueType 文件通常以 `.ttf` 扩展名结尾。

OpenType 由微软和 Adobe 于 1996 年首次发布。与 TrueType 相比，其改进之处在于能够将替代字形、间距选项等多种特性嵌入到单个文件中。OpenType 文件通常以 `.otf` 扩展名结尾。

更改字体

默认情况下，LuaLaTeX 和 XELaTeX 使用 Latin Modern，它是 LaTeX 原始字体家族 Computer Modern 的一种 OpenType 版本。虽然这些都是高质量的字体，但它们并不是你唯一想使用的字体。对于其他字体，我们转向 `fontspec` 宏包：

```
\documentclass{article}

\usepackage{fontspec}
```

*One of Adobe's original claims to fame, PostScript is a language for defining and drawing computer graphics, including type. It remains in widespread use today.

†Mac versions of LaTeX also support Apple's AAT, but let's limit this discussion to more common formats.

```
\setmainfont[Ligatures=TeX]{Source Serif Pro}
\setsansfont[Ligatures=TeX]{Source Sans Pro}
\setmonofont{Source Code Pro}

\begin{document}
Hello, Source type family! Neat---no? \\
\sffamily Let's try sans serif! \\
\ttfamily Let's try monospaced!
\end{document}
```

应该产生类似于*

你好，源类型家族！很棒——不是吗
？我们试试无衬线字体！
Let's try monospaced!

`Ligatures=TeX` 选项允许您使用第六章中的标点快捷方式 (`--` 用于短破折号，`` ` `` 和 `''` 用于弯引号等)，而无需强制输入相应的字符，这些字符可能不在您的键盘上。不过，使用等宽字体时通常不希望进行这些替换。使用等宽字体的文本——例如代码——通常是为了逐字打印。`"Hello!"` 不应变成 “`Hello!`“。

选择字体文件

`fontspec` 通常可以找到某一给定字体家族所需的文件，尤其是在你只想要基本的一组正体、斜体、粗体和粗斜体字体时。但字体家族的样式可能远不止这些。例如，本书中使用的 Futura 版本就包含 light、book、medium、demi、bold 和 extra bold 等字重。每一种字重也都有对应的倾斜体字体。一个字体家族还可能有其他变体，比如小型大写字母†，或多种光学尺寸（见第 5 章）。

*Assuming, of course, that you have Adobe's open-source fonts installed.[6]

†OpenType allows some styles, like small caps, to be placed in the same file(s) as the "main" glyphs for a given weight. If your font supports this, `fontspec` will automatically switch to them whenever you use `\textsc` or `\scshape`. But for TrueType fonts, and for OpenType fonts that don't leverage this feature, you will have to specify separate files.

我们可能希望手动挑选字重，以实现某种特定的外观，或更好地与文档中的其他字体相匹配。* 继续以 Futura 为例，假设我们希望将“book”作为默认字重，而将“demi”用于粗体。假设字体文件的命名为：

- `Futura-Boo` 直立书籍重量
- `Futura-BooObl` 用于斜放书籍的重量
- `FuturaSC-Boo` 用于小型大写字母，书本字重
- `Futura-Dem` 用于正体半粗（加粗）
- `Futura-DemObl` 用于倾斜半粗体

我们的设置可能类似于：

```
\usepackage{fontspec}
\setmainfont[
    Ligatures=TeX,
    UprightFont = *-Boo,
    ItalicFont = *-BooObl,
    SmallCapsFont = *SC-Boo,
    BoldFont = *-Dem,
    BoldItalicFont = *-DemObl
]{Futura}
```

请注意，与其逐个输入`Futura-Boo`、`Futura-BooObl`等，我们可以使用`*`来插入基名。†

*Compare how the light, book, **and medium weights** of Futura look next to the rest of the type on this page.

†This is a place where XeLaTeX and LuaLaTeX differ. The former uses system libraries—such as FontConfig on Linux—to find font files. The latter has its own font loader, based on code from FontForge.[7] Because the two look for files in different ways, the expected name of a font might differ between the two engines. See the `fontspec` package manual for details.

缩放

使用多种字体来创建统一的设计很棘手，尤其是因为它们在相同字号下可能看起来完全不同。`fontspec` 可以通过将字体缩放以匹配你的主字体的 x 高度或大写高度*，分别使用 `Scale=MatchLowercase` 或 `Scale=MatchUppercase` 来提供帮助。但绕开这一问题的一种方法，是一开始就使用更少的字型。只需一到两种，谨慎使用，就能产生惊艳的效果。

OpenType 特性

正如本章开头所提到的，OpenType 字体提供了许多可以开启和关闭的特性。在 LaTeX 中，我们通过给 `\setmainfont` 及其相关命令添加可选参数来实现这一点。也可以使用 `\addfontfeature` 为当前分组设置特性。让我们来看看一些常见的特性。

连字

许多字体使用连字，将多个字符合并为一个字形.† OpenType 将它们分为三类：

标准连字可以解决某些字符之间的间距问题。考虑小写字母 f 和 i：在许多字体中，它们会组合形成连字 fi——这可以避免 f 的升部与 i 的点之间出现尴尬的间距（fi）。英语写作中其他常见的连字包括 ff、ffi、fl 和 ffl。标准连字默认启用。

某些字体提供可选连字，例如 ct。默认情况下是禁用的，但可以通过 `Ligatures=Discretionary` 启用。

历史上的连字是指那些已经不再常用的连字，例如带长s的连字（例如，　）。这些连字默认是禁用的，但可以通过`Ligatures=Historic`启用。

*Refer back to chapter 5 for an explanation of these heights.

†Ligatures fell out of style during the 20th century due to limitations of printing technology and the increased popularity of sans serif typefaces, which often lack them. Today they are making a comeback, thanks in no small part to their support in OpenType.

多个选项可以组合在一起。假设你想要自由连字。在很可能你也想要 `Ligatures=TeX` 的情况下，你可以用 `Ligatures={TeX,Discretionary}` 同时启用两者。连字也可以通过相应的 `*Off` 选项来禁用。如果你想在某个段落中停止使用自由连字，

```
{\addfontfeature{Ligatures=DiscretionaryOff}...}
```

奏效。

有些词在没有连字时看起来反而更好——shelfful 就是一个经典例子。 你可以通过插入一个零宽空格来手动阻止连字，例如 `shelf\hspace{0pt}ful`。或者，既然人生苦短，你也可以让 `selnolig` 宏包为你处理这些情况。

图

在设置数字时，*你需要做出两个选择：排版方式与文本方式，以及比例数字与表格数字。排版数字，有时称为标题数字，其高度类似于大写字母：

A B C D 1 2 3 4

文本数字，或旧式数字，与小写字母有更多相似之处：

盘腿坐在地板上……25或6到4?

用于正文时，两种选择都可以，但不要把大写字母与文本数字（旧式数字）混用。“F-15C”看起来很怪，“V2.3 Release”也是如此。

proportional 和 tabular 这两个术语指的是间距。等宽数字采用统一的宽度，因此 1 与 8 占用相同的空间。顾名思义，这非常适合表格以及其他需要数字按列对齐的场景：

Item	Qty.	Price
Gadgets	42	$5.37
Widgets	18	$12.76

*_Figures_ is typography-speak for what we might also call _digits_ (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

比例数字则恰恰相反——它们的间距，嗯……与每个数字的宽度成比例。它们在正文中看起来更好一些：1837 在这里看起来比 1837 更自然。

你可以使用以下选项选择图形：

```
Numbers=  Lining / Uppercase
          OldStyle / Lowercase
          Proportional
          Tabular / Monospaced
```

与连字选项类似，这些也可以组合使用：比例等高数字使用 `Numbers={Proportional,Lining}` 设置，表格旧式数字使用 `Numbers={Tabular,OldStyle}` 设置。每个选项还都有一个对应的 `*Off` 变体。*

最后，一些字体提供上标和下标数字，用于排版序数（1st，2nd 3rd，……）、分数（2⁄ ）等。它们与字体中其他字符具有相同的字重，比将全尺寸数字缩小得到的版本在视觉上更一致。（将上面的示例与它们的“冒牌货”比较：1^{st}、2^{nd}、3^{rd} 和 $^{25}/_{624}$。注意第二组与周围文字相比显得过轻。）上标数字使用 `VerticalPosition=Superior` 排版，下标数字则使用 `VerticalPosition=Inferior`。

靠你自己

- 了解 `fontspec` 如何根据点大小选择光学尺寸，可以从 OpenType 字体中嵌入的范围自动选择，或使用 `SizeFeatures` 手动选择。

- 尝试使用 `LetterSpace` 选项来调整字距——或字距跟踪。大多数情况下不需要增加字距，但它可以让小型大写字母更易读一些。

*This is especially useful since different fonts have different defaults. Some fonts use lining figures by default and enable text figures with `Numbers=OldStyle`. Others default to text figures and require `Numbers=Lining`.

10. 微排版

微排版通过细微、潜意识的调整提高文本的可读性。它是

> [⋯]增强文档外观和可读性的艺术，同时表现出最小程度的视觉干扰。它关注的是字符、单词或行之间或边缘发生的情况。而文档的宏观排版方面（即其布局）即使对未经训练的眼睛来说也能清楚地看到，而微观排版的细节理想情况下应该是无法察觉的。也就是说，您可能觉得文档看起来很美，但可能无法确切说明为什么：良好的微观排版实践力图减少所有可能干扰读者的潜在因素。

在 LaTeX 中，微排版通过 `microtype` 包控制。其使用是自动的——对于绝大多数文档，您应该添加

```
\usepackage{microtype}
```

查看您的前言并继续。但让我们快速看看这个包实际上做了什么。

字符突出

默认情况下，LaTeX 将行对齐到完全直的边距。这是一个显而易见的默认设置，但它容易受到一种恼人的视觉错觉的影响：以小字符（如句点、逗号或连字符）结尾的行，看起来比没有这些字符的行要短。* `microtype` 通过将这些字符突出到边距之外来补偿。

*Many other optical illusions come up in typography. For example, if a circle, a square, and a triangle of equal heights are placed next to each other, the circle and triangle look smaller than the square. For this reason, round or pointed characters (like O and A) must be slightly taller than “flat” ones (such as H and T) for all to appear the same height.[10]

字体扩展

为了让段落间距更加均匀并减少连字符断行，`microtype` 可以在水平方向拉伸字符。你可能会认为这样扭曲字形会立刻显而易见，但你正在阅读的一本书在每一页都这样做！这种称为字体扩展的效果应用得非常轻微——默认情况下，字符宽度的改变不超过百分之二。*

此功能目前不适用于 XeLaTeX。如果你想使用它，你需要 LuaLaTeX。

靠你自己

一如既往，请参阅该软件包的手册以了解如何调整这些功能。`microtype` 还有一些其他技巧，但其中有几个只在较旧的 LaTeX 引擎上才能使用。† 我们关心的那些——例如字距调整——可以通过 `fontspec` 或其他软件包来处理。

*Of course, you can use package options to change this limit, or disable the feature entirely.

†i.e., pdfTeX

11. 国际字体排印

令人惊讶的是，除了英语之外还存在其他语言。你可能会想用它们来写作。

Unicode

将书面语言数字化是一个复杂的主题，自 LaTeX 诞生以来已经发生了显著的发展。如今，大多数软件使用 Unicode 来表示文本。简而言之，

- Unicode 文本文件是一系列码点。每个码点表示一个要绘制的字符、一个与相邻字符组合的重音或附加符号，或格式信息，例如指示将后续文本从右向左显示的指令。
- 一个或多个这些代码点组合在一起表示一个字形簇或字形，字体中的形状我们非正式地称之为“字符”。

  你好 你好 你

  你看到了多少个字符？多少个码点？

- 现代字体格式包含一些表，用于将码点映射到文件中包含的字形。

LuaLaTeX 和 XELaTeX 具备对 Unicode 的良好支持，并且能很好地与 Unicode 文本文件配合使用。* 请确保你选择的字体包含你所需要的字形——许多字体只支持拉丁语言。

*LuaLaTeX accepts UTF-8 files. XeLaTeX also accepts UTF-16 and UTF-32.

多语言学

当你的文档包含英语以外的语言时，考虑使用 `polyglossia` 宏包。它将自动：

- 加载特定于语言的断字规则和其他约定。
- 为每种语言切换到用户指定的字体。
- 翻译文档标签，例如“章节”、“节”等。
- 根据特定语言的惯例格式化日期。
- 在拥有自身数字系统的语言中格式化数字。
- 对于包含从右向左书写语言的文档，请使用 `bidi` 软件包。
- 设置具有脚本和语言标签的 OpenType 字体。

要使用 `polyglossia`，请指定文档的主要语言，以及它使用的任何其他语言。某些语言还可以将地区方言作为可选参数：

```
\usepackage{polyglossia}
\setdefaultlanguage[variant=american]{english}
\setotherlanguage{french}
```

一旦设置完成，`polyglossia` 会为所请求的语言定义环境。每个环境都会自动将其语言的规范应用到其中的文本。例如，法语会在标点符号周围留出额外的空格，所以

```
Dexter cried,
\begin{french}
«Omelette du fromage!»
\end{french}
```

给出

德克斯特哭了，“Omelette du* fromage！”

*Yes, it's *omelette au fromage*. Direct all complaints to Genndy Tartakovsky.

靠你自己

- 请参阅 `polyglossia` 手册以获取语言特定的命令。
- 考虑将 `babel` 软件包作为 `polyglossia` 的替代方案.*
- 尝试使用 `xeCJK` 或 `luatex-ja` 软件包来排版日文或中文。

*`polyglossia` has better support for OpenType font features via `fontspec`. However, it is newer and has a few known bugs. `babel` is a fine substitute if you run into trouble.

12. 当好类型变坏时

如果运气不错，你已经用 LaTeX 有了一个扎实的开端。但和任何复杂的工具一样，你最终都会遇到麻烦。下面列出了一些常见问题以及你可以尝试的解决办法。

修复溢出

当 LaTeX 无法将一个段落断成间距良好的行时，它就会放弃并溢出到页边距。你有时可以通过一些“应急拉伸”来补救。如果你在导言区添加 `\emergencystretch=<width>`，LaTeX 会尝试第二次排版那些棘手的段落，在每一行中将总空白拉伸或收缩，幅度最多不超过所提供的宽度。* 如果这仍然无济于事，就调整问题段落的措辞。这可能令人沮丧，但另一种选择是让 LaTeX 生成过于松散的间距——单词之间出现很大的空隙——或者过于紧密，单词被别扭地挤在一起。

避免孤行和寡行

良好的版式会避免孤行和寡行（也称为“俱乐部行”）：即被页面边界与其段落其余部分分隔开的行。LaTeX 会尝试防止这些情况，但它的页面拆分算法远不如其段落拆分算法成熟。†你可以让 LaTeX 更努力地避免孤行和寡行：

*LaTeX has pretty sane defaults for how much it stretches and shrinks spacing. You probably don't want to make `<width>` larger than an em or two.

†This is because 1980s computers didn't have enough RAM to do so. Seriously—Knuth wrote at the time, "The computer doesn't have enough high-speed memory capacity to remember the contents of several pages, so TeX simply chooses each page break as best it can, by a process of 'local' rather than 'global' optimization."[11]

```
\widowpenalty=<penalty>
\clubpenalty=<penalty>
```

`<penalty>` 是一个介于 0 到 10000 之间的值。当这些值被最大化时，LaTeX 在任何情况下都不允许留下孤行或寡行。* 这可能会产生一些非常奇怪的版式，因此如果你选择较大的惩罚值，一定要检查你的页面。

处理语法错误

如果你把 LaTeX 搞糊涂了——比如发出了不存在的命令，或者忘记结束一个环境——它会打印一条错误消息,† 然后显示一个以 `?` 开头的交互式提示符。在这里你可以输入关于如何继续的指示。曾经，当计算机慢上数千倍、而 LaTeX 重新运行也要花那么久的时候，这种方式更有用。如今，我们大概只想退出，等修复好文档后再试一次。要退出该提示符，输入 `x`，然后按 Enter。更好的办法是，你可以在运行引擎时使用 `-halt-on-error` 选项，让 LaTeX 一遇到问题就放弃：

```
$ lualatex -halt-on-error myDocument.tex
```

祝你好运！

*When considering a given layout, LaTeX assigns penalties, or "badness", to anything that arguably makes a document look worse. It chooses whichever layout it can find with the least badness.

†Usually this contains a succinct summary of the problem and the number of the line(s) it occurred on. Occasionally, LaTeX gets *really* confused and emits something so cryptic it gives C++ template errors a run for their money. As you use LaTeX, you will start to get a feel for what sorts of mistakes cause these rare, but enigmatic messages.

A. LaTeX 简史

唐纳德·克努斯（Donald Knuth）是计算机科学之父之一，以开创算法分析以及其仍在持续进行的鸿篇巨著《计算机程序设计艺术》而闻名。1968 年 taocp 第一卷发行时，其印刷方式仍沿用自世纪之交以来的做法：采用热金属排版。字母由熔融的铅浇铸而成，然后排列成行。这些行被夹紧组合成页面，随后上墨并压印到纸张上。

1977年3月，克努斯已准备好对《计算机程序设计艺术》（TAOCP）第2卷进行第二次印刷，但当他收到校样时却大为震惊。使用热金属排版既昂贵又复杂，而且耗时，因此出版社——包括克努斯的出版社在内——转而采用照相排版，这是一种将字符投射到胶片上的工艺。新技术虽然更便宜、更快捷，却未能提供克努斯所期望的质量。12

一般的作者可能会对这种变化听之任之、继续前行，但 Knuth 对自己书籍的排版引以为豪，尤其是在数学内容方面。受到他最近对不断发展的数字排版领域的接触所启发，Knuth 踏上了史上最伟大的“削牦牛”*之一。他把其他工作搁置一旁，在这个问题上苦干了一年多。到 1978 年尘埃落定时，Knuth 向世界介绍了 TEX.†

很难体会 TEX 曾经是一场多么深刻的革命，尤其是从今天回望——如今任何一个拥有网页浏览器的人都可以成为自己的桌面出版者。Adobe 的 pdf 还要再过十年才会出现，因此 Knuth 和他的研究生们设计了他们自己的设备无关文档格式 dvi。可缩放字体并不常见，于是他创建了 METAFONT，将字形栅格化为页面上的点。也许最重要的是，Knuth 及其学生设计了能够自动进行断字和两端对齐的算法，把文本排成美观的段落。‡

LATEX 是 Lamport TEX 的简称，由 Leslie Lamport 于 20 世纪 80 年代初开发，作为一组用于常见文档布局的命令。他于 1986 年在其指南《LaTeX：文档准备系统》中介绍了它。开发

*Programmers call seemingly-unrelated work needed to solve their main problem “yak shaving”. The phrase is thought to originate from an episode of *The Ren & Stimpy Show*.[13]

†The name “TEX” comes from the Greek τέχνη, meaning *art* or *craft*.[14]

‡These same algorithms went on to influence the ones Adobe uses in its software today.[15]

至今仍在继续，既以用户提供的TEX和LATEX的包的形式，也作为对TEX排版程序本身的改进。有四个版本，或称为引擎：

TEX 是唐纳德·克努斯创建的原始系统。克努斯在 1990 年 3 月的 3.0 版之后停止增加新功能——此后的版本只包含错误修复。随着每次发布，版本号通过增加一个额外的数字而渐近地逼近 π。最新的版本 3.1 41592653 于 2021 年发布。

pdf TEX 是 TEX 的一个扩展，提供直接的 PDF 输出、对 PostScript 和 TrueType 字体的原生支持，以及第 10 章中讨论的微排版特性。它最初由 Hàn Thế Thành 开发，作为其在捷克共和国布尔诺马萨里克大学博士论文的一部分。1

XETEX 是 TEX 的进一步扩展，增加了对 Unicode 和 OpenType 的原生支持。它最初由 Jonathan Kew 于 2000 年代初开发，并在 2007.1 获得了完整的跨平台支持。

LuaTEX 在原生 Unicode 和现代字体支持方面与 XETEX 相似。它还将 Lua 脚本语言嵌入到引擎中，为宏包和文档作者提供接口。它最早于 2007 年出现，由 Hans Hagen、Hartmut Henkel、Taco Hoekwater 和 Luigi Scarso 组成的核心团队开发。1

构建TEX今天是一个…有趣的任务。当它在1970年代末编写时，计算机科学学生没有大型、文档齐全的开源项目可以学习，所以Knuth决定创建一个。作为这项努力的一部分，TEX以他称之为“文学化编程”的风格编写：与大多数程序不同——这些程序中文档通常穿插在代码中——Knuth将TEX写成一本书，代码被插入到段落之间。这种英语与代码的混合被称为WEB。

不幸的是，现代系统并没有为 TEX 所使用的 20 世纪 70 年代的 Pascal 方言提供良好的工具，因此当今的发行版使用另一个程序 web2c，将其 WEB 源码转换为 C 代码。pdf TEX 和 XETEX 是通过将该结果与其他 C 以及 C++ 源码组合来构建的。作为这种复杂方法的替代方案，LuaTEX 的作者将 Knuth 的 Pascal 手工翻译成了 C。他们自 2009.1 起一直使用该代码。

*Knuth also released a pair of companion programs named TANGLE and WEAVE. The former extracts the book—as TEX, of course—and the latter produces TEX's Pascal source code.

B. 附加资源

适用于 LaTeX

正如一开始所承诺的，本书并不完整。简而言之，LaTeX 的一些主要特性——如插图、图注、表格、图形和参考文献——尚未讨论。使用以下一些资源来填补空白：

LaTeX 维基书，位于 https://en.wikibooks.org/wiki/LaTeX；《不那么简短的 LaTeX 入门》，可在 https://www.ctan.org/tex-archive/info/lshort/english/ 获取；ShareLaTeX 知识库，位于 https://www.sharelatex.com/learn；TEX Stack Exchange，位于 https://tex.stackexchange.com/

用于排版

我们在这里把大部分时间都花在介绍你可以用 LaTeX 做什么上，而较少讨论你应该如何使用它来创建设计良好的文档。请继续阅读：

《实用排版》，作者 Matthew Butterick。可在 https://practicaltypography.com 免费获取！

别再偷羊了：弄清字体是如何运作的，作者：埃里克·斯皮克曼

用字体思考，作者：艾伦·卢普顿

《塑造文本》，扬·米登多普著

《文字排印风格要素》，作者：罗伯特·布林赫斯特

《字体排印中的细节》，作者：约斯特·霍赫利

注释

1. 扬·米登多普，《塑造文本》（阿姆斯特丹，2014），71

2. Lynne Truss,《吃、射击与留白》（纽约，2003）

3. Matthew Butterick,“连字符与破折号”，实用排版学, https://practicaltypography.com/hyphens-and-dashes.html

4. 唐纳德·E·克努特和迈克尔·F·普拉斯，《将段落拆分为行》（斯坦福，1981年）

5. Franklin Mark Liang, 计算机分词（斯坦福，1983）, http://www.tug.org/docs/liang/

6. Adobe的开源字体可以在 https://github.com/adobe-fonts免费下载 7. LuaTEX参考手册（版本1.0.4，2017年2月），10

8. Knuth，《The TEXbook》（Addison-Wesley，1986），19

9. R Schlicht, 微型排版包 (v2.7a, 2018年1月14日), 4 10. Jost Hochuli, 排版细节 (Éditions b42, 2015), 18–19 11. Knuth,《TEX手册》, 110 12. Knuth,《数字排版》(斯坦福大学, 1999), 3–5 13.“yak shaving”,《术语文件》, www.catb.org/~esr/jargon/html/Y/yak-shaving.html

14. Knuth，《TEXbook》，1

15. 一些来源（http://www.tug.org/whatis.html, https://tug.org/interviews/thanh.html, http://www.typophile.com/node/34620）提到TEX对

由 Peter Karow 和 Hermann Zapf 开发的 hz 程序，得益于 Knuth 与 Zapf 的合作。hz 随后被 Adobe 收购，并用于创建 InDesign 的段落格式化系统。16. Hàn Thế Thành，《TEX 排版系统的微排版扩展》（马萨里克大学布尔诺，2000 年 10 月）17. Jonathan Kew，“XETEX Live”，TUGboat 29，第 1 期（2007）18. http://www.luatex.org 19. Taco Hoekwater，LuaTEX 告别 Pascal（MAPS 39，EuroTEX 2009），https://www.tug.org/TUGboat/tb30-3/tb96hoekwater-pascal.pdf

书志

本指南使用 LuaLaTeX 排版，采用 Robert Slimbach 的 Garamond Premier 字体，这是对 16 世纪法国字模雕刻师 Claude Garamond 所创作的罗马体的一次复兴。斜体则汲取自 Garamond 同时代人 Robert Granjon 的作品。
等宽项目采用 Drive Mono 排版，由 Black Foundry 的 Elliott Amblard 和 Jérémie Hornus 设计。

说明文字使用 Neue Haas Grotesk，这是由 Christian Schwartz 修复的 Helvetica。早期对这种无处不在的瑞士字体进行的数字化，基于为 Linotype 行铸机和照相排字机制作的字体，导致数字版本继承了前两代印刷技术中不必要的妥协和拼凑式补丁。Schwartz 的工作以 Helvetica 的原始设计图为基础，打造出忠实于原始冷金属活字的设计。

URW Futura 偶有客串亮相。由保罗·雷纳设计并于 1927 年首次发布的 Futura 几乎无处不在，从广告和政治宣传到登月。道格拉斯·托马斯最近关于这款字体的历史著作《Never Use Futura》读来精彩。

各种非拉丁文本片段采用 Noto 排版，这是谷歌推出的一个字体家族，覆盖了 Unicode 标准中的所有字形。

最后，Latin Modern——全书中使用的 Knuth 的 Computer Modern 的 OpenType 版本——以及 TEX Gyre Termes——第 1 页所见 Times Roman 的免费替代字体——均出自 Grupa U ytkowników Systemu TEX（波兰 TEX 用户组）之手。关于这些出色项目的概览可在 gust 网站上找到：

http://www.gust.org.pl/projects/e-foundry/latin-modern
http://www.gust.org.pl/projects/e-foundry/tex-gyre.