福昕PDF编辑器 个人版

• 永久 • 轻巧 • 自由

立即下载

购买会员

永久使用

无限制使用次数

极速轻巧

超低资源占用，告别卡顿慢

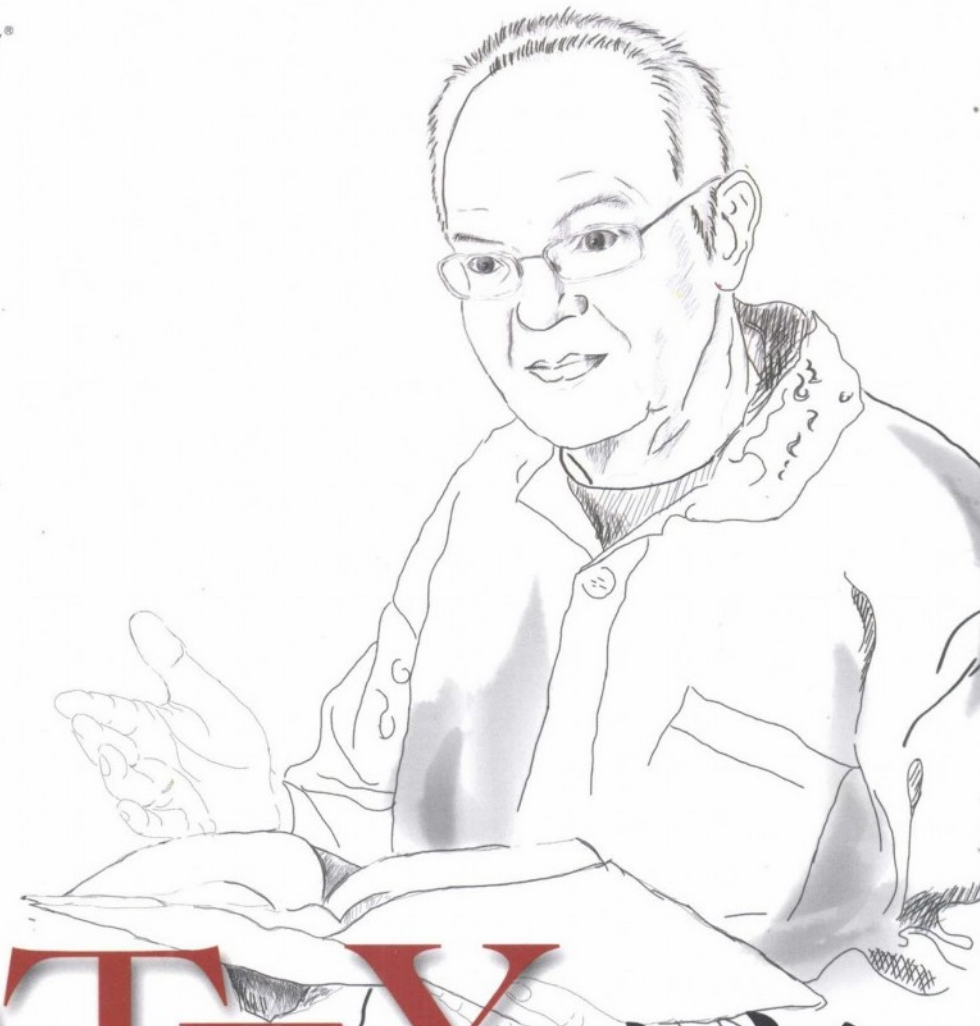
自由编辑

享受Word一样的编辑自由

扫一扫，关注公众号

http://edit.foxitreader.cn

Broadview®
www.broadview.com.cn

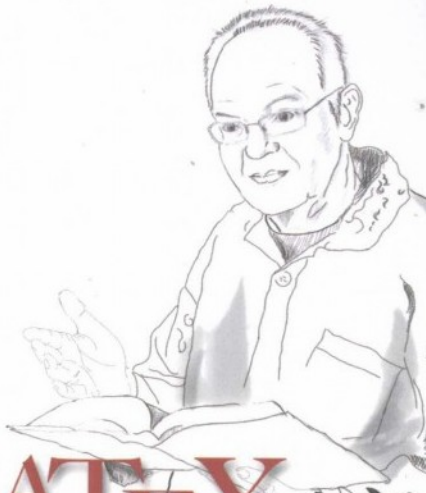


LaTeX入门

◎ 刘海洋　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Broadview®
www.broadview.com.cn
博文视点·IT出版旗舰品牌
技术凝聚实力·专业创新出版



LaTeX入门

◎本书从介绍TeX到LaTeX的发展开始，详细讲述了使用LaTeX撰写科技文档以及幻灯片演示的基本方法和技巧。书的内容深入浅出，对于LaTeX初学者以及科技工作者都是一本很好的参考书籍。

刘利刚，中国科学技术大学，教授

◎本书两大特色，“全”与“新”：不仅全面涵盖了LaTeX的基础知识，而且包含了近几年LaTeX的最新发展，内容丰富，层次分明，是一本很好的新手入门教程，对有一定使用经验的老手而言，也是一本完善的案头参考书。

韩建成，CTeX，版主

◎很多LaTeX用户对LaTeX的认识依然很模糊，这本书会让LaTeX入门变得清晰而专业。
虽然我只阅读了样稿，不得不说作者写得非常用心、认真。对LaTeX入门来说，这是不可多得的好书、必读书。

王昭礼，ChinaTeX，版主

策划编辑：张月萍
责任编辑：高洪霞
封面设计：张　昱

上架建议：计算机／办公软件

ISBN 978-7-121-20208-7

定价：79.00 元

LaTeX 入门

刘海洋 编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

LaTeX 已经成为国际上数学、物理、计算机等科技领域专业排版的实际标准，其他领域（化学、生物、工程、语言学等）也有大量用户。本书内容取材广泛，涵盖了正文组织、自动化工具、数学公式、图表制作、幻灯片演示、错误处理等方面。考虑到 LaTeX 也是不断进化的，本书从数以千计的 LaTeX 工具宏包中进行甄选，选择较新而且实用的版本来讲解排版技巧。

为了方便读者的学习，本书给出了大量的实例和一定量的习题，并且还提供了案例代码。书中的示例大部分来自作者多年的实际排版案例，读者不断练习，肯定能掌握 LaTeX 的排版技能。

本书适合数学、物理、计算机、化学、生物、工程等专业的学生、工程师和教师阅读，也适合中学数学教师。此外，本书还适合对 LaTeX 排版有兴趣的人员。

图书在版编目（CIP）数据

LaTeX 入门 / 刘海洋编著. —北京：电子工业出版社，2013.6
ISBN 978-7-121-20208-7

Ⅰ. ①L… Ⅱ. ①刘… Ⅲ. ①排版－应用软件 Ⅳ. ①TS803.23

中国版本图书馆 CIP 数据核字（2013）第 079359 号

策划编辑：张月萍
责任编辑：高洪霞
印　　刷：北京京科印刷有限公司
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：36.25　字数：632 千字
印　　次：2013 年 6 月第 1 次印刷
印　　数：4000 册　定价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

序

看了本书的样稿后使人感到印象深刻。本书充分反映了 TEX 的最新进展，尽管 TEX 的生命力是顽强的，TEX 的基本命令系统也是稳定的，但是它对非西方语言的扩展以及输出格式等都随着计算机技术的发展以及科技文献传播方式的变化而不断推陈出新，这也正是 TEX 能经久不衰的生命力所在。因此推广 TEX 的书也需要与时俱进。我们写的《LATEX 入门与提高》的第二版至今已有 7 年了，可惜它的作者或者已退休，或者兴趣转移，不可能再作更新。我一直期待能有人出来写一本反映最新发展的 TEX 入门书作为我们那本书的补充及更新。现在看到了刘海洋的《LATEX 入门》，觉得这正是我所期望的，甚至超过了我的期望。本书文笔活泼，阅读起来像是面对一位向你细细讲解的和蔼老师，他了解你的需求和会遇到的难点，使你爱不释手，而不像一般的软件说明书，只管板着脸罗列一大堆用法，不管你是否需要或是否理解。但是本书作者又很严谨，许多内容都有出处，好像一篇科研论文。不过说到底，这是一本面向读者需求的学习指导书，并非 TEX 的说明书。这正是想学习 TEX 的人最想要的书。而且第 8 章还讲到了更深入的技巧。因此本书的适用范围可以从初学者直至想自己设计版面或宏的高级应用者。大家都能从本书学到很多东西。尽管国内在 TEX 的普及与发展方面与西方发达国家相比还有很大的差距，但是感谢许多热心的 TEX 爱好者及他们的网站的努力，TEX 在中国的推广也是富有成效的。越来越多的研究生用 TEX 写作论文或向期刊投稿，并且在答辩或演示时也广泛使用 TEX 生成的 PDF。希望本书的出版能为 TEX 在中国的普及作出新的贡献。

陈志杰
华东师范大学数学系教授

2013 年 3 月 5 日

前言

提到 LaTeX，便不能不说起它的基础 TeX。TeX 是诞生于 20 世纪 70 年代末到 80 年代初的一款计算机排版软件，用来排版高质量的书籍，特别是包含有数学公式的书籍[124; 126]。TeX 以追求高质量为目标，很早就实现了矢量描述的计算机字体、细致的分页断行算法和数学排版功能，因其数学排版能力得到了学术界的广泛使用，也启发了不少后来复杂的商业计算机排版软件。有趣的是，这样一款排版软件却并非在排版业界产生，而是由计算机科学家高德纳教授在修订其七卷本巨著《计算机程序设计艺术》的前三卷[127–129] 时，为了排版这一部书籍而产生的。这是一部花费高德纳几乎毕生精力的巨著，直到今天仍在撰写，然而在照相排版技术刚刚兴起的 1976 年，新的计算机系统却无法提供与传统手工排版相媲美的质量。面对这种情况，高德纳抱怨道[130]：

> 我不知道怎么办。我花了整整 15 年写这些书，可要是这么难看，我就再也不写了。我怎么能对这样的作品引以为豪呢？

从翌年开始，高德纳就在其学生、友人的帮助下，开发 TeX 排版软件。直到 8 年后 TeX 软件功能完备，他才又回到撰写书籍的工作中去。这段历史一直被引为 TeX 和高德纳的传奇，有“十年磨一剑”之称。TeX 原本是用于个人的排版软件，这也引出了 TeX 与其他专业排版软件的一点重大的区别，就是 TeX 主要是由书籍、文章的作者本人来使用的，它是面向作者的。因此，TeX 有许多方便作者的自定义功能，使用也简单方便，很快受到作者们的青睐，排版自己的学术书籍。

LaTeX 肇始于 20 世纪 80 年代初，也是 Leslie Lamport 博士为了编写他自己的一部书籍而设计的[137]。LaTeX 实际上就是用 TeX 语言编写的一组宏代码，拥有比原来的 TeX 格式（Plain TeX）更为规范的命令和一整套预定义的格式，隐藏了不少排版方面的细节，可以让完全不懂排版理论的学者们也可以比较容易地将书籍和文稿排版出来。LaTeX 一出，很快更为风靡，在 1994 年 LaTeX 2_ε 完善之后，现在已经成为国际上数学、物理、计算机等科技领域专业排版的事实标准，其他领域（化学、生物、工程、语言学等）也有大量用户。相关专业的学术期刊也都主要接受 LaTeX 作为投稿格式。

既然 TeX/LaTeX 主要是面向作者本人的排版软件，本书的目标对象也就以学术文章的作者为主，也就是需要经常编写 LaTeX 稿件的高校师生和科研院所的研究人员。本书的内容选择以满足学术排版需求为准，阅读本书后读者应该不仅能应对各种学术投稿的简单需要，也将有能力排版一般的学术书籍，并使用 LaTeX 完成简单的学术报告幻灯片。不过，本书也力图广泛取材，让排版公司的工人、中学数学教师或是用 LaTeX 作笔

记的电脑程序员都能有所得。

本书虽然名为“入门”，假定读者没有任何使用 TEX 的经验，但为了避免读者逡巡于门外而不入，也力图使内容详实可靠，为更深入地使用 LATEX 打好基础。在编写本书时，作者追求以下几个目标：

- **内容广泛** 本书从软件安装和最基本的示例讲起，然后按正文组织、自动化工具、数学公式、图表制做、幻灯片演示、错误处理等方面详述 LATEX 的功能和使用，最后收束于 LATEX 的扩展、相关工具和资源。LATEX 的基本内容并不多，功能也很有限，但经过 20 多年的发展，现代 LATEX 文档的一大特点是大量使用工具宏包来完成复杂的工作。本书也力图体现这一特点，全书过半的篇幅都在讲解各种重要的 LATEX 宏包和工具。本书正文共有 566 页，作为一本入门书已是嫌多，但仍不可能包罗 LATEX 的所有方面，未免有遗珠之憾，只能留待读者自己学习了。
- **取材从新** TEX 最初的一个设计目标是良好的稳定性，希望在多年前编写的文档在最新的系统中排版仍能得到完全相同的结果，各种排版命令的语义保持稳定，TEX 也确实做到了这一点。然而 LATEX 是一个更为开放的系统，与其他软件一样，它是在不断进化的。不仅其内核从最初的 LATEX 2.09 到 LATEX 2_ε 再到正在开发中的 LATEX3 不断变化，而且还有数以千计的工具宏包在不断更新，完成各种复杂的排版功能。实现 TEX 语言的 TEX 引擎，也在不断增添新的功能。为了反映这种变化，本书作者也尽量对内容加以甄别，选取较新并且实用的软件工具加以介绍。
- **切合实用** 为了增强实用性，本书给出了大量实例和一定量的习题。第 1 章和第 6 章提供了两段完整的文档源代码，而其他章节也给出了大量的代码示例。代码示例和习题很多都源自作者历年来收集的各类实际的排版问题，相信对于本书的读者也会有所裨益。

为了照顾不同层次的读者，本书按 LATEX 的不同功能编排章节，章节之间没有严格的顺序关系，阅读本书也不必完全依照章节顺序。

- 希望快速上手的初学者应首先阅读第 1 章，安装好 TEX 软件并在 1.2 节学习基本的实例，然后就可以模仿实例编写自己的 LATEX 文档了，等到实际需要时再翻到对应的章节了解具体内容。
- 希望系统学习 LATEX 的读者可以从前往后依次阅读。书中一些段落前，或整个一节之前有一个危险标记，说明这一段或一节内容较难或者依赖后面章节的内容，在初次阅读时可以略过，可以在读完基本内容后再来了解这部分内容。还有一些段落前有两个危险标记，则表示这些内容中部分已经超出本书的范围，通常需要参见书中引用的其他文档才能完全理解。

- 具有一定 LaTeX 经验的读者可以根据自己的需要查找有用的内容，书后的索引将有助于找到特定的概念或命令，而每章末尾的注记与书后的文献列表则可以帮助读者找到本书中未能详述的内容。

本书使用不同的字体表示不同的内容。在正文中，使用等宽字体表示代码，如 `\alpha` 命令、`equation` 环境；用无衬线字体表示宏包名称，如 amsmath 宏包、beamer 文档类；用尖括号内的楷体（西文斜体）表示参数，如 ⟨长度⟩、⟨*key*⟩。在表示 LaTeX 命令或环境的语法形式时，则使用加粗的等宽字体，如：

\usefont⟨编码⟩⟨族⟩⟨系列⟩⟨形状⟩

书中给出了大量示例代码。大部分示例以左右对照的方式给出，左侧灰色框中是代码，右侧白色框中是代码的排版效果，例如：

0-1

```
$\Delta = b^2 - 4ac$
```

$\Delta = b^2 - 4ac$

较长的示例则以上下对照的方式给出，如：

0-2

```
\[
  x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]
```

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

还有一些代码示例没有直接的排版结果，则只给出源代码。如上所示，示例通常会有一个编号以方便引用。本书中所有带编号的示例和第 1 章、第 6 章的两个大的例子会随书附带，也可以在 CTeX 论坛网站上获取。

书中在部分章节后面安排了一些题外的内容，在标题前用书籍符号标示（如右），内容用楷体字印刷。这些内容游离于本书的主线之外，主要介绍一些背景知识，读者可根据自己的兴趣选择阅读。

此外，在部分章节后还设置了少量的练习题，用铅笔符号标示（如右），读者可据此检查自己是否掌握了正文中的内容。这些题目并非为了把读者难住，大部分练习在书末都有解答或提示。

在本书编写过程中，许多朋友都为作者提供了无私的帮助。韩建成阅读了本书早期的草稿和初稿，在结构和内容方面都提出了宝贵的意见和建议；赵劲松和李清阅读了本书的初稿，并在内容上给出了详细的建议与勘误；江疆和王越在阅读初稿后，对

本书的内容和格式都提出了宝贵的意见。本书的编写一直受到在 CTEX 论坛与水木社区 TEX 版上网友们的关注和支持，论坛中对 LATEX 具体问题的大量讨论时常能启发作者的思路，为成书提供了重要的素材。在此，作者向所有关心本书的人们致以真诚的感谢！

作者已尽力使本书准确可靠，但受精力和水平所限，书中的错误在所难免。欢迎读者指出书中的技术上的、文字上的或是排版上的任何错误。有关本书的各种问题，可发送电子邮件至 info@dozan.cn 联系本书出版策划。

刘海洋

目录

第1章

熟悉 LaTeX

LaTeX 是一种基于 TeX 的文档排版系统。TeX 只这么交错起伏的几个字母，便道出了“排版”二字的几分意味：精确、复杂、注重细节和品位。而 LaTeX 则为了减轻这种写作、排版一肩挑的负担，把大片排版的格式细节隐藏在若干样式之后，以内容的逻辑结构统帅纷繁的格式，遂成为现在最流行的科技写作——尤其是数学写作的工具之一。

无论你是因为心慕 LaTeX 漂亮的输出结果，还是因为要写论文投稿被逼上梁山，都不得不面对一个事实：LaTeX 是一种并不简单的计算机语言，不能只点点鼠标就弄好一篇漂亮的文章，也不是一两个小时的泛泛了解就尽能对付得过去的[①]。还得拿出点上学搞研究时的那股钻研劲儿，才能通过手指下的键盘，编排出整齐漂亮的文章来。

LaTeX 的读音和写法

TeX 一名源自 technology 的希腊词根 τεχ，TeX 之父高德纳教授[②]近乎固执地要求[126] 它的发音必须是（按国际音标）[tɛx]，尽管英语中它常被读做 [tɛk]。（同样，高德纳教授也近乎固执地要求别人说他的姓 Knuth 时不要丢掉“K”，叫他 Ka-NOOTH，尽管在英语环境他时常会变成 Nooth 教授。）对比汉语，TeX 的发音近似于“泰赫”，而且可以用汉语拼音准确地拼出来：têh（或许老一辈的人习惯用注音：ㄊㄝˋㄏ）。

① 是的，有一个著名的入门教程就叫《112 分钟学会 LaTeX》[187]。不过这个分钟其实是以页码计算的，粗粗浏览一遍还远算不上学会。而且即使掌握了这个教程中的内容，仍然可能在实际写作中遇到许多难以解决的问题。本书同样不打算让你能迅速变成一个高手。

② Donald Ervin Knuth，Stanford 大学计算机程序设计艺术荣誉教授，Turing 奖和 von Newmann 奖得主。高德纳是他的中文名字。TeX 系统就是高德纳为了排版他的七卷本著作《计算机程序设计艺术》而编制的。

LaTeX 这个名字则是把 LaTeX 之父 Lamport 博士[①]的姓和 TeX 混合得到的。所以 LaTeX 大约应该读成“拉泰赫”。不过人们仍然按着自己的理解和拼写发音习惯去读它：['lɑːtɛk]、['leɪtɛk] 或是 [lɑː'tɛk]，甚至不怎么合理的 ['leɪtɛks]。好在 Lamport 并不介意 LaTeX 到底被读做什么。“读音最好由习惯决定，而不是法令。”——Lamport 如是说 [136, § 1.3]。

两个创始人对于名称和读音的不同态度或许多少说明了这样一个事实：LaTeX 相对原始的 TeX 更少关注排版的细节，因此 LaTeX 在很多时候并不充当专业排版软件的角色，而只是一个文档编写工具。而当人们在 LaTeX 中也抱以追求完美的态度并用到一些平时不大使用的命令时，通常总说这是在 TeX 层面排版——尽管 LaTeX 本身正是运行于 TeX 之上的。

类似地，TeX 和 LaTeX 字母错位的排印也体现出一种面向排版的专业态度，即使在字符难以错位的场合，也应该按大小写交错写成 TeX 和 LaTeX。

现在我们使用的 LaTeX 格式版本为 2_ε，意思是超出了第 2 版，接近却没有达到第 3 版，因此写成 LaTeX 2_ε。在只能使用普通字符的场合，一般写成 LaTeX2e。

1.1 让 LaTeX 跑起来

学习 LaTeX 的第一步就是上手试一试，让 LaTeX 跑起来。首先安装 TeX 系统及其他一些必要的软件，然后跑一个测试的例子。下面的几节包含了一大堆具体软件安装和使用的内容，虽然有些烦琐，但这是使用 LaTeX 进行写作的必要前提。如果你早已做好这些准备，或者在读本书以前就已经迫不及待地做了不少尝试的话，可以直接跳到第 32 页 1.2 节开始第一个实际规模的例子。

1.1.1 LaTeX 的发行版及其安装

TeX/LaTeX 并不是单独的程序，现在的 TeX 系统都是复杂的软件包，里面包含各种排版的引擎、编译脚本、格式转换工具、管理界面、配置文件、支持工具、字体及数以千计的宏包和文档。一个 TeX 发行版（Distribution）就是把所有这样的部件都集合起来，打包发布的软件。

尽管内容庞杂，但现在的 TeX 发行版的安装还是非常方便的。下面将介绍两个最为流行的发行版，一是 1.1.1.1 节的 CTeX 套装，二是 1.1.1.2 节的 TeX Live。前者是

① Leslie Lamport 博士，微软研究院资深研究员，Dijkstra 奖得主。

Windows 系统下的软件，后者则可以用在各种常见的桌面操作系统上。对 Windows 用户来说，两个发行版并没有显著的优劣之分，你可以任选一个安装使用。

请注意：下面介绍的发行版都是在写作本书时最新的版本。然而当你读到这一段时，软件可能已经更新，界面也可能会有些不同。不过不用担心，安装的过程和使用方法大体上都是一样的。

1.1.1.1 CTeX 套装

CTeX 套装是由中国科学院的吴凌云制作并维护的一个面向中文用户的 Windows 系统下的发行版。这个发行版事实上是对另一个发行版 MiKTeX 的再包装，除了 MiKTeX 主体以外，CTeX 套装增加了 WinEdt 作为主要编辑器，以及 PDF 预览器 SumatraPDF，PostScript 文件预览器 GSview，PostScript 解释器 GhostScript，一些旧的中文支持包和工具（如 CCT 系统）和其他一些有关中文的额外配置（如额外中文字体配置）。

CTeX 套装或许是中文 LaTeX 用户最常用的发行版了。它一直以安装简单、容易上手著称。CTeX 套装有基本版和完全版之分，基本版只包含一些基本安装的 MiKTeX 系统，实际使用中缺少的宏包会在编译时自动下载安装，或由用户自己选择手工安装；而完全版则包含了完整的 MiKTeX 所有组件。对于一般用户，建议使用完全版的 CTeX 套装，这不仅避免了编译时因缺少宏包还要临时下载的问题，而且完全版中包含的诸多文档资料对于用户也很有用。只要从 `http://www.ctex.org/CTeXDownload` 下载对应版本的安装文件，就可以直接进行安装，见图 1.1。

CTeX 套装安装好后，会在“开始”菜单增加一个项目，里面有多个子项目。其中 WinEdt[①]和 MiKTeX 目录下的 TeXworks 是最主要的 LaTeX 编辑器，多数时间我们都将在这两个编辑器之中工作。如果你已经完成安装，现在就可以跳到第 13 页 1.1.2 节开始熟悉使用编辑器了。

“开始”菜单中的其他项目也值得注意。

☞ **FontSetup**

为 CTeX 套装重新安装 CJK 宏包使用的中文字体。CTeX 套装使用 Windows 操作系统所安装的中文字体进行配置，默认支持宋体、黑体、仿宋、楷体、隶书、幼圆 6 种，其中前 4 种是中文版 Windows 预装的字体，后两种是中文版 Office 系统预装的字体。如果系统没有安装对应的字体，则不能进行配置安装。

☞ **Uninstall CTeX**

卸载 CTeX 套装。

① WinEdt 是商业共享软件，用户可以免费试用一个月。

图 1.1 在 Windows 7 中安装 CTeX 套装 2.9

☞ GhostScript

GhostScript 程序是 PostScript 的解释器，许多 TeX 程序都依赖它工作。在命令行下经常还可以使用它转换一些图像格式。

☞ Ghostgum

这个目录里面是 PostScript 文件 .ps 和 .eps 的查看工具 GSview[①]，类似于 TeX Live 中的 PS_View。也可以用它来查看 PDF 文件，不过效果没有 Adobe Reader 好。安装后 .ps 和 .eps 文件会与这个程序关联。

☞ Help

里面是一些由 CTeX 套装所附带的额外的帮助文档。包括一个常见问题集[308]（CTeX FAQ）、《LaTeX 2ε 插图指南》[204]（Graphics）、一个入门文档 lshort[187]

① GSview 是一个发布于 AFPL 协议下的开源免费软件，运行时可能会有注册的弹窗，但软件本身是无须注册的，不影响使用。

（LaTeX Short）、一个 LaTeX 参考手册[23]（LaTeX2e Reference Manual）、《LaTeX Companion》第八章数学公式部分[166]（Mathematics）、一份符号大全[192]（Symbols）和英文的常见文题集[270]（UK TeX FAQ）。

不过遗憾的是，这里提供的部分资料有些陈旧。CTeX 的常见问题集已经几年没有更新，关于中文处理的内容大大落后于现在的实际情况；《LaTeX 2ε 插图指南》也是翻译自几年前的文档，个别内容已经有所变化。本书涵盖了上面内容文档中除符号表外的大多数内容。但无论如何，这里选取的几个文档可以说是日常使用中最实用的一些，还是值得一看的。

☞ **MiKTeX**

MiKTeX 目录下有好几个项目。Previewer 是 MiKTeX 的 DVI 文件预览器，叫做 Yap，类似于 TeX Live 中的 DVIOUT，不过我们很少会用它；TeXworks 是一个小巧好用的编辑器；Help 目录下是 MiKTeX 这个发行版本身的文档；Maintenance 和 Maintenance (Admin) 目录中是 MiKTeX 的对 Windows 当前用户和所有用户的配置工具；而 MiKTeX on the Web 目录中则是 MiKTeX 网站的快捷方式。

这里需要详细说明的是 MiKTeX 的配置工具（Maintenance，如图 1.2 所示）。其中有三项：Package Manager 是 MiKTeX 的包管理工具；Settings 将打开 MiKTeX 的配置选项 MiKTeX Options；而 Update 则是 MiKTeX 的在线升级程序。

☞ **Package Manager**

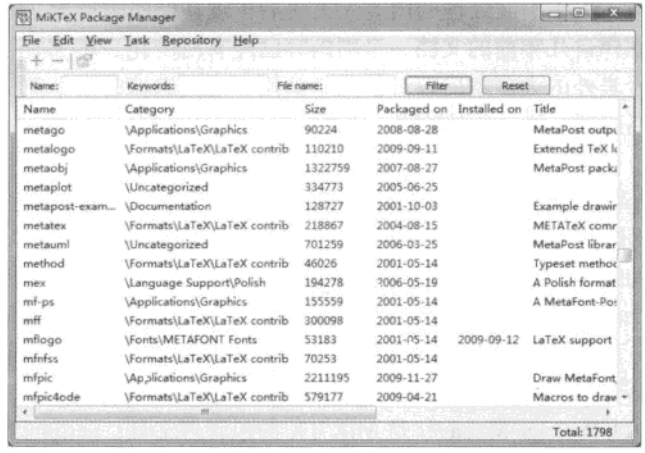
利用包管理器（Package Manager）可以查看和检索 MiKTeX 共有哪些宏包，已经安装了哪些宏包，也可以在线安装和删除各种宏包。所有宏包都有一个简单的介绍和分类，对于喜欢刨根问底，打算了解自己计算机上到底安装了什么东西的人来说，包管理器是一个很好的切入口。如果要安装新宏包，请注意首先选好 MiKTeX 的软件仓库（Repository）并进行同步（Synchronize）。软件仓库通常选取一个 CTAN 网站镜像的 MiKTeX 目录，如 CTeX 网站的镜像。

☞ **Settings**

MiKTeX 选项（MiKTeX Options）里面是一些关于 MiKTeX 发行版的整体配置。

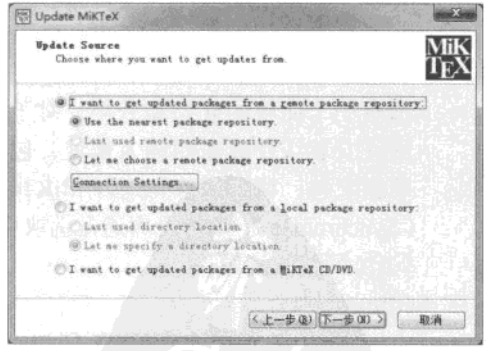
在 General 选项卡中，可以刷新文件名数据库（Refresh FNDB）或更新格式（Update Formats），这通常用在手工安装或更新了宏包和工具的时候；可以设置默认的纸张大小；也可以设置在编译时缺少宏包时是不是自动在线安装（这是 MiKTeX 系统的特色功能）。

在 Roots 选项卡中，可以查看、改变或增加 TeX 的根目录。每个 TeX 根



(a) 包管理器（Package Manager）

(b) 选项设置（Options）



(c) 更新

图 1.2 MiKTeX 配置工具

目录下的目录树结构都是基本相同的，只有按照这种结构放置的文件才能被正确找到并使用。这种树结构一般称为 TDS 结构（TEX Directory Structure，参见 [269]）。一般用户自己编写的文件和一些从第三方得到的宏包、字体、文档等，都放在单独的 TDS 根目录中，在 CTEX 套装中安装目录下的 CTEX 目录就是这样一个 TDS 根目录。

Formats 选项卡用来管理 TEX 系统的编译格式。TEX 和相关的宏语言可能有多种格式（format），INITEX 等程序为每个格式以预编译的方式生成一些二进制格式的信息，并与对应的编译命令（如 pdflatex、mpost 等）结合起来。一般没有必要修改这里的内容。

Language 选项卡可以管理一些语言（不包括中文，主要是西方语言）的支持文件。Packages 选项卡与包管理器的功能类似。可以查看和修改已安装的 MiKTEX 包。

☞ **Update MiKTeX**

这是 MiKTEX 的升级程序，可以用于更新宏包或升级整个 MiKTEX 系统。CTEX 套装的主体就是 MiKTEX，因此可以不重装 CTEX 套装，直接使用 MiKTEX 的升级程序完成除旧式中文支持和编辑器配置外的大部分升级工作。

1.1.1.2　TEX Live

TEX Live 是由 TUG（TEX User Group，TEX 用户组）发布的一个发行版。TEX Live 可以在类 UNIX/Linux、Mac OS X 和 Windows 等不同的操作系统平台下安装使用，并且提供相当可靠的工作环境[①]。TEX Live 可以安装到硬盘上运行，也可以经过便携（portable）方式安装刻录在光盘上直接运行（故有“Live”之称）。

有两种安装 TEX Live 的方式：一是从 TEX Live 光盘进行安装，二是从网络在线安装。不同操作系统下安装设置 TEX Live 的方式基本一样，这里仍以 Windows 操作系统为例进行演示。

一、从光盘安装

TEX Live 一般以安装光盘镜像的方式在互联网上发布。光盘镜像文件可以从 TUG[②] 或 CTAN[③] 网站上下载。可以把镜像文件刻录到 DVD 光盘上使用，也可以直接加载到

① 例如在中文支持方面，旧版本 MiKTEX 的中文字体配置一直有一些错误，所以 CTEX 套装做了进一步配置才正确支持中文；而 TEX Live 就没有这种问题。

② http://www.tug.org/texlive/

③ CTAN 有很多镜像网站，参见 8.3.2 节，国内常用的镜像是 CTEX 网站的 FTP 镜像：ftp://ftp.ctex.org/mirrors/CTAN/systems/texlive/Images/。

虚拟光驱上进行安装。

装入光盘后，安装程序会自动运行（见图 1.3）。如果系统禁用了自动运行，可以手动执行光盘根目录的 install-tl.bat 安装。只要选择好安装的位置，不断单击“下一步”按钮就可以安装 TeX Live 了。

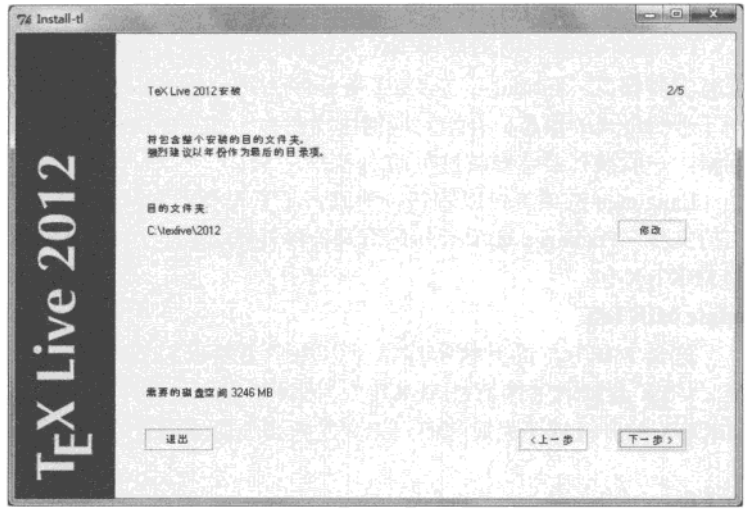


图 1.3 在 Windows 7 下安装 TeX Live 2012

如果对 TeX 系统已经比较熟悉，还可以运行光盘根目录的 install-tl-advanced.bat 进行可定制安装（见图 1.4）。此时，除了安装的位置以外，还可以从预置的几种安装方案中选择某种进行安装，可以选择安装的语言、宏包、工具、文档集合，或进行进一步的安装配置。例如，如果要在服务器上安装后台服务，不想让 TeX 系统占用太大的空间，可以去掉所有的文档和源代码，只选择安装少量必需的宏包和工具，只用原来几分之一的硬盘空间安装一份基本可用的系统。

对于 Linux 系统的用户，还需要设置环境变量并为 XeTeX 配置字体。设置 Linux 环境变量的方式参见 [25, § 3.4]，我建议偷懒的用户在安装时选择在标准路径下创建符号链接的选项，这样就不必设置环境变量了。下面则需要为 XeTeX 配置字体，让操作系统的 fontconfig 库能找到 TeX Live 附带的字体，按下面步骤操作：

1. 进入 TeX Live 的 TEXMFSYSVAR/fonts/conf/ 目录（其中 TEXMFSYSVAR 是一个变量，在定制安装时选定。其默认值为 /usr/local/texlive/2012/texmf-var/），将里面的 texlive-fontconfig.conf 文件改名为 09-texlive.conf，复制到 /etc/fonts/conf.d/ 目录。可以在命令行下（参见 1.1.2.3 节）执行命令：

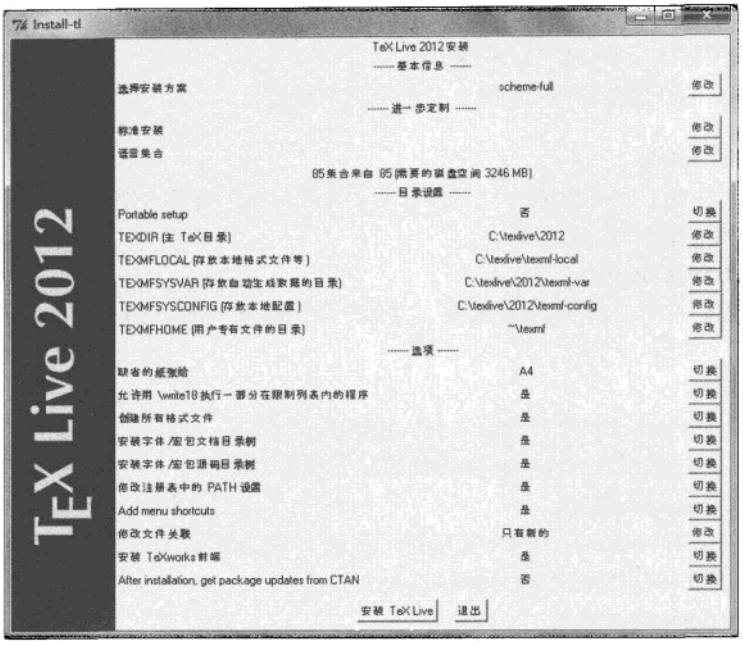


图 1.4　定制安装 TeX Live 2012

```
sudo cp /usr/local/texlive/2012/texmf-var/etc/fonts/texlive-fontconfig.conf \
  /etc/fonts/conf.d/09-texlive.conf
```

2. 刷新 fontconfig 的字体缓存，执行命令：

```
sudo fc-cache -fsv
```

如果一切正常，你会看到屏幕上提示缓存了 TeX Live 一些目录中的字体。

这一配置过程也将使你可以在其他程序中使用 TeX Live 所安装的几百种字体。在类 UNIX 系统下安装 TeX Live 的过程比在 Windows 下略显复杂，希望这个情况在以后能有所改观。

此外，如果希望 pdfTeX、dvipdfmx 等程序能正确找到操作系统中安装的字体，或让 XeTeX 能按字体文件名找到系统字体，还需要设置正确的 OSFONTDIR 变量。TeX Live 会对 Windows 系统自动设置这一变量，对 Linux 等系统也需要手工修改。新建或修改在 TeX Live 安装目录（如 /usr/local/texlive/2012/）下的配置文件 texmf.cnf，在里面修改 OSFONTDIR 变量的值，典型的值如：

```
OSFONTDIR = /usr/share/fonts//;/usr/local/share/fonts//;~/.fonts//
```

程序安装好后，会在桌面上增加 TeX 编辑器 TeXworks 和 PostScript 文件查看工具 PS_View 的图标[①]，现在就可以进行工作了。

TeX Live 的开始菜单相对简单。它包括以下项目：

☞ **TeXworks editor**

这是 TeX Live 预装的一个的 TeX 文件编辑器，简单方便。大部分工作都可以在这个编辑器中完成。

☞ **DVIOUT DVI viewer**

这是一个 DVI 文件预览器，类似于 MiKTeX 中的 Yap。不过我们很少会用到它。

☞ **PS_View**

这是 PostScript 文件查看工具，和桌面上的图标一样。也可以用它来查看 PDF 文件，不过效果没有 Adobe Reader 好。安装后 `.ps` 和 `.eps` 文件会与它关联。

☞ **TeX Live command line**

它打开 Windows 的命令提示符（参见 1.1.2.3 节），并设置好必要的环境变量，可以在其中使用命令行编译处理 TeX 文档。

☞ **TeX Live documentation**

这是一个 HTML 页面的链接，里面是 TeX Live 系统中所有 PDF 或 HTML 格式的文档列表。在首页你可以找到几种语言（包括简体中文）的 TeX Live 发行版文档，以及到近 2000 份各种文档的列表的链接——这份有一公里长的列表多少说明了 TeX Live 是一个多么复杂的系统，以及它在完全安装时为什么占用了这么大的空间。当然，你不需要读完里面的所有文档才能学会使用 LaTeX，不过你会发现在工作中总需要时不时地查看里面的东西（参见 8.3.1 节）。

☞ **TeXdoc GUI**

这是一个常用文档的列表，不过以图形界面的方式把文档分成若干类别，还可以搜索（见图 1.5）。这里面直接列出的宏包数量较少，用来简单浏览可以，但如果要查看更多的内容，最好使用其文件搜索功能或利用命令行 texdoc 工具（参见 8.3.1 节）。

☞ **TeX Live Manager**

这是 TeX Live 管理工具的图形界面（见图 1.6），简称 tlmgr。管理工具也可以在命令行下用 `tlmgr` 命令运行，用 `tlmgr gui` 可以在命令行下打开图形界面。

① TeXworks 和 PS_View 是在 Windows 下安装的附加软件。在其他操作系统如 Linux 中，通常都已经安装或容易从其他途径安装类似的软件，如 Kile 和 Evince。

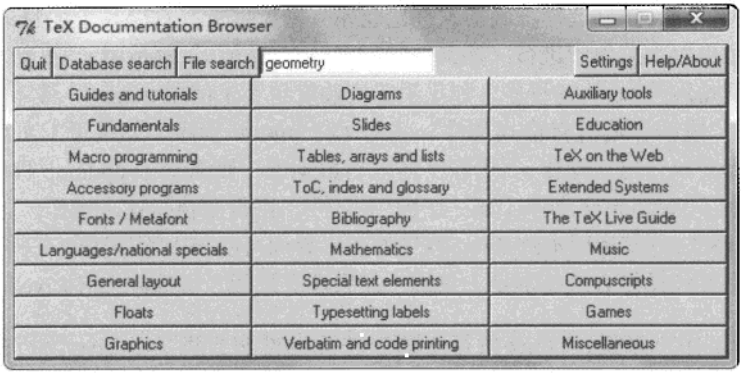


图 1.5　TeXdoc GUI

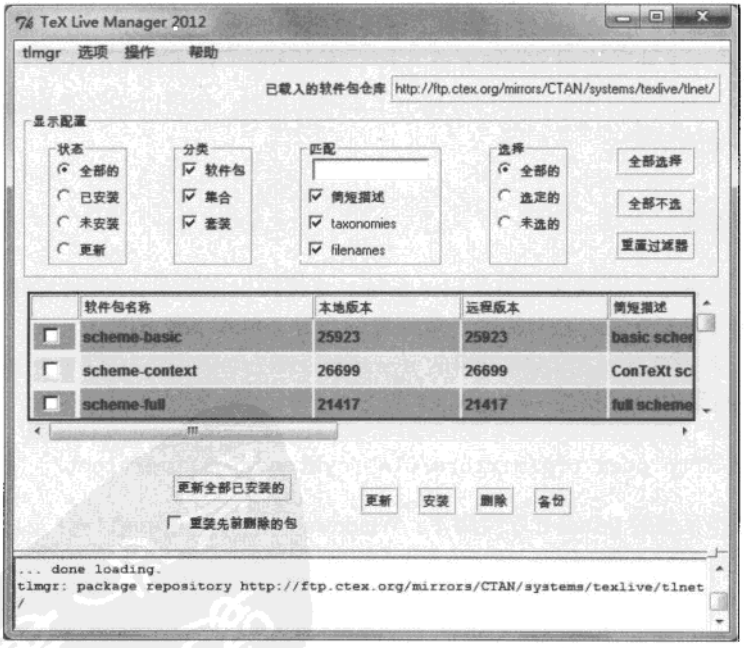


图 1.6　TeX Live Manager（TeX Live 管理工具）

可以用 tlmgr 从网络上或光盘中安装、删除或更新宏包及组件，在开始安装或更新组件前，注意选择正确的软件包仓库（光盘目录或 CTAN 上的目录）并载入。

也可以在菜单中进行一些其他的配置。在“操作”菜单中，“更新文件名数据库”就是运行 texhash 程序，如果手工安装宏包（未使用 tlmgr），就需要执行这个操作；“重新创建所有格式文件”就是运行 fmtutil 程序，如果手工更新了一些程序，需要执行这个操作；“更新字体映射数据库”则对应于 updmap 程序，如果手工安装了 PostScript 字体（如一些商用字体），则需要执行这个操作。

TeX Live 较新版本的 tlmgr 程序的图形界面可能与上面描述的有所不同，但配置的内容和操作方法基本是一致的。如果还有疑问，可参阅 TeX Live 的手册[25]。

对 Linux 用户来说，Linux 发行版的软件源也可能会将 TeX Live 另行打包，以方便通过 Linux 的软件源安装，例如 Ubuntu Linux 的软件源里面就有若干以 texlive 开头的 apt 包。操作系统自带的 TeX Live 往往比较陈旧或被分割简化，特别是难以利用 CTAN 源更新，不过好处是安装起来更容易些。我建议最好还是自己安装[25]。许多 Linux 软件依赖 TeX 系统（如 TeX 编辑器 Kile），在安装时要求先安装操作系统的 texlive 包，与自己安装的 TeX Live 发行冲突。解决这类包依赖问题可以使用虚拟包（dummy package），或在手动下载相关包手在命令行下强制安装，或直接从源代码安装依赖 TeX Live 的软件，不过这方面的内容已经超出了本书的范围，你可以在你的 Linux 发行版的社区请教相关的专家。

二、从网络安装

也可以从网络上在线安装 TeX Live 系统。这样可以保证安装的组件都是最新版本，而且如果进行定制安装，就只需要下载需要的部分，节省下载时间。

网络安装需要先从 CTAN 镜像的 `systems/texlive/tlnet/` 目录下载安装工具。如 CTeX 网站的 CTAN 镜像（参见 8.3.2 节）：

```
http://ftp.ctex.org/mirrors/CTAN/systems/texlive/tlnet/
```

下载对应操作系统的 install-tl 安装脚本：Windows 用户下载 `install-tl.zip`，Linux 和其他类 UNIX 用户下载 `install-tl.tar.gz`。

从下载的压缩包解压得到安装工具后，安装过程与在光盘上安装完全一样。Windows 用户只要双击执行解压出的 `install-tl.bat` 或 `install-tl-advanced.bat` 就会出现图 1.3 或图 1.4 的安装界面了，按提示进行安装，程序会自动从网络上下载所需的文件进行安装。如果网络比较的话，用这种方式安装不比用光盘安装慢多少。

默认情况下，安装程序会自动选择较近的 CTAN 镜像服务器，不过教育网用户可能不方便访问国外的网站，需要在命令行手工指定国内的 CTAN 镜像服务器地址。例如运行如下命令从 CTeX 网站安装 TeX Live：

```
install-tl -repository http://ftp.ctex.org/mirrors/CTAN/systems/texlive/tlnet/
```

1.1.2　编辑器与周边工具

1.1.2.1　编辑器举例——TeXworks

像其他计算机语言一样，LaTeX 使用纯文本描述，因而任何能编辑纯文本的编辑器都能编辑 LaTeX 文档，如 Windows 系统的记事本、写字板，Linux 下的 VI、GEdit。不过，使用专门为 LaTeX 设计或配置的编辑器，进行语法高亮、命令补全、信息提示、文档排版等工作，会使工作方便许多。

LaTeX 代码编辑器有很多，大致可以分为两类：一是主要为 TeX/LaTeX 代码编辑而专门设计的编辑器，二是可以为 TeX/LaTeX 代码编辑配置或安装插件的通用代码编辑器。前者如 WinEdt、TeXworks、TeXMaker、Kile，后者如 Emacs、VIM、Eclipse、SciTE 等。通常前一种编辑器配置和使用更简单些，下面主要以 TeXworks 为例说明编辑器的一些简单配置。其他大部分编辑器在基本功能和设置上都大同小异，不难举一反三。

TeXworks 是 MiKTeX 和 Windows 系统下 TeX Live 预装的编辑器，也是国际 TeX 用户组（TUG）发布并推荐的入门级编辑器。Linux 系统下 TeX Live 没有自动安装 TeXworks 编辑器，你可以到 TeXworks 的网站[①]自己下载安装。

TeXworks 的界面非常简洁（见图 1.7）：它分为两部分，左侧是 TeX 源文件的编辑器窗口，右侧是生成的 PDF 文件的预览窗口。左边的编辑器窗口最上面是标题栏和标准菜单项，接着是工具栏，中间最大的编辑区，最下面则是显示行列号的状态栏。右边的预览窗口把编辑区换成了 PDF 预览区。

除了文本编辑区，编辑器窗口中最常用的是工具栏。工具栏的最左边的按钮是整个编辑器最为重要的“排版”按钮，它调用具体的命令把输入的 TeX 源文件编译为对应的 PDF 结果，刷新右边 PDF 文件的显示。紧靠排版按钮右边的下拉菜单用来选择排版时所使用的命令，通常对应一条单一的命令（如 TeX Live 中的版本或自己单独下载安装的版本），但也可以配置为好几条复合命令（如 CTeX 套装或纯 MiKTeX 中的版本）。通常我们使用最多的排版命令是“XeLaTeX”或“PDFLaTeX”，视具体情况而定。使用排版按钮时，未保存的文档会自动保存。工具栏剩下的按钮则是一系列常见的标准按钮：新建、打开、保存；撤销、重做；剪切、复制、粘贴；查找和替换，不必多说。

① http://code.google.com/p/texworks/

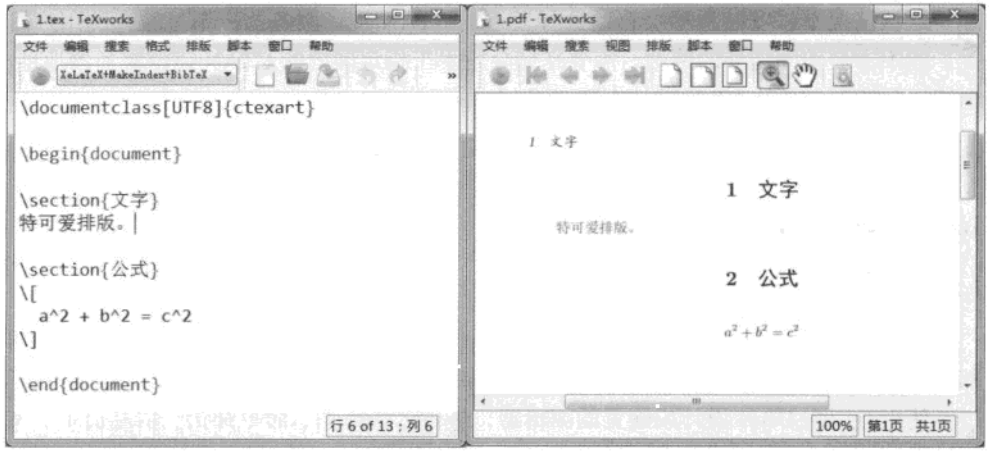


图 1.7 CTEX 套装中的 TeXworks 界面

PDF 预览窗口的工具栏也是一排按钮。最前面的排版按钮与编辑区的功能一样。右面是 4 个向前后翻页的按钮；而后是显示比例的按钮；再后面是放大工具、滚屏工具；最后是 PDF 文本查找工具。

使用 TeXworks 也非常简单：

1. 在编辑区输入 TEX 源文件（如在图 1.7 中的编辑区看到的就是一个简单的例子）；
2. 单击“保存”按钮，给源文件起名并保存在正确的位置（如 `1.tex`）；
3. 在排版按钮旁的下拉菜单中选择“XeLaTeX”，单击排版按钮，查看结果。

编译时在文本编辑区下方的“控制台输出”面板会显示编译进度和信息。如果编译过程有错误或提示输入，程序会停下来等待处理。如果编译结束无误，控制台输出面板会自动关闭，而在预览窗口会显示新的 PDF 的结果。

在文本编辑区或 PDF 预览区用 Ctrl 加鼠标左键单击可以从源文件查找 PDF 文件的对应位置；或反过来从 PDF 文件查找 TEX 源文件的位置。这个功能称为 TEX 文档的正反向查找，对编写长文档特别有用。正反向查找是由 SyncTeX 机制实现的，需要源代码编辑器、PDF 阅读器和 TEX 输出程度的共同参与，一些旧的发行版或程序可能并不支持。

TeXworks 支持自动补全功能。输入一个助记词或命令的一部分，再按 Tab 键，则 TeXworks 会根据配置补全整个命令或是环境；连续按 Tab 键可以切换补全的不同形式。例如，输入 `\doc` 再按 Tab 键，会补全命令 `\documentclass{}`；使用 `beq` 补全则可以得到公式环境：

```
\begin{equation}

\end{equation}•
```

光标移动在环境中央等待输入，再次按下 Ctrl + Tab 组合键则可以跳转到后面的圆点处继续下面内容的输入，而不需要使用方向键。

下面来看 TeXworks 中的一些常见的配置。

刚刚安装的 TeXworks 通常会使用很小的字体，而且可能没有语法高亮等功能，给编辑工具带来许多不便。在 TeXworks 的“格式”菜单中，“字体”项可以用来临时更改显示的字体，而“语法高亮显示”项可以临时控制如何进行语法高亮。如图 1.7 中设置的就是 12 磅的 Consolas 字体。要使字体和语法高亮的设置对所有文档生效，则应该改变 TeXworks 的默认选项。单击 TeXworks “编辑”菜单的最后一项“选项”，将弹出 TeXworks 首选项窗口（见图 1.8）。在“编辑器”选项卡中，可以设置编辑器默认的字体及字号，下面则有语法高亮、自动缩进等格式。

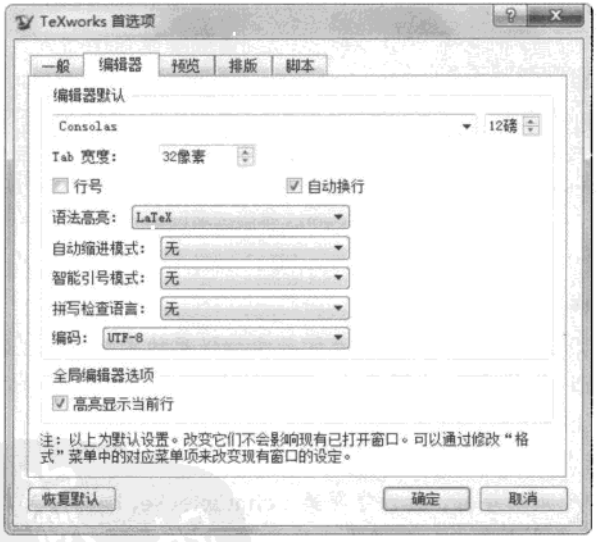
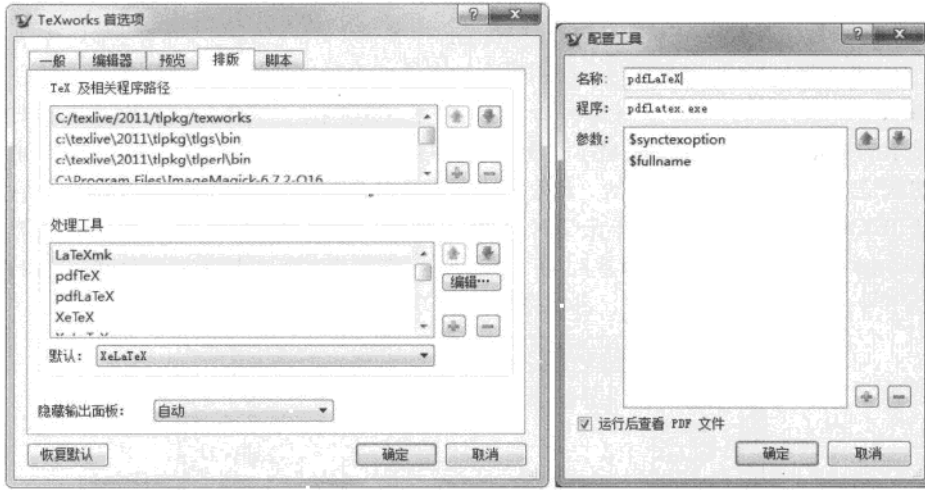


图 1.8　TeXworks 编辑器选项设置

TeXworks 支持多种语言界面和多种文字编码。TeXworks 默认的界面会与操作系统的默认语言（Locale 设置）一致，可以在选项设置窗口的“一般”（General）选项卡中设置程序的界面语言为中文。在“编辑器”选项卡中则有“编码”选项（见图 1.8），一般应该选择 TeXworks 的默认值，即 UTF-8 编码，编辑器保存和打开文件将默认使用此

编码。

TeXworks 选项设置窗口的“排版”选项卡（见图 1.9）可以用来设置 TeXworks 的“排版”按钮所执行的命令。在图1.9(a) 中选择对应的处理工具，单击“编辑…”按钮，就可以在弹出的窗口（见图1.9(b)）中设定对应的命令及参数。参数中使用的变量，可参见 TeXworks 的帮助文档。



(a) TeXworks 排版选项　　(b) TeXworks 工具配置：PDFLaTeX

图 1.9　TeXworks 排版选项设置

文字编码与 Unicode

在使用 TEX 编辑器时，必须注意的是文档保存的文字编码。如果编码使用错误，轻则遇到“乱码”，重则干脆程序运行错误。我们前面所说的“UTF-8”编码，就是现在最为常用的编码之一。

文字在计算机内部都是以数字的形式表示、存储和传输的，人们圈定一些在计算机中使用的字符，称为字符集（character set），一个字符通常就用它在字符集中的序号来表示。不过由于在计算机中数字的二进制表示也有不同的格式，因而相同的字符集也可能有不同的二进制表示方式，也就是字符编码（character encoding）。IBM 公司以前给自己系统中每种编码编一个号，即所谓代码页（code page），后来其他计算机厂商如微软、Oracle 都把自己的字符编码用代码页的方式给出，不过使用的代码页编号都不一样。我们通常见到的代码页，都是微软公司的编号。字符集、字符编码、代码页这些概念，在

很多时候都不加区分，可以混用。

最早的字符编码可以追溯到前计算机时代的电脑码，莫尔斯码（Morse code）和国际二号电码（ITA2）是以前最常用的电报编码。莫尔斯码是变长编码，常用字符短些，不常用的长些；国际二号电码则是定长编码，一共 32 个字符，每个字符用 5 位滴答表示（相当于 5 位二进制数）。汉字电报码是 1869 年发明的，以后略加修改，用 4 位十进制数表示一个汉字，直到今天汉语电报还用的是这套编码（不过现在还用电报的人已经几乎没有了）。

计算机领域早期编码的典范是美国信息交换标准代码（American Standard Code for Information Interchange），也就是大名鼎鼎的 ASCII 编码。ASCII 使用 7 位二进制数表示 128 个符号（包括数字、字母、标点符号和一些控制符）。现代计算机使用 8 位的字节，用一个字节表示一个符号，可以把剩下的一位用作校验，也可以扩展为 256 个符号，表示其他一些西方语言中的字符或图形符号。ASCII 码可算是计算机中使用最为广泛的编码方式，高德纳最早的 TeX 程序使用的就是 8 位 ASCII 编码处理文档。

除了 IBM 等大公司在不断为不同的语言发展不同的字符集和编码，国际标准化组织（ISO）也试图确定标准的 ACSII 扩展方式，于是有了 ISO 8859 标准。ISO 8859 给不同的字母语言使用不同的扩展编码，实际产生了 16 种编码，如 ISO 8859-1（西欧）、ISO 8859-2（中欧）、ISO 8859-5（西里尔字母，如俄语）。为不同的语言使用不同的扩展方式，在当年大约是为了把编码限定在一个字节之内，不过现在看来几乎是在把水搅浑，因为不同的编码实在是太多了。

8 位编码并不足以表示像汉字这样庞大的字符集。中国、日本和韩国这些使用表意文字的国家都纷纷推出自己的字符集。中国于 1980 年推出了国家标准 GB 2312，包括 7445 个字符，其中有 6763 个汉字；GB 2312 字符集通常使用 EUC-CN 编码，也常被称为 GB 2312 编码。这些字符用来排版当然不够用（方正公司就为自己的产品研发了专用的 748 编码），1993 年发布了与 Unicode 1.1 相当的 GB 13000。GB 13000 没有得到推广，实际应用的却是与 GB 13000 字符数量相当的 GBK（K 是扩展的意思），有 21 886 个字符。GBK 编码不是正式标准，然而应用十分广泛，Windows 95 之后的微软操作系统都支持 GBK 编码。2000 年 3 月又发布了标准 GB 18030-2000，Windows XP 就支持这一字符集；2005 年 11 月的 GB 18030-2005 则成为强制执行的标准，于是从 Windows Vista 之后的版本都是基于 GB 18030 编码的系统。GB 18030 两个版本的字符集实际来自 Unicode 3.0 和 4.1，前者包括 27 533 个字符，后

者包括 76 556 个字符。GB 18030 到 GBK 到 GB 2312 到 ASCII，编码都是向下兼容的。

ISO 于 1990 年推出了通用字符集（Universal Character Set，UCS）标准 ISO 10646，包括 UCS-2 和 UCS-4 两种长度的编码；1991 年一个叫做通用码协会（Unicode Consortium）的组织发布了 Unicode 1.0 标准。两个组织都打算把全世界所有文字的符号都用同一套字符集和编码统一起来。后来两个组织协作起来，从 Unicode 2.0 起，Unicode 就符合 ISO 10646 了。2012 年 1 月发布的 Unicode 6.1 已经定义了 110 181 个字符，包括世界上 100 种文字，而且还在不断修订扩充之中。Unicode 已渐次成为字符编码的新方向，包括 GB 18030 也可以看做是 Unicode 字符集的一种编码格式。除了 UCS-2 和 UCS-4，Unicode 标准还提出了多种编码形式，称为 Unicode 交换格式（Unicode Transformation Format, UTF）：主要包括变长的 UTF-8、UTF-16 和定长的 UTF-32 编码。UTF-8 编码与 ASCII 编码向下兼容，因而最为常用。

TeX 系统原本只支持 ASCII 编码。但只要设置好超过 127 的数字对应的符号，所有扩展 ASCII 编码都能正确排版，如 ISO 8859 的各种标准。汉字编码 GB 2312、GBK 和 UTF-8 都是兼容 ASCII 的多字节编码，因而在 LaTeX 中通过 CJK 宏包也可以通过特殊的方式，把多个字符对应到一个汉字上，支持中文的排版。

CJK 宏包这种支持多字节编码的方法是一种黑客手段。后来 TeX 的新实现 XeTeX 和 LuaTeX 都直接支持 UTF-8 编码，新的中文排版方式也自 2007 年起随着这两种新排版引擎应运而生。LuaLaTeX 的本地化支持目前还暂处在起步阶段，本书将着重介绍基于 XeTeX 的方式。

1.1.2.2 PDF 阅读器

CTeX 套装和 TeX Live 都已经预装了 DVI 文件和 PostScript 文件的阅读器。然而却没有安装最重要的 PDF 格式阅读器。现在使用 TeX 系统基本上最终都将输出 PDF 格式的结果，而且 TeX 系统中的大部分文档也都是 PDF 格式的，因此一个 PDF 阅读器是不可或缺的。

PostScript 阅读器 GSview 和 PS_View 也都可以当做 PDF 阅读器来使用，不过效果不是很好。我们使用 PDF 阅读器主要有两个目的，一是在编辑文档过程中随时查看编译的效果，这对于编辑复杂公式、插图以及幻灯片来说都非常重要；二是为了阅读 PDF 格式的文档资料，或查看自己编写文档的最终效果。这两个目的各有不同的要求，前者

要求快捷方便，最好还能在 PDF 的效果与 TeX 源文件之间方便地切换检索；后者则要求显示准确美观、功能全面。

要满足第一个要求，编辑器 TeXworks 内置的显示功能最为方便（见图 1.7）。TeXworks 把源代码和 PDF 结果左右排开，对照显示，TeX 代码编译后右边的 PDF 文件就会更新。而且可以用 Ctrl 加鼠标左键单击进行从源文件到 PDF 文件或从 PDF 文件到源文件的正反向查找。

CTeX 套装还预装了 SumatraPDF 阅读器用来在 WinEdt 中预览 TeX 文件的编译结果。SumatraPDF 是一个很小的 PDF 阅读器，同样支持用鼠标双击进行 PDF 的反向搜索。

CTeX 套装和 TeX Live 都没有预装能满足第二个要求的阅读器。我们建议使用 Adobe Reader 最新的版本（在 Linux 系统中通常称为 acroread）。Adobe Reader 是官方免费提供的 PDF 格式的阅读器，通常它的显示效果最好，支持全面的 PDF 特性（如 JavaScript 脚本、动画、3D 对象等，而且这些很可能在 LaTeX 制作的幻灯中用到）。其他一些常见的 PDF 阅读器，如 Foxit Reader，则可能在一些功能上有所欠缺。

如果你还没有安装 Adobe Reader，可以直接从 Adobe 的官方网站，或几乎任何软件下载站点得到并安装。不必抱怨它占用上百 MB 的安装空间：除了一些高级功能的插件，Adobe Reader 还提供了许多种高质量的 OpenType 中西文字体，这些都可以在你未来的排版中用到。

PS 格式和 PDF 格式

PS 是 PostScript 的简称。PostScript 是 Adobe 公司于 1984 年发布的一种页面描述语言，自 1985 年苹果公司的 LaserWriter 打印机开始，此后的很多高档激光打印机都带有 PostScript 语言的解释器，可以直接打印 PostScript 语言描述的文档。PostScript 遂逐渐成为电子与桌面出版的标准格式，并一直延伸到整个出版业，风靡一时。就连国内的北大方正公司的排版系统也是以变形的 PostScript 格式输出，并沿用至今。

“PostScript”这个名字多少体现了这个语言的特点：它是一种基于后缀表达式和栈操作的解释型计算机语言[4]。如表达式 1 + 2 就被写成

```
1 2 add
```

而

```
0 0 moveto
100 100 lineto
```

则是在描述从坐标 (0,0) 到 (100,100) 的直线路径。使用这种后缀语法原本是为了方便计算机芯片高效解释 PostScript 这种复杂的语言，大部分 PostScript 也都是由其他计算机程序自动生成的。不过，富于经验的老手可以就凭着这种看起来有些怪异的语法直接画出图来，这种技艺也一直延伸到 5.5.2 节将要讲到的 PSTricks 宏包。

PostScript 拥有强大的图形能力，可以用一段 PostScript 语言的代码表示很复杂的图形。然而作为一门完整的计算机语言 PostScript 过于复杂，因而有了所谓封装的 PostScript（Encapsulated PostScript）格式，即 EPS 格式。EPS 格式的文件也是一段 PostScript 代码，但只能表示一页，而且加上了诸多限制，成为一种专门用来存储可以嵌入其他应用中的图形格式。TeX 的许多输出引擎都支持这种图形格式，我们将在 5.2 节回到这个话题。

由于在电子出版领域的地位，PostScript 一度成为 TeX 系统最重要的输出格式，至今在网络上仍能见到大量 TeX 产生的 PostScript 格式的书籍和文章，一些期刊也一直要求以能生成 PostScript 格式的 TeX 文档投稿。然而随着新一代廉价的喷墨打印机的出现，需要复杂解释芯片的 PostScript 打印机逐渐式微；而网络技术的发展进一步催生了电子文档交换的需求，PDF——Portable Document Format（可移植文档格式）便应运而生。PDF 由 Adobe 公司于 1993 年发布，它是 Adobe Acrobat 系列产品的原生文件格式，并随着文件格式的公开和阅读器 Adobe Reader 免费的发放，迅速风靡起来。

PDF 与 PostScript 使用相同的 Adobe 图形模型，可以得到与 PostScript 相同的输出效果，而在程序语言方面则比 PostScript 大为削减，并增强文档格式结构化，可以更迅速地由计算机处理。尽管 PDF 最初只是 PostScript 削减功能适应电子文档处理的结果，但 PDF 转而在电子文档功能如交互式表单、多媒体嵌入等方面大下功夫，并不断进行各方面的扩充，最终成为一种比 PostScript 还复杂的格式（描述 PDF 1.7 的手册 [5] 比描述 PostScript 1.3 的文档 [4] 要厚得多）。PDF 也继 PostScript 之后成为现在新一代的电子出版业的事实标准。

现代的 TeX 输出引擎几乎都以 PDF 为输出格式。同时 PDF 格式也可以像 EPS 格式一样作为图形格式被 TeX 和其他软件使用。现在能够输出 PDF 图形的软件和支持嵌入 PDF 图形的 TeX 引擎比 EPS 格式的还要多些，PDF 也成为现在 TeX 系统中最重要的图形格式。

1.1.2.3　命令行工具

一、命令行

尽管大多数常用编译操作可以在编辑器中完成，CTeX 和 TeX Live 也都给出了一些图形界面的配置工具，但 TeX 发行版的主体仍然是命令行下的程序。不了解命令行，就难以了解 TeX 的处理流程，也不能很好地使用诸如 Makeindex 这样的基本 LaTeX 工具。因此，有必要对命令行和一些命令行工具的使用作一了解。

命令行是以文字方式与计算机交互的方式，与图形方式相对。在 Windows 系统[①]中，命令行通常由命令解释程序 cmd.exe 处理；在 Linux 及其他类 UNIX 操作系统中，命令解释程序称为壳（Shell），最常见的壳是 Bash。在命令行下可以执行一些基本的文件操作，也可以运行其他程序，批处理脚本也是由命令行解释程序执行的。Linux 中 shell 的使用一般远比在 Windows 中频繁，因此这里仍以 Windows 为例。

Windows 中默认的命令行解释程序 cmd.exe 可以从“开始”菜单的“附件”项中找到，叫做“命令提示符”[②]（见图 1.10）；也可以直接运行（利用“开始”菜单的“运行”项或组合键 Win + R）cmd.exe 进入。如果使用频繁，可以在桌面或快速启动栏建立快捷方式，或设定快捷键随时使用。

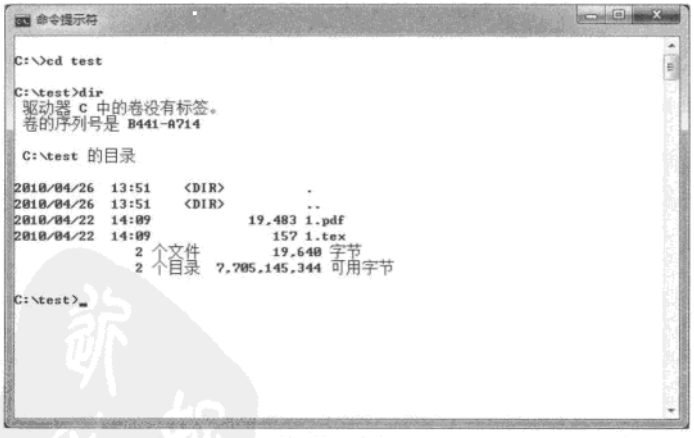


图 1.10　Windows 命令提示符（默认是黑底白字，这里为显示清晰改为白底黑字）

使用 TeX 经常需要在特定文件所在的目录（文件夹）进行命令行操作，可以把进入命令行的操作添加到 Windows 资源管理器鼠标右键菜单中。这可以通过修改

① 指 Windows NT 及 Windows 2000 以后的版本，包括 Windows XP、Windows Vista、Windows 7 等。

② 从 DOS 系统开始使用计算机的人时常称 Windows 的命令解释程序为 DOS 窗口，这对于 Windows 9x 及更早的 Windows 是对的，但以后版本的 Windows 则不再包括 DOS。新的命令行解释器只是在外观和命令上像是 DOS。

Windows 注册表来完成。将下面的内容保存到一个后缀为 .reg 的文件中，双击导入注册表，或手工按其中的内容建立对应的注册表项，可以为当前用户添加“进入命令行”的右键菜单：

```
Windows Registry Editor Version 5.00
[HKEY_CURRENT_USER\Software\Classes\*\shell\进入命令行\command]
@="cmd"
[HKEY_CURRENT_USER\Software\Classes\Folder\shell\进入命令行\command]
@="cmd /k cd \"%1\""
```

打开命令行窗口后，会显示命令提示符。默认的命令提示符由当前盘符、目录（即文件夹）和一个大于号 > 组成，如

```
C:\>
```

表示当前目录是 C 盘的根目录 >。后面的光标等待输入命令，Windows 命令行命令和文件名不区分大小写，输入一行命令后按回车键即开始执行。

使用最频繁的命令是列文件列表命令 dir（directory 的缩写），直接输入 dir 后按回车键就会显示当前目录下所有文件的详细列表。dir 命令后可以指定要列出的盘符、目录和文件名，如

```
dir C:\WINDOWS
```

将列出 C 盘 WINDOWS 目录下的所有文件。

目录和文件名可以使用 ? 和 * 作为通配符。? 可以代替任意一个字符，* 可以代替任意多个字符。例如，命令

```
dir book*.tex
```

将列出所有以 book 开头，后缀为 .tex 的文件。目录和文件名可以用 Tab 键自动补全，如输入 book 后，连按 Tab 键将交替地补全当前目录所有以 book 开头的文件。有两个特殊的目录名 . 和 ..，分别用来表示当前目录（可省略不写）和当前目录的上一层目录。

cd 命令（或 chdir，change directory 的缩写）用来改变当前所在的目录。如

```
cd pictures
```

将进入当前目录下的 pictures 目录（如果有的话），而从 C 盘用命令

```
cd \WINDOWS\Fonts
```

则进入 Windows 的字体目录。注意更换盘符不能用 cd 命令，而要单独使用 (盘符) 后加符号：进入，如输入

```
D:
cd \test
```

将进入 D 盘根目录下的 test 目录。

把多个命令行写到一个文件里面，保存为后缀为 .bat 或 .cmd 的文件，就得到一个批处理文件（又称批处理脚本）。在命令行下可以像运行其他程序一样调用批处理文件，也可以在图形界面鼠标点击批处理文件执行。批处理可以一次完成多项任务，如完成多道工序的 TeX 源文件编译工作。批处理还提供命令行参数、变量定义、条件判断等简单的编程功能，详细内容可参见微软的联机帮助。

二、GhostScript

GhostScript 是一种 PostScript 的解释器，它的主体也是命令行工具。Windows 版本的 MiKTeX 和 TeX Live 都附带安装了一个简化版本的 GhostScript，CTeX 套装则另行安装了一份完全的 GhostScript。

MiKTeX 附带的 GhostScript 程序名为 mgs，TeX Live 中的程序则名为 rungs。一般无论是 Linux 用户还是 Windows 用户，最好还是单独下载安装完全版本的 GhostScript，因为一些 LaTeX 输出引擎有时仍会调用它，Linux 用户可以使用系统软件源中的版本，Windows 用户可以在

```
http://code.google.com/p/ghostscript/downloads/list
```

下载安装包。

可以用 GhostScript 查看 PostScript 或 PDF 格式的文件，PostScript 文件查看器 GSview 和 PS_View 都是调用 GhostScript 工作的。GhostScript 更常用的功能则是进行文档格式转换，做 PS、PDF 格式的相互转换，或把它们转换为点阵图片格式，如 PDF 输出引擎 DVIPDFMx 就会在处理 EPS 图片时自动调用 GhostScript。

GhostScript 为一些常用的转换提供简单的命令行，最常见的是从 .ps 到 .pdf 文件的转换，可以用 ps2pdf 命令完成，如：

```
ps2pdf foo.ps
```

命令会将 foo.ps 文件转换为 foo.pdf 文件。类似的命令还包括 pdf2ps 和 eps2eps 等。

GSview 程序为 GhostScript 的格式转换功能提供了一些图形界面的接口，在 File 菜单下的“PS to EPS”项目，就是用来把 .ps 文件转换为 .eps 文件的；而 File 菜单下的“Convert...”项目（见图 1.11），则可以完成 GhostScript 支持的各种转换。

图 1.11 使用 GSview 转换文件格式

所有显示和转换的工作都可以通过 GhostScript 的主程序完成。GhostScript 的主程序是一个命令行程序，在 Windows 下名叫 gswin32c.exe（64 位的版本下名字为 gswin64c.exe），在 Linux 等系统下通常就叫做 gs，也可以使用 TeX Live 的 rungs 或 MiKTeX 的 mgs，这里统一用 GS 表示。一个调用 GS 的命令通常带有许多命令行参数，以完成各种复杂的操作，例如，

```
GS -dBATCH foo.eps
```

将使 GhostScript 在屏幕上显示 foo.eps 的内容并退出；下面的命令（第一行末的 \ 并不存在，只表示延续到下一行）：

```
GS -q -sDEVICE=png256 -dEPSCrop -r128 -dGraphicsAlphaBits=4 \
   -dTextAlphaBits=4 -o bar.png foo.eps
```

则把 foo.eps 转换为 256 色 PNG 图像 bar.png，使用 128 dpi 的分辨率，剪裁到适当大小，并对文字和图像做边缘抗锯齿处理。关于 GhostScript 的详细命令行参数可以参考 GhostScript 的联机文档。

三、ImageMagick

ImageMagick 是一款优秀的基于命令行的位图处理软件，可以在超过 100 种不同的图像格式之间转换，或对图像进行各种变换和处理。熟悉平面设计的人可以把它看做是 Adobe Photoshop 这类软件的一些图像滤镜的命令行版本。ImageMagick 并不直接与 TeX 相关，但 TeX 用户经常用它来做一些有关图形转换的工作，个别与 TeX 相关的软件（如 `Asymptote`）也会调用 ImageMagick。

ImageMagick 是自由软件，可以在

```
http://www.imagemagick.org/
```

下载安装。Windows 用户一般下载标注为“Win32 dynamic at 16 bits-per-pixel”的版本安装，Linux 等类 UNIX 操作系统可以下载二进制包或源代码编译，也可以直接使用系统软件源中的版本。

在 Windows 下安装 ImageMagick 会在开始菜单项中找到它的帮助文档和 ImageMagick 中唯一的图形界面程序 IMDisplay。但注意 IMDisplay 只是一个图片查看器，并不具备任何 ImageMagick 的图像处理功能，我们主要还是在命令行下使用 ImageMagick。

ImageMagick 是个很复杂的软件，包括 10 多个不同的命令行工具，具有 200 多种不同的命令行参数。这里只介绍 ImageMagick 最基本的图像类型转换功能，也是最常用的功能，更详细的功能可以参见 ImageMagick 的联机帮助文件。

命令 `convert` 用于图像的转换，即把一幅图像转换为另一幅图像，尽管功能复杂，但基本的使用方法是十分简明的，如：

```
convert foo.bmp bar.png
```

是将 BMP 格式的图像 `foo.bmp` 转换为 PNG 格式的图像 `bar.png`，类似地，

```
convert foo.eps bar.pdf
```

则是把 EPS 格式的图片 `foo.eps` 转换为 JPG 格式的图片 `bar.pdf`。不过 ImageMagick 在处理涉及 PostScript 和 PDF 格式的图片时，内部实际还是调用 GhostScript 来完成的，这时可以把它看做是 GhostScript 命令的一种方便的变形。

获取命令行帮助

熟练的用户使用命令行完成一些工作比使用图形界面的软件更高效快捷。不过对于刚接触命令行不久的人来说，命令行的最大问题就是记不住命令的用法，因此应该了解如何在命令行下获取帮助信息。

专门的联机文档或在线文档是比较通用的帮助形式，如在 Windows 下，由“开始”菜单进入联机帮助，以“命令行”、“cmd”等关键字搜索，很容易就能得到详尽的命令行帮助。有时帮助文档则以专门的文件存储，如 GhostScript 和 ImageMagick 在 Windows 下都提供网页形式的帮助文档，可以在“开始”菜单找到。也有许多程序提供 CHM、PDF 等格式的文档。

另一种方式是直接在命令行下得到帮助，这通常是通过特殊的命令行参数得到的。通常，Windows 命令行的基本命令可以在命令后加 /? 参数获得帮助。如输入

```
dir /?
```

将会在屏幕上得到 dir 命令的帮助信息：

```
显示目录中的文件和子目录列表。

DIR [drive:][path][filename] [/A[[:]attributes]] [/B] [/C] [/D] [/L] [/N]
  [/O[[:]sortorder]] [/P] [/Q] [/S] [/T[[:]timefield]] [/W] [/X] [/4]

  [drive:][path][filename]
              指定要列出的驱动器、目录和/或文件。
.....................
```

来自类 UNIX 系统的程序命令行选项以 - 开头，命令行帮助通常可在命令后加 --help 得到。如输入

```
convert --help
```

将会在屏幕上得到 ImageMagick 的所有命令行选项的说明（会非常长）。

类 UNIX 系统在命令行下有一个 man 命令，可以用来调出文档。如用

```
man ls
```

将在命令行中直接调出 ls 命令（相当于 Windows 中的 dir 命令）的详细帮助，类似的文档程序还有 info。在 Windows 中，用 help 命令可以达到类似的效果，不过效果和使用 /? 选项相同。TeX 系统继承了 UNIX 中 man 的用法，也提供了一个 texdoc 程序，可以在命令行下调出 TeX 宏包、工具和字体等的文档，参见 8.3 节。

1.1.3 “Happy TeXing”与“特可爱排版”

在做完所有的准备工作以后，我们来一起运行一个简单的例子，测试整个系统。

首先，打开你的 TeX 编辑器，如 TeXworks，新建一个文件，输入下面的内容（不包括行号）：

```
1 \documentclass{article}
2
3 \begin{document}
4 This is my first document.
5
6 Happy \TeX ing!
7 \end{document}
```

1-1-1

新建一个测试用的目录，将刚刚输入的文件保存到这个目录里面，选择 PDFLaTeX 或 XeLaTeX 的命令，点击编辑器上的对应的排版按钮（见图 1.7）。如果一切顺利，将在 PDF 预览窗口看到编译的结果，内容类似下面的样子：

This is my first document.
Happy TeXing!

如果你使用的不是 TeXworks，而是 WinEdt 这类不带 PDF 预览功能的编辑器，点击排版按钮后可能会弹出一个 PDF 阅读器的窗口，显示出上面的页面，也可能你需要手工再点击打开 PDF 文件的按钮来查看排版的结果。

这个文件中有一些以反斜杠 \ 开头的语句，大多没有出现在最终的 PDF 文档中。虽然我们以前并没有接触过这些语句，不过不难猜测其涵义：\documentclass{article} 声明了文档的类型是一篇文章；\begin{document} 和 \end{document} 语句标识出正文的范围；至于正文中的 \TeX，看结果就知道它表示“TeX”这个高低不平的符号。这就是我们的第一个例子，看起来很简单。

可是，如果你马上兴致勃勃地把里面的内容换成汉字，再点击按钮看结果时，就会发现汉字并没有出现在 PDF 文档中，只有英文字符出现。这是因为 TeX 原本是面向西文写作的，默认并没有加载中文字体。

通过更换文档类型，下面这个稍稍复杂的例子可以正确显示出中文（这正是图 1.7 和图 1.12 中的例子）：

```
1 \documentclass[UTF8]{ctexart}
2 \begin{document}
```

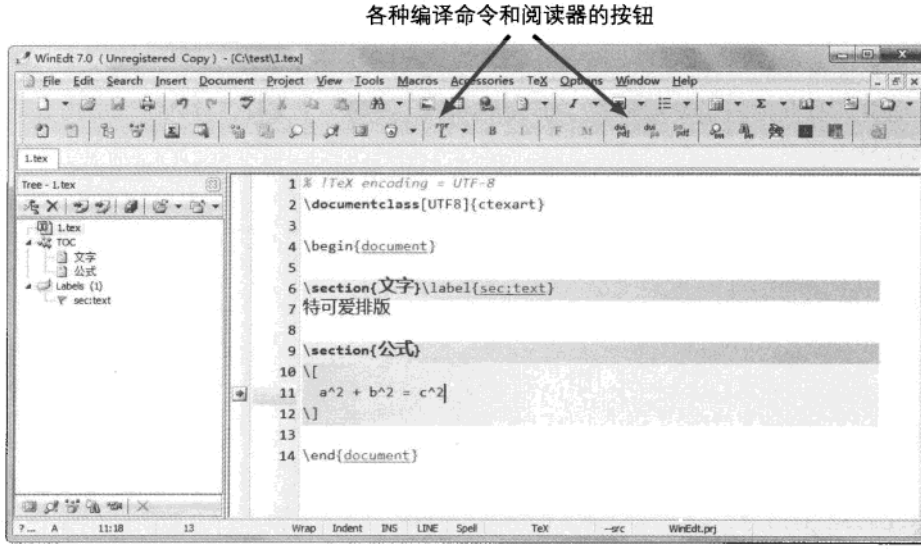


图 1.12 CTEX 套装 2.9 中的 WinEdt 7 编辑器。WinEdt 的界面比 TeXworks 要复杂得多，有各种命令的编译按钮和许多额外的工具

```
3 \section{文字}
4 特可爱排版。
5 \section{数学}
6 \[
7   a^2 + b^2 = c^2
8 \]
9 \end{document}
```

1-1-2

注意文档保存时要使用 UTF-8 编码，这是 TeXworks 的默认值，但 WinEdt 可能需要在保存时选择[①]，编译后的结果如图 1.13 所示。

这段代码也不难看懂[②]：文档类换成了 ctexart，即中文 TEX 的文章（article）类型，这个文档类使得中文可以正确地显示[③]；在 ctexart 前面的 [UTF8] 是使用这个文档类的选项，表明了中文所使用的编码；两个 \section 命令各自生成了一节的标题；

① 不同版本的 WinEdt 设置不同，WinEdt 7 开始以 UTF-8 为默认编码，但 CTEX 套装可能仍然配置为本地的 GBK 编码。不同版本在使用时需要仔细查看。

② 唔……也许使你最费解的是“特可爱”。这其实是 TEX 的谐音，也有双关的意味。

③ ctexart 文档类默认使用中文版 Windows 所预装的字体。对于 Linux 等其他系统的命令，可能需要不同的设定才能正确显示汉字，参见 2.4.1 节。

1　文字

特可爱排版。

2　数学

$$a^2 + b^2 = c^2$$

图 1.13　最简单的 LaTeX 文档

唯一不大直观的是由 \[和 \] 包裹起来的数学公式，不过 LaTeX 数学公式的能力太出名了，你一定早听说过它了。

上面两个简单的例子给了我们一个 LaTeX 的直观印象，而且正确运行它们或许能增强你学习 LaTeX 的信心。粗略地看，LaTeX 是一种标记式排版语言[①]，有相关背景的人大概会觉得 LaTeX 的代码与 HTML 代码有很多相似之处，整个文档通过一些标记（命令）分成结构化的部分。LaTeX 的命令以反斜线 \ 开头，命令一般用英文单词命名，有的可以带参数。通过一个程序的处理，我们称为编译过程，LaTeX 源代码就能生成对应的输出结果，通常就是一个 PDF 文档。

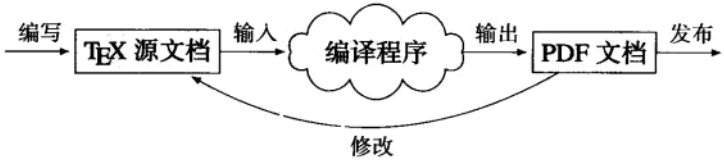


图 1.14　LaTeX 文档的写作流程

LaTeX 文档的写作流程见图 1.14。通常这个过程都是自动化完成的，编写 TEX 源文档通常是在专门的 TEX 编辑器中进行，例如 TeXworks 和 WinEdt，而后按下一个按钮，源文件就被送给 TEX 的编译程序进行处理，输出 PDF 文件，此时编辑器调用 PDF 阅读器查看结果。如果出了问题，需要根据输出的结果或程序的错误信息修改源文件或编译方式。

练 习

① 严格来说，TEX/LaTeX 并不是 HTML、XML 那样的标记语言，而是主要基于字符串代换的宏语言。不过 TEX，尤其是 LaTeX 的格式与标记语言的用法很像，在很多情况下也可以把它看做标记语言。

1.1 不使用专用编辑器，只用普通的文本编辑器录入上面的例子，然后在命令行下编译 TeX 文档，查看运行的结果。

编译程序

TeXworks、WinEdt 等编辑器里面给出了许多编译程序的按钮，往往让人有目不暇接的感觉，如果你留心来自各种书籍、文档和网络的资料，上面介绍的编译方法五花八门。如果是使用命令行编译，则输入起来更觉头疼，那么，这些不同的编译程序做了什么？该如何选择和使用呢？

高德纳设计的 TeX 原本只是一个相对简单的程序，命令 tex 就会调用最基本的 TeX 程序[①]。它使用高德纳在 [126] 中描述的一个相对简单的格式 Plain TeX 进行排版。tex 读入 TeX 源文档，输出一种称为“设备无关”的（DeVice Independent）格式，即 DVI 文件，DVI 文件在过去是 TeX 的标准输出格式，但功能比较受限，不能嵌入字体和图形等，在 PostScript 和 PDF 流行之后，DVI 格式就主要成为一种到 PS 或 PDF 格式的中间格式了。

程序 Dvips 将 DVI 文件转换为 PostScript 文件，可以直接拿到支持 PostScript 的打印机上打印，也可以通过 GhostScript 的 ps2pdf 或 Adobe Acrobat 提供的 Distiller 等程序再从 PostScript 文件转换为 PDF 文件。PDF 流行以后又有了能把 DVI 文件直接转换为 PDF 文件的 dvipdf 程序，之后出现了更为先进的 dvipdfm 和 dvipdfmx，可以支持更丰富的 PDF 功能和东亚字体等，现在新的发行版中主要还在使用的是 dvipdfmx（常写做 DVIPDFMx）。这类把 DVI 文件转换为其他实用格式的程序常被称为 TeX 输出的驱动（driver）。

除了最初的 TeX 程序，后来有许多人对 TeX 进行了扩展。先是有了 ε-TeX，后来在 ε-TeX 的基础上，Hàn Thế Thành 设计了能直接输出 PDF 格式的 pdfTeX。不过 pdfTeX 程序也保留了输出 DVI 格式的能力，因而现在很多输出 DVI 格式的命令内部也是使用的 pdfTeX 程序。pdfTeX 的后继是 LuaTeX，这是一种把脚本语言 Lua 和 TeX 结合起来的程序。ε-TeX 的另一发展则是 XeTeX（纯文本写成 XeTeX）程序，它将中间层 DVI 格式扩充为更强大的 xdv 格式，一般会直接调用 dvipdfmx 的后继 xdvipdfmx，直接输出 PDF 格式。LuaTeX 和 XeTeX 都将原来 TeX 支持的 ACSII 编码改为 UTF-8 编码，并且可以更方便地使用各种字体。TeX 程序连同这些扩展常被称为不同的 TeX 引擎（engine）。

有关 TeX 的各种软件及其关系的更详细的说明，可参见 Trautmann [266]。

① 有的发行版是使用 Plain TeX 格式并输出 DVI 文件的 pdfTeX 程序。

不同的引擎都可以编译 Plain TeX、LaTeX 或是 ConTeXt[①] 等不同格式的文档，不同的组合就使用不同的命令，见表 1.1，我们主要关注 pdfTeX 和 XeTeX 引擎使用 LaTeX 格式的命令。

表 1.1　各类引擎和格式使用的 TeX 命令

命令 / 格式 / 引擎	Plain TeX	LaTeX	ConTeXt	
TeX/ε-TeX	tex / etex			输出 DVI
pdfTeX	tex	**latex**		输出 DVI
	pdftex	**pdflatex**	texexec	输出 PDF
XeTeX	xetex	**xelatex**	特殊参数	输出 PDF
LuaTeX	luatex	lualatex	context	输出 PDF

使用 LaTeX 格式的排版得到 PDF 文件的方式也有好几种（见图 1.15）。其中使用 latex + dvips 的方式最为古老，不便于中文文档的排版，现在一些西文期刊仍然要求这样排版。其他几种方式都能较好地进行中文排版。用 latex 和 pdflatex 命令排版在处理中文时都使用 CJK 宏包的机制，而 xelatex 则使用新的 xeCJK 宏包的机制。功能上 xelatex 最为方便，尤其是在处理中文时；而用 pdflatex 编译，一些宏包的兼容性更好一些。不过本书的大部分内容并不限于图 1.15 中的任何一种模式，只是在处理中文时，将主要讨论 xelatex。

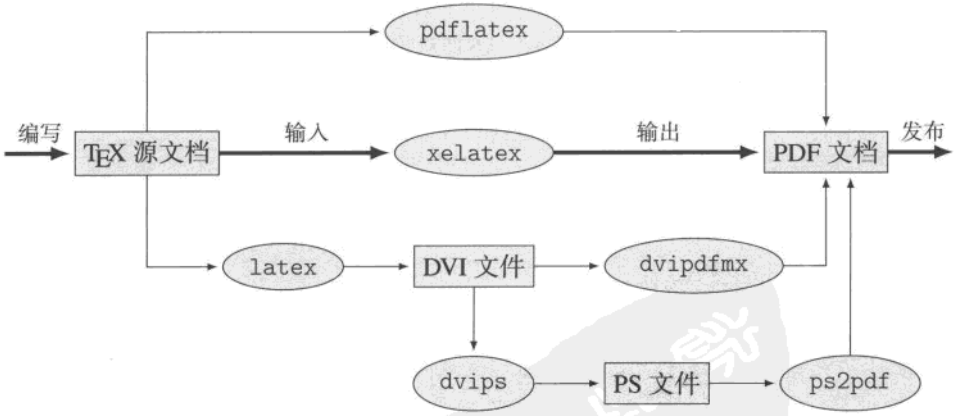


图 1.15　使用各种引擎编译 LaTeX 文档的简要流程。在处理中文时，我们以 xelatex 为主

① 不同于 LaTeX，ConTeXt 是一种把 TeX 和脚本语言紧密结合的格式，LuaTeX 程序就主要用于 ConTeXt 格式。

1.2 从一个例子说起

这一节将研究一个相对实际的例子。在这个简化的例子中，我们将看到在真正的写作排版工作中时常遇到的一些模式、问题的解决思路。有一些代码或许一时难以理解，不要担心，我们将在后续的章节里面详细讨论。

1.2.1 确定目标

现在来把话题限定在初等平面几何，假定我们要写一篇关于勾股定理的短文，短文是一般的科技论文的模式，结构上包括标题、摘要、目录、几节的正文和最后的参考文献；内容包括文字、公式、图形、表格等。短文的格式很平凡，没有什么特别的地方，但也足够实际，可以代表大多数使用 LaTeX 的人日常接触最多的文档类型，只不过现实中的例子在内容上比这里的例子更丰富、更深刻。

为了能在书中方便地显示这个例子，我们把短文的页面设置得很小，四页拼成一页，完成后的样子见图 1.16。如果你以前已经对 LaTeX 有一些基础，不妨自己动手试排一下这个小例子（不偷看本章后面的说明），看看你能否准确高效地完成这个例子；即使你对 LaTeX 的实际了解还仅限于 1.1.3 节中的简单介绍，也不妨考虑一下，在这个极其简单的例子中，有哪些内容需要表现，它们对应的形式是什么，需要注意哪些问题。

1.2.2 从提纲开始

无论是对已经写好的文章进行排版，还是从零开始直接写文章，从提纲开始都是一个好主意。写出 LaTeX 文档的框架，进行必要的基本设置，然后再填入内容就方便了。

我们的例子《杂谈勾股定理》的提纲如下：

```
%-*- coding: UTF-8 -*-
% gougu.tex
% 勾股定理
\documentclass[UTF8]{ctexart}

\title{杂谈勾股定理}
\author{张三}
\date{\today}

\bibliographystyle{plain}
```

杂谈勾股定理

张三

2010 年 11 月 5 日

摘要

这是一篇关于勾股定理的小短文。

目录

1 勾股定理在古代

西方称勾股定理为毕达哥拉斯定理，将勾股定理的发现归功于公元前 6 世纪的毕达哥拉斯学派[1]。该学派得到了一个法则，可以求出可排成直角三角形三边的三元数组。毕达哥拉斯学派没有书面著作，该定理的严格表述和证明则见于欧几里德[1]《几何原本》的命题 47："直角三角形斜边上的正方形等于两直角边上的两个正方形之和。"证明是用面积做的。

我国《周髀算经》载商高（约公元前 12 世纪）答周公问：

> 勾广三，股修四，径隅五。

又载陈子（约公元前 7-6 世纪）答荣方问：

> 若求邪至日者，以日下为勾，日高为股，勾股各自乘，并而开方除之，得邪至日。

都较古希腊更早。后者已经明确道出勾股定理的一般形式。图 1 是我国古代对勾股定理的一种证明[2]。

[1]欧几里德，约公元前 330–275 年。

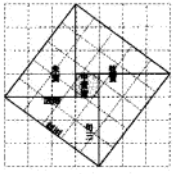


图 1: 宋赵爽在《周髀算经》注中作的弦图（仿制），该图给出了勾股定理的一个极具对称美的证明。

2 勾股定理的近代形式

勾股定理可以用现代语言表述如下：

定理 1 (勾股定理) 直角三角形斜边的平方等于两腰的平方和。

可以用符号语言表述为：设直角三角形 ABC，其中 $\angle C = 90°$，则有

$$AB^2 = BC^2 + AC^2. \tag{1}$$

满足式 (1) 的整数称为勾股数。第 1 节所说毕达哥拉斯学派得到的三元数组就是勾股数。下表列出一些较小的勾股数：

直角边 a	直角边 b	斜边 c	
3	4	5	$(a^2 + b^2 = c^2)$
5	12	13	

参考文献

[1] 克莱因. 古今数学思想. 上海科学技术出版社, 2002.

[2] 曲安京. 商高、赵爽与刘徽关于勾股定理的证明. 数学传播, 20(3), 1998.

[3] 矢野健太郎. 几何的有名定理. 上海科学技术出版社, 1986.

图 1.16 完整排版的小例子

```
11
12 \begin{document}
13
14 \maketitle
15 \tableofcontents
16 \section{勾股定理在古代}
17 \section{勾股定理的近代形式}
18 \bibliography{math}
19
20 \end{document}
```

源文件中的一些东西我们已经见过了，也有一些是没见过的。但可以看出整个文章的框架，现逐条进行说明：

- 前面以百分号 % 开头的行是注释。在 TEX 中，源文件一行中百分号后面的内容都会被忽略。这里有三行注释，第 1 行表明了这个文件的编码是 UTF-8，这对中文文档往往非常有用[①]；第 2 行是源文件的文件名 gougu.tex；第 3 行则说明了源文件的内容。注释并不是 TEX 源文件必需的，这里对文件内容的注释似乎与文档标题重复，不过对于比较大的文档，源文件往往分成多个文件，这类说明性的文字就十分重要了。
- 第 4 行是文档类，因为是中文的短文，所以使用 **ctexart**，并用 [UTF8] 选项说明编码。（参见 2.4.1 节）
- 第 6 行至第 8 行，声明了整个文章的标题、作者和写作日期，其中 \today 当然是“今天”的日期。这些信息并不马上出现在编译的结果中，而要通过第 14 行的 \maketitle 排版。（参见 2.3.1 节）
- 第 10 行的 \bibliographystyle 声明参考文献的格式。（参见 3.3.1 节）

  以上在 \begin{document} 之前的部分称为导言区（**preamble**），导言区通常用来对文档的性质做一些设置，或自定义一些命令。
- 第 12 行和第 20 行以 \begin{document} 和 \end{document} 声明了一个 document 环境，里面是论文的正文部分，也就是直接输出的部分。
- 第 14 行的 \maketitle 命令实际输出论文标题。（参见 2.3.1 节）
- 第 15 行的 \tableofcontents 命令输出目录。（参见 3.1.1 节）

① 这里用的其实是一种特定格式的特殊注释，源自 Emacs 编辑器，WinEdt 与一些 UNIX 下的编辑器可以根据这种注释自动判断文件的编码，不过这里使用这种格式主要为了好看。TeXworks 也有这种功能的注释，只是格式不同。

- 第 16 至 17 行两个 \section 开始新的一节。(参见 2.3.2 节)
- 最后第 18 行的 \bibliographystyle{math} 则是提示 TeX 从文献数据库 math 中获取文献信息，打印参考文献列表。(参见 3.3.1 节)

为了格式上的清晰，源文件中适当使用了一些空行作为分隔。在正文外的部分，空行不表示任何意义。

这里的提纲非常简单，整个文档也没有什么复杂的层次结构。编译提纲将得到只有一些标题的文件。我们并没有写任何编号或数字，所有编号，包括目录和页码都是自动生成的。注意这里要生成目录至少需要编译两次，让 LaTeX 有机会读完整个论文来计算目录结构。

1.2.3　填写正文

```
1 西方称勾股定理为毕达哥拉斯定理，将勾股定理的发现归功于公元前␣6␣世纪的
2 毕达哥拉斯学派。该学派得到了一个法则，可以求出可排成直角三角形三边的三
3 元数组。毕达哥拉斯学派没有书面著作，该定理的严格表述和证明则见于欧几里
4 德《几何原本》的命题␣47：“直角三角形斜边上的正方形等于两直角边上的两
5 个正方形之和。”证明是用面积做的。
6
7 我国《周髀算经》载商高（约公元前␣12␣世纪）答周公问……
```

填写正文的部分看起来比较容易，就是直接填写大段的文字，不过仔细查看代码，也有如下一些要注意的地方 (这里用 ␣ 表示空格)。

- 使用空行分段。单个换行并不会使文字另起一段，而只是起到使源代码更易读的作用 (上面的代码每行 35 个汉字)。空白行，也就是至多有空格的行，会使文字另起一段。空行只起分段作用，使用很多空行并不起任何增大段间距的作用。
- 段前不用打空格，LaTeX 会自动完成文字的缩进。即使手工在前面打了空格，LaTeX 也会将其忽略，事实上它会忽略每行开始的所有空格。也不要使用全角的汉字空格，这通常会使排版的效果变得糟糕。
- 通常汉字后面的空格会被忽略，其他符号后面的空格则保留，因而用左␣右就得到连续的“左右”，但 left␣right 则输出有空格的“left right”。单个的换行就相当于一个空格，因此源代码中大段文字可以安全地分成短行。空格只起分隔单词或符号的作用，使用很多空格并不起任何增大字词间距的作用。

  使用 xelatex 编译文档时，ctexart 文档类会调用 xeCJK 宏包，自动处理汉字与其他符号之间的距离，无论你有没有在它们之间加上正确的空格，这是十分方

便的。不过，在源代码中仍然可以给汉字与其他符号之间加上一个空格，这会令代码更加清晰。

换行与空格的使用，正是在 LaTeX 中文字排版最基本的部分，却也是最容易被忽略的。现在你的心思可能早已经飘到脚注和《周髀算经》的引用这些显眼的地方了，但在进行下一步之前最好还是巩固一下前面的内容。

练 习

1.2 从你最喜欢的小说中找几段文字，使用 LaTeX 排版。如果有某些特殊符号（比如注释符号 %）造成了问题，可以暂时将其去掉。

1.2.4 命令与环境

继续排版短文的第 1 节，我们来处理脚注和引用内容。

脚注是在正文“欧几里德”的后面用脚注命令 \footnote 得到的（参见 2.2.7 节）：

```
……见于欧几里德\footnote{欧几里德，约公元前 330--275 年。}《几何
原本》的……
```

在这里，\footnote 后面花括号内的部分是命令的参数，也就是脚注的内容。

文中还使用 \emph 命令改变字体形状，表示强调（emphasis）的内容：

```
……的整数称为\emph{勾股数}。
```

一个 LaTeX 命令（宏）的格式为：

无参数： **\command**

有 n 个参数： **\command**$\langle arg_1\rangle\langle arg_2\rangle\ldots\langle arg_n\rangle$

有可选参数： **\command**$[\langle arg_{opt}\rangle]\langle arg_1\rangle\langle arg_2\rangle\ldots\langle arg_n\rangle$

命令都以反斜线 \ 开头，后接命令名，命令名或者是一串字母，或是单个符号。命令可以带一些参数，如果命令的参数不止一个字符（不包括空格），就必须用花括号括起来。可选参数如果出现，则用方括号括起来。这里的脚注命令 \footnote 就是带有一个参数的命令，前面见到的 \documentclass 命令就是一个能带可选参数的命令。

引用的内容则是在正文中使用 quote 环境得到的：

```
……答周公问：
\begin{quote}
```

```
勾广三，股修四，径隅五。
\end{quote}
又载陈子（约公元前 7--6 世纪）答荣方问：
\begin{quote}
若求邪至日者，以日下为勾，日高为股，勾股各自乘，并而开方除之，得邪至日。
\end{quote}
都较古希腊更早。……
```

quote 环境即以 \begin{quote} 和 \end{quote} 为起止位置的部分。它将环境中的内容单独分行，增加缩进和上下间距排印，以突出引用的部分（参见 2.2.2 节）。

不过，如果只使用 quote 环境，并不能达到预想的效果：quote 环境并不改变引用内容的字体。因此还需要再使用改变字体的命令，即：

```
\begin{quote}
\zihao{-5}\kaishu 引用的内容。
\end{quote}
```

> 引用的内容。

这里，\zihao 是有一个参数的命令，选择字号（-5 就是小五号）；而 \kaishu 则是没有参数的命令，把字体切换为楷书，注意用空格把命令和后面的文字分开（参见 2.1.3 节和 2.1.4 节）。

类似地，文章的摘要也是在 \maketitle 之后用 abstract 环境生成的：

```
\begin{abstract}
这是一篇关于勾股定理的小短文。
\end{abstract}
```

摘要环境预设的格式已经满足我们的要求，不必再修改了。

上面使用的选择字体字号的命令与之前的脚注命令不同。\footnote{内容} 只在原地发生效果，即生成脚注；但 \zihao{字号} 与 \kaishu 命令则会影响后面的所有文字，直到整个分组结束，这种命令又称为声明（declaration）。

分组限定了声明的作用范围。一个 LaTeX 环境自然就是一个分组（group），因此前面的字号、字体命令会影响整个 quote 环境。最大的分组是表示正文的 document 环境，也可以用成对的花括号 { } 产生一个分组。

LaTeX 环境（environment）的一般格式是：

\begin{环境名}

```
\begin{⟨环境名⟩}
⟨环境内容⟩
\end{⟨环境名⟩}
```

有的环境也有参数或可选参数，格式为：

```
\begin{⟨环境名⟩}[⟨可选参数⟩]⟨其他参数⟩
⟨环境内容⟩
\end{⟨环境名⟩}
```

quote 环境是无参数的，后面我们很快会在制作表格时遇到有参数的环境。

文章第二节的定理，是用一类定理环境输出的（参见 2.2.4 节）。定理环境是一类环境，在使用前需要先在导言区做定义：

```
\newtheorem{thm}{定理}
```

这就定义了一个 thm 的环境。定理环境可以有一个可选参数，就是定理的名字，于是前面的勾股定理就可以由新定义的 thm 环境得到：

```
\begin{thm}[勾股定理]
直角三角形斜边的平方等于两腰的平方和。

可以用符号语言表述为……
\end{thm}
```

最后来注意一个小细节，前面在表示起迄年份时，用了两个减号 --，这在 LaTeX 中将输出一个“en dash”，即宽度与字母“n”相当的短线，通常用来表示数字的范围[12、35]。

1.2.5 遭遇数学公式

现在来看我们最关心的问题——输入数学公式，这大概是多数使用 LaTeX 的人花费精力最多的地方了。

最简单的输入公式的办法是把公式用一对美元符号 $ $ 括起来，如使用 $a+b$ 就得到漂亮的 $a+b$，而不是直接输入 a+b 得到的干巴巴的 a+b。这种夹在行文中的公式称为“正文公式”（in-text formula）或“行内公式”（inline formula）。

对比较长或比较重要的公式，一般则单独居中写在一行；为了方便引用，经常还给公式编号。这种公式被称作“显示公式”或“列表公式”（displayed formula），使用 equation 环境就可以方便地输入这种公式：

```
\begin{equation}
  a(b+c) = ab + ac
\end{equation}
```

$$a(b+c) = ab + ac \quad (1.1)$$

1-2-1

键盘上没有的符号，就需要使用一个命令来输入。例如表示“角”的符号 ∠，就可以用 \angle 输入。命令的名字通常也就是符号的名字，“角”的符号是 \angle，希腊字母 π 也就用其拉丁拼写 \pi[①]。用命令表示的数学符号在 LaTeX 中使用起来与用键盘输入的数学符号用起来并没有什么区别：

```
$\angle ACB = \pi / 2$
```

$\angle ACB = \pi/2$

1-2-2

数学公式不止是符号的堆砌，还具有一定的数学结构，如上下标、分式、根式等。在勾股定理的表述中，就用到了上标结构表示乘方：

```
\begin{equation}
AB^2 = BC^2 + AC^2.
\end{equation}
```

$$AB^2 = BC^2 + AC^2. \quad (1.2)$$

1-2-3

符号 ^ 用来引入一个上标，而 _ 则引入一个下标，它们用起来差不多等同于一个带一个参数的命令，因此多个字符的上下标需要用花括号分组，如 `$2^{10}=1024$` 得到 $2^{10} = 1024$。

怎么输入 90°？如果去查 4.3 节的数学符号表，你可能一无所获，由于 LaTeX 默认的数学字体中，并没有一个专用于表示角度的符号，自然也没有这个命令。角度的符号 ° 是通过上标输入的：`$^\circ$`。这里 \circ 其实是一个通常用来表示函数复合的二元运算符 “∘”，我们把它的上标借用来表示角度，90° 可以使用 `$90^\circ$` 输入。

这篇小短文用到的数学公式暂且就只有这么多，我们将在第 4 章再来深入讨论这个话题。

1.2.6　使用图表

准备图表比起输入文字和公式就要麻烦一些了，很多人能驾驭十分复杂的数学公式，却往往在图表问题上一筹莫展。这篇关于勾股定理的短文使用的图表形式都比较简单，但也是典型的。

① ISO 标准对科技文档要求常数 π 使用直立体，不能用斜体，这个例子不考虑这些。相关问题见 4.3.1 节。

首先来看插图。在 LaTeX 中使用插图有两种途径，一是插入事先准备好的图片，二是使用 LaTeX 代码直接在文档中画图。大部分情况下都是使用插入外部图片的方式，只在一些特别的情况大量用代码作图（如数学的交换图）。

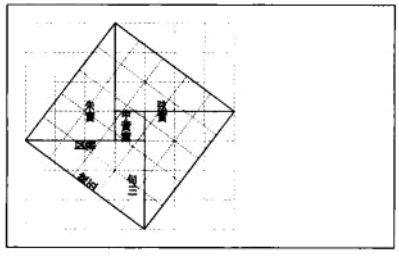
插图功能不是由 LaTeX 的内核直接提供，而是由 graphicx 宏包提供的。要使用 graphicx 宏包的插图功能，需要在源文件的导言区使用 \usepackage 命令引入宏包：

```
\documentclass{ctexart}
\usepackage{graphicx}
% ……导言区其他内容
```

引入 graphicx 宏包后，就可以使用 \includegraphics 命令插图了：

1-2-4

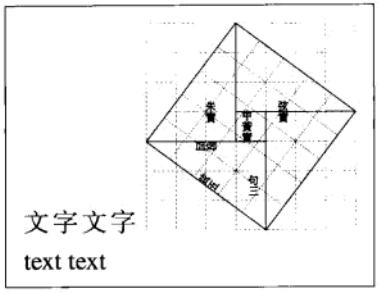
```
\includegraphics[width=3cm]{xiantu.pdf}
```



这里 \includegraphics 有两个参数，方括号中的可选参数 width=3cm 设置图形在文档中显示的宽度为 3 cm，而第二个参数 xiantu.pdf 则是图形的文件名（放在源文件所在目录）。有最常见的情况，图形使用其他画图工具做好，但在制作的时候尺寸不符合文章的要求，需要在插图时设置参数缩放到指定的大小。还有一些类似的参数，如 scale=⟨放缩因子⟩、height=⟨高度⟩ 等，我们在这篇小短文中实际使用的是 scale=0.6。插图命令支持的图形文件格式与所使用的编译程序有关，这篇中文文章使用 xelatex 命令编译，支持的图形格式包括 PDF、PNG、JPG、EPS 等，这里的图形实际是利用 Asymptote 语言制作的（参见 5.5.3 节）。

插入的图形就是一个有内容的矩形盒子，在正文中和一个很大的字符没有多少区别。因此如果把插图和文件混在一起，就会出现这样的情况：

```
文字文字
\includegraphics[width=3cm]{xiantu.pdf}
text text
```



除了一些很小的标志图形，我们很少把插图直接夹在文字之中，而是使用单独的环境列出。而且很大的图形如果固定位置，会给分页造成困难。因此，通常都把图形放在一个可以变动相对位置的环境中，称为浮动体（float）。在浮动体中还可以给图形加入说明性的标题，因此，在《杂谈勾股定理》中实际是使用下面的代码插图的：

```
1 \begin{figure}[ht]
2   \centering
3   \includegraphics[scale=0.6]{xiantu.pdf}
4   \caption{宋赵爽在《周髀算经》注中作的弦图（仿制），该图给出了勾股定
5   理的一个极具对称美的证明。}
6   \label{fig:xiantu}
7 \end{figure}
```

在上面的代码中，第 1 行和第 7 行使用了 figure 环境，就是插图使用的浮动体环境。figure 环境有可选参数 [ht]，表示浮动体可以出现在环境周围的文本所在处（here）和一页的顶部（top）。figure 环境内部相当于普通的段落（默认没有缩进）；第 2 行用声明 \centering 表示后面的内容居中；第 3 行插入图形；第 4 行和第 5 行使用 \caption 命令给插图加上自动编号和标题；第 6 行的 \label 命令则给图形定义一个标签，使用这个标签就可以在文章的其他地方引用 \caption 产生的编号（编号引用我们会在后面讲到）。这段插图的代码非常格式化，在绝大多数情况下，文章中的插图都是用与这里几乎完全相同的代码插入的。

下面再来看表格。插图可以用其他软件做好插入，但表格一般都还是直接在 LaTeX 里面完成的。制作表格，需要确定的是表格的行、列对齐模式和表格线，这是由 tabular 环境完成的：

```
1 \begin{tabular}{|rrr|}
2 \hline
3 直角边 $a$ & 直角边 $b$ & 斜边 $c$ \\
```

```
\hline
        3 &          4 &         5 \\
        5 &         12 &        13 \\
\hline
\end{tabular}
```

直角边 a	直角边 b	斜边 c
3	4	5
5	12	13

tabular 环境有一个参数，里面声明了表格中列的模式。在前面的表格中，|rrr| 表示表格有三列，都是右对齐，在第一列前面和第三列后面各有一条垂直的表格线。在 tabular 环境内部，行与行之间用命令 \\ 隔开，每行内部的表项则用符号 & 隔开。表格中的横线则是用命令 \hline 产生的。

表格与 \includegraphics 命令得到的插图一样，都是一个比较大的盒子。一般也放在浮动环境中，即 table 环境，参数与大体的使用格式也与 figure 环境差不多，只是 \caption 命令得到的标题是“表”而不是“图”。在《杂谈勾股定理》中，我们稍稍改变了一下 figure 环境通常的内容：

```
\begin{table}[H]
\begin{tabular}{|rrr|}
\hline
直角边 $a$ & 直角边 $b$ & 斜边 $c$\\
\hline
3 & 4 & 5 \\
5 & 12 & 13 \\
\hline
\end{tabular}%
\qquad
($a^2 + b^2 = c^2$)
\end{table}
```

1-2-5

直角边 a	直角边 b	斜边 c
3	4	5
5	12	13

$(a^2 + b^2 = c^2)$

这里并没有给表格加标题，也没有把内容居中，而是把表格和一个公式并排排开，中间使用一个 \qquad 分隔。命令 \qquad 产生长为 2 em(大约两个“M”的宽度) 的空白。因为我们已经使用 \qquad 生成足够长度的空格了，所以再用 \end{tabular} 后的注释符取消换行产生的一个多余的空格，这正好达到我们预想的效果。

之所以使用这种方式放置表格，是因为在正文中表格前面写道：

……下表列出一些较小的勾股数：

也就是说表格和正文是直接连在一起的，而且后面的公式也说明了表格的意义，自然就不再需要多余的标题了，这么一来表格就与正文连在一起，不允许再浮动了，因而这里本来是不应该使用浮动的 table 环境的，但我们仍然用了 table 环境，在表示位置的参数处使用了 [H]，表示“就放在这里，不浮动”。[H] 选项并不是标准 LaTeX 的 table 环境使用的参数，而是由 float 宏包提供的特殊功能。因此要让上面的代码正确运行，还要在导言区使用 \usepackage{float}。在这种表格很小 (不影响分页)，行文又要求连贯的场合，float 宏包的这种不浮动的图表环境是很有用的。

1.2.7 自动化工具

到目前为止，《杂谈勾股定理》这篇小文的大部分内容已经排完了，如果把前面提到的所有代码结合起来，装进一个文件中，差不多就能得到一篇完整的文章——这里说“差不多”，其实还缺少一些重要的东西，最明显的就是参考文献列表。

你一定已经注意到了，在前面文档的提纲中，我们已经用 \bibliographystyle 命令声明了参考文献的格式，又用 \bibliography 命令要求打印出参考文献列表。不过，这只是使用 BibTeX 处理文献的一个空架子，我们尚没有定义“参考文献数据库”，自然也不会产生任何文献列表。

BibTeX 使用的参考文献数据库其实就是一个后缀为 .bib 的文件。我们的《杂谈勾股定理》使用了一个包含 3 条文献的数据库文件 math.bib，内容如下：

```
% This file was created with JabRef 2.6.
% Encoding: UTF8

@BOOK{Kline,
  title = {古今数学思想},
  publisher = {上海科学技术出版社},
```

```
  year = {2002},
  author = {克莱因}
}

@ARTICLE{quanjing,
  author = {曲安京},
  title = {商高、赵爽与刘徽关于勾股定理的证明},
  journal = {数学传播},
  year = {1998},
  volume = {20},
  number = {3}
}

@BOOK{Shiye,
  title = {几何的有名定理},
  publisher = {上海科学技术出版社},
  year = {1986},
  author = {矢野健太郎}
}
```

正如上面所看到的，一个文献数据库文件的格式并不复杂，每则文献包括类型、引用标签、标题、作者、出版年、出版社等信息，可以直接手工输入。不过，正像前面数据库文件的注释（以 % 开头的两行）所显示的那样，这个数据库文件并不是直接输入上面的文件内容得到的，而是使用文献管理工具 JabRef 制作的（见图 1.17），参见 3.3.2 节。

在现实中，BibTeX 数据库经常并不需要我们自己录入，而可以从相关学科的网站直接下载或是从其他类型的文献数据库转换得到。即使是在需要我们自己录入的情况下，使用 JabRef 这种软件来管理也更方便，不易出错。

BibTeX 是一个专用于处理 LaTeX 文档文献列表的程序，使用 BibTeX 处理文献时，编译 gougu.tex 这一个文档的步骤就增加为四次运行程序（或点击四次按钮）：

```
xelatex gougu.tex
bibtex gougu.aux
xelatex gougu.tex
```

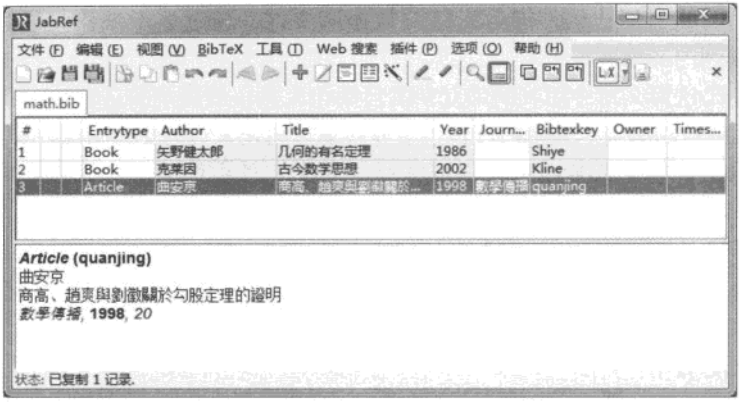


图 1.17　在 Windows 7 下用 JabRef 打开 math.bib

```
xelatex gougu.tex
```

第一次运行 xelatex 为 BibTeX 准备好辅助文件，确定数据库中的哪些文献将被列出来。然后 bibtex 处理辅助文件 gougu.aux，从文献数据库中选取文献，按指定的格式生成文献列表的 LaTeX 代码。后面两次 xelatex 再读入文献列表代码并生成正确的引用信息。这种利用多趟编译处理辅助文件的方式看起来有些复杂，但这是使用自动文献生成的代价之一，不过好处也是明显的，文献的管理、文献列表的排序和排版格式等都能高效漂亮地完成。

如果现在就使用上面的步骤编译，你仍然会一无所获，因为还没有选择要列出的文献。LaTeX 只选择被引用的文献。引用文献的方法是在正文中使用 \cite 命令，如：

```
西方称勾股定理为毕达哥拉斯定理，将勾股定理的发现归功于公元前 6 世纪的
毕达哥拉斯学派 \cite{Kline}。

……是我国古代对勾股定理的一种证明 \cite{quanjing}。
```

\cite 命令的参数 Kline 和 quanjing 分别是其中两篇的引用标签，也就是在 math.bib 中每个条目第一行出现的东西。使用 \cite 命令会在引用的位置显示文献在列表中的编号（它在第 3 次 xelatex 编译后才能确定），同时在辅助文件中说明某文献将被引用。如果要在列表中显示并不直接引用的文献，可以使用 \nocite 命令，一般是把它放在 \bibliography 之前，像我们这篇文章中一样：

```
\nocite{Shiye}
\bibliography{math}
```

有了上面的引用代码，加上完整的数据库文件，通过多步编译，最终就能得到 1.2.1 节中看到的文献列表了。

BibTeX 是这篇文章中用到的最复杂的自动化工具，最简单的自动化工具则是页码、定理和公式的自动编号，其余的还包括生成目录与图表公式的交叉引用。

目录也是自动从章节命令中提取并写入目录文件中的，我们在提纲中就使用了 \tableofcontents 命令，它将在第二次 xelatex 编译时生效。

引用不仅限于参考文献。图表、公式的编号，只要事先设定了标签，同样可以通过辅助文件为中介引用。基本的交叉引用命令是 \ref，它以标签为参数，得到被引用的编号。例如，在插图时已经用 \label 命令为弦图定义了标签 fig:xiantu，于是，在正文中就可以使用

```
图 \ref{fig:xiantu} 是我国古代对勾股定理的一种证明 \cite{quanjing}。
```

来对弦图的编号进行引用。

公式编号的引用也可照此办理，不过需要先在公式中定义标签：

```
\begin{equation}\label{eq:gougu}
AB^2 = BC^2 + AC^2.
\end{equation}
```

而后在正文中以（\ref{eq:gougu}）引用。实际中引用公式非常常用，数学宏包 amsmath 就定义了 \eqref 命令，专门用于公式的引用，并能产生括号：

```
% 导言区使用 \usepackage{amsmath}
满足式 \eqref{eq:gougu} 的整数称为\emph{勾股数}。
```

练 习

1.3 除了引用文献，《杂谈勾股定理》中共有三处交叉引用，除了引用插图和公式，在第 2 节的定理后面还引用了第 1 节的编号，试写出相关的代码。

1.2.8 设计文章的格式

写到这里，原先的提纲骨架已经变成一篇完整的文章，似乎已经没有什么可说的了。然而，TeX 的精神是精益求精、追求完美，如果我们对比 1.2.1 节的目标来审视我们现在排版的结果，还是会发现有些不同，如标题的字体还需要修正，目录中少了“参考文献”一项，插图标题的字体、字号和对齐都不正确等。更重要的还有，1.2.1 节预

设的文章页面很小，页边距也非常紧凑，与现在宽大的页面大相径庭。这些都属于文章的整体格式，需要进一步的设计完善。

绝大部分设计工作是在文章的导言区通过一些命令定义和参数设定来完成的，但往往相当复杂，好在其中的大多数工作可以通过使用一些宏包来简化，前面已经用到过 graphicx、float、amsmath 几种宏包完成一些工作，这里也要用到几种。

设计页面尺寸可以使用 geometry 宏包（参见 2.4.2 节）：

```
\usepackage{geometry}
\geometry{a6paper,centering,scale=0.8}
```

这是最简单的设定方式，定义页面使用 A6 纸大小，版心居中，长宽占页面的 0.8 倍。

改变图表标题格式可以使用 caption 宏包（参见 5.3.2 节）：

```
\usepackage[format=hang,font=small,textfont=it]{caption}
```

设定图表所有标题使用悬挂对齐方式（即编号向左突出），整体用小字号，而标题文本使用斜体（对汉字来说就是楷书）。

增加目录的项目则可以用 tocbibind 宏包：

```
\usepackage[nottoc]{tocbibind}
```

宏包默认会在目录中加入目录项本身、参考文献、索引等项目。这里使用 nottoc 选项取消了在目录中显示目录本身。

标题和作者的字体可以直接在 \title、\author 命令中设定，因为标题本身就是用这些命令在导言区定义的：

```
\title{\heiti 杂谈勾股定理}
\author{\kaishu 张三}
\date{\today}
```

其中 \heiti 是和 \kaishu 类似的中文字体命令，把字体切换为黑体。

这篇短文到这里就全部排完了，不过，正文中表示引用的 quote 环境里面还夹杂着字体命令，这种散落在各处的格式设置很难看清，而且不方便修改。为了解决这个问题，可以利用 \newenvironment 命令定义一个新的环境，在原来 quote 的基础上再增加格式控制：

```
\newenvironment{myquote}
  {\begin{quote}\kaishu\zihao{-5}}
  {\end{quote}}
```

这里，\newenvironment 有三个参数，第一个参数是环境的名字，后两个参数分别是在环境开始和末尾处的代码，因此，就可以用新环境

```
\begin{myquote}
勾广三，股修四，径隅五。
\end{myquote}
```

来代替原来的 quote 环境了。如果此时需要更改引用的格式，那么只需要在导言区修改 myquote 的定义，而不必在全文中搜索所有的 quote 环境的使用了。

类似地，原来数学公式中角度的单位 ^\circ 也很不直观，可以用 \newcommand 命令定义一个新的命令 \degree：

```
\newcommand\degree{^\circ}
```

其中，\newcommand 命令的两个参数分别是新命令和新命令的定义，于是我们就可以用 $90\degree$ 来代替原来不直观的 90° 了。

在整篇文章编排结束之际，我们还是使用自定义的环境 myquote 和自定义的命令 \degree 代替了文中出现的特殊格式控制。类似地，在设定插图标题的字体时，并没有把字体、字号的命令塞进 \caption 命令的参数中，而是使用 caption 宏包统一设置。这样看起来比最“直接”的做法要多绕一道弯子，但好处是更清晰和更容易修改格式。这篇短文排在普通 A4 大小的纸张上只有一两页，还看不出什么特别的好处，但当你开始编写和维护几十上百页的长文档时，在设计阶段所付出的精力就会得到回报了。

LaTeX 是一种结构化的排版语言，在填写标准格式的模板时（就像我们填写 1.2.2 节所列的提纲一样）可以忽略编号、格式等许多具体细节。在文档排版中应该主动追求内容与格式的分离，在 document 环境之内避免直接使用诸如字体字号、对齐缩进的格式控制命令，而代之以有具体意义的环境和命令，让文档变得清晰。这种模式化的操作能提高工作效率，许多 LaTeX 的拥护者把这种工作方式称为“所想即所得[①]”。可是不要忘记，机器还远没有智能化到想人之所想的程度，LaTeX 也不能阻止我们编排出效果糟糕、代码混乱的文章，要得到好的文章，无论是在内容上还是排版形式上，都得靠我们自己。

练 习

1.4 收集这一节关于《杂谈勾股定理》的全部代码，把它们整理成完整的文档，编译运行，看看能不能得到与 1.2.1 节完全一样的效果。

① What you think is what you get. 区别于图形化工具的“所见即所得”（What you see is what you get）模式。尽管关于孰好孰坏多有争论，但实际上两种方式各有其优缺点和适用场合。

本章注记

关于 TeX 的经典文献是 Knuth [126]。关于 LaTeX 的经典文献是 Lamport [136]、Mittelbach and Goossens [166]。

LaTeX 有很多优秀的简短入门书籍或文档可以免费获得，尤其常见的如 Indian TeX Users Group [112]、Oetiker et al. [188]，其中 [188] 有中文译本 [187]；中文的短篇文档如 黄新刚（Alpha Huang）[318]。这些书籍或文档只有本书的几分之一篇幅，可能更容易读完。

与本书程度接近的英文书籍可参见 van Dongen [65]、Grätzer [90]、Kopka and Daly [134]。

关于 MiKTeX 2.9 的进一步信息可参见 Schenk [226]；关于 TeX Live 2012 的详细信息可参见 Berry [25]。有关 CTeX 中文套装的信息和讨论，可前往 CTeX 论坛[①]查询。

① http://bbs.ctex.org/

第2章 组织你的文本

从这一章开始，我们将要深入 LaTeX 编辑的各个方面。首先来看 LaTeX 文档中文本的编辑和组织，在探究中，我们会从一个空格、一个标点开始分析，但深入细节的同时也不要忘记整个文档的结构和组织性，要见树木更见森林。

2.1 文字与符号

2.1.1 字斟句酌

简单正文的输入没有太多特别之处，TeX 传统上使用扩展 ASCII 字符集[①]，较新的 TeX 引擎使用 UTF-8 编码。在接受的字符集之内，除了个别特殊符号，大部分字符可以直接录入。

2.1.1.1 从字母表到单词

LaTeX 可以从标准键盘上直接打出 26 个字母的大小写形式，当然，从字母表到单词只有一步之遥，即用空格和标点把字母分开。但事实上总免不了要遇到一些稀奇古怪的词汇或人名，试试输入下面的词：

café　Gödel　Antonín Dvořák　Øster Vrå　Kırkağaç

或者这些：

① 旧的中文处理方式使用 CJK 宏包，它的 GBK 等编码支持是实际也是在扩展 ASCII 字符集上工作的，只是使用特殊的方式把两个扩展 ASCII 字符拼接为一个汉字。

χαϊδεύης　　Крюкова

上面的例子中，前一组词都是来自拉丁字母，但明显增加了许多符号；后一组则是希腊语词汇和俄语人名。我们常用的字母表包括扩展的拉丁字母、希腊字母和西里尔字母（Cyrillic alphabet），这比标准键盘上所能直接输入的 52 个字母要多得多。不过不用担心，数以万计的汉字我们也搞定了，区区几个字母也不在话下。

解决输入超过 ASCII 码范围的字母的问题有传统和现代两类方案，先来看一下现代的方案。

现代的方案就是使用 UTF-8 编码直接输入。在 XeLaTeX 这样的排版引擎下，UTF-8 编码是原生的，不需要任何多余的设置：

```
% UTF-8 编码
café\quad Gödel\quad Antonín Dvořák

χαϊδεύης\qquad Крюкова
```

café　Gödel　Antonín Dvořák
χαϊδεύης　Крюкова

2-1-1

不过，要想正确地输入、显示并输出所有这些符号，仍然不是一件简单的事。

输入特殊的字母需要计算机键盘布局或输入法的支持，例如在法语键盘中（一般可以在操作系统中设置），标准键盘 7890 位置上的字符分别是 è _ ç à，这种键盘布局对需要大量录入法语的人来说特别有用。如果不方便使用特殊键盘来设置，操作系统或编辑器往往还提供了“字符映射表”一类的程序或插件，可以用鼠标选取一些字符输入；中文输入法的软键盘也可以用来输入一小部分特殊的外文字母。

显示 ASCII 以外的字母表需要编辑器所使用的字体的支持。大部分西文字体都支持扩展拉丁字母，如 è、ç，但不是所有字体都有希腊字母和西里尔字母，即使是一些罕用的拉丁字母（如 ř）也有可能有缺失。编辑器常用的等宽字体中，Windows 和 Mac 系统都预装的 Courier New，Windows 下的 Consolas、Lucida Sans Typewriter 等都能显示上面所列的所有字母，Linux 下常用的 Bitstream Vera Sans Mono、Inconsolata 则只支持到扩展拉丁字母，因此配置编辑器时也需要仔细选择。

最后但却最重要的是，TeX 系统输出时所使用的字体，也必须要能直接显示这些字母。LaTeX 默认的 Computer Modern 在一个字体中只覆盖很小的字符集，XeLaTeX 下默认的 Latin Modern 字体要好得多，一般都支持到拉丁字母（这通常就足够了），但希腊、西里尔等字母需要更换其他字体。要完整显示其他语言的字母，就必须频繁地更换不同的字体，或在 TeX 中使用覆盖更大字符集的字体（更换字体参见 2.1.3 节）。比如，为了能正确显示前面例子中的三种字母，我们在前面已经设置了 Windows 系统预装的 Times New Roman 字体。

而传统的解决方案是使用特殊的命令，给字母加上重音的标记，使用特殊字母或者整体更换字母表。

所谓重音（accents），是指加在字母上的标记，实际包括抑音符、锐音符、抑扬符等等多种符号。LaTeX 支持的重音记号及其命令见表 2.1，命令名称大多取象形的符号。将重音加之于普通的拉丁字母上，可以得到一大批扩展的拉丁字母。

表 2.1 LaTeX 中的重音命令，以字母 o 为例

ò	\`o	ó	\'o	ô	\^o	ö	\"o
õ	\~o	ō	\=o	ȯ	\.o	ǒ	\u{o}
ǒ	\v{o}	ő	\H{o}	o͡o	\t{oo}	o̊	\r{o}
o̧	\c{o}	ọ	\d{o}	o̲	\b{o}		

使用命令可以输入的 LaTeX 字母，见表 2.2。其中 ı, ȷ 就是字母 i, j，只是在加上重音命令时需要去掉上面的点。ij, IJ, SS 是字母连写，在默认的字体中没有区别。

表 2.2 LaTeX 中的特殊字母

Å	\AA	å	\aa	Æ	\AE	æ	\ae
Œ	\OE	œ	\oe	SS	\SS	ß	\ss
IJ	\IJ	ij	\ij	Ł	\L	ł	\l
Ø	\O	ø	\o	ı	\i	ȷ	\j

如果还要输入希腊字母和西里尔字母，默认的字体就不够用了，需要更换其他编码和字体。为此，LaTeX 提供了 babel 宏包，可以方便同时访问多种语言的字母表。babel 宏包可带有一个或多个语言的可选参数，支持不同的语言，如

2-1-2
```
\usepackage[greek,english]{babel}
```

将使用英语和希腊语，其中最后一个参数的英语是默认语言，此时希腊语就可以用 ASCII 字符代替：

```
\textgreek{abcde}
```

上述代码输出 αβςδε，需要少量俄文的西里尔字母，可以换用 OT2 编码的字体（参见 2.1.3 节）得到，例如：

2-1-3
```
% 导言区 \usepackage[OT2,OT1]{fontenc}
{\fontencoding{OT2}\selectfont ABCabc}
```

АБЦабц

如果排版全文是俄文的文章，也可以用 russian 参数使用 babel 宏包，不过输入时就不使用 ASCII 码，而使用俄文专用的编码。由于这些语言较少使用，这里不做更多说明，可参见 [31、245、278]。

使用 pdfTEX 这样的传统排版引擎也可以使用 UTF-8 编码输入文字，此时需要使用 inputenc 宏包并选用 utf8 选项，它会将 UTF-8 的输入编码自动转换为当前字体编码所对应的符号，字体编码的设置仍然与原来一样，如：

```
% coding: utf-8
% pdflatex 命令编译
\documentclass{article}
\usepackage[OT2,T1]{fontenc}
\usepackage[utf8]{inputenc}
\begin{document}
café \qquad Gödel
{\fontencoding{OT2}\selectfont Крюкова}
\end{document}
```

2-1-4

LATEX 在排版中会将单词中的一些字母连写为一个符号，即连字（ligature）。连字的有无和多少一般是由使用的字体决定的，在默认的 Computer Modern 或 Latin Modern 字体中，小写字母组合 ff, fi, fl, ffi, ffl 都有连字[①]：

```
differ find flight difficlut ruffle
```

differ find flight difficlut ruffle

2-1-5

本书中主要使用的 Times 字体则只有 fi 和 fl 的连字（fi, fl），而一些专业字体可能使用的更多：

ﬀ ﬁ ﬂ ﬃ ﬄ ﬅ ﬆ

偶尔出于意义或美观的考虑，需要取消连字。此时可以使用空的分组，或借用 \/ 命令（参见 2.1.3 节）：

```
shelfful shelf{}ful shelf\/ful
```

shelfful shelfful shelfful

2-1-6

使用 XETEX 引擎、OpenType 字体时，可以方便地使用 fontspec 宏包的 Ligature 字体选项选择连字的有无和程度，参见 [212]。

① 这也是排版中最为基本的五种连字。大部分连字都出现在字母 f 之后，这是因为字母 f 的实际内容通常会超出自己的字符边界，不使用连字时可能会与相邻字母挤在一起。

练 习

2.1 使用 LaTeX 输入前面特殊拉丁字母的例子：

café Gödel Antonín Dvořák Øster Vrå Kırkağaç

2.1.1.2 正确使用标点

在键盘上，可以直接使用的标点符号有 16 种：

```
,  .  ;  :  !  ?  `  '  (  )  [  ]  -  /  *  @
```

标点 , . ; : ! ? 用来分隔句子或部分句子，在每个标点之后应该加上空格，以保证正确的距离和换行。在特殊情况下，这些标点与空格还有一些更微妙的关系，参见 2.1.1.3 节。

引号在 LaTeX 中用 ` 和 ' 两个符号表示。单引号就用一遍，双引号用两遍。如果遇到单引号与双引号连续出现的情形，则在中间用 \, 命令分开：

2-1-7

```
``\,`A' or `B?'\,'' he asked.
```

“‘A’ or ‘B?’” he asked.

这里 \, 命令产生很小的间距，注意 LaTeX 并不会忽略以符号命名的宏前后的空格，所以在它前后都不要加多余的空格。符号 ' 同时也是表示所有格和省字的撇号（apostrophe，如 “It's Knuth's book.”）。

引号和括号通常要在前后加空格分隔单词。逗号、句号等标点何时放在引号和括号内，何时放在引号和括号外，可参见英文写作的格式指导 [12、282]。

除了在数学模式中表示减号，符号 - 在 LaTeX 正文中也有多种用途：单独使用时它是连字符（hyphen）；两个连用（--），是 en dash，用来表示数字范围；三个连用（---），是 em dash，即破折号[①]：

2-1-8

```
An inter-word dash or, hyphen, as in X-ray.

A medium dash for number ranges, like 1--2.

A punctuation dash---like this.
```

An inter-word dash or, hyphen, as in X-ray.
A medium dash for number ranges, like 1–2.
A punctuation dash—like this.

① en 和 em 指符号的宽度，详见 2.1.5 节。

不过，按中文写作的习惯，表示数字范围也常使用符号 ～（数学模式的符号 $\sim$），有时也用汉字的全角破折号或半个汉字破折号。

西文的省略号（ellipsis）使用 \ldots 或 \dots 命令产生，相比直接输入三个句号，它所略微拉开的间距要合理得多：

```
Good: One, two, three\ldots

Bad: One, two, three...
```

Good: One, two, three…
Bad: One, two, three...

2-1-9

\ldots 与 \dots 命令在正文中是等价的，它们会在每个点后面增加一个小的间距，因而直接在 \ldots 后面再加逗号、句号、叹号等标点，也能得到正确的间距。西文省略号的用法在不同的格式手册中往往有详细规定[12、35]，通常在句中使用时，前后要加空格，而在句末使用则应该使用 4 个点。这是因为 \ldots 的后面也有间距，所以使用 H\ldots. 能得到正确的 “H....”，但直接使用 H \ldots\ H 却将得到错误的间距 “H … H”（后一个间距比前一个大）。解决的办法是把省略号放进数学模式：

```
She $\ldots$ she got it.

I've no idea\ldots.
```

She . . . she got it.
I've no idea….

2-1-10

标准键盘上不能直接录入的标点符号有 10 个，它们占据了主键盘上面一排的一大半：

~ # $ % ^ & { } _ \

它们都有特殊作用，其中的许多我们已经熟知：数学模式符号 $、注释符 %、上标 ^、分组 { }、宏命令 \。剩下的符号中，~ 是带子，# 用在宏定义中，& 用于表格对齐，而 _ 表示数学模式的下标，我们也将会在后面的章节中陆续遇到。要在正文中使用这些符号，大部分是在前面加 \，只有个别例外：

```
\# \quad \$ \quad \% \quad \& \quad
\{ \quad \} \quad \_ \quad
\textbackslash
```

$ % & { } _ \

2-1-11

可以用没有字母的重音 \~{} 和 \^{} 输出 ~ 和 ^，但这两个符号一般不直接在普通正文中出现，而出现在其他地方：可以是重音符号；可以出现在程序代码中（就像现在看到的 ~ 和 ^），参见 2.2.5 节使用抄录；此外还有一个数学符号 ～，参见 4.3.3 节。

符号

| < > + =

虽然可以接受，但它们一般用在数学公式中，其文本形式的效果不好或有错，一般并不使用它们。键盘上的双引号 " 一般也极少使用在正文中，而常被另外定义移作他用。

中文使用的标点与西文标点不同，中文写作使用全角标点：

句号	。或.	逗号	，	顿号	、	分号	；
冒号	：	问号	？	感叹号	！	间隔号[1]	·
单引号	‘ ’	双引号	“ ”	单书名号	〈〉	双书名号	《》
括号	（）［］〔〕						
省略号	……	破折号	——				

在计算机中使用中文输入法录入全角标点是通常是很直接的。特别需要说明的是破折号（——）和省略号（……），它们都占两个中文字符，在大部分输入法中可以使用 Shift 键加减号 - 和数字 6 键得到。

在科技文章中，为与数字、字母区分，中文的句号一般也使用一个圆点表示，此时应该使用全角的“．”而非混用西文句号。这个标点在大部分中文输入法下可能不易输入，可以先统一使用句号“。”，最后统一替换。

也有一种科技文章的写作风格，是中文与西文统一使用西文的标点，只有顿号、破折号和省略号仍用中文标点。但这样做可能造成标点大小、位置与汉字对不准，以及字体风格的不统一，应该小心使用。

LaTeX 并不会自动处理好汉字标点的宽度和间距，甚至不能保证标点的禁则（如句号不允许出现在一行的开始）。使用 XeLaTeX 作为排版引擎时，中文标点一般是由 xeCJK 宏包控制的。xeCJK 提供了多种标点格式[2]，默认是全角式，即所有标点占一个汉字宽度，只在行末和个别标点之间进行标点挤压。还支持其他一些标点格式，可以使用 \punctstyle 命令修改：

① 在较早版本的 xeCJK 中间隔号被看做是西文标点，需要用 \xeCJKsetcharclass{‘·’}{‘·’}{1} 命令把它设置为汉字符号。参见 2.2.5 节。

② 旧的中文处理方式使用 CJK 宏包处理汉字，CJKpunct 宏包处理标点的禁则和间距。CJKpunct 也可以使用 \punctstyle 命令设定标点的格式，用法和 xeCJK 宏包类似。

\punctstyle{quanjiao} 全角式，所有标点全角，有挤压。例如，“标点挤压”。又如《标点符号用法》。
\punctstyle{banjiao} 半角式，所有的标点半角，有挤压。例如，“标点挤压”。又如《标点符号用法》。
\punctstyle{kaiming} 开明式，部分的标点半角，有挤压。例如，“标点挤压”。又如《标点符号用法》。
\punctstyle{hangmobanjiao} 行末半角式，仅行末挤压。例如，“标点挤压”。又如《标点符号用法》。
\punctstyle{plain} 无格式，只有禁则，无挤压。例如，“标点挤压”。又如《标点符号用法》。

2.1.1.3 看不见的字符——空格与换行

文本中的空格起分隔单词的作用，任意多个空格与一个空格的功能相同；只有字符后面的空格是有效的，每行最前面的空格则被忽略，这样有利于复杂代码的对齐；单个换行也被看做是一个空格。例如（我们仍用 ␣ 表示空格）：

```
This␣is␣␣␣␣␣a␣short
sentence.␣␣␣This␣is
␣␣␣␣␣␣␣␣␣␣␣another.
```

This is a short sentence. This is another.

2-1-12

以字母命名的宏，后面空格会被忽略。如果需要在命令后面使用空格，可以使用 \␣，它表示两个普通单词间的空格距离；也可以在命令后加一个空的分组 {}，有时也可以把命令用一个分组包裹起来：

```
Happy␣\TeX␣ing.␣Happy␣\TeX\␣ing.
Happy␣\TeX{}␣ing.␣Happy␣{\TeX}␣ing.
```

Happy TEXing. Happy TEX ing.
Happy TEX ing. Happy TEX ing.

2-1-13

有一种不可打断的空格，在 TEX 中被称为带子（ties），用 ~ 表示。TEX 禁止在这种空格之间分行，因而可以用来表示一些不宜分开的情况，例如[126, Chapter 6]：

```
Question~1          % 名称与编号间
Donald~E. Knuth     % 教名之间，但姓可以断行
Mr.~Knuth           % 称谓缩写与名字间
function~$f(x)$     % 名字后的短公式
1,~2, and~3         % 序列的部分符号间
```

2-1-14

西文的逗号、句号、分号等标点后面应该加空格，这不仅能保证正确的间距，也能保证正确的换行。这是因为标点后如果没有空格，就不能换行。LaTeX 在西文句末（包括句号 . 问号 ? 和叹号 !）后面使用的距离会比单词间的距离大些，这在上面的例子中已经可以看到。更确切地说，LaTeX 把大写字母后的点看做是缩写标记，把小写字母后的点看做是句子结束，并对它们使用不同的间距；但偶尔也有大写字母结束的句子，或小写字母的缩写，这时就必须明确告诉 LaTeX 使用普通单词间的空格 \␣，或用 \@. 指明 . 是大写字母后的句末。例如：

2-1-15

```
A␣sentence.␣And␣another.

U.S.A.␣means␣United␣States␣Army?

Tinker␣et␣al.\␣made␣the␣double␣play.

Roman␣number␣XII\@.␣Yes.
```

A sentence. And another.
U.S.A. means United States Army?
Tinker et al. made the double play.
Roman number XII. Yes.

有时也需要整体禁止这种在标点后的不同的间距，法语排版的习惯就是如此。此时可以使用 \frenchspacing 命令来禁止标点后的额外间距。

汉字后的空格会被忽略[①]。使用 xelatex 编译中文文档时，汉字和其他内容之间如果没有空格，xeCJK 宏包会自动添加[②]：

2-1-16

```
中文和English的混排效果
并不依赖于␣space␣的有无。
```

中文和 English 的混排效果并不依赖于 space 的有无。

个别时候需要忽略汉字与其他内容之间由 xeCJK 自动产生的空格，这时可以把汉字放进一个盒子里面：

2-1-17

```
\mbox{条目}-a 不同于条目-b
```

条目-a 不同于条目 -b

① 旧的中文处理方式使用 CJK 宏包。它的 CJK* 环境忽略汉字后面的空格，而 CJK 环境则保留。

② 使用 CJK 方式处理中文时，汉字与其他内容之间的间距必须手工控制，常用的办法是使用 CJK* 环境，并用 \CJKtilde 命令改变符号 ~ 的意义，表示汉字与其他内容间的距离，TeX 中的带子改用 \nbs 命令表示。可以手工在合适的地方加 ~ 表示间距，或用命令行工具 cctspace 添加。这是 ctex 宏包及文档类不使用 XeLaTeX 时的默认设置。另一种较好的方式是使用 CJKspace 宏包，它忽略汉字间的空格，保留其他空格，从而可以用较自然的方式书写，但这类方式都不如在 XeLaTeX 引擎下方便。

还有时需要完全禁用汉字与其他内容之间的空格（例如在本书所有 TEX 代码中），这时可以使用 \CJKsetecglue 手工设置汉字与其他内容之间的内容为空（默认是一个空格）：

```
\CJKsetecglue{}
汉字word
```

汉字word　　2-1-18

在空格之中，最神奇的一种可能就是被称为幻影（phantom）的空格。幻影命令 \phantom 有一个参数，作用是产生与参数内容一样大小的空盒子，没有内容，就像是参数的一个幻影一样。偶尔可以使用幻影完成一些特殊的占位和对齐效果：

```
幻影\phantom{参数}速速隐形

幻影参数速速显形
```

幻影　　速速隐形
幻影参数速速显形　　2-1-19

类似地有 \hphantom 和 \vphantom，分别表示水平方向和垂直方向的幻影（在另一个方向大小为零）。

空行，即用连续两个换行表示分段，段与段之间会自动得到合适的缩进。任意多个空行与一个空行的效果相同。

分段也可以用 \par 命令生成，这种用法一般只在命令或环境定义的内部使用，而普通行文中不宜出现。与连续的空行类似，连续的 \par 命令也只产生一次分段效果。

除了分段，也可以让 LATEX 直接另起一行，并不分段。有两种相关的命令：\\ 命令直接另起一行，上一行保持原来的样子；而 \linebreak 则指定一行的断点，上一行仍按完整一行散开对齐：

```
这是一行文字\\另一行

这是一行文字\linebreak 另一行
```

这是一行文字
另一行
这是一行文字
另一行　　2-1-20

\\ 一般用在特殊的环境中，如排版诗歌的 verse 环境（2.2.2 节），特别是在对齐（2.2.6 节）、表格（5.1.1 节）和数学公式（4.4 节）中使用广泛，但很少用在普通正文的行文中；\linebreak 命令则用在对个别不太合适分行的手工精细调整上。为了完成精细调整分行的功能，\linebreak 可以带一个从 0 到 4 的可选参数，表示允许断行的

程度，0 表示不允许断行，默认的 4 表示必须断行。类似地，也有一个 \nolinebreak 命令，只是参数意义与 \linebreak 相反。注意在正常的行文中，这两个命令都不会被用到。

\\ 命令可以带一个可选的长度参数，表示换行后增加的额外垂直间距。如 \\[2cm]。因此必须注意在命令 \\ 后面如果确实需要使用方括号（即使括号在下一行），则应该在 \\ 后面加空的分组以示分隔，否则会发生错误，这种情况在数学公式中非常常见：

2-1-21

```
\begin{align*}
[2 - (3+5)]\times 7 &= 42 \\{}
[2 + (3-5)]\times 7 &= 0
\end{align*}
```

$$[2-(3+5)]\times 7 = 42$$
$$[2+(3-5)]\times 7 = 0$$

练 习

2.2 LaTeX 用命令 \# \$ \% 表示符号 # $ %，为什么给反斜线 \ 用了 \textbackslash 这么复杂的名称？

2.1.2 特殊符号

除了一般的文字，有时还需要使用一些特殊的符号，其中在正文中最为常用的如表 2.3 所示。

表 2.3 正文常用的部分特殊符号

§	\S	†	\dag	‡	\ddag	¶	\P
©	\copyright	®	\textregistered	™	\texttrademark	£	\pounds
•	\textbullet						

上面的几种符号中，不带 \text 前缀的是在文本模式和数学模式通用的。特殊符号依赖于当前使用的字体，上面所列举的是在 LaTeX 默认字体设置下的效果。

LaTeX 定义了一批常用的符号命令，但实际使用时仍然捉襟见肘。在不同的字体包中，也定义了其他大量的符号，例如最常用的 LaTeX 的基本工具宏包 textcomp 就定义了大量用于文本的符号，例如欧元符号 \texteuro (€)、千分符 \textperthousand (‰) 等；tipa 宏包提供了国际音标字体的访问（比如 ['lɑtɛk]）；dingbat、bbding、pifont 等宏包提供了许多指示、装饰性的小符号，如 ☞，等等。在这里一一枚举所有这些宏包和

命令既冗长无味，也没有必要，Pakin 编写的“LaTeX 符号大全”[192] 是查找各种古怪符号的绝佳参考，它收集了约 70 个宏包的近 6000 个文本或数学符号，日常使用的各种符号一般都能在上面找到，同时这个文档也讲述了符号字体的一些一般知识及制作新符号的办法。

使用宏包来调用特殊符号时，需要注意的是，有些宏包只提供符号命令（如 bbding），可以随意使用；有些宏包提供一套符号字体的选择方式（如 tipa），可以通过与此符号对应的 ASCII 符号或符号的数字编码来使用符号；有些宏包则实际上是完整的正文字体包（如 fourier），使用它会整体改变全文的默认字体，同时提供一些额外的符号。“符号大全”[192] 对这些情况并没有仔细区分，可能需要再查看相应符号宏包的文档才能进一步了解它们的用法和注意事项。

xunicode 宏包重定义了 XeTeX 的 UTF-8 编码下（EU1 字体编码）的大量符号的命令，使得在新编码下，\textbullet 之类的命令也能正常工作，而 \'e 这样的复合重音标记也会自动转为字体中带重音的字母。xunicode 宏包会自动被 fontspec 或 ctex 宏包载入，但在旧的系统下可能需要手工完成。不过，当字体本身缺少需要的符号时，需要的符号就无法显示，即使使用的是 \'e 这样的复合形式，你也可能需要更换其他字体或切换编码（参见 2.1.3 节）来显示这些符号。

其实，使用 XeTeX 引擎时，输入特殊符号最简捷的办法就是在 UTF-8 编码下直接输入：

```
® © £ § ¶ † ‡ • ™ € ‰
```

® © £ § ¶ † ‡ • ™ € ‰

2-1-22

当然，与输入特殊字母一样，这种方式也要求输入法、编辑器字体和输出字体的共同支持。通过更换字体（参见 2.1.3 节），可以用输入任何可以找得到的符号。

即使并非使用 XeTeX 引擎，依然可能直接输入标准 ASCII 编码以外的特殊符号。这依然需要使用比 ASCII 更大的输入编码。inputenc 宏包[116] 允许使用诸如 ascii（默认值，ASCII）、latin1（ISO 8859-1）、utf8（UTF-8）等参数表示使用的输入编码。用这种方式也可以直接输入特殊字母与特殊符号，不过这里的 UTF-8 编码也只支持西方的拼音文字，无法支持汉字。

使用 XeTeX 或其他引擎直接指定字体输入特殊符号时，一个很大的问题是，并非所有符号都可以用键盘输入，即使借助特殊输入法、字符映射表等工具输入了这些符号，在源代码编辑器中仍然可能无法显示。为此，LaTeX 提供了命令 \symbol 来直接用符号在字体中的编码来输入符号。\symbol 命令带有一个参数，就是一个表示字符编码的数字。在 TeX 中数字除了普通的十进制，也可以用八进制、十六进制或编码对应的字符本身来表示，其语法形式如表 2.4 所示。其中字符形式中的字符如果是特殊字

符，需要在前面加 \ 转义，如用 \symbol{‘\%} 就得到% 本身，与 \% 类似。

表 2.4 TEX 中不同的数字表示形式

表示法	语法形式	例
十进制	⟨数字⟩	90
十六进制	"⟨数字⟩	"5A
八进制	’⟨数字⟩	’132
字符形式	‘⟨字符⟩	‘Z

要得到字符的编码，对于传统的 TEX 字体，可以参见字体的符号码表，许多字体宏包（如 txfonts）在文档中都列出了很长的字体符号码表，选定字体后，就可以利用 \symbol 命令输入用键盘难以输入的符号，例如：

2-1-23

```
\usefont{T1}{t1xr}{m}{n}
\symbol{"DE}\symbol{"FE}
```

Þþ

而对于 XeTEX 调用的 OpenType、TrueType 字体，一般编码均为 Unicode，可以查通用的 Unicode 码表或利用字符映射表等外部工具，查看符号的编码，输入对应的符号，例如：

2-1-24

```
\symbol{28450}\symbol{35486}
```

漢語

使用这种方式，就可以只使用 ASCII 编码的源文件，排版出任何字体中的任何符号。

2.1.3 字体

2.1.3.1 字体的坐标

当我们说“换一个字体”时，指的是什么呢？可能是想换掉文字的整体感觉，如 TEXT 与 TEXT；可能是想把直立的文字改为倾斜的，如 TEXT 与 *TEXT*；可能是想把细的文字改为粗的，如 TEXT 与 **TEXT**；可能是想把包含一些符号的字体改为包含另一些符号的，如 TEXT 与 θεχθ；还可能只是想改变字体的大小，如 TEXT 与 TEXT。——尽管所使用的文字内容都是一样的（TEXT）。

上面说的五种不同的性质，在 LATEX 中一起决定了文字的最终输出效果。字号（font size），即文字大小，常常被独立出来，看做不同于字体的单独的性质（参见 2.1.4 节）；

字体编码（font encoding），即字体包含的符号也较少直接设定。使用最多的是其他的三种性质，即字体族（font family）、字体形状（font shape）、字体系列[①]（font series）。

LaTeX 提供了带参数和命令和字体声明两类修改字体的命令，前者用于少量字体的更换，后者用于分组或环境中字体的整体更换。例如：

```
\textit{Italic font test}

{\bfseries Bold font test}
```

Italic font test
Bold font test

2-1-25

预定义命令的字体族有 3 种：罗马字体族（roman family）、无衬线字体族（sans serif family）和打字机字体族（typewriter family）。其命令为：

字体族	带参数命令	声明命令	效果
罗马	\textrm{⟨文字⟩}	\rmfamily	Roman font family
无衬线	\textsf{⟨文字⟩}	\sffamily	Sans serif font family
打字机	\texttt{⟨文字⟩}	\ttfamily	Typewriter font family

正文默认使用罗马字体族。字体族一般对应一组风格相似，适于一起使用的成套字体。

预定义命令的字体形状有 4 种：直立形状（upright shape，也称为 roman shape）、意大利形状（italic shape）、倾斜形状（slanted shape）、小型大写形状（small capitals shape）。其命令为：

字体形状	带参数命令	声明命令	效果（Latin Modern Roman 字体族）
直立	\textup{⟨文字⟩}	\upshape	Upright shape
意大利	\textit{⟨文字⟩}	\itshape	*Italic shape*
倾斜	\textsl{⟨文字⟩}	\slshape	*Slanted shape*
小型大写	\textsc{⟨文字⟩}	\scshape	SMALL CAPITALS SHAPE

正文默认使用直立字体形状。注意其中倾斜形状和意大利形状的区别，倾斜形状差不多是直接对符号倾斜产生的，而通常所说的“斜体”往往是指意大利形状，它类似于更加圆滑的手写体。因为数学公式中的字体一般使用意大利形状，因而与数学混排时倾斜形状不会与公式中的字母混淆；在标题、参考文献中也有使用倾斜形状的。并非所有字体族都有这么多种形状，除了 LaTeX 默认的 Computer Modern 和 Latin Modern，大多数字体都只有意大利形状与倾斜形状中的一种（比如本书使用的 Times 字体）；很多字体也可能缺少小型大写字母的符号。另一方面，一些其他字体可能也会提供更多的

① 通常指字体的重量（weight，即粗细）和宽度（width）。

字体形状，如 Venturis ADF 系列字体[206] 就提供倾斜的小型大写（italic small capitals）、空心（outline）等形状。

预定义命令的字体系列有中等（medium）和加宽加粗（bold extended）两类：

字体系列	带参数命令	声明命令	效果
中等	\textmd{⟨文字⟩}	\mdseries	Medium series
加宽加粗	\textbf{⟨文字⟩}	\bfseries	**Bold extended series**

正文默认使用中等字体系列。两个命令表示的意义对不同套的字体可能有所区别，如命令 \textbf 和 \bfseries 对默认的字体选择加宽加粗的字体系列，但对一些字体则是选择加粗（bold）或半粗（demi-bold）字体系列。

字体的这三种性质有如确定字体的三维坐标，同一维度内的性质不能重叠，但不同类的性质则可以叠加。三种性质的组合效果见表 2.5。

表 2.5 字体的坐标：族、形状和系列。字体族、形状和系列用声明形式的字体命令表示。这里以 Latin Modern 字体为例，这是 XeLaTeX 的字体默认值。表中实际共有 17 种不同的字体，如果使用其他的字体，表中的缺项可能会有所不同

字体族	\rmfamily		\sffamily		\ttfamily	
形状＼系列	\mdseries	\bfseries	\mdseries	\bfseries	\mdseries	\bfseries
\upshape	Text	**Text**	Text	**Text**	Text	**Text**
\itshape	*Text*	***Text***	同 *slanted*	同 ***slanted***	*Text*	同 ***slanted***
\slshape	*Text*	***Text***	*Text*	***Text***	*Text*	***Text***
\scshape	TEXT	缺	缺	缺	TEXT	缺

除了上面列举的这些字体命令，还有 \textnormal{⟨文字⟩} 和 \normalfont 命令用来把字体设置为“普通”的格式。默认情况下，普通字体相当于 \rmfamily \mdseries \upshape 的效果。普通字体特别适用于在复杂的字体环境中恢复普通的字体，尤其是在宏定义这类不知道外部字体设置的情况下，如：

2-1-26

```
\sffamily
\textbf{This is a paragraph of bold and
\textit{italic font, sometimes returning
to \textnormal{normal font} is necessary.}}
```

This is a paragraph of bold and *italic font, sometimes returning to* normal font *is necessary.*

使用斜体声明（\itshape、\slshape）时，最后一个倾斜的字母会超出右边界，使得后面的文字与它相距过紧，而用带参数的命令（\textit、\textsl）就可以自动修正这个距离，也可以手工使用 \/ 命令进行这种倾斜校正（italic correction），如：

```
{\itshape M}M

\textit{M}M

{\itshape M\/}M
```

*M*M
*M*M
*M*M

2-1-27

倾斜校正命令 \/ 会在字母后面加上一个小的距离，其大小由具体的字体和符号来决定[126, Chapter 4]。倾斜校正一般只用在声明形式的斜体命令，但偶尔也使用它取消连字（参见 2.1.1.1 节），也可以用来校正一些粗体字母和引号之间的距离（当然最好还是使用带参数的命令形式）：

```
Bold ‘{\bfseries leaf}’

Bold ‘{\bfseries leaf\/}’

Bold ‘\textbf{leaf}’
```

Bold ‘**leaf**’
Bold ‘**leaf**’
Bold ‘**leaf**’

2-1-28

在很少的情况下，\textit 自动加入的倾斜校正是不必要的，此时可以使用 \nocorr 命令禁止校正，如：

```
\textit{M}M

\textit{M\nocorr}M
```

*M*M
*M*M

2-1-29

也可以使用 \renewcommand 重定义 \nocorrlist 命令设置不对特定字符校正，LaTeX 默认定义不对句号和逗号校正，相当于已经定义了：

```
\newcommand\nocorrlist{,.}
```

中文字体通常没有西文字体那样复杂的成套的变体，各个字体之间一般都是独立的，只有少数字体有不同重量的成套字体。因此，对于中文字体，一般只使用不同的字体族进行区分。xeCJK 和 CJK 宏包机制下，中文字体的选择命令和西文字体是分离的，选择中文字体族使用 \CJKfamily 命令，如：

2-1-30

```
{\CJKfamily{hei}这是黑体}

{\CJKfamily{kai}这是楷书}
```

这是黑体
这是楷书

中文的字体族，根据不同的系统和使用方式各有不同。在 ctex 宏包及文档类下有一些预定义，在默认情况下（winfonts 选项）针对 Windows 常用字体配置了的四种字体族：song（宋体）、hei（黑体）、kai（楷书）、fs（仿宋）[①]；如果使用了其他选项，则可能会有不同的字体[②]，为了方便使用，ctex 宏包提供了简化的命令：

2-1-31

```
{\songti 宋体}  \quad {\heiti 黑体} \quad
{\fangsong 仿宋} \quad {\kaishu 楷书}
```

宋体 **黑体** 仿宋 楷书

ctex 宏包及文档类（如 ctexart）另外定义了一些组合字体，可以让中文也具备使用粗体（\bfseries）和意大利体（\itshape）的功能，并且重定义 \rmfamily 使它同时对中文起作用。默认的中文字体族是 rm，其正常字体是宋体，粗体是黑体，意大利体是楷体，如：

2-1-32

```
% ctex 宏包下默认相当于 \CJKfamily{rm}
% \rmfamily 或 \textrm 也会同时设置此字体
中文字体的\textbf{粗体}与\textit{斜体}
```

中文字体的**粗体**与*斜体*

类似地，\sffamily（对应 sf 中文字体族）和 \ttfamily（对应 tt 中文字体族）也可以同时作用于西文和中文，分别相当于幼圆和仿宋体。

LaTeX 的这种利用相互正交的几种性质来区分字体的方式，称为新字体选择方案[142]（New Font Selection Scheme, NFSS）。当然，NFSS 最早于 1989 年发布，现在也并不新了；它是针对 Plain TeX 和旧版本的 LaTeX 2.09 中直接指定具体字体旧方案而说的。在旧的字体选择方式中，一个字体命令对应一个单一的字体，甚至有些字体命令对应的字体的大小也是确定的，如在 Plain TeX 和旧的 LaTeX 2.09 格式中，命令 \rm, \bf, \it 分别表示使用 Computer Modern 的普通罗马字体、粗罗马字体和意大利体，但使用 \bf\it 并不能得到加粗的意大利体，而仍然只是一个 \it 的效果。NFSS 改变了

① 对 CJK 方式还有 li（隶书）以及 you（幼圆）。

② 本书描述 ctex 宏包 1.02c 版，使用 XeTeX 方式处理中文时的字体选择。除了默认的 winfonts 选项，还有 adobefonts 和 nofonts 选项。在 adobefonts 选项下有宋、黑、楷、仿宋四种字体可用；而 nofonts 选项则没有预定义的字体族和命令，只有自行定义后才能使用，参见 2.1.3.2 节。

这种不便的使用方式。现在，出于对旧文档的兼容考虑，现在的版本 LaTeX 2ε 在标准文档类中也保留了 \rm, \bf, \it, \sc, \sl, \sf, \tt 等命令（以及数学模式下的 \mit, \cal 命令），请不要在文档中使用它们。

NFSS 为字体划分了编码、族、系列、形状、尺寸等多个正交属性，这些属性各自可以用一个简短的符号来表示，如字体编码有 OT1, T1, OML, OMS, OMX, U 等；字体族有 cmr, cmss, cmtt, cmm, cmsy, cmex 等；字体系列有 m, b, bx, sb, c 等；字体形状有 n, it, sl, sc 等，由具体的字体可以有不同的定义，常用的标准定义可参见 NFSS 的标准文档 [142]、Adobe PostScript 字体文档 [229, § 8]。LaTeX 提供了更原始的命令：

\fontencoding⟨编码⟩
\fontfamily⟨族⟩
\fontseries⟨系列⟩
\fontshape⟨形状⟩
\fontsize⟨大小⟩⟨基本行距⟩　　（纯数字，单位是 pt）

通过这些命令来使用这些基本属性，需要在后面加 \selectfont 命令使它们生效，如：

```
\fontencoding{OT1}\fontfamily{pzc}
\fontseries{mb}\fontshape{it}
\fontsize{14}{17}\selectfont
PostScript New Century Schoolbook
```

PostScript New Century Schoolbook

2-1-33

也可以使用

\usefont⟨编码⟩⟨族⟩⟨系列⟩⟨形状⟩

一次性选择某个字体，如：

```
\usefont{T1}{pbk}{db}{n}
PostScript Bookman Demibold Normal
```

PostScript Bookman Demibold Normal

2-1-34

2.1.3.2　使用更多字体

使用 \rmfamily, \bfseries, \itshape 或 \kaishu 这类命令，只能选择预定义的少数几种字体。对于西文，就是 Computer Modern 或 Latin Modern 系列的几款成套的字体；对于中文，就是 6 种预定义的 Windows 字体。这些对于简单的文章并没有问题，

但如果想要改变文章的字体风格（For example, Palatino），或者缺少预设的字体文件（如中文 Linux 用户就不会有 Windows 预装的中文字体），那么就需要用到预设之外的更多字体。

选择非默认的字体有两类方法：当不使用 XeTeX 引擎时，可以通过字体宏包或 LaTeX 底层字体命令调用在 TeX 系统中预先安装好的字体；当使用 XeTeX 引擎时，可以调用操作系统安装的中西文字体。由于质量可靠的中文字体几乎都是商用的，因此 TeX 系统中只安装了很少几种质量较差的中文字体，一般总要调用操作系统所安装的字体来使用 XeTeX 引擎的功能。

一个 TeX 发行版通常都同时安装了大量西文字体，直接使用 LaTeX 底层命令（参见 2.1.3.1 节末尾）指定字体通常比较难用，特别是一些字体同时还包括对应的数学字体和符号命令的设置，非常烦琐，因此很多西文字体都做成了方便调用的字体宏包，可以直接更换整套的西文字体（或数学字体）。其中，最为常见的要求是把整套字体换为 Times Roman 的衬线字体（罗马体）或 Helvetica 的无衬线字体，Times 字体也能与中文宋体能很好地配合。有好几个宏包可以达到这个目的，最简单的是 times 宏包[229]，只更换正文字体，没有更换配套的数学字体，很少使用；mathptmx 在 times 宏包[229] 的基础上增加了数学字体的支持；效果最好的免费字体则是 txfonts 宏包[220]，对整套西文字体和数学符号给出了完整的解决方案。使用字体宏包非常简单，通常只要导入此宏包即可：

2-1-35

```
\documentclass{article}
\usepackage{txfonts}
\begin{document}
Test text
\end{document}
```

txfonts 的效果见图 2.1。

Theorem 1 (Cauchy's Theorem) Let f be holomorphic on a closed disc $\overline{D}(z_0, R)$, $R > 0$. Let C_R be the circle bounding the disc. Then f has a power series expansion

$$f(z) = \sum_{n=0}^{\infty} \frac{(z - z_0)^n}{2\pi \mathrm{i}} \int_{C_R} \frac{f(\zeta)}{(\zeta - z_0)^{n+1}} \mathrm{d}\zeta.$$

图 2.1 Times 字体效果（txfonts）

为文章的不同部分选用字体应该协调配合，因此通常带有配套数学字体的字体包是最为常用的，但有时也需要分别定义正文字体和与之配套的数学字体，此时就必须手工指定不同的字体包。例如使用高德纳的 Concrete 正文字体与 Zapf 的 Euler 数学字

体配合时（这正是 Concrete 字体设计时的用法[87]），就需要综合使用字体包 ccfonts 和 euler，效果见图 2.2：

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{ccfonts,eulervm}
\begin{document}
Test text
\end{document}
```

2-1-36

Theorem 1 (Cauchy's Theorem) Let f be holomorphic on a closed disc $\overline{D}(z_0, R)$, $R > 0$. Let C_R be the circle bounding the disc. Then f has a power series expansion

$$f(z) = \sum_{n=0}^{\infty} \frac{(z - z_0)^n}{2\pi i} \int_{C_R} \frac{f(\zeta)}{(\zeta - z_0)^{n+1}} d\zeta.$$

图 2.2　Concrete 和 Euler 字体效果（T1 编码，ccfonts, euler）

这一字体组合比较清晰，它与无衬线字体包（如 arev, cmbright 等）都比较适合于在幻灯片演示中使用。

在使用 Concrete 与 Euler 字体时，我们使用了一个新的宏包 fontenc 来选择字体的编码。fontenc 宏包可以包含多个选项，表示文档所使用的正文字体编码，最后一个选项的编码是文档默认使用的编码。字体的编码决定字体包含的符号，同一族的字体可能会有不同编码的版本。除了 XeLaTeX 使用的 Unicode 编码（在 NFSS 中一般是 EU1），传统的 LaTeX 字体编码有一般正文字体 OT1、扩展正文字体 T1、数学字母 OML、数学符号 OMS、数学符号扩展 OMX 等。需要手工设置的通常只有正文字体的编码，默认的正文字体编码是 OT1，而扩展编码 T1 则包含更多的符号，特别是带重音的拉丁字母等。许多字体包都要求使用 T1 编码，才能获得最好的效果，在字体包的说明文档中一般都会提及。注意字体编码是指符号在字体中位置的编码，与输入文档时使用的文档编码不是一回事。

初用 LaTeX 的最大的问题往往是并不知道 LaTeX 中有哪些字体可用，因为作为一门排版语言，LaTeX 并没有把可用的字体放在一个下拉菜单中等待选择，因此必须自己查看 TeX 发行版的字体安装目录[①]或综述性的字体文档。一个带有说明、例子、使用方法和详细功能比较的综述性字体文档是 Hartke [100]，里面介绍了 20 多种带有数学支持

① 一般是 TeX 安装目录下的 fonts 目录，不过除了 OpenType 和 TrueType 格式的字体，其他字体很不好认。Type 1 格式的字体名可参见 Berry [22]，但使用方法仍然需要另外查看说明。

的免费 TeX 字体。该文档在 TeX Live 和 CTeX 套装中都有收录，是使用成套字体的很好的参考。一个收录更为全面的 LaTeX 字体目录是 Jørgensen [118]，它展示了 Linux 源中的 TeX Live 所能使用的近 200 种西文字体族。

下面来看现代的方法，使用 XeTeX 来选择字体。在 XeLaTeX 中，主要使用 fontspec 宏包的机制来调用字体。最基本的是设置正文罗马字体族、无衬线字体族和打字机字体族的命令：

\setmainfont[⟨可选选项⟩]{⟨字体名⟩}
\setsansfont[⟨可选选项⟩]{⟨字体名⟩}
\setmonofont[⟨可选选项⟩]{⟨字体名⟩}
（可用的可选选项参见 fontspec 文档 Robertson and Hosny [212]）

例如：

```
% 在导言区设置全文字体为 Windows 提供的
% Times New Roman, Verdana, Courier New 字体
\usepackage{fontspec}
\setmainfont{Times New Roman}
\setsansfont{Verdana}
\setmonofont{Courier New}
```

2-1-37

此时 \rmfamily, \sffamily 和 \ttfamily 就分别对应设置的三种字体，而且 fontspec 会自动找到并匹配对应的粗体、斜体等变体，尽量使 \bfseries, \itshape 等命令也有效。

也可以定义新的字体族命令：

\newfontfamily⟨命令⟩[⟨可选选项⟩]{⟨字体名⟩}

例如为 Java 运行库附带的 Lucida Sans 字体定义一个命令 \lucidasans：

```
% 导言区使用
\newfontfamily\lucidasans{Lucida Sans}
% 正文使用
{\lucidasans This is Lucida Sans.}
```

2-1-38

This is Lucida Sans.

XeLaTeX 下中文字体的设置使用 xeCJK 宏包[310]（ctex 宏包或文档类会自动调用它）。xeCJK 提供了与 fontspec 对应的中文字体设置命令：

```
\setCJKmainfont[⟨可选选项⟩]{⟨字体名⟩}
\setCJKsansfont[⟨可选选项⟩]{⟨字体名⟩}
\setCJKmonofont[⟨可选选项⟩]{⟨字体名⟩}
\setCJKfamilyfont{⟨中文字体族⟩}[⟨可选选项⟩]{⟨字体名⟩}
```

这些命令对于 Linux 等系统下的中文用户特别有用，因为 ctex 宏包默认是针对 Windows 的字体配置的，可以用 nofonts 选项禁用预定义的中文字体设置而来自己定义字体。例如使用文鼎公司免费发布的字体：

```
% 在导言区设置全文字体为“文鼎ＰＬ报宋二GBK”
% 并设置中文的 kai 字体族为“文鼎ＰＬ简中楷”
\setCJKmainfont{AR PLBaosong2GBK Light}
\setCJKfamilyfont{kai}{AR PL KaitiM GB}
```

此后使用 \CJKfamily{kai} 就会使用文鼎的楷体。

fontspec 所能使用的字体是 fontconfig 库所能找到的所有字体，也就是在 TEX 发行版和操作系统中所安装的字体，通常是 OpenType 和 TrueType 的字体，也包括一些 PostScript 格式字体，需要注意的就是要使用正确的字体名。字体的列表可以在命令行下使用 fc-list 命令来列出，通常列出的字体信息会非常多，不能在一屏内完全显示，Windows 下还会因编码不统一而出现乱码，此时可以利用命令重定向操作符把结果输出到文件中：

```
fc-list > fontlist.txt
```

fc-list 命令输出的结果类似这样（这里只选取了一小部分，并稍做重新排列）：

```
Minion Pro:style=Bold
Minion Pro:style=Bold Italic
Minion Pro:style=Italic
Minion Pro:style=Regular
Times New Roman:style=cursiva,kurzíva,kursiv,Πλάγια,Italic,
  Kursivoitu,Italique,Dőlt,Corsivo,Cursief,kursywa,Itálico,
  Курсив,İtalik,Poševno,nghiêng,Etzana
Times New Roman:style=Negreta cursiva,tučné kurzíva,
  fed kursiv,Fett Kursiv,Έντονα Πλάγια,Bold Italic,
  Negrita Cursiva,Lihavoitu Kursivoi,Gras Italique,
  Félkövér dőlt,Grassetto Corsivo,Vet Cursief,Halvfet Kursiv,
```

```
  Pogrubiona kursywa,Negrito Itálico,Полужирный Курсив,
  Tučná kurzíva,Fet Kursiv,Kalın İtalik,Krepko poševno,
  nghiêng đậm,Lodi etzana
Times New Roman:style=Negreta,tučné,fed,Fett,Έντονα,Bold,
  Negrita,Lihavoitu,Gras,Félkövér,Grassetto,Vet,Halvfet,
  Pogrubiona,Negrito,Полужирный,Fet,Kalın,Krepko,đậm,Lodia
Times New Roman:style=Normal,obyčejné,Standard,Κανονικά,
  Regular,Normaali,Normál,Normale,Standaard,Normalny,Обычный,
  Normálne,Navadno,thường,Arrunta
宋体,SimSun:style=Regular
黑体,SimHei:style=Normal,obyčejné,Standard,Κανονικά,Regular,
  Normaali,Normál,Normale,Standaard,Normalny,Обычный,
  Normálne,Navadno,Arrunta
文鼎ＰＬ报宋二GBK,AR PLBaosong2GBK Light:style=Regular
文鼎ＰＬ简中楷,AR PL KaitiM GB:style=Regular
```

fc-list 列出的是字体名（对应于字体族）和同族字体变体（对应于 LaTeX 的字体系列和形状）。在冒号前面的是字体族名称，style= 后面的是字体的变体，用逗号分隔开的是同一个字体（或变体）在不同语言下的名称。如这里的 Minion Pro 字体就有 Regular（对应于 \mdseries\upshape）、Italic（对应于 \mdseries\itshape）、Bold（对应于 \bfseries\upshape）、Bold Italic（对应于 \bfseries\itshape）这 4 种变体，而几种中文字体都没有变体。通常 fontspec 宏包会自动处理好字体的变体，因此只要知道前面的字体名称。

fc-list 可以带许多参数来控制输出的格式。例如，我们通常只需要字体族名，而不需要变体的名称，就可以加 -f "%{family}\n" 选项只输出字体族；又如，使用 :lang=zh 选项可以只输出中文字体，而 :outline 则可以只显示矢量字体。下面的命令就可以将所有中文字体族的列表输出到 zhfont.txt 中：

```
fc-list -f "%{family}\n" :lang=zh > zhfont.txt
```

fontspec 宏包为字体，尤其是 OpenType 字体提供了多种字体性质的选项。例如 Minion Pro 字体使用带斜线的数字 0，以与字母 O 区分：

```
\newfontfamily\minion[Numbers=SlashedZero]{Minion Pro}
\minion 100, OK.
```

100, OK.

2-1-39

字体选项的选择可参见 Goossens and Rahtz [86]、Robertson and Hosny [212]，OpenType 字体提供的性质可通过命令行工具 otfinfo 或 opentype-info 宏包查看。

对中文字体来说，尤其有用的一组 fontspec 命令选项是设置斜体、粗体、粗斜体的选项 ItalicFont, BoldFont, BoldItalicFont，这几个选项可以把字体的变体用另一种字体来代替。如可以设置正文字体为宋体，其粗体为黑体、斜体为楷体、粗斜体为隶书，以弥补中文字体一般没有一族成套变体的问题：

```
\setCJKmainfont[BoldFont=SimHei,ItalicFont=KaiTi,
  BoldItalicFont=LiSu]{SimSun}
```

ctex 宏包默认设置 Windows 字体时正是采用类似的方法。如果不指定中文字体的变体形式，ctex 调用 xeCJK 时会增加选项使用伪粗体和伪斜体代替。不过中文排版没有斜体的概念，因此在非 Windows 环境一般也都需要设置这种中文的复合字体，避免斜体的使用。

fontspec 主要用于设置正文字体，通常不用来设置数学字体，不过一些数学字体（如 \mathrm）默认与正文字体一致，则正文字体的设置会影响到数学字体。fontspec 的 \setmathrm, \setmathsf, \setmathtt 等命令可以用来以与正文同样的方式设置这些受影响的数学字体，它们的用法和 \setmainfont 等命令类似。此外，fontspec 宏包的 no-math 选项可以禁用其对数学字体的所有影响（包括 \setmahtrm 等命令的作用），在加载许多数学字体包时，这个选项都会自动生效，对于不自动加载 no-math 选项的宏包，我们可以自己加上这个选项。为了使用正确的字体编码，fontspec 应该放在数学字体包的后面。当然，fontspec 几乎总会令字体宏包原来的正文字体设置失效，可以使用 fontspec 的方式再另行设置搭配的字体，例如：

```
\documentclass{article}
\usepackage{ccfonts}% 公式使用 Concrete 系列字体
\usepackage[no-math]{fontspec}% \mathrm 等也使用 Concrete 系列字体
\setmainfont{Latin Modern Mono Prop}% 正文使用 Ladin Modern Mono 字体
```

2-1-40

处理中文时的情况可能更复杂一些。如果使用 xeCJK 或 ctex/ctexcap 宏包，可以在它们之前使用数学字体包，需要时可以带 no-math 选项加载 fontspec，用法和前面西文的文档类似。使用 ctex 中文文档类时，如果需要，fontspec 的 no-math 选项可以传递给文档类。当 fontspec 是由文档类加载时，就不可能在这之前加载数学字体包了，但由于个别字体包会改变字体编码，此时可能需要在数学字体包之后以 EU1 编码为最后一个选项加载 fontenc 宏包，以保证使用正确的字体编码，例如：

```
\documentclass[no-math]{ctexart}
\usepackage[utopia]{mathdesign}% 数字字体使用 mathdesign 与 Utopia
```

```
\usepackage[EU1]{fontenc}% 恢复正文字体的 Unicode 编码
\setmainfont{Utopia Std}% 正文字体使用 OpenType 格式的 Adobe Utopia
```

2-1-41

在使用数学字体包时同时加载对应的 OpenType 或 TrueType 格式的正文字体，是在 XeLaTeX 下使用复杂字体的一般方法。

一种更方便的方式是使用传统的字体包设置全文的字体（数学或正文字体），只把其他字体（如汉字）留给 fontspec 和 xeCJK。也就是说，我们要将旧的字体选择机制和新的字体选择机制混合使用，这同样可以做到，与前面的区别仅仅是使用 fontenc 宏包将传统的非 Unicode 字体编码设置为默认的编码，即最后一个选项[①]。这种方法不影响使用 fontspec 中 \newfontfamily 声明的字体和 xeCJK 声明的汉字字体。但 \setmainfont, \setsansfont 和 \setmonofont 这三个命令会因字体编码而失效，需要时在文档中使用 \fontencoding 命令临时切换编码，或者直接重定义 \rmfamily、\sffamily 和 \ttfamily，让它们增加切换编码的功能。下面给出一个在设置字体方面比较复杂的例子，以西文的 T1 编码为默认编码调用 fourier 字体包，西文用 fontspec 的两种方式定义其他字体，汉字也使用 OpenType 的字体：

```
\documentclass[UTF8]{ctexart}
% 西文正文和数学字体
\let\hbar\relax % 解决 xunicode 与 fourier 的符号冲突
\usepackage{fourier}
% 设置默认编码为 T1，以支持 fourier 宏包
\usepackage[T1]{fontenc}
% 定义新的西文 Times 字体族
\newfontfamily\times{Times New Roman}
% 设置西文等宽字体，并重定义 \ttfamily 来切换到 EU1 编码
\setmonofont{Consolas}
\let\oldttfamily\ttfamily
\def\ttfamily{\oldttfamily\fontencoding{EU1}\selectfont}
% 设置中文字体
\setCJKmainfont{Adobe Kaiti Std}
\begin{document}
Utopia text and $\sum math$ fonts.
```

① 这种方法在旧版本的 fontspec 宏包中会失效，如果遇到问题，最好更新 TeX 系统。

```
汉字楷书与 {\times Times New Roman} 字体。

\texttt{Consolas 0123}
\end{document}
```

2-1-42

XeLaTeX 通常不影响符号字体包的使用，但偶尔会因为字体编码不同而造成问题；注意 LaTeX 在同一时刻只能使用一种字体编码，多种字体编码要手工切换。例如，通常由 fontspec 自动加载的 xunicode 宏包会把国际音标字体包 tipa[314] 的功能对应到 Unicode 编码字体上，但这会使原来的 \textipa 命令改用 fontspec 管理的 Unicode 编码字体。因此默认的正文字体 Latin Modern 因为符号不全就不能用于国际音标输出，而应该改用包含音标符号的字体，如 Linux Libertine O、Times New Roman 等。例如：

```
% XeLaTeX 编译
\documentclass[UTF8]{ctexart}
% 不需要 tipa 宏包，xunicode 已经实现其功能
\setmainfont{CMU Serif} % Computer Modern Roman 的 Unicode 版本
\begin{document}
\LaTeX{} 读音为 \textipa{["lA:tEx]}。
\end{document}
```

2-1-43

如果不愿意使用 OpenType 或 TrueType 格式的 Unicode 编码国际音标字体，仍然可以在 XeLaTeX 下使用 T3 编码的 tipa 的功能：

```
% XeLaTeX 编译
\documentclass[UTF8]{ctexart}
\usepackage{tipa}% 宏包已经正确加载 fontenc
% \mytipa 的定义参考原来的 \textipa 的旧定义，手工切换编码
\newcommand\mytipa[1]{{\fontencoding{T3}\selectfont#1}}
\begin{document}
\LaTeX{} 读音为 \mytipa{["lA:tEx]}。
\end{document}
```

2-1-44

有时调用一个字体并没有对应的字体包，就必须通过文档了解字体在 NFSS 下的坐标，手工进行选定。在 2.1.3.1 节中我们已经见到了这一点，如使用 Computer Modern 的 Fibonacci 字体[169]，就可以直接设置字体族：

```
\fontfamily{cmfib}\selectfont
Computer Modern Fibonacci Roman
```

Computer Modern Fibonacci Roman

2-1-45

而如果要将 Fibonacci 字体设置为全文的默认字体，并且让 \rmfamily 和 \textrm 都指向 Fibonacci 字体，就需要在导言区重定义默认的罗马字体族 \rmdefault：

2-1-46
```
% 导言区修改全文默认字体 Computer Modern Fibonacci 字体族
\renewcommand\rmdefault{cmfib}
```

类似地，可以重定义 \sfdefault 和 \ttdefault 来修改 \rmfamily 和 \ttfamily 表示的字体族。进一步，如果希望让全文的默认字体是无衬线的字体族，那么还可以重定义 \familydefault，如：

2-1-47
```
% 导言区修改全文默认字体为无衬线字体族 phv (Helvetica)
\renewcommand\familydefault{\sfdefault}
\renewcommand\sfdefault{phv}
```

这正是 helvet 宏包的主要工作。当然，在有对应宏包的情况下，还是应该尽量使用宏包提供的功能，这样可能有针对特定字体的更详细的设置。

类似地，较新版本的 xeCJK 也提供了 \CJKrmdefault, \CJKsfdefault, \CJKttdefault 和 \CJKfamilydefault 命令，其用法与 NFSS 中的几个命令类似。在排版中文文档时，如果重定义了 \familydefault 设置全文为无衬线字体族，那么也应该同时设置中文：

2-1-48
```
\renewcommand\CJKfamilydefault{\CJKsfdefault}
```

最后介绍一个有用的宏包 fonttable[290]，可以用它输出字体的符号表，这对于使用 \symbol 命令时查找符号代码特别有用。在导言区使用

```
\usepackage{fonttable}
```

之后，就可以使用命令

\fonttable{⟨原始字体名⟩}
\xfonttable{⟨编码⟩}{⟨族⟩}{⟨系列⟩}{⟨形状⟩}

得到字体的符号表。例如，我们可以列举出 OT1 编码下 Latin Modern Roman 的字体符号表：

2-1-49
```
\xfonttable{OT1}{lmr}{m}{n}
```

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7	″0x
´01x	Φ 8	Ψ 9	Ω 10	ﬀ 11	ﬁ 12	ﬂ 13	ﬃ 14	ﬄ 15	
´02x	ı 16	ȷ 17	` 18	´ 19	ˇ 20	˘ 21	¯ 22	° 23	″1x
´03x	¸ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31	
´04x	˗ 32	! 33	” 34	# 35	$ 36	% 37	& 38	’ 39	″2x
´05x	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47	
´06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	″3x
´07x	8 56	9 57	: 58	; 59	¡ 60	= 61	¿ 62	? 63	
´10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	″4x
´11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79	
´12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	″5x
´13x	X 88	Y 89	Z 90	[91	“ 92	] 93	ˆ 94	˙ 95	
´14x	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	″6x
´15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111	
´16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	″7x
´17x	x 120	y 121	z 122	– 123	— 124	˝ 125	˜ 126	¨ 127	
´20x	Ă 128	Ą 129	Ć 130	Č 131	Ď 132	Ě 133	Ę 134	Ğ 135	″8x
´21x	Ĺ 136	Ľ 137	Ł 138	Ń 139	Ň 140	Ő 141	Ő 142	Ŕ 143	
´22x	Ř 144	Ś 145	Š 146	Ş 147	Ť 148	Ţ 149	Ű 150	Ů 151	″9x
´23x	Ÿ 152	Ź 153	Ž 154	Ż 155	IJ 156	İ 157	đ 158	§ 159	
´24x	ă 160	ą 161	ć 162	č 163	ď 164	ě 165	ę 166	ğ 167	″Ax
´25x	í 168	ı 169	ł 170	ń 171	ň 172	ŋ 173	ő 174	ŕ 175	
´26x	ř 176	ś 177	š 178	ş 179	ť 180	ţ 181	ű 182	ů 183	″Bx
´27x	ÿ 184	ź 185	ž 186	ż 187	ij 188	· 189	‚ 190	£ 191	
´30x	À 192	Á 193	Â 194	Ã 195	Ä 196	Å 197	« 198	Ç 199	″Cx
´31x	È 200	É 201	Ê 202	Ë 203	Ì 204	Í 205	Î 206	Ï 207	
´32x	Đ 208	Ñ 209	Ò 210	Ó 211	Ô 212	Õ 213	Ö 214	» 215	″Dx
´33x	‰ 216	Ù 217	Ú 218	Û 219	Ü 220	Ý 221	Þ 222	SS 223	
´34x	à 224	á 225	â 226	ã 227	ä 228	å 229	_ 230	ç 231	″Ex
´35x	è 232	é 233	ê 234	ë 235	ì 236	í 237	î 238	ï 239	

'36x	ð 240	ñ 241	ò 242	ó 243	ô 244	õ 245	ö 246	÷ 247	"Fx
'37x	∅ 248	ù 249	ú 250	û 251	ü 252	ý 253	þ 254	„ 255	
	"8	"9	"A	"B	"C	"D	"E	"F	

2.1.3.3 强调文字

现在回到我们最初认识的第一个字体命令：\emph。\emph 命令表示强调，用于把直立体改为意大利体，把意大利体改为直立体：

2-1-50

```
You \emph{should} use fonts carefully.

\textit{%
You \emph{should} use fonts carefully.}
```

You *should* use fonts carefully.
You should *use fonts carefully.*

与其他字体命令一样，\emph 也有一个声明形式，可以用在分组或环境中：

2-1-51

```
This is {\em emphasized\/} text.
```

This is *emphasized* text.

但注意此时要在合适的地方使用倾斜校正命令 \/。

在西文中通常使用意大利体表示夹在正文中的强调词句[35]，这种轻微的字体变化不像粗体那样显眼突兀，因而与正文可以良好的切合。不过，有时仍然使用大写、小型大写或粗体进行更醒目的强调，比如在参考文献的一些项目、书籍索引中的部分页码，或其他类似的内容。假设我们需要用粗体表示比 \emph 更强烈的强调，就可以为此定义一个新的 \Emph 命令，实际内容就是 \textbf：

2-1-52

```
\newcommand\Emph{\textbf}
This is \Emph{emphasized} text.
```

This is **emphasized** text.

下画线是另一种颇具手稿风格的强调方式，LaTeX 命令 \underline 可以给文字或公式加下画线：

2-1-53

```
\underline{Emphasized} text and
\underline{another}.
```

Emphasized text and another.

不过 \underline 的一个很大的缺点是下画线的部分不能换行，如果仔细看上面的例子还会发现下画线与文字的距离不整齐。ulem 宏包[17] 的 \ulem 命令解决了这些问题，使用并且把默认的 \emph 命令也改为使用下画线的方式：

```
% 导言区用 \usepackage{ulem}
\uline{Emphasized} text and \uline{another}.

A \emph{very very very very very very very
very very very very very} long sentence.
```

Emphasized text and another.
A very very very very very very very very very very very very long sentence.

2-1-54

如果不希望用下画线代替标准的 \emph 命令定义，可以给 ulem 宏包加 normalem 参数，或使用 \normalem 和 \ULforem 命令切换两种强调。

除了下画线，ulem 宏包也提供了其他一些修饰文字的命令：

```
\uuline{urgent}\qquad \uwave{boat}\qquad
\sout{wrong}\qquad \xout{removed}\qquad
\dashuline{dashing}\qquad \dotuline{dotty}
```

urgent　boat　~~wrong~~　~~removed~~　dashing　dotty

2-1-55

CJKfntef 宏包[244] 对汉字也提供了类似的功能，同时也进行了一些扩充[①]：

```
\CJKunderdot{汉字，下加点}\\
\CJKunderline{汉字，下画线} \\
\CJKunderdblline{汉字，下画线} \\
\CJKunderwave{汉字，下画线} \\
\CJKsout{汉字，删除线}\\
\CJKxout{汉字，删除线}
```

汉字，下加点
汉字，下画线
汉字，下画线
汉字，下画线
~~汉字，删除线~~
~~汉字，删除线~~

2-1-56

此外，CJKfntef 还提供了指定宽度，让汉字分散对齐的环境：

```
\begin{CJKfilltwosides}{5cm}
汉字，分散对齐
\end{CJKfilltwosides}
```

汉　字，　分　散　对齐

2-1-57

使用 CJKfntef 宏包后 \emph 命令也被改为下画线的格式，同样可以用 \normalem 改回原来的意大利体定义。在 ctex 宏包及文档类中，可以使用 fntef 选项调用 CJKfntef，此时 \emph 的定义不会被改变为下画线格式。同时也可以使用 \CTEXunderline 等以 \CTEX 开头的命令代替以 \CJK 开头的命令，如：

① 新版本的 CJKfntef 还提供了自定义符号和自定义距离的功能，参见其源代码。

2-1-58

```
\emph{汉字，强调}\\
\CTEXunderdot{汉字，加点}
```

汉字，*强调*
汉字，加点

练 习

2.3 soul 宏包[81] 提供了 \hl 命令实现对参数的强调，当没有彩色时它与 ulem 一样用下画线强调，而有彩色支持时它的效果则是荧光高亮显示。但是，soul 宏包的实现机制较为脆弱，与现在各种中文支持方式都无法一起正常使用。查看文档 Arseneau [17]，试利用 ulem 宏包实现用黄色高亮强调的命令 \hl。（提示：利用下画线）

从 METAFONT 到 OpenType

使用各种字体是任何排版软件都应具备的重要功能。今天 TEX 引擎可以使用很多格式的字体，如 METAFONT, PostScript, TrueType, OpenType 等，这些又都是些什么呢？

高德纳教授最初设计 TEX 系统时，也同时为 TEX 设计了配套的字体格式和字体描述语言，这就是 METAFONT。TEX 默认 Computer Modern 系列字体就是使用 METAFONT 描述的。METAFONT，顾名思义，就是“元字体”，它是一门通过曲线路径描述字体信息的宏语言[125]，把字符用坐标绘图的方式“画出来”。例如 cryst 字体中的一个箭头符号 → 就是用下面的代码画出来的：

```
beginchar(9,120u#,68u#,0);
"2, 0 Grad";
z1=(0u,37u); z2=(88u,37u); z3=(88u,31u); z4=(0,31u);
z5=(116u,34u); z6=(73u,17u); z7=(72u,19u);
z8=(84u,34u); z9=(72u,49u); z10=(73u,51u);
fill z10{dir -28}..{dir -15}z5..z5{dir -165}..{dir -152}z6
     --z7--z8--z9--cycle;
fill z1--z2--z3--z4--cycle;
labels(range 1 thru 6);
endchar;
```

METAFONT 使用坐标作图的命令描述字符的曲线，然后把曲线计算为三次 Bézier 样条，再转换为光栅格式的通用字体文件（generic font）.gf，同时生成 TEX 字体度量文件（TEX font metric）.tfm。TEX 引擎从 .tfm 文件中读

入字符尺寸信息，生成 DVI 文件；驱动程序在打印输出或屏幕显示时从 .gf 文件（一般转换为压缩的 .pk 文件）中读入字符图形信息，生成最终打印或显示的结果。尽管 METAFONT 在描述字符时使用的是曲线方式，但生成的结果却是光栅点阵形式的，在字体放大和使用电子文档时质量不够好；而且这种用数学坐标的方式描述符号很不直观，因此，现在 METAFONT 格式的字体使用得越来越少了。倒是 METAFONT 这种绘图方式，催生了绘图语言 METAPOST 的产生。

Adobe 公司推出 PostScript 时，也定义了 PostScript 字体的格式，Type 1 和 Type 3 是其中的两个类型[4]。很多高质量的商业字体都是 PostScript Type 1 格式的，它支持字体信息的微调（hinting），后来当 Adobe 开放 Type 1 格式的使用后，TEX 中原来用 METAFONT 描述的字体也都纷纷转换为 Type 1 字体了。Type 3 格式不支持微调，不过也可以表示点阵字体，当输出 PostScript 或 PDF 文件时，METAFONT 生成的 PK 字体就被转换为 Type 3 格式。Type 1 字体一般使用 .pfb 后缀作为字体扩展名，用 .afm 作为字体度量文件的扩展名，不过 TEX 使用时仍然要使用转换得到的 .tfm 度量文件。目前输出 PostScript 或 PDF 文件的 TEX 驱动和引擎都支持 PostScript 字体。

TrueType 是苹果公司与 Adobe 竞争而于 1991 年发布的字体格式，微软得到授权后，它已成为 Windows 操作系统中最为常见的字体格式；OpenType 是微软公司和 Adobe 公司于 1996 年发布的字体格式，它继承了 PostScript 字体的许多特性。TrueType 字体以 .ttf 和 .ttc 为后缀，OpenType 字体以 .otf 和 .ttf 为后缀。新的驱动和引擎 dvipdfmx, pdfTEX, XETEX 和 LuaTEX 都支持 TrueType 和 OpenType 字体格式，不过 dvipdfmx 和 pdfTEX 使用 TrueType 和 OpenType 时仍然需要事先生成 .tfm 文件并进行配置，功能上也有一些限制，而 XETEX 和 LuaTEX 则支持直接读取系统字体及全部 OpenType 字体特性。

2.1.4　字号与行距

字号指文字的大小，它原本是字体的性质，也被 NFSS 作为字体的坐标之一。例如 Computer Modern Roman 字体族的中等直立体就有 5, 6, 7, 8, 9, 10, 12, 17 点共 8 种大小的不同字号，其中点（point, pt）是长度单位，为 1/72.27 英寸（inch）。不过当可缩放的矢量字体流行起来以后，字体只有一款，通过缩放达到不同的尺寸，字号也就经常作为独立于字体的单独性质了。

基本的 LaTeX 提供了 10 个简单的声明式命令调整文字的大小：

\tiny	tiny	\Large	larger
\scriptsize	script size	\LARGE	even larger
\footnotesize	footnote size	\huge	huge
\small	small	\Huge	largest
\normalsize	normal size		
\large	large		

例如：

2-1-59

```
The text can be {\Large larger}.
```

The text can be larger.

字号命令表示的具体尺寸随着所使用的文档类和大小选项的不同而不同（见表 2.7）。在标准 LaTeX 文档类 article, report 和 book 中，可以设置文档类选项 10pt, 11pt 和 12pt，全局地设置文档的字号，默认为 10pt，即 \normalsize 的大小为 10 pt，这个数值表示字体中一个 \quad 的长度，通常也就是整个符号所占盒子的高度（对汉字来说，一般也相当于汉字宽度）。

上述字号命令除了会修改文字大小外，同时对 \normalsize、\small 和 \footnotesize 也会修改文字的基本行距（默认是文字大小的 1.2 倍）、列表环境的各种间距（参见 2.2.3.3 节）、显示数学公式的垂直间距（参见 4.5.3 节）。因此这些命令比原始的 \fontsize、\zihao 更方便使用。

中文字号可以使用同样的命令设置（字号命令不区分中西文）。不过为了明确字号的具体大小，也可以使用 ctex 宏包或 ctexart 等文档类提供的 \zihao 命令设置。\zihao 命令带一个参数，表示中文的字号，它影响分组或环境内命令后面所有的文字。\zihao 命令可用的参数见表 2.6。

类似地，ctexart, ctexrep 和 ctexbook 文档类也提供了两个选项 c5size 和 cs4size 影响全文的文字大小和 \normalsize 等命令的意义（见表 2.7）。c5size 和 cs4size 分别表示 \normalsize 为五号字和小四号字，默认为 c5size。

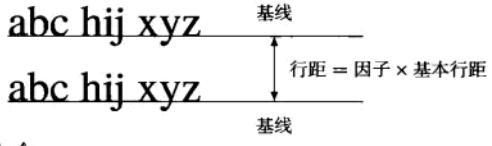
LaTeX 中的行距是与字号直接相关的。在设置字号时，同时也就设置了基本行距为文字大小的 1.2 倍。行距一词指的是一行文字的基线（base line）到下一行文字的基线的距离。

表 2.6　中文字号

命令	大小（bp）	意义
\zihao{0}	42	初号
\zihao{-0}	36	小初号
\zihao{1}	26	一号
\zihao{-1}	24	小一号
\zihao{2}	22	二号
\zihao{-2}	18	小二号
\zihao{3}	16	三号
\zihao{-3}	15	小三号
\zihao{4}	14	四号
\zihao{-4}	12	小四号
\zihao{5}	10.5	五号
\zihao{-5}	9	小五号
\zihao{6}	7.5	六号
\zihao{-6}	6.5	小六号
\zihao{7}	5.5	七号
\zihao{8}	5	八号

表 2.7 不同文档类选项下的字号命令。这里给出不同字号命令对应的字号尺寸。正文的默认字号是 \normalfont。西文文档类默认选项是 10pt，字体尺寸以 pt 为单位；ctex 文档类的默认选项是 c5size，使用中文字号

命令	10pt	11pt	12pt	c5size	cs4size
\tiny	5	6	6	七	小六
\scriptsize	7	8	8	小六	六
\footnotesize	8	9	10	六	小五
\small	9	10	10.95	小五	五
\normalsize	10	10.95	12	五	小四
\large	12	12	14.4	小四	小三
\Large	14.4	14.4	17.28	小三	小二
\LARGE	17.28	17.28	20.74	小二	二
\huge	20.74	20.74	24.88	二	小一
\Huge	24.88	24.88	24.88	一	一



可以使用命令

\linespread{⟨因子⟩}

来设置实际行距为基本行距的倍数[①]。与许多其他底层字体命令一样，它也在 \selectfont 命令（或等效的字体变更）后生效。对 article 等标准文档类来说，默认值为 1，即行距是字号大小的 1.2 倍；而对 ctexart 等中文文档类来说，默认值为 1.3，即行距是字号大小的 1.56 倍。

setspace 宏包[265] 提供了一组命令和环境，用于在修改行距因子的同时保证数学公式、浮动体、脚注间距的值也相对合理。基本的命令是 \setstretch{⟨因子⟩}，大致相当于使用了 \linespread 后再加 \selectfont 生效。除此之外，setspace 也提供了几个环境用来产生宏包定义的单倍、一倍半和双倍行距，不过要注意其“倍数”的意义并不是 \setstretch 的行距因子，而是指行距相比字号尺寸的倍数，这也与其他一些软件（如微软的字处理器 Word[②]）不同，见表 2.8。

① 一些较早的书籍会使用重定义 \baselinestretch 命令的方式设置行距因子，这种方法并非 LaTeX 2ε 推荐的方式，我们这里使用更容易掌握的方式。

② Word 中段落设置 *n* 倍行距的概念与标准 LaTeX 的 \linespread 是一样的，也是加之于基本行距的因子。

表 2.8　setspace 提供的命令和环境

命令形式	环境形式	意义
\setstretch	spacing	与 \linespread 功能相当
\singlespacing	singlespace	正常行距
\onehalfspacing	onehalfspace	基线间距是字号大小的 1.5 倍
\doublespacing	doublespace	基线间距是字号大小的 2 倍

可以使用 2.1.3.1 节的 \fontsize 直接指定文字的字号和基本行距，注意这里字号和基本行距是 NFSS 坐标中的一部分，只有在使用了 \selectfont 后才能生效。

除了设置两行基线间的距离，TEX 还提供了两个基本尺寸，用来设置两行文字内容间的距离。TEX 中行距默认由基线间的距离 \baselineskip 控制（LATEX 的 \fontsize 和 \linespread 间接控制 \baselineskip）。\lineskiplimit 是一个界限值，当前一行盒子的底部与后一行盒子的顶部距离小于这个界限时，行距改由 \lineskip 控制，它将设置行距，使得前一行底到后一行顶的距离等于 \lineskip。在文档中设置合适的 \lineskiplimit 和 \lineskip 值可以避免两行因包含太高的内容（如分式 $\frac{1}{2}$）而距离过紧，如本书的设置为：

```
\setlength{\lineskiplimit}{2.625bp} % 五号字 1/4 字高
\setlength{\lineskip}{2.625bp}
```

2-1-60

即要求两行间至少有 ¼ 五号字字号高度的距离。

2.1.5　水平间距与盒子

2.1.5.1　水平间距

现在我们来继续 2.1.1.3 节有关空格的话题，来说说如何正确地产生和使用水平间距。

说到长度，就不能不说说单位。在 TEX 中，可以使用长度单位有以下几种：

pt　point 点（欧美传统排版的长度单位，有时叫做“磅”）
pc　pica（1 pc = 12 pt，相当于四号字大小）
in　inch 英寸（1 in = 72.27 pt）

bp big point 大点（在 PostScript 等其他电子排版领域的 point 都指大点，1 in = 72 bp）
cm centimeter 厘米（2.54 cm = 1 in）
mm millimeter 毫米（10 mm = 1 cm）
dd didot point（欧洲大陆使用，1157 dd = 1238 pt）
cc cicero（欧洲大陆使用，pica 的对应物，1 cc = 12 dd）
sp scaled point（TEX 中最小的长度单位，所有长度都是它的倍数，65 536 sp = 1 pt）
em 全身（字号对应的长度，等于一个 \quad 的长度，也称为“全方”。本义是大写字母 M 的宽度）
ex x-height（与字号相关，由字体定义。本义是小写字母 x 的高度）

一个长度必须数字和单位齐全，正确的长度如下所示：

```
2cm    1.5pc    -.1cm    + 72.dd    1234567sp
```

在正文中可以使用下面的命令表示不可换行的水平间距：

\thinspace 或 \,	0.1667 em	→|←
\negthinspace 或 \!	-0.1667 em	→←
\enspace	0.5 em	→|←
\nobreakspace 或 ~	空格	→|←

可以使用下面的命令表示可以换行的水平间距：

\quad	1 em	→| |←
\qquad	2 em	→| |←
\enskip	0.5 em	→|←
\␣	空格	→|←

使用水平间距的命令要注意适用，例如 \, 是不可断行的，因而就不适用于分隔很长的内容，但用来代替逗号给长数字分段就很合适[①]：

```
1\,234\,567\,890
```

1 234 567 890

2-1-61

负距离 \! 则可以用来细调符号距离，或拼接两个符号，如把两个“剑号”拼起来：

① 当然，对于输出数字如果要更多的效果，还可以使用 fancynum, numprint 等宏包。使用这类专门的宏包通常总会更方便，效果也更好。

```
\newcommand\dbldag{\dag\!\dag}
\dbldag\ versus \dag\dag
```

‡‡ versus ††　2-1-62

而正如前面已经多次见到的，分隔词组经常使用可断行的而距离也比较大的 \quad 和 \qquad。

当上面的命令中没有合适的距离时，可以用 \hspace{距离} 命令来产生指定的水平间距（这里 \, 则用来分隔数字和单位）：

```
Space\hspace{1cm}1\,cm
```

Space　1 cm　2-1-63

\hspace 命令产生的距离是可断行的。\hspace 的作用是分隔左右的内容，在某些只有一边有内容的地方（如强制断行的行首），LaTeX 会忽略 \hspace 产生的距离，此时可以用带星号的命令 \hspace*{距离} 阻止距离被忽略：

```
text\\
\hspace{1cm}text\\
\hspace*{1cm}text
```

text
text
　　text　2-1-64

\hspace 可以产生随内容可伸缩的距离，即所谓胶（glue）或者橡皮长度（rubber length，又称弹性长度）。橡皮长度的语法是：

⟨普通长度⟩ **plus** ⟨可伸长长度⟩ **minus** ⟨可缩短长度⟩

单词间的空格、标点后的距离都是橡皮长度，这样才能保证在分行时行末的对齐，因此在定义经常出现的距离时应该使用橡皮长度。如：

```
\newcommand\test{longggggggg%
  \hspace{2em plus 1em minus 0.25em}}
\test\test\test\test\test\test\test\test
```

longggggggg　longggggggg
longggggggg　longggggggg
longggggggg　longggggggg
longggggggg　longggggggg　2-1-65

有一种特殊的橡皮长度 \fill，它可以从零开始无限延伸，此时橡皮长度就真的像是一个弹簧，可以用它来把几个内容均匀排列在一行之中：

```
left\hspace{\fill}middle%
\hfill right
```

left　middle　right　2-1-66

\hfill 命令是 \hspace{\fill} 的简写，还可以使用 \stretch{⟨倍数⟩} 产生具有指定“弹力”的橡皮长度，如 \stretch{2} 就相当于两倍的 \fill：

2-1-67
```
left\hspace{\stretch{2}}$2/3$%
\hspace{\fill}right
```
left 2/3 right

\hrulefill 和 \dotfill 与 \hfill 功能类似，只是中间的内容使用横线或圆点填充：

2-1-68
```
left\hrulefill middle\dotfill right
```
left middle ..... right

LaTeX 预定义了一些长度变量控制排版的参数，可以通过 \setlength 命令来设置。例如段前的缩进：

2-1-69
```
\setlength{\parindent}{8em}
Paragraph indent can be very wide in \LaTeX.
```
Paragraph indent can be very wide in LaTeX.

我们在后面的章节会陆续见到这类长度变量。还可以用 \addtolength 命令在长度变量上做累加，如：

2-1-70
```
Para\par
\addtolength{\parindent}{2em}Para\par
\addtolength{\parindent}{2em}Para\par
```
Para
Para
Para

长度变量的改变在当前分组或环境内起效，因此不必担心在一个环境内的设置会影响到外面的内容，也可以使用分组使变量的改变局部化。

可以用 \newlength 命令定义自己的长度变量，这样就可以在不同的地方反复使用：

```
\newlength\mylen  \setlength{\mylen}{2em}
```

这在自定义的一些环境中是十分有用的。

2.1.5.2 盒子

盒子（box）是 TeX 中的基本处理单位，一个字符、一行文字、一个页面、一张表格在 TeX 中都是一个盒子。回到活字印刷时代或许有助于理解盒子的概念：一个活字

就表示一个字符，一行活字排好就用钢条分隔固定成为一行，一整页排完也固定在金属框内。TEX 也是这样，组字成行，组行为页，小盒子用胶粘连成为大盒子，逐步构成完整的篇章。

TEX 可以处在不同的工作模式下，在不同的工作模式下产生不同的盒子。最基本的模式的水平模式（如在组字成行的时候）和垂直模式（如在组行成页的时候），水平模式下把小盒子水平排列组成大盒子，垂直模式下把小盒子垂直排列组成大盒子；此外还有更为复杂的数学模式，此时小盒子会构成复杂的数学结构。通常 TEX 根据内容自动切换不同的模式，完成这些复杂的工作；不过也可以使用命令进入指定的模式生成盒子，我们现在只看最简单的水平模式下的盒子。

最简单的命令是 \mbox{内容}，它产生一个盒子，内容以左右模式排列。可以用它表示不允许断行的内容，如果不在行末看不出它与其他内容的区别：

```
\mbox{cannot be broken}
```

cannot be broken　2-1-71

\makebox 与 \mbox 类似，但可以带两个可选参数，指定盒子的宽度和对齐方式：

\makebox[⟨宽度⟩][⟨位置⟩]{⟨内容⟩}

对齐参数可取 **c**（中）、**l**（左）、**r**（右）、**s**（分散），默认居中。

例如：

```
\makebox[1em]{\textbullet}text \\
\makebox[5cm][s]{some stretched text}
```

• text
some　stretched　text　2-1-72

甚至可以使用 \makebox 产生宽度为 0 的盒子，产生重叠（overlap）的效果：

```
% word 左侧与盒子基点对齐
\makebox[0pt][l]{word}文字
```

文字word

不过，LATEX 已经提供了两个命令来专门生成重叠的效果，即 \llap 和 \rlap。这两个命令分别把参数中的内容向当前位置的左侧和右侧重叠：

```
语言文字\llap{word}\\
\rlap{word}语言文字
```

语言文字word　2-1-73
语言文字word

命令 \fbox 和 \framebox 产生带边框的盒子，语法与 \mbox 和 \makebox 类似：

2-1-74
```
\fbox{framed} \\
\framebox[3cm][s]{framed box}
```
framed
framed box

边框与内容的距离由长度变量 \fboxsep 控制（默认为 3 pt）：

2-1-75
```
\setlength{\fboxsep}{0pt} \fbox{tight}
\setlength{\fboxsep}{1em} \fbox{loose}
```
tight loose

边框线的粗细则由长度变量 \fboxrule 控制（默认为 0.4 pt）。

可以用 \newsavebox{命令} 声明盒子变量，用

\sbox{命令}{⟨内容⟩}
\savebox{命令}[⟨宽度⟩][⟨对齐⟩]{⟨内容⟩}
\begin{lrbox}{⟨命令⟩} ⟨内容⟩ \end{lrbox}

给盒子变量赋值，在文章中使用 \usebox{内容} 反复使用：

2-1-76
```
\newsavebox{\mybox} % 通常在导言区定义
\sbox{\mybox}{test text}
\usebox{\mybox} \fbox{\usebox{\mybox}}
```
test text test text

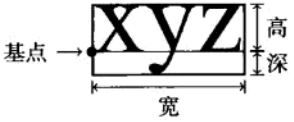
\savebox 与 \sbox 的区别类似 \makebox 与 \mbox 的区别，基本功能相同，只是增加了宽度和对齐的选项。

盒子变量中一般保存比较复杂的内容。特别是可以使用 lrbox 环境保存抄录环境（参见 2.2.5 节）等难以放在其他命令参数中的内容作进一步的处理[①]（见 124 页例 2-2-73）：

2-1-77
```
\newsavebox{\verbbox} % 通常在导言区定义
\begin{lrbox}{\verbbox}
\verb|#$%^&{}|
\end{lrbox}
\usebox{\verbbox} \fbox{\usebox{\verbbox}}
```
#$%^&{} #$%^&{}

① 不过，这个功能可以直接使用 fancyvrb 宏包的 \SaveVerb 和 \UseVerb 命令。

TEX 中的每个盒子都有其宽度（width）、高度（height）和深度（depth），高度和深度以基点（base）为界：



可以用

```
\settowidth{⟨长度变量⟩}{⟨内容⟩}
\settoheight{⟨长度变量⟩}{⟨内容⟩}
\settodepth{⟨长度变量⟩}{⟨内容⟩}
```

把内容的宽度、高度或深度赋值给长度变量。也可以用

```
\wd{⟨盒子变量⟩}　　\ht{⟨盒子变量⟩}　　\dp{⟨盒子变量⟩}
```

分别得到盒子的宽度、高度和深度。

除此之外，在 \makebox、\framebox 等盒子命令的参数中，也可以使用 \width、\height、\depth、\totalheight 来分别表示盒子内容的自然宽度、高度、深度，以及高度和深度之和。例如，下面的例子产生一个带边框的盒子，其总宽度恰好是文字自然宽度的 2 倍：

```
\framebox[2\width]{带边框}
```

带边框

2-1-78

2.2　段落与文本环境

2.2.1　正文段落

我们已经知道，LATEX 使用空行表示分段，在自定义命令中，也常用 \par 命令分段。自然段是 LATEX 最基本的正文分划方式。

在 2.1.5 节已经看到，每个自然段在第一行有一个固定的缩进，可以由长度变量 \parindent 控制。在西文标准文档类（如 article）中，每个章节的第一段是不缩进的，而中文文档类（如 ctexart）则每段缩进，并自动设置段落缩进为两个汉字宽。如果要在某一段开头临时禁用缩进，可以在段前使用 \noindent 命令；而如果要在本来没有缩

进的地方使用缩进，可以用 \indent 命令产生一个长为 \parindent 的缩进。在西文文档中，可以使用 indentfirst 宏包启用章节首段的缩进。

除了段落首行缩进，另一个关于分段的重要参数是段与段之间的垂直距离，这由变量 \parskip 控制。\parskip 的默认值是橡皮长度 0pt plus 1pt，在中文排版中常常使用

2 2-2-1

```
\setlength{\parskip}{0pt}
```

把段间距定义为固定长度，禁止段落间距离伸长。当然有时为了文字清晰，也常设置一个较大的 \parskip。

段落最明显的属性是对齐方式。LaTeX 的段落默认是两端均匀对齐的，也可以改为左对齐、右对齐或居中格式。\raggedright 命令设置段落左对齐（ragged right 意谓右边界参差不齐），这在双栏文档一行非常窄的时候特别有用，此时强求均匀对齐会使单词间的距离过大，十分难看，如下所示：

```
English words like 'technology' stem from a
Greek root beginning with the letters τεχ\dots
```

English words like 'technology' stem from a Greek root beginning with the letters τεχ…

而如果使用左对齐就可以解决这个问题：

2-2-2

```
\raggedright
English words like 'technology' stem from a
Greek root beginning with the letters τεχ\dots
```

English words like 'technology' stem from a Greek root beginning with the letters τεχ…

类似地，右对齐使用 \raggedleft 命令，居中使用 \centering 命令。右对齐常常用于排版签名、日期、格言警句等；居中的段落则有强调的意味。

LaTeX 提供了三个环境来排版不同对齐方式的文字：flushleft 环境左对齐、flushright 环境右对齐和 center 环境居中。这几个环境会在段落前后增加一小段垂直间距以示强调，对于少量的段落，它们不会影响前后的文字，因此比 \raggedright 等命令更常用一些。不过如果不想要额外的垂直间距，则不应该使用它们，如：

```
\begin{center}
居中
\end{center}
```

居中

2-2-3

在默认的段落设置下，LaTeX 可能会在单词的两个音节中间断行，并在前一行末尾加上连字符，即所谓断词（用连字号连接，hyphenation）。例如：

This is a hyphen-
ation test.

TeX 的断词算法通常工作得很好，不需要人为干预，不过仍然可能会有一些特殊的单词是 TeX 不能正确处理的，此时可以在单词中使用 \- 命令告诉 LaTeX 可能的断点，如 man\-u\-script。还可以使用 \hyphenation 命令在导言区全局地设置断点列表，如：

```
\hyphenation{man-u-script com-pu-ter gym-na-sium}
```

2-2-4

断词只在均匀对齐的段落中起作用，左对齐的段落就没有断词。使用 \sloppy 命令可以允许段落中更大的空格，从而禁用断词功能；与之相对的命令是 \fussy，让段落恢复默认的较严格的间距。更多地是使用等效的 sloppypar 环境，把允许更宽松间距的文本段放在环境中。以 none 选项使用 hyphenat 宏包[284] 可以更好地禁用断词，宏包也提供了有关打字机字体的断词功能。ragged2e 宏包[230] 则可以在左对齐（\RaggedRight）、右对齐（\RaggedLeft）或居中（\Centering）的段落中使用断词，这样可以得到更合理的段落：

```
% 导言区 \usepackage{ragged2e}
\RaggedRight
English words like 'technology' stem from a
Greek root beginning with the letters τεχ\dots
```

English words like 'tech-
nology' stem
from a Greek
root beginning
with the letters
τεχ…

2-2-5

ragged2e 宏包还提供了 \justifying 命令回到均匀对齐的段落，以及对应的 Center, FlushLeft, FlushRight, justify 环境，作为标准 LaTeX 命令和环境的补充。

对付西文文本参差不齐的小短行，一个有力的工具是 microtype 宏包[227]。这个宏包利用 pdfTeX 的功能，可以通过调整单词内的字母间距改善 TeX 的分行效果。当排版西文文档时可以总打开此宏包的功能，不过当前版本暂不支持 XeTeX[①]。

可以使用长度变量 \leftskip 和 \rightskip 来控制段落的宽度，如果是局部修改，注意把整个段落放在一个分组里面，或定义为单独的环境。例如：

2-2-6

```
\setlength{\leftskip}{4em}
\setlength{\rightskip}{1em}
These parameters tell \TeX{} how much glue
to place at the left and at the right end
of each line of the current paragraph.
```

These parameters tell TeX how much glue to place at the left and at the right end of each line of the current paragraph.

上面介绍的只是基本的段落形状。使用原始 TeX 命令 \parshape（参见 Abrahams et al. [1]、Knuth [126]），LaTeX 还可以排版出更为特殊形状的段落，但 \parshape 命令使用起来比较复杂，这里不作介绍。较为常用的一种特殊段落是“悬挂缩进”，它可以由命令 \hangafter 和 \hangindent 控制，例如：

2-2-7

```
\hangindent=5pc \hangafter=-2
These two parameters jointly specify
``hanging indentation'' for a paragraph.
The hanging indentation indicates to \TeX{}
that certain lines of the paragraph should
be indented and the remaining lines should
have their normal width.
```

These two parameters jointly specify “hanging indentation” for a paragraph. The hanging indentation indicates to TeX that certain lines of the paragraph should be indented and the remaining lines should have their normal width.

① microtype 宏包 v2.5 以后的版本支持 XeTeX 引擎，但目前没有正式发布，需要用户从 http://tlcontrib.metatex.org/ 等网站单独安装测试版本。

正的 \hangindent 作用于段落左侧，负的 \hangindent 作用于段落右侧；正的 \hangafter 作用于段落的 n 行之后，负的 \hangafter 作用于段落的前 n 行。\hangindent 和 \hangafter 的设置只对当前段起作用。

lettrine 宏包[79] 利用特殊的段落形状产生首字下沉的效果，例如：

```
% 导言区 \usepackage{lettrine}
\lettrine{T}{he} \TeX{} in \LaTeX{} refers
to Donald Knuth's \TeX{} typesetting system.
The \LaTeX{} program is a special version of
\TeX{} that understands \LaTeX{} commands.
```

THE TEX in LATEX refers to Donald Knuth's TEX typesetting system. The LATEX program is a special version of TEX that understands LATEX commands.

2-2-8

shapepar 宏包[15] 提供了 \parshape 命令的一个较为方便的语法接口，特别是定义了一些预定义的形状，可以方便地排出一些有趣的效果：

```
% 导言区 \usepackage{shapepar}
\heartpar{%
  绿草苍苍，白雾茫茫，有位佳人，在水一方。
  绿草萋萋，白雾迷离，有位佳人，靠水而居。
  我愿逆流而上，依偎在她身旁。无奈前有险滩，道路又远又长。
  我愿顺流而下，找寻她的方向。却见依稀仿佛，她在水的中央。
  我愿逆流而上，与她轻言细语。无奈前有险滩，道路曲折无已。
  我愿顺流而下，找寻她的足迹。却见仿佛依稀，她在水中伫立。}
```

2-2-9

绿草苍苍，白　　雾茫茫，有位
佳人，在水一方。绿　草萋萋，白雾迷离，
有位佳人，靠水而居。我愿逆流而上，依偎
在她身旁。无奈前有险滩，道路又远又长。
我愿顺流而下，找寻她的方向。却见依稀
仿佛，她在水的中央。我愿逆流而上，与
她轻言细语。无奈前有险滩，道路曲
折无已。我愿顺流而下，找寻她
的足迹。却见仿佛依稀，
她在水中伫立。
♡

2.2.2 文本环境

有几种常用的特殊文本段落类型，在 LaTeX 中以文本环境的形式给出，它们是引用环境、诗歌环境和摘要环境。

引用环境有两种，分别是 quote 环境和 quotation 环境。quote 环境在段前没有首行的缩进，每段话的左右边距比正文大一些，通常用于小段内容的引用：

2-2-10

```
前文……
\begin{quote}
学而时习之，不亦说乎？
有朋自远方来，不亦乐乎？
\end{quote}
后文……
```

前文……

> 学而时习之，不亦说乎？有朋自远方来，不亦乐乎？

后文……

而 quotation 环境则在每段前有首行缩进，因而适用于多段的文字引用：

2-2-11

```
前文……
\begin{quotation}
学而时习之，不亦说乎？
有朋自远方来，不亦乐乎？

默而识之，学而不厌，诲人不倦，何有于我哉？
\end{quotation}
后文……
```

前文……

> 学而时习之，不亦说乎？有朋自远方来，不亦乐乎？
>
> 默而识之，学而不厌，诲人不倦，何有于我哉？

后文……

verse 环境用来排版诗歌韵文：

2-2-12

```
\begin{verse}
在一段内使用 \verb=\\= 换行，\\
分段仍用空行。

过长的长会在折行时悬挂缩进，
就像现在这一句。
\end{verse}
```

> 在一段内使用 \\
> 换行，
> 分段仍用空行。
>
> 过长的长会在折
> 行时悬挂缩
> 进，就像现在
> 这一句。

摘要环境 abstract 是 article 和 report 文档类（包括中文的 ctexart 和 ctexrep）定义的，它产生一个类似 quotation 的小号字环境，并增加标题：

```
\begin{abstract}
本书讲解 \LaTeX{} 的使用。
\end{abstract}
```

摘要

本书讲解 LaTeX 的使用。

2-2-13

摘要的标题由 \abstractname 定义，英文默认是“Abstract”，中文是“摘要”。可以通过重定义 \abstractname 来设置，在 ctex 文档类中也可以使用 \CTEXoptions 设置[58]，如：

```
\CTEXoptions[abstractname={摘\quad 要}]
```

2-2-14

2.2.3　列表环境

2.2.3.1　基本列表环境

列表是常用的本文格式。LaTeX 标准文档类提供了三种列表环境：编号的 enumerate 环境、不编号的 itemize 环境和使用关键字的 description 环境。在列表环境内部使用 \item 命令开始一个列表项，它可以带一个可选参数表示手动编号或关键字。

enumerate 环境使用数字自动编号：

```
\begin{enumerate}
  \item 中文
  \item English
  \item Français
\end{enumerate}
```

1. 中文
2. English
3. Français

2-2-15

itemize 环境不编号，但会在每个条目前面加一个符号以示标记：

```
\begin{itemize}
  \item 中文
  \item English
  \item Français
\end{itemize}
```

- 中文
- English
- Français

2-2-16

description 环境总是使用 \item 命令的可选参数，把它作为条目的关键字加粗显示：

2-2-17

```
\begin{description}
  \item[中文] 中国的语言文字
  \item[English] The language of England
  \item[Français] La langue de France
\end{description}
```

中文 中国的语言文字

English The language of England

Français La langue de France

上面三种列表环境可以嵌套使用（至多四层），LaTeX 会自动处理不同层次的缩进和编号，如 enumerate 环境：

2-2-18

```
\begin{enumerate}
  \item 中文
  \begin{enumerate}
    \item 古代汉语
    \item 现代汉语
    \begin{enumerate}
      \item 口语
      \begin{enumerate}
        \item 普通话
        \item 方言
      \end{enumerate}
      \item 书面语
    \end{enumerate}
  \end{enumerate}
  \item English
  \item Français
\end{enumerate}
```

1. 中文
   (a) 古代汉语
   (b) 现代汉语
      i. 口语
         A. 普通话
         B. 方言
      ii. 书面语
2. English
3. Français

所有条目都以 \item 命令开头。使用 \item 命令的可选参数可以为 enumerate 环境或 itemize 环境临时手工设置编号或标志符号，如：

```
\begin{enumerate}
  \item 中文
  \item[1a.] 汉语
  \item English
\end{enumerate}
```

1. 中文
1a. 汉语
2. English

2-2-19

```
\begin{itemize}
  \item[\dag] 中文
  \item English
  \item Français
\end{itemize}
```

† 中文
• English
• Français

2-2-20

2.2.3.2 计数器与编号

enumerate 环境的编号是由一组计数器（counter）控制的。当 LaTeX 进入一个 enumerate 环境时，就把计数器清零；每遇到一个没有可选参数的 \item 命令，就让计数器的值加 1，然后把计数器的值作为编号输出，达到自动编号的目的。4 个不同嵌套层次的 enumerate 环境使用不同的计数器，分别是 enumi, enumii, enumiii 和 enumiv。LaTeX 的计数器都有一个对应的命令 \the计数器名，用来输出计数器的值，如第一级 enumerate 环境使用 \theenumi：

```
\begin{enumerate}
  \item 这是编号 \theenumi
  \item 这是编号 \theenumi
\end{enumerate}
```

1. 这是编号 1
2. 这是编号 2

2-2-21

而 enumerate 环境又定义了命令 \labelenumi、\labelenumii、\labelenumiii、\labelenumiv 输出条目实际的标签，一般就是在编号后面增加一个标点。有关 enumerate 编号命令的汇总见表 2.9。

LaTeX 计数器的值可以使用命令 \arabic, \roman, \Roman, \alph, \Alph 或 \fnsymbol 带上计数器参数输出，它们分别表示阿拉伯数字、小大写罗马数字、小大写字母、特殊符号[①]：

① ctex 宏包及文档类还提供了 \chinese 命令，生成汉字的数字。

表 2.9 enumerate 环境的编号和标签

嵌套层次	计数器	计数器输出		条目标签	
		命令	默认值	命令	默认值
1	enumi	\theenumi	\arabic{enumi}	\labelenumi	\theenumi.
2	enumii	\theenumii	\alph{enumi}	\labelenumii	(\theenumii)
3	enumiii	\theenumiii	\roman{enumi}	\labelenumiii	\theenumiii.
4	enumiv	\theenumiv	\Alph{enumi}	\labelenumiv	\theenumiv.

2-2-22

```
\begin{enumerate}
\item 编号
  \arabic{enumi}, \roman{enumi}, \Roman{enumi},
  \alph{enumi}, \Alph{enumi}, \fnsymbol{enumi}
\item 编号
  \arabic{enumi}, \roman{enumi}, \Roman{enumi},
  \alph{enumi}, \Alph{enumi}, \fnsymbol{enumi}
\item 编号
  \arabic{enumi}, \roman{enumi}, \Roman{enumi},
  \alph{enumi}, \Alph{enumi}, \fnsymbol{enumi}
\end{enumerate}
```

1. 编号 1, i, I, a, A, *
2. 编号 2, ii, II, b, B, †
3. 编号 3, iii, III, c, C, ‡

因此，可以通过重定义 \theenumi（默认定义是 \arabic{enumi}）和 \labelenumi（默认定义是 \theenumi.）等命令，控制 enumerate 环境的编号：

2-2-23

```
\renewcommand\theenumi{\roman{enumi}}
\renewcommand\labelenumi{(\theenumi)}
\begin{enumerate}
  \item 使用中文
  \item Using English
\end{enumerate}
```

(i) 使用中文

(ii) Using English

计数器在 LaTeX 中非常常用，除了列表环境的编号以外，页码、章节和图表的编号等也是由计数器控制的。如页码的计数器是 page，因此现在这句话在第 \thepage 页，即在第 100 页。

可以自定义计数器完成一些工作。新定义计数器用 \newcounter 命令，给计数器赋值用 \setcounter 命令，计数器自增用 \stepcounter 命令，给计数器的值加上一个数用 \addtocounter 命令。例如我们仿照 enumerate 环境的编号过程：

```
% 计数器设置，通常在导言区
\newcounter{mycnt}
\setcounter{mycnt}{0}                    % 默认值就是 0
\renewcommand\themycnt{\arabic{mycnt}}  % 默认值就是阿拉伯数字
% 计数器使用，通常做成自定义命令的一部分
\stepcounter{mycnt}\themycnt 输出计数器值为 1；
\stepcounter{mycnt}\themycnt 输出计数器值为 2；
\addtocounter{mycnt}{1}\themycnt 输出计数器值为 3；
\addtocounter{mycnt}{-1}\themycnt 输出计数器值为 2；
\addtocounter{mycnt}{-1}\themycnt 输出计数器值为 1。
```

2-2-24

\refstepcounter 命令与 \stepcounter 功能类似，并且会将当前 \label 命令设置为对应的计数器（参见 3.2 节），因而在自动编号的命令或环境的定义中更为常用。在 \addtocounter 的参数等需要用到数字的地方，可以使用 \value{⟨计数器⟩} 使用计数器的数值[①]。

\newcounter 命令还可以在后面增加一个可选参数，内容是一个已有的计数器，表示新计数器会随着旧计数器的自增而自动清零。这特别适合定义每个章节独立的编号，例如：

```
\newcounter{quiz}[section]
\renewcommand\thequiz{\thesection-\arabic{quiz}}
```

2-2-25

amsmath 提供了 \numberwithin{⟨计数器$_1$⟩}{⟨计数器$_2$⟩} 命令，扩展了 \newcounter 自动清零的功能，用来让已有的计数器 ⟨计数器$_1$⟩ 随 ⟨计数器$_2$⟩ 的增加而清零重编号，同时定义其编号格式[7]，如让数学方程按节编号：

```
\usepackage{amsmath}
\numberwithin{equation}{section}
```

2-2-26

chngcntr 宏包[289] 提供了类似功能的命令 \counterwithin，以及相反功能的命令 \counterwithout，用来取消计数器间的关联。

① 事实上，\value 命令得到的是 LaTeX 计数器对应的原始 TeX 数字寄存器。

计数器不仅可以用于编号，也能用于复杂的条件控制和循环，ifthen 宏包就提供了有关条件判断和循环的功能，而 calc 宏包则提供了有关长度（参见 2.1.5 节）和计数器的一些简单运算功能。可参见文档 Carlisle [47]、Thorup et al. [262]。

2.2.3.3 定制列表环境

与 enumerate 环境（见表 2.9）类似，itemize 环境也使用一组命令来控制前面的标签，见表 2.10。不过 itemize 环境比较简单，没有编号。

表 2.10 itemize 环境的标签

嵌套层次	命令	默认值	效果
1	\labelitemi	\textbullet	•
2	\labelitemii	\normalfont\bfseries \textendash	–
3	\labelitemiii	\textasteriskcentered	*
4	\labelitemiv	\textperiodcentered	·

description 环境相对比较简单，它使用 \descriptionlabel 命令来控制标签的输出格式。\descriptionlabel 在标准文件类中的原始定义如下[139]：

```
\newcommand*\descriptionlabel[1]{\hspace\labelsep
                                \normalfont\bfseries #1}
```

这是一个带有一个参数的命令，参数是标签的文本。标签前面的间距 \labelsep 默认为 0.5 em。可以修改 \descriptionlabel 命令得到不同效果的 description 环境：

```
{% 使用分组让 \descriptionlabel 的修改局部化
\renewcommand\descriptionlabel[1]{\normalfont\Large\itshape
  \textbullet\ #1}
\begin{description}
  \item[标签] 可以修改 \verb=\descriptionlabel= 改变标签的格式。
  \item[其他] 其他格式的也可以参考后面的列表环境修改。
\end{description}
}
```

2-2-27

*本节内容初次阅读可略过。

- **标签** 可以修改 \descriptionlabel 改变标签的格式。
- **其他** 其他格式的也可以参考后面的列表环境修改。

基本列表环境 enumerate, itemize, description 都是由广义列表环境 list 生成的，list 环境语法为：

```
\begin{list}{⟨标签⟩}{⟨设置命令⟩}
    ⟨条目⟩
\end{list}
```

其中 ⟨标签⟩ 中给出标签的内容，如果是编号列表可以使用计数器；⟨设置命令⟩ 则可以设置列表使用的计数器和一些长度参数。\usecounter{⟨计数器⟩} 表示使用指定的计数器编号。下面是一个简单的编号列表的例子：

```
\newcounter{mylist}
\begin{list}{\#\themylist}%
  {\usecounter{mylist}}
  \item 中文
  \item English
\end{list}
```

#1 中文

#2 English

2-2-28

与 list 环境相关的参数有很多，见图 2.3。可以在 ⟨设置命令⟩ 项中使用 \setlength 等命令设置，可以这样产生一个紧凑的类 itemize 环境：

```
\begin{list}{\textbullet}{%
  \setlength{\topsep}{0pt}  \setlength{\partopsep}{0pt}
  \setlength{\parsep}{0pt}  \setlength{\itemsep}{0pt}}
  \item 中文

    又一段中文

  \item English
  \item Français
\end{list}
```

2-2-29

- 中文

  又一段中文

- English
- Français

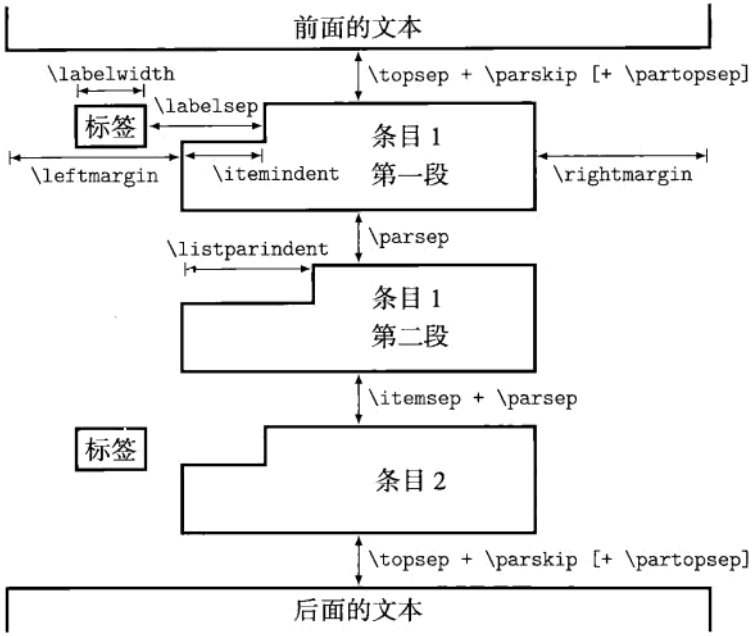


图 2.3 list 列表环境的长度参数。其中 \partopsep 在列表环境开始新的一段（即前面有空行）时生效

list 环境语法比较烦琐，一般并不直接使用，而是用它来定义新的环境，例如前面的紧凑列表可以定义为 myitemize 环境：

```
\newenvironment{myitemize}{%
    \begin{list}{\textbullet}{%
      \setlength{\topsep}{0pt}  \setlength{\partopsep}{0pt}
      \setlength{\parsep}{0pt}  \setlength{\itemsep}{0pt}}}
  {\end{list}}
```

2-2-30

还有一种平凡列表环境 trivlist，可以生成空标签的列表。trivlist 一般不单用，而是用来生成一些与列表看起来没什么关系的环境，例如 center 环境就是由 trivlist 加上 \centering 命令生成的[33]，等价定义为：

```
\newenvironment{mycenter}
```

```
{\begin{trivlist}
   \centering
   \item[]}
{\end{trivlist}}
```

2-2-31

使用 list 环境定制列表比较复杂，特别是对 enumerate 等环境更是难以修改其格式。使用 enumitem 宏包[28] 可以方便地对各种列表环境的标签、尺寸进行定制，也可以用它来定义新的列表①。使用 enumitem 宏包，可以直接在 enumerate 等环境后加可选参数来定制参数，如：

```
% \usepackage{enumitem}
\begin{enumerate}[itemsep=0pt,parsep=0pt,
    label=(\arabic*)]
  \item 中文
  \item English
  \item Français
\end{enumerate}
```

(1) 中文
(2) English
(3) Français

2-2-32

也可以使用 \setlist 命令整体设置参数，用 \newlist 命令定义新列表，如：

```
\usepackage{enumitem}
% 仿照 enumerate 环境定义可二级嵌套的 mylist
\newlist{mylist}{enumerate}{2}
% 分别定义每级的格式
\setlist[mylist,1]{itemsep=0pt,parsep=0pt,label=(\arabic*)}
\setlist[mylist,2]{itemsep=0pt,parsep=0pt,label=(\alph*)}
```

2-2-33

enumitem 宏包接受的参数大多与图 2.3 中对应，其中标签的参数 label 用星号 * 表示计数器，对非标准计数器输出（如 \chinese）还要单独进行设置，如：

```
% \usepackage{enumitem}
\AddEnumerateCounter{\chinese}{\chinese}{}
\begin{enumerate}[label={\chinese*、},labelsep=0pt]
  \item 内容清晰
  \item 格式美观
\end{enumerate}
```

2-2-34

① enumitem 宏包继承并扩展了在此之前 enumerate、mdwlist，特别是 paralist[224] 宏包的功能，提供了更为强大的功能。

一、内容清晰

二、格式美观

有关 enumitem 更多功能的详细内容可参见宏包文档 Bezos [28]。

2.2.4 定理类环境

定理类环境是 LaTeX 中的一类重要的文本环境，它可以产生一个标题、编号和特定格式的文本，我们在 1.2.4 节中已经看到了定理类环境的定义使用：

2-2-35
```
\newtheorem{thm}{定理} % 一般在导言区
\begin{thm}
直角三角形斜边的平方等于两腰的平方和。
\end{thm}
```

定理 1 直角三角形斜边的平方等于两腰的平方和。

在上面的例子中，命令 \newtheorem 用来声明一个新的定理类环境，两个参数分别是定理类环境名和定理输出的标题名。因而，\newtheorem{thm}{定理} 就定义了一个 thm 环境，效果是输出标题为“定理”的一段文字。新定义的 thm 环境允许带有可选参数表示定理的小标题：

2-2-36
```
\begin{thm}[勾股定理]
直角三角形斜边的平方等于两腰的平方和。
\end{thm}
```

定理 2 (勾股定理) 直角三角形斜边的平方等于两腰的平方和。

\newtheorem 命令可以在最后使用一个可选的计数器参数 ⟨*Ctr*⟩，用来表示定理编号是 ⟨*Ctr*⟩ 的下一级编号，并会随 ⟨*Ctr*⟩ 的变化而清零。这通常用于让定理按章节编号，如：

2-2-37
```
\newtheorem{lemma}{引理}[chapter]% 按章
\begin{lemma}偏序集可良序化。\end{lemma}
\begin{lemma}实数集不可数。  \end{lemma}
```

引理 2.1 偏序集可良序化。

引理 2.2 实数集不可数。

也可以在两个参数之间使用一个可选的计数器参数 ⟨*Ctr*⟩，表示定理使用 ⟨*Ctr*⟩ 进行编号。这通常用于让几个不同的定理类环境使用相同的编号（定理类环境的计数器就是环境名），如：

```
\newtheorem{prop}[thm]{命题}
\begin{prop}
直角三角形的斜边大于直角边。
\end{prop}
```

命题 3 直角三角形的斜边大于直角边。

2-2-38

默认的定理类环境的格式是固定的，但这往往并不符合我们的期望。theorem 宏包[159] 扩展了定理类环境的格式，可以方便地修改定理类环境的格式，它提供了 \theoremstyle{格式} 命令来选择定理类环境的格式，可用的预定义格式有：

plain	默认格式；
break	定理头换行；
marginbreak	编号在页边，定理头换行；
changebreak	定理头编号在前文字在后，换行；
change	定理头编号在前文字在后，不换行；
margin	编号在页边，定理头不换行。

定理类环境的默认字体是 \slshape，定理头默认是 \bfseries，可以通过 \theorembodyfont{字体} 和 \theoremheaderfont{字体} 分别设置定理内容和定理头的字体，并可以设置定理前后的垂直间距变量 \theorempreskipamount 和 \theorempostskipamount，例如：

```
% 导言区
\usepackage{theorem}
\theoremstyle{changebreak}
\theoremheaderfont{\sffamily\bfseries}
\theorembodyfont{\normalfont}
\newtheorem{definition}{定义}[chapter]
```

2-2-39

```
\begin{definition}
有一个角是直角的三角形是\emph{直角三角形}
。
\end{definition}
```

2.1 定义
有一个角是直角的三角形是*直角三角形*。

2-2-40

ntheorem 宏包[153] 进一步扩充了 theorem 宏包的功能，它增加了下面几种 \theoremstyle：

nonumberplain　类似 **plain** 格式，但没有编号；

nonumberbreak 类似 **break** 格式，但没有编号；
empty 没有编号和定理名，只输出可选参数。

可以输出无编号的定理。也可以用 \newtheorem* 来定义无编号的定理。

ntheorem 增加了更多的设置命令和大量辅助的功能，如给宏包增加 [thmmarks] 选项后可以使用 \theoremsymbol 命令设置定理类环境末尾自动添加的符号，这对定义证明环境表示证毕符号特别有用：

2-2-41
```
% 导言区
\usepackage[thmmarks]{ntheorem}
{   % 利用分组，格式设置只作用于证明环境
  \theoremstyle{nonumberplain}
  \theoremheaderfont{\bfseries}
  \theorembodyfont{\normalfont}
  \theoremsymbol{\mbox{$\Box$}} % 放进盒子，或用 \ensuremath
  \newtheorem{proof}{证明}
}
```

2-2-42
```
\begin{proof}
证明是显然的。
\end{proof}
```

证明 证明是显然的。 □

ntheorem 宏包的功能很多，除了详细定制定理格式，还可以重定义已有的定理类环境、分类管理定理格式、生成定理目录等；具体的使用中也有很多注意事项（如与 amsmath 宏包使用时要加 [amsmath] 选项），读者可查阅宏包的文档 May and Schedler [153] 获得进一步的信息。

除了 theorem 和 ntheorem 宏包，美国数学会发布的 amsthm 宏包[10] 也常用于定理类环境的定制。在使用 XeLaTeX 时，amsthm 必须在 fontspec 宏包之前载入，而由于 ctex 文档类中的 xeCJK 宏包会自动调用 fontspec，因此不能在 ctexart 等文档类中直接使用 amsthm，而只能在 article 等标准文档类中使用 amsthm 宏包用再使用 ctexcap 宏包，比较不便。

amsthm 提供了预定义的 proof 环境用来表示证明并自动添加证毕符号，但这个环境的可定制性较差，只能通过重定义 \proofname 宏修改证明的“Proof”字样，或重定义 \qedsymbol 宏修改证毕符号，例如：

```
\usepackage{amsthm}
\renewcommand\proofname{证明}
\renewcommand\qedsymbol{\ensuremath{\Box}}
```

2-2-43

amsthm 的证毕符号没有 ntheorem 的智能，在显示公式中无法正确判断位置，此时需要手工使用 \qedhere 命令添加证毕符号，如：

```
% \usepackage{amsthm}
\begin{proof}
最后我们有
\[
  f(x) = 0. \qedhere
\]
\end{proof}
```

2-2-44

amsthm 预定义了一些格式，可以使用 \theoremstyle 命令选择，但如果用 \newtheoremstyle 命令定义新的格式则语法比较复杂。amsthm 与 ntheorem 宏包的功能大致相当，但 ntheorem 宏包的语法容易理解一些，也没有 XeLaTeX 下与 fontspec 的冲突问题，因此最好使用 ntheorem 包。有关 amsthm 定制定理格式的进一步用法可以详细内容参见宏包的文档 American Mathematical Society [10]，这里不做过多介绍了。

练 习

2.4 Schwarz 的 thmtools 宏包为 amsthm 或 ntheorem 提供了一个更为友好的基于键值语法的用户界面。试参阅文档 [231]，使用 thmtools 宏包的功能重写例 2-2-39。

2.2.5 抄录和代码环境

2.2.5.1 抄录命令与环境

在 2.1.2 节中我们已经看到，LaTeX 输入特殊符号相当复杂。然而我们有时候必须经常性地使用特殊符号，例如在排版计算机程序源代码时（特别是排版有关 TeX 的文章时）就不可避免地使用大量在 LaTeX 中有特殊意义的符号，此时就需要使用抄录（verbatim，即逐字）功能。

\verb 命令可用来表示行文中的抄录，其语法格式如下：

\verb⟨符号⟩⟨抄录内容⟩⟨符号⟩

在 \verb 后，起始的符号和末尾的符号相同，两个符号之间的部分将使用打字机字体逐字原样输出：

2-2-45
```
\verb"\LaTeX \& \TeX" \qquad
\verb!\/}{#$%&~!
```

\LaTeX \& \TeX \qquad \/}{#$%&~

使用带星号的命令 \verb* 则可以使输出的空格为可见的 ␣：

2-2-46
```
显示空格 \verb*!1 2  3   4!
```

显示空格 1␣2␣␣3␣␣␣4

大段的抄录则可以使用 verbatim 环境：

2-2-47
```
\begin{verbatim}
#!usr/bin/env perl
$name = "guy";
print "Hello, $name!\n";
\end{verbatim}
```

```
#!usr/bin/env perl
$name = "guy";
print "Hello, $name!\n";
```

同样，可以使用带星号的 verbatim* 环境输出可见空格：

2-2-48
```
\begin{verbatim*}
#include <stdio.h>
main() {
    printf("Hello, world.\n");
}
\end{verbatim*}
```

```
#include␣<stdio.h>
main()␣{
␣␣␣␣printf("Hello,␣world.\n");
}
```

抄录命令和环境都属于特殊命令，一般不能作为其他命令的参数出现，例如使用 \fbox{\verb!abc!} 将会产生错误。可以使用 lrbox 环境把它们保存在自定义盒子中再进行使用，参见 2.1.5 节，这也可以由 fancyvrb 宏包[301] 提供的命令实现，例如：

2-2-49
```
% \usepackage{fancyvrb}
\SaveVerb{myverb}|#$%^&|
\fbox{套中 \UseVerb{myverb}}
```

套中 #$%^&

值得一提的是 cprotect 宏包[80]，它定义了 \cprotect 等命令，可以方便地在其他命令参数中使用 \verb 命令或 verbatim 环境。\cprotect 的用法与 \protect 有些类似，把它用在带参数的命令前面，来“保护”参数中有抄录的命令：

```
% \usepackage{cprotect}
\cprotect\fbox{套中 \verb|#$%^&|}
```

套中 #$%^&　2-2-50

尽管使用方便，cprotect 宏包的使用限制会多一些，在一些命令中（如 \parbox）仍然需要用先保存再使用的方式。

verbatim 宏包[233] 提供了 verbatim 环境的一些扩展。特别是定义了 \verbatiminput{文件名} 命令用于逐字抄录整个文件的内容。

shortvrb 宏包[161] 提供了 \verb 命令的简写形式，可以使用 \MakeShortVerb{符号} 来定义这种简写，或用 \DeleteShortVerb{符号} 取消定义，如在导言区定义：

```
\usepackage{shortvrb}
\MakeShortVerb|
```

那么就可以使用竖线 | 作为简写方式了：

```
verbatim |\LaTeX|
```

verbatim \LaTeX　2-2-51

fancyvrb 宏包提供了一系列 verbatim 环境的扩展，它提供的 Verbatim 环境可以修改字体、边框、填充颜色、行号等多种格式，甚至在抄录环境中插入任意 LaTeX 代码，功能强大。读者可参考宏包的文档 Zandt [301]，这里不作详细介绍了。

2.2.5.2　程序代码与 listings

可以使用 verbatim 环境排版程序代码。不过，如果还想在程序代码中增加语法高亮功能，那么 verbatim 环境就捉襟见肘了，比如要排版下面的效果：

```
1 /* hello.c
2  *  A 'hello world' program. */
3 #include <stdio.h>
4 int main()
5 {
6     printf("Hello, world.\n");
7     return 0;
8 }
```

这种带语法高亮的程序代码可以使用 listings 宏包[174] 排版。

listings 宏包提供的基本功能是 lstlisting 环境，可以把它看做是加强的 verbatim 环境。不过在未做任何设置时，lstlisting 环境并没有语法高亮的效果，而且看起来比直接使用 verbatim 还难看些，如下所示：

```
% 导言区使用 \usepackage{listings}
\begin{lstlisting}
/* hello.c */
#include <stdio.h>
main() {
    printf("Hello.\n");
}
\end{lstlisting}
```

```
/* hello.c */
#include <stdio.h>
main() {
    printf("Hello.\n");
}
```

要想得到漂亮的排版效果，必须仔细设置 lstlisting 环境的格式。可以使用 lstlisting 环境的可选参数设置格式，也可以使用 \lstset{⟨设置⟩} 进行全局设置。最基本的参数是 language，设置代码使用的语言，如：

2-2-52

```
\begin{lstlisting}[language=C]
/* hello.c */
#include <stdio.h>
main() {
    printf("Hello.\n");
}
\end{lstlisting}
```

```
/* hello.c */
#include <stdio.h>
main() {
    printf("Hello.\n");
}
```

此时可以看到 C 语言的注释用斜体排出，关键字用粗体排出，字符串中的空格也排版为可见的。listings 宏包支持的语言非常多，几乎包括了各种常见的语言（见表 2.11）。

前面的例子中字体并不是很好看，可以用 basicstyle 选项设置 lstlisting 环境的整体格式，或用 keywordstyle 设置关键字的格式，用 identifierstyle 设置标识符格式，用 stringstyle 设置字符串的格式，用 commentstyle 设置注释的格式，等等。以上这些都是影响 lstlisting 环境输出效果最明显的一些参数，例如：

表 2.11　listings 宏包预定义的语言。这里一些语言的定义只是初步的，如 HTML 和 XML，带下画线的方言是默认的方言

ABAP (R/2 4.3, R/2 5.0, R/3 3.1, R/3 4.6C, R/3 6.10)
ACSL
Ada (2005, 83, 95)
Algol (60, 68)
Ant
Assembler (Motorola68k, x86masm)
Awk (gnu, POSIX)
bash
Basic (Visual)
C (ANSI, Handel, Objective, Sharp)
C++ (ANSI, GNU, ISO, Visual)
Caml (light, Objective)
CIL
Clean
Cobol (1974, 1985, ibm)
Comal 80
command.com (WinXP)
Comsol
csh
Delphi
Eiffel
Elan
erlang
Euphoria
Fortran (77, 90, 95)
GCL
Gnuplot
Haskell
HTML
IDL (empty, CORBA)
inform
Java (empty, AspectJ)
JVMIS
ksh
Lingo
Lisp (empty, Auto)
Logo
make (empty, gnu)
Mathematica (1.0, 3.0, 5.2)
Matlab
Mercury
MetaPost
Miranda
Mizar
ML
Modula-2
MuPAD
NASTRAN
Oberon-2
OCL (decorative, OMG)
Octave
Oz
Pascal (Borland6, Standard, XSC)
Perl
PHP
PL/I
Plasm
PostScript
POV
Prolog
Promela
PSTricks
Python
R
Reduce
Rexx
RSL
Ruby
S (empty, PLUS)
SAS
Scilab
sh
SHELXL
Simula (67, CII, DEC, IBM)
SPARQL
SQL
tcl (empty, tk)
TeX (AlLaTeX, common, LaTeX, plain, primitive)
VBScript
Verilog
VHDL (empty, AMS)
VRML (97)
XML
XSLT

2-2-53

```
\lstset{ % 整体设置
  basicstyle=\sffamily,
  keywordstyle=\bfseries,
  commentstyle=\rmfamily\itshape,
  stringstyle=\ttfamily}
\begin{lstlisting}[language=C]
/* hello.c */
#include <stdio.h>
main() {
    printf("Hello.\n");
}
\end{lstlisting}
```

```
/* hello.c */
#include <stdio.h>
main() {
    printf("Hello.\n");
}
```

listings 宏包默认将字符等宽显示，这样可以使文字的不同列容易对齐，但也会使得一些字体看上去非常怪异。可以通过设置 column 选项为 flexible（默认为 fixd），或使用 flexiblecolumns 选项来把字符列设置为非等宽的。采用非等宽的列可以让默认的 Roman 字体看上去也比较顺眼：

2-2-54

```
\lstset{flexiblecolumns}% column=flexible
\begin{lstlisting}[language=C]
/* hello.c */
#include <stdio.h>
main() {
    printf("Hello.\n");
}
\end{lstlisting}
```

```
/* hello.c */
#include <stdio.h>
main() {
    printf ("Hello.\n");
}
```

可以设置 numbers 选项为 left 或 right 在左右增加行号（默认为 none），使用 numberstyle 选项设置行号格式：

```
\lstset{columns=flexible,
  numbers=left,numberstyle=\footnotesize}
\begin{lstlisting}[language=C]
/* hello.c */
#include <stdio.h>
main() {
    printf("Hello.\n");
}
\end{lstlisting}
```

```
1  /* hello.c */
2  #include <stdio.h>
3  main() {
4      printf ("Hello.\n");
5  }
```

2-2-55

listings 宏包还提供了 \verb 命令的对应物 \lstinline，这样可以直接在行文段落中使用带语法高亮的代码：

```
\lstset{language=C,flexiblecolumns}
语句 \lstinline!typedef char byte!
```

语句 **typedef char** byte

2-2-56

listings 宏包对汉字等非 ASCII 字符的支持不是很好。使用 XeLaTeX 时，一般需要对汉字用逃逸字符处理，这样才能得到正确的结果[①]：

```
\lstset{language=C,flexiblecolumns,
  escapechar=`} % 设置 ` 为逃逸字符
\begin{lstlisting}
int n;  // `一个整数`
\end{lstlisting}
```

int n; // 一个整数

2-2-57

listings 将逃逸字符之间的内容当做普通的 LaTeX 代码处理，因而汉字可以得到正确处理。事实上这种方法也可以用来输出一些特殊效果的代码，比如在代码中使用数学公式：

```
\lstset{language=C,flexiblecolumns,
  escapechar=`}
\begin{lstlisting}
double x = 1/sin(x); // `$\frac1{\sin x}$`
\end{lstlisting}
```

double x = 1/sin(x); // $\frac{1}{\sin x}$

2-2-58

① 不使用 XeLaTeX 而使用 CJK 方式处理汉字时，还需要设置 extendedchars=false 选项。

上面介绍的选项只是 listings 宏包各种功能中的很少一部分，事实上，使用 listings 宏包还可以设置背景颜色、强调内容、标题、目录等。请读者自行查阅宏包文档 Moses [174]。

2.2.6 tabbing 环境

tabbing 环境用来排版制表位，即让不同的行在指定的地方对齐。在 tabbing 环境中，行与行之间用 \\ 分隔，使用 \= 命令来设置制表位，用 \> 命令则可以跳到下一个前面已设置的制表位，因而可以用 tabbing 环境制作比较简单的无线表格（真正的表格参见 5.1 节）：

2-2-59

```
\begin{tabbing}
格式\hspace{3em} \= 作者 \\
Plain \TeX  \> 高德纳 \\
\LaTeX      \> Leslie Lamport
\end{tabbing}
```

格式	作者
Plain TEX	高德纳
LATEX	Leslie Lamport

\kill 命令与 \\ 类似，它会忽略这一行的内容，只保留制表位的设置。使用 \kill 可以做出 tabbing 环境的样本行：

2-2-60

```
\begin{tabbing}
格式\hspace{3em} \= 作者 \kill
Plain \TeX  \> 高德纳 \\
\LaTeX      \> Leslie Lamport
\end{tabbing}
```

Plain TEX	高德纳
LATEX	Leslie Lamport

关于制表位的其他命令如下：

\'	使它前后的文字以当前制表位为中心对齐
\`	使后面的文字右对齐
\<	与 \> 相反，跳到前一个制表位
\+	后面的行开始都右跳一个制表位
\-	后面的行开始都左跳一个制表位
\pushtabs	保存当前制表位
\poptabs	恢复由 \pushtabs 保存的制表位

由于在 tabbing 环境中原来表示字母重音的命令 \=、\'、\` 被重定义为制表位的操作，所以在 tabbing 环境中重音命令改为 \a 后加 =、`、'，如在 tabbing 中的 \a=o 就得到 ō。

下面给出一个相对复杂的例子，使用上面的命令排版一个算法，并说明这些命令的意义：

```
\newcommand\kw{\textbf} % 表示描述算法的关键字
\begin{tabbing}
\pushtabs
算法：在序列 $A$ 中对 $x$ 做二分检索 \\
输入：$A$, $x$ 及下标上下界 $L$, $H$ \\
\qquad\=\+\kw{integer} $L$, H, M, j$ \\
\kw{while} \=\+ $L \leq H$ \kw{do} \` $L$ 与 $H$ 是左右分点 \\
          $M \gets \lfloor(L+H)/2\rfloor$ \` $M$ 是中间分点 \\
          \kw{case} \=\+\\
                  condition \= foo \+\kill
                  $x > A[M]$:\' $H \gets M-1$ \\
                  $x < A[M]$:\' $H \gets M+1$ \\
                   \kw{else}:\' \= $j \gets M$ \` 找到 $x$，返回位置\\
                                \> \kw{return}$(j)$ \\
          \<\< \kw{endcase} \-\-\-\\
$j \gets 0$ \\
\kw{return}$(j)$ \-\\
\poptabs
算法示例：\\
$A = \{2, 3, 5, 7, 11\}$, $x=3$\\
\qquad\=\+ $M$\qquad \= $L$\qquad \= $H$\qquad \= \\
          无      \> 1       \> 5          \> 初始值，进入循环 \\
          3       \> 1       \> 2          \> $H$ 变化 \\
          2       \> 无      \> 无         \> 找到 $x$，输出位置 2
\end{tabbing}
```

2-2-61

算法：在序列 A 中对 x 做二分检索
输入：A, x 及下标上下界 L, H
　　integer L, H, M, j

```
    while L ≤ H do                                  L 与 H 是左右分点
        M ← ⌊(L + H)/2⌋                             M 是中间分点
        case
          x > A[M]: H ← M − 1
          x < A[M]: H ← M + 1
                else: j ← M                         找到 x，返回位置
                      return(j)
        endcase
    j ← 0
    return(j)
```

算法示例：

$A = \{2, 3, 5, 7, 11\}, x = 3$

M	L	H	
无	1	5	初始值，进入循环
3	1	2	H 变化
2	无	无	找到 x，输出位置 2

尽管使用 tabbing 环境可以很自由地排版复杂的算法，但时刻考虑制表位来调整算法结构有时实在太考验人的耐心了。排版算法这类结构化的伪代码可以使用专门的算法宏包。clrscode[57] 是许多算法宏包中最为简单的一种，它可以按著名教材《算法导论》[56] 中的格式进行算法排版，事实上它就是用 tabbing 环境实现的；另一个使用广泛的算法宏包是 algorithm2e[78]，它提供丰富的命令和复杂的定制功能；algorithmicx[117] 也提供了类似的易定制的算法排版功能，有需要的读者可以参见这些算法宏包的说明文档来做进一步的选择。

练 习

2.5 参考 clrscode 或 algorithm2e 宏包的文档，重新排版例 2-2-61 的算法。

2.2.7 脚注与边注

LaTeX 使用 \footnote{⟨脚注内容⟩} 产生脚注，例如[1]。这是脚注的默认格式，使用一个阿拉伯数字的上标作为编号，脚注内容出现在页面底部，以 \footnotesize 的字

[1] 这是一个脚注。

号输出。脚注的内容可以是大段的文字，但注意前后应该都和正文紧接着，避免出现多余的空格，特别要注意标点前后的位置，像前面的例子：

```
例如\footnote{这是一个脚注。}。
```
2-2-62

\footnote 是自动编号的，例如[1]。\footnote 可以带一个可选的参数，表示手工的数字编号，例如用 \footnote[1]{又是一？} 的效果[1]，它不改变原来的脚注编号[2]。

脚注的使用有一些限制，它通常只能用在一般的正文段落中，不能用于表格、左右盒子（如一个 \mbox 的参数中）等受限的场合。此外，脚注也不能直接用在 \section 等章节命令的参数中，也不能用在 \parbox 中。不过可以把脚注用在 minipage 环境（参见 2.2.8 节）里面，此时脚注出现在 minipage 环境产生的盒子的底部，并使用局部的编号（默认按字母编号）：

```
\begin{minipage}{8em}
这是小页环境\footnote{脚注。}中的脚注。
\end{minipage}
```

这是小页环境[a]中的脚注。

[a]脚注。

2-2-63

脚注使用的计数器是 footnote，在 minipage 环境中则使用 mpfootnote 计数器，可以修改 \thefootnote 和 \thempfootnote 改变编号的格式。一种常见的修改是使用符号编号，即定义：

```
\renewcommand\thefootnote{\fnsymbol{footnote}}
```
2-2-64

得到的效果是[‡]。另一种常见修改是带圈的数字，这可以利用 \textcircled 命令生成的带圈文字来完成：

```
\renewcommand\thefootnote{\textcircled{\arabic{footnote}}}
```

得到的效果是[④]。使用 pifont 宏包[229] 提供的带圈数字符号效果更好些，使用 pifont 宏包的 \ding 命令可以输出符号表中的符号，查表找到阳文带圈数字从 172 号符号开始，此时需要 ε-TEX 的 \numexpr 命令支持数字的计算：

```
\usepackage{pifont}
\renewcommand\thefootnote{\ding{\numexpr171+\value{footnote}}}
```
2-2-65

[1]另一个脚注。
[1]又是一？
[2]继续前面编号。
[‡]符号编号的脚注。
[④]带圈数字编号脚注。

得到的效果是[①]，这也是本书使用的编号符号。

在不能使用脚注的位置，例如盒子、表格中，可以使用命令 \footnotemark 和 \footnotetext 分开输入脚注的编号和内容。\footnotemark[⟨数字⟩] 命令产生正文中的脚注编号，如果没有可选参数，脚注计数器自动自增；\footnotetext[⟨数字⟩]{⟨内容⟩} 命令产生脚注的内容，脚注计数器不自增，如果有可选参数就使用手工编号。这里在表格中使用脚注的方法是典型的（也可以把表格放在 minipage 环境中，直接使用 \footnote 命令即可）：

2-2-66
```
\begin{tabular}{r|r}
  自变量 & 因变量\footnotemark \\ \hline
  $x$ & $y$
\end{tabular}
\footnotetext{$y=x^2$。}
```

自变量	因变量[2]
x	y

如果要在章节或图表标题中使用脚注，则要用 \protect\footnote 代替 \footnote，因为这里 \footnote 是脆弱命令（fragile command），当脆弱命令用在所谓活动的命令参数（moving arguments）中时，就必须使用 \protect 命令保护起来[136, C.1.3]（参见 8.1.2 节）。此外还要注意使用 \section 等命令的可选参数避免把脚注符号装进页眉和目录中：

2-2-67
```
\section[节标题]{节标题\protect\footnote{标题中的脚注}}
```

实际中主要遇到的活动参数就是会生成目录项的命令，即章节命令和 \caption 等。

几条脚注之间的距离用长度变量 \footnotesep 设置。脚注线由命令 \footnoterule 定义，默认的定义是长为 2 in 的一条线，可以重定义 \footnoterule 改变脚注线，如可以使用命令 \renewcommand\footnoterule{} 定义脚注线为空。

更多脚注的格式的定制可以使用 footmisc 宏包[71] 完成。例如，LaTeX 的脚注默认每章重新清零编号（如果没有 \chapter 一级，则全文统一编号），可以使用

2-2-68
```
\usepackage[perpage]{footmisc}
```

让脚注每页清零编号，这也是中文排版更常见的样式。footmisc 还提供了控制脚注段落形状的许多选项，以及一些杂项命令，可参考文档 Fairbairns [71] 获得更多信息。

[①] 更好的带圈脚注。
[2] $y = x^2$。

LaTeX 还提供了边注的命令 \marginpar{⟨内容⟩}，用来给文档添加在页边的旁注。边注具有手稿风格。

边注的使用方法和脚注差不多，只是边注不编号，内容出现的位置与正文更接近。页边距通常很窄，所以不要在边注中长篇大论，最好把它留给个别需要强调指出的地方。

在单面模式 onecolumn 下（参见 2.4.2 节），边注在页面的右侧；在双面模式 twocolumn 下，边注在页面的外侧（即奇数页在右，偶数页在左）。\marginpar 命令可以带一个可选参数，设置出现在偶数页左侧的边注，而原来的参数表示在右侧的边注，例如：

```
有边注的文字\marginpar[\hfill 左 $\rightarrow$]{$\leftarrow$ 右}
```

2-2-69

有边注的文字

← 右

可以使用命令 \reversemarginpar 改变边注的左右（或内外）位置，或者用命令 \normalmarginpar 复原边注的左右位置。

边注主要由三个长度变量控制：\marginparwidth 是边注的长宽，\marginparsep 是边注与正文的距离，\marginparpush 是边注之间的最小距离。这些长度都可以用 \setlength 设置，不过前面两个变量也可以使用 geometry 宏包设置，这样更为方便（参见 2.4.2 节）。

\marginpar 产生的边注是浮动的，这样当边注内容较多不能直接放下时，实际内容会与插入命令的位置略有变化，如果不想要浮动的边注，可以改用 marginnote 宏包[132] 提供的 \marginnote 命令，禁止不需要的浮动。有时候，浮动的边注在双面模式下会被放在错误的一侧，可以使用 mparhack 宏包[234] 进行修正。

除了边注和脚注，endnotes 宏包还提供了尾注功能，一般用在注释特别多或长，不适于用脚注的情形，详细说明可参见宏包的文档 Lavagnino [144]。

2.2.8　垂直间距与垂直盒子

垂直间距的命令与水平间距的命令是对应的，类似用 \hspace 和 \hspace* 生成水平间距，可以用 \vspace{⟨长度⟩} 和 \vspace*{⟨长度⟩} 生成垂直间距。垂直间距也是弹性距离，可以用 \vfill 表示 \vspace{\fill}。

\vspace 的参数可以是长度的值，也可以是像 \parskip、\itemsep 这样的长度变量。LaTeX 也有一些预定义的垂直间距命令，\smallskip、\medskip、\bigskip 分别表示较小的、中间的和较大的垂直间距：

\smallskip　\medskip　\bigskip

这三个间距的大小由长度变量 \smallskipamount、\medskipamount、\bigskipamount 定义；其具体的值在文档类中定义，默认 \smallskipamount 的值是 3pt plus 1pt minus 1pt，\medskipamount 和 \bigskipamount 分别是它的 2 倍和 4 倍。

\addvspace{⟨长度⟩} 与 \vspace{⟨长度⟩} 的功能类似，只是它在重复使用时只起一个的作用，即 \addvspace{⟨*s1*⟩} \addvspace{⟨*s2*⟩} 相当于 \addvspace{max(⟨*s1*⟩, ⟨*s2*⟩)} 的功能。

为了保证正确的间距效果，这些间距命令一般放在后面一段的开头，而不是前面一段的末尾[①]。

LaTeX 使用两种机制处理断页问题，可以使用命令 \raggedbottom 告诉 LaTeX 让页面中的内容保持它的自然高度，把每一页的页面底部用空白填满。相反，\flushbottom 则让 LaTeX 将页面高度均匀地填满，使每一页的底部直接对齐。在标准文档类中，LaTeX 会为单面输出的文档（oneside 选项）设置 \raggedbottom，而为双面输出的文档（twoside 选项）设置 \flushbottom。当排满一页后，页面剩余空间比较大的时候，如果还要排版一个很高的内容（如多行的公式或表格），就会造成难看的断页，通常这是由浮动环境（参见 5.3.1 节）解决的，但在无可避免的时候就需要在两种断页机制下选择一种：双面印刷的书籍使用 \flushbottom 可以保证摊开时左右两页对称，但如果有太多过于松散的页面就不如使用 \raggedbottom 了。

\pagebreak 可以指定页面断页的位置，它可以带一个 0 到 4 的可选参数，表示建议断页的程度，0 不允许分页，默认值 4 表示必须断页。\nopagebreak 与 \pagebreak 功能和参数的意义相反。例如可以使用 \pagebreak[3] 标示一个特别适合断页的地方，LaTeX 会优先考虑在此处断页。

在遇到一页最后只剩孤立的一行没有排完时，还可以临时用 \enlargethispage{⟨长度⟩} 增加当前页版心的高度，把剩下的一点内容装到当前页，避免难看的断页（尤其是在书籍一章的末尾）。还可以用带星的命令 \enlargethispage*，此时不仅增加页面版心高度，还会适当缩小行距。

可以使用 \newpage 直接对文字手工分页。分页不同于简单的断页，这也是通常我们习惯上所说的换一页的意义。\newpage 实际相当于首先强制分段，然后使用 \vfill 把页面填满，最后用 \pagebreak 换页。连续使用多个 \newpage 并不会多次分页产生空白页，如果需要多次手工分页，可以在空白页使用一个空的盒子 \mbox{} 占位。

\clearpage 与 \newpage 功能类似，也完成填充空白并分页的工作，不同的是它还会清理浮动体（参见 5.3.1 节）；\cleardoublepage 与 \clearpage 类似，只是在双面文档（twoside，参见 2.4.1 节）的奇数页会多分一页，使新的一页也在奇数页。

① 即在分段或 \par 之后。在盒子构造的场合，有时也可以不分段。

在 2.1.5 节我们已经知道了盒子的基本概念。不过，2.1.5 节介绍的盒子都是水平盒子，文字不能在其中分行分段（除非嵌套其他内容）。使用 \parbox 命令或 minipage 环境生成的子段盒子就没有这种限制，垂直盒子在 LaTeX 中也称为子段盒子（parbox），其语法格式如下：

\parbox[⟨位置⟩][⟨高度⟩][⟨内容位置⟩]{⟨宽度⟩}{⟨盒子内容⟩}

\begin{minipage}[⟨位置⟩][⟨高度⟩][⟨内容位置⟩]{⟨宽度⟩}
⟨盒子内容⟩
\end{minipage}

\parbox 和 minipage 环境必须带有一个宽度参数，表示内容的宽度，超出宽度的内容会自动换行：

```
前言\parbox{2em}{不搭后语}。
```

前言不搭
后语。

2-2-70

在垂直盒子中会自动设置一些间距，如段落缩进 \parindent 会被设置为 0pt，段间距会被设置为 0pt（正文默认为 0pt plus 1pt）。

\parbox 和 minipage 环境还可以带三个可选参数，分别表示盒子的基线位置、盒子的高度以及（指定高度后）盒子内容在盒子内的位置。位置参数可以使用 c（居中）、t（顶部）、b（底部），默认为居中；内容位置参数可以使用 c、t、b、s（垂直分散对齐）。其中 s 参数只在有弹性间距时生效。而 t 选项指按第一行的基线对齐，而不是盒子顶端。例如：

```
前言\parbox[t]{2em}{不搭后语}。
后语\parbox[b]{2em}{不搭前言}。
```

不搭
前言不搭。后语前言。
后语

2-2-71

```
\begin{minipage}[c][2.5cm][t]{2em} 两个 \end{minipage}\quad
\begin{minipage}[c][2.5cm][c]{3em} 黄鹂鸣翠柳， \end{minipage}\quad
\begin{minipage}[c][2.5cm][b]{3em} 一行白鹭上青天。 \end{minipage}\quad
\begin{minipage}[c][2.5cm][s]{4em}
\setlength{\parskip}{0pt plus 1pt}% 恢复正文默认段间距
窗含西岭千秋雪，\par
门泊东吴万里船。
\end{minipage}
```

2-2-72

两个黄鹂鸣翠柳，一行白鹭上青天。窗含西岭千秋雪，门泊东吴万里船。

在例 2-1-77 中我们看到了如何利用 lrbox 环境把 \verb 命令产生的抄录内容放在盒子里面使用，但如果把里面的 \verb 换成 verbatim 就会失败，因为 lrbox 里面只能放置水平模式的内容，verbatim 环境则是在垂直模式数行内容。此时，可以在 verbatim 环境外再套一层 minipage，将整个抄录环境的内容看做一个整体[①]：

2-2-73

```
\newsavebox{\verbatimbox} % 通常在导言区定义
\begin{lrbox}{\verbatimbox}
\begin{minipage}{10em}
\begin{verbatim}
#!/bin/sh
cat ~/${file}
\end{verbatim}
\end{minipage}
\end{lrbox}
\fbox{\usebox{\verbatimbox}}\quad\fbox{\usebox{\verbatimbox}}
```

```
#!/bin/sh
cat ~/${file}
```

```
#!/bin/sh
cat ~/${file}
```

除了水平盒子和垂直盒子，还有一种重要的盒子，称为标尺（rule）盒子。标尺盒子用 \rule 命令产生：

\rule[⟨升高距离⟩]{⟨宽度⟩}{⟨高度⟩}

一个标尺盒子就是一个实心的矩形盒子，不过通常只是使用一个细长的标尺盒子画线，这正是“标尺”一词的由来，例如：

2-2-74

```
\rule{1pt}{1em}Middle\rule{1pt}{1em} \\
Left\rule[0.5ex]{2cm}{0.6pt}Right \\
\rule[-0.1em]{1em}{1em} 也可以用作证毕符
号
```

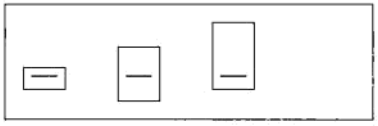
Middle
Left——————Right
■ 也可以用作证毕符号

① 不过，这个功能可以直接使用 fancyvrb 的 SaveVerbatim 环境和 \UseVerbatim 命令。

这里参数 (升高距离) 默认为 0pt，它实际上是设置盒子深度的相反数，效果就像把盒子做了移位一样。

类似幻影 \phantom，LaTeX 有一种宽或高为零的盒子，专门用来占位，称为“支架”(strut)。LaTeX 有预定义的垂直支架 \strut，占有当前字号大小的高度和深度；有时也可以使用长或宽为零的标尺盒子来表示任意大小的支架，例如：

```
\fbox{---}\qquad
\fbox{\strut---}\qquad
\fbox{\rule{0pt}{2em}---}
```



2-2-75

可以用 \raisebox 造成升降的 (水平) 盒子：

\raisebox{(距离)}[(高度)][(深度)]{(内容)}

其中距离为正时盒子里面的内容上升，距离为负时下降。例如我们可以使用 \raisebox 和 \hspace 自己定义 TeX 的标志：

```
% 这与实际 \TeX 的定义基本等价
\mbox{T\hspace{-0.1667em}%
\raisebox{-0.5ex}{E}%
\hspace{-0.125em}X}
```

TEX

2-2-76

\parbox 命令和 minipage 环境产生的垂直盒子必须事先确定宽度，这对于自动折行是十分必要的。然而在个别的场合，我们只是需要盒子的内容能手工分成几行，却又希望它保持按手工分行的“自然”宽度，这时可以改用 varwidth 宏包[18] 提供的 varwidth 环境代替 minipage 环境。varwidth 对应 minipage 的宽度参数，表示盒子的最大宽度，例如：

```
\fbox{\begin{varwidth}{10cm}
自然\\宽度
\end{varwidth}}
```

自然
宽度

2-2-77

不过在 5.1.2 节我们将看到，在一些简单的情况下这可以使用一个单列的表格完成。

TeX 标志

高德纳教授为他的 TeX 软件设计了一个高低起伏的标志：“字母‘E’是不平的，错位的‘E’提醒我们 TeX 是关于排版的，并且也把 TeX 与其他系

统的名字区分开来。”[126] 与此同时，与 TEX 同时设计的字体描述语言 METAFONT[125] 也被赋予了独特的写法——使用以 METAFONT 语言描述并绘制的专用字体排印。

这一独特的命名趣味随即被 TEX 的各种相关工具所效仿。LATEX 的标志是在 TEX 的标志前面加上错排的 L, A 两个字母，而其当前版本 LATEX 2ε 在后面加上 2ε 来表示一个超过 2 而又接近 3 的数字。受 METAFONT 影响而产生的绘图语言 METAPOST 则使用了与前者如出一辙的标志；美国数学会（AMS）编写的各种宏包、字体被冠以 $\mathcal{A}_{\mathcal{M}}\mathcal{S}$ 的称号，这实际是用与 TEX 同样方法错位排布的三个数学花体字；XETEX 与 XELATEX 的标志中有一个反写的字母 E；BIBTEX 中 BIB 三个字母（bibliography 的缩写）则用了小型大写字母，等等。甚至连绘图包 XY-pic、基于 LATEX 的文档处理系统 LYX、TEXMACS 等，也有自己独特的名字和标志。TEX 相关软件的作者们喜欢以此彰显自己的不同。

当然，所有这些标志都不难用 TEX 本身排版出来，例 2-2-76 就给出了 TEX 标志的排版方式，使用类似的方法不难实现各种相关软件的标志。尽管本书充斥着这类软件标志，但它们在实际的书籍文章中其实很少会用到。表 2.12 给出了一些常见标志在 LATEX 中的命令。

表 2.12 TEX 及部分相关软件所使用的标志

标志	命令	额外的宏包
TEX	\TeX	
LATEX	\LaTeX	
LATEX 2ε	\LaTeXe	
METAFONT	\MF	mflogo
METAPOST	\MP	mflogo
$\mathcal{A}_{\mathcal{M}}\mathcal{S}$	\AmS	amsmath
XETEX	\XeTeX	metalogo 或 hologo
XELATEX	\XeLaTeX	metalogo 或 hologo
BIBTEX	\BibTeX	doc 或 hologo

2.3　文档的结构层次

纲举而目张，科技文档首重结构条理。LaTeX 鼓励结构化的文档编写，并提供了许多设置文档结构的手段。这一节我们将看到如何正确使用 LaTeX 提供的命令将文档的内容与格式分离，使文档不仅有良好的输出效果，也有清晰整齐易于维护的源文件。

2.3.1　标题和标题页

LaTeX 的标题文档类提供专门的命令输入文章或书籍的标题。在 LaTeX 中，使用标题通常分为两个部分：声明标题内容和实际输出标题。每个标题则由标题、作者、日期等部分组成。

声明标题、作者和日期分别使用 \title、\author 和 \date 命令。它们都带有一个参数，里面可以使用 \\ 进行换行。标题的声明通常放在导言区，也可以放在标题输出之前的任何位置，例如：

```
\title{杂谈勾股定理\\——勾股定理的历史与现状}
\author{张三\\九章学堂}
\date{庚寅盛夏}
```

2-3-1

\author 定义的参数可以分行，一般第一行是作者姓名，后面是作者的单位、联系方式等。如果文档有多个作者，则多个作者之间用 \and 分隔，例如：

```
\author{张三\\九章学堂 \and 李四\\天元研究所}
```

2-3-2

张三　　李四
九章学堂　　天元研究所

\date 命令可以省略，如果省略，就相当于定义了 \date{\today}，即设置为当天的日期。\today 输出编译文档当天的日期，在英文文档类中默认使用英文（如 April 28, 2013），在中文 ctexart 等文档类中默认使用中文（如 2013 年 4 月 28 日）。使用 ctex 宏包可以用 \CTEXoptions 来设置 \today 的输出格式[58]：

\CTEXoptions[today=small]	2013 年 4 月 28 日
\CTEXoptions[today=big]	二〇一三年四月二十八日
\CTEXoptions[today=old]	April 28, 2013

在声明标题和作者时，可以使用 \thanks 命令产生一种特殊的脚注，它默认使用特殊符号编号，通常用来表示文章的致谢、文档的版本、作者的详细信息等，例如：

```
\title{杂谈勾股定理\thanks{本文由九章基金会赞助。}}
\author{张三\thanks{九章学堂讲师。}\\九章学堂}
```

2-3-3

使用 \maketitle 命令可以输出前面声明的标题，在 1.2.2 节中我们已经看到它的使用，通常 \maketitle 是文档中 document 环境后面的第一个命令。整个标题的格式是预设好的，在 article 或 ctexart 文档类中，标题不单独成页；在 report, book 或 ctexrep, ctexbook 文档类中，标题单独占用一页。也可以使用文档类的选项 titlepage 和 notitlepage 来设置标题是否单独成页。

对于标准文档类，标题的格式是固定好的，LaTeX 并没有提供更多的修改标题格式的命令和选项，如果只是修改字体，可以直接把字体命令放进 \title、\author 等命令中（参见 1.2.8 节）。对于更复杂的标题格式，由于标题在文档中只出现一次，而且不影响文档其他部分的内容，所以文档标题可以通过直接手工设置文字和段落格式排版得到，可以不使用 \title, \maketitle 等命令提供的功能。

单独成页的标题格式通常形式多变，可以在 titlepage 环境中排版。titlepage 环境提供没有页码的单独一页，并使后面的内容页码从 1 开始计数，例如：

```
% 手工排版的标题页
\begin{titlepage}
  \vspace*{\fill}
  \begin{center}
  \normalfont
  {\Huge\bfseries 杂谈勾股定理}

  \bigskip
  {\Large\itshape 张三}

  \medskip
  \today
  \end{center}
  \vspace{\stretch{3}}
\end{titlepage}
```

2-3-4

除了直接手工排版标题，或在声明标题时就加上了字体、字号等命令，也可以使用 titling 宏包详细设置标题、作者、日期在使用 \maketitle 时的输出方式。这个宏包适合模板作者使用，可参见宏包文档 Wilson [291]。

2.3.2 划分章节

LaTeX 的标准文档类可以划分多层章节。在 1.2.2 节中，我们只看到了 \section（节）的层次，事实上，在 LaTeX 中可以使用 6 到 7 个层次的章节，如表 2.13 所示。

表 2.13 章节层次

层次	名称	命令	说明
−1	part（部分）	\part	可选的最高层
0	chapter（章）	\chapter	report, book 或 ctexrep, ctexbook 文档类的最高层
1	section（节）	\section	article 或 ctexart 类最高层
2	subsection（小节）	\subsection	
3	subsubsection（小小节）	\subsubsection	report, book 或 ctexrep, ctexbook 类默认不编号、不编目录
4	paragraph（段）	\paragraph	默认不编号、不编目录
5	subparagraph（小段）	\subparagraph	默认不编号、不编目录

一个文档的最高层章节可以是 \part，也可以不用 \part 直接使用 \chapter（对 book 和 report 等）或 \section（对 article 等）；除 \part 外，只有在上一层章节存在时才能使用下一层章节，否则编号会出现错误。在 \part 下面，\chapter 或 \section 是连续编号的；在其他情况下，下一级的章节随上一节的编号增加会清零重新编号。

例如，在 book 类中，可以使用如下的提纲：

```
\documentclass{book}
\title{Languages}\author{someone}
\begin{document}
\maketitle
\tableofcontents
% 这里用缩进显示层次
\part{Introduction}                          % Part I
  \chapter{Background}                       %   Chapter 1
\part{Classification}                        % Part II
  \chapter{Natural Language}                 %   Chapter 2
  \chapter{Computer Languages}               %   Chapter 3
    \section{Machine Languages}              %     3.1
```

```
13    \section{High Level Languages}          %     3.2
14      \subsection{Compiled Language}        %       3.2.1
15      \subsection{Interpretative Language}  %       3.2.2
16      \subsubsection{Lisp}
17        \paragraph{Common Lisp}
18        \paragraph{Scheme}
19      \subsubsection{Perl}
20  \end{document}
```

2-3-5

可以使用带星号的章节命令（如 \chapter*）表示不编号、不编目的章节。例如教材每章后面的习题往往不编号，也不必放在目录中，就可以用 \section*{习题} 开始习题这一节。

有时，我们希望在正文中的章节题目与在目录、页眉中的不同——正文中使用完整的长标题，目录和页眉使用短标题。可以给命令添加可选参数来做到这一点，如：

```
\chapter[展望与未来]{展望与未来：畅想新时代的计算机排版软件}
```

计数器 secnumdepth 控制除 \part 外，对章节进行编号的层次数，它的默认值为 3，也就是对 book, report 类编号到 \subsection；对 article 类编号到 \subsubsection（见表 2.13）。可以在导言区修改这个计数器的值来修改编号的层次数。

计数器 tocdepth 控制除 \part 外，对章节编入目录的层次数，它的默认值为 3（见表 2.13）。可以在导言区修改此计数器的值。

章节的计数器与其命令同名，如 \chapter 的计数器就是 chapter、\section 的计数器就是 section。直接重定义这些计数器的输出格式可以一定程度上的修改章节格式，详细的设置参见 2.3.4 节。

\appendix 命令用来表示附录部分的开始。命令 \appendix 后面的所有章（对于 book、report 等）或节（对于 article 等）都将改用字母进行编号：如编号的“Chapter 1”（中文文档类为“第一章”）改为“Appendix A”（中文文档类为“附录 A”）。例如某教材的习题解答作为附录的一章，可以用

```
% ...
\appendix
\chapter{习题解答}
% ...
```

标准文档类对附录部分章节格式是固定的，即适当修改了相应的章节格式。可以自己可以重定义 \appendix 命令来设置自己的附录格式（参见 2.3.4 节）。

也可以使用 appendix 宏包[286] 设置附录格式，它提供了更多的命令和选项来控制 \appendix 及相关命令的行为，这里不再赘述。

对于 book 或 ctexbook 类，还可以把全书划分为正文前的资料（front matter）、正文的主要部分（main matter）、后面的附加材料（back matter）。这是由 \frontmatter, \mainmatter 和 \backmatter 控制的，例如：

```
\documentclass{ctexbook}
\title{语言}
\author{张三 \and 李四}
\begin{document}

\frontmatter
\maketitle
\tableofcontents
\chapter{序}                    % 不编号
% ...

\mainmatter                     % 页码重新计数
\chapter{自然语言}
% ...
\chapter{计算机语言}
% ...

\backmatter
\chapter{进一步的参考资料}  % 不编号
% ...
\end{document}
```

这三个命令都会使用 \clearpage 或 \cleardoublepage 另起新页（参见 2.2.8 节、5.3.1 节），输出以前未处理的浮动图表。\frontmatter 会令页码按小写罗马数字编号，并关闭 \chapter 的编号；\mainmatter 会令页码按阿拉伯数字编号；\backmatter 会关闭 \chapter 的编号。

2.3.3 多文件编译

对一篇只有几页纸的文章，把所有的内容都放进一个 TEX 源文件就足够了。但如果要排版更长的内容，例如与本书篇幅相当的文档，单一文件的编译方式就不那么方便了。更好的方式是按文档的逻辑层次，把整个文档分成多个 TEX 源文件，这样文档的内容更便于检索和管理，也适合大型文档的多人协同编写。

LATEX 提供的 \include{(文件名)} 命令可用来导入另一个文件的内容作为一个章节，文件名不用带 .tex 扩展名。\include 命令会在之前和之后使用 \clearpage 或 \cleardoublepage 另起新页（参见 2.2.8 节），同时将这个文件的内容贴到 \include 命令所在的位置[①]。于是，可以按下面的方式组织一本书：

```
% languages.tex
%   整个文档的主文件
\documentclass{ctexbook}
\title{语言}
\author{张三 \and 李四}
\begin{document}
\maketitle
\tableofcontents
\include{lang-natural}
\include{lang-computer}
\end{document}
```

```
% lang-natural.tex
%   “自然语言”一章，不能单独编译
\chapter{自然语言}
……
```

```
% lang-computer.tex
%   “计算机语言”一章，不能单独编译
\chapter{计算机语言}
……
```

① 这个命令实际完成的工作还要多一些，如写不同的辅助文件、判断个别例外不读入等。

这样，就把一本书籍划分成了三个文件：一个主文件 languages.tex，它是文档的中心，编译文档时只要对主文件编译即可；两章内容的文件 lang-natural.tex 和 lang-computer.tex，每个文件只有一章的内容。如果这本书由张三和李四分工编写，在他们共同写好 languages.tex 这个提纲以后，就可以分头编写两章的内容了，二人的工作可以互不影响。

使用 \include 划分文档以后有一个特别的便利，就是可以通过修改主文件的几行来选择编译整个文档的某一章或某几章。当然可以把不要的章节注释掉来达到这个目的，不过更好的办法是使用 \includeonly{⟨文件列表⟩} 命令，其中 ⟨文件列表⟩ 是用英文逗号隔开的若干文件名。在导言区使用 \includeonly 命令以后，只有文件列表中的文件才会被实际地引入主文件。更好的是，如果以前曾经完整地编译过整个文档，那么在使用 \includeonly 选择编译时，原来的章节编号、页码、交叉引用等仍然会保留为前一次编译的效果，例如：

```
% languages.tex
%   整个文档的主文件
\documentclass{ctexbook}
\title{语言}
\author{张三 \and 李四}
\includeonly{lang-natual}        % 只编译“自然语言”一章
\begin{document}
\maketitle
\tableofcontents
\include{lang-natural}
\include{lang-computer}
\end{document}
```

使用 \include 命令需要注意的是，最好不要在子文件中新定义计数器、声明新字体，否则在使用 \includeonly 时，会因为找不到出现在辅助文件中而在源文件中缺失的计数器而出错。

比 \include 命令更一般的是 \input 命令，它直接把文件的内容复制到 \input 命令所在的位置，不做其他多余的操作。\input 命令接受一个文件名参数，文件名可以带扩展名，也可以不带扩展名（此时认为扩展名是 .tex）。例如，如果一个文档的导言区设置非常多，可以把导言区的设置都放在一个 preamble.tex 文件中，然后在主文件中就可以这样写：

```
% main.tex
```

```
%   主文档
\documentclass{ctexart}
\input{preamble}    % 复杂的导言区设置
\begin{document}
…… (文档的内容)
\end{document}
```

除了导言区，经常也把复杂图表代码放在一个单独的文件中，然后在主文件中使用 \input 命令插入，这样可以使文档的正文部分看起来比较清爽，图表的代码也可以被另外的专用主文档引入单独进行测试。另外，如果需要引入的文件只是 article 中的一节，不需要多余的换页时，也可以用 \input 命令代替 \include 命令引入不换页的小的章节文件。

在被引入的文件末尾，可以使用 \endinput 命令显式地结束文件的读入。在 \endinput 命令的后面，就可以直接写一些注释性的文字，而不必再加注释符号，例如：

```
% lang-natural.tex
\chapter{自然语言}
……
\endinput
这是“自然语言”一章，不能单独编译。要编译文档，直接编译主文件：
  xelatex languages.tex
```

主文件中的 \end{document} 命令也有类似的功能。

对于大型文档，在编写中有时只需要 LaTeX 编译程序检查语法，并不需要实际输出。此时可以使用 syntonly 宏包的 \syntaxonly 命令[168]：

```
% languages.tex
%   整个文档的主文件
\documentclass{ctexbook}
\usepackage{syntonly}
\syntaxonly     % 只检查语法，不输出 DVI/PDF 文件
\begin{document}
……
\end{document}
```

2-3-6

使用 syntonly 宏包编译文档时不输出 DVI/PDF 文件，速度比直接编译输出要快一些，可以节约文档编写的时间。

2.3.4　定制章节格式

LaTeX 标准文档类 book, report, article 的章节格式是固定的，章节标题用什么字体、字号，前后的间距如何，用怎样的编号方式等，都是预定好的，并没有提供直接进行控制的命令，一般都要通过其他的宏包完成相关的定制。

ctex 宏包的三个文档类 ctexbook, ctexrep, ctexart 使用的默认章节格式与英文标准文档类的格式略有区别。ctex 宏包还提供了 \CTEXsetup 命令来设置章节标题的格式[58, § 2.5]，其语法格式如下：

\CTEXsetup[⟨选项*1*⟩=⟨值*1*⟩, ⟨选项*2*⟩=⟨值*2*⟩, ...]{⟨对象类型⟩}

其中 ⟨对象类型⟩ 可以是 part, chapter, section, subsection, subsubsection, paragraph, subparagraph; ⟨选项*1*⟩, ⟨选项*2*⟩ 等是设置的选项，包括 name, number, format, nameformat, numberformat, titleformat, aftername, beforeskip, afterskip, indent；而 ⟨值*1*⟩, ⟨值*2*⟩ 等是对应选项的设置内容。

下面逐条解释 \CTEXsetup 的选项：

☞ **name={⟨前名⟩,⟨后名⟩}**

用来设置章节的名字，包括章节编号前面和后面的词语，前后两个名字之间用西文逗号分开，例如：

```
\CTEXsetup[name={第,节}]{section}
```

将设置 \section 的标题的名字类似“第 1 节”。

☞ **number={⟨编号格式⟩}**

相当于设置章节的计数器的 \the⟨*counter*⟩ 命令，例如：

```
\CTEXsetup[number={\chinese{section}}]{section}
```

如果配合前面章节名的设置，将产生类似“第一节”的名字。

☞ **format={⟨格式⟩}**

用于控制章节标题的全局格式，作用域为章节名字和随后的标题内容。常用于控制章节标题的对齐方式，例如设置节标题左对齐，整体粗体：

*本节内容初次阅读可略过。

```
\CTEXsetup[format={\raggedright\bfseries}]{section}
```

☞ **nameformat={⟨格式⟩}**

类似 format，控制章节名和编号的格式，不包括后面的章节标题。

☞ **numberformat={⟨格式⟩}**

类似 format，仅控制编号的格式，不包括前后章节名和章节标题。

☞ **titleformat={⟨格式⟩}**

类似 format，仅控制章节标题的格式，不包括前面的章节名和编号。

☞ **aftername={⟨代码⟩}**

控制章节名和编号（如“第一章”）与后面标题之间的内容。通常设置为一定间距的空格，或换行（换行仅对 chapter 和 part 有效），例如：

```
\CTEXsetup[aftername={\\\vspace{2ex}}]{part}
```

☞ **beforeskip={⟨长度⟩}**

控制章节标题前的垂直距离。

☞ **afterskip={⟨长度⟩}**

控制章节标题后的垂直距离。

☞ **indent={⟨长度⟩}**

控制章节标题的缩进长度。

ctex 的文档类中，\CTEXsetup 的选项的默认值可进一步参见文档 ctex.org [58]。

在 \CTEXsetup 命令中，可以使用 += 代替 =，表示在默认选项的基础上增加格式设置。例如，如果要让每小节 \subsection 的标题使用仿宋体，就可以用命令：

```
\CTEXsetup[titleformat+={\fansong}]{subsection}
```

ctex 宏包提供的 \CTEXsetup 只能用于中文文档，而且支持的格式还不够多变自由，难以生成更复杂的章节标题格式。在西文文档中，或是要设置更复杂的章节标题（如本书），可以使用 titlesec 宏包。这里只对一些初级命令做简单介绍，详细的设置请参见 Bezos [27]。

titlesec 宏包使用时可以带几个可选参数，用来全局地修改标题的字体、对齐方式。选项

```
rm  sf  tt  bf  up  it  sl  sc
```

用来设置标题使用的字体族、字体系列和字体形状，默认是 bf，即普通字体的粗体系列。选项

```
big  medium  small  tiny
```

用来设置标题的字号，默认是最大的 big，最小号的 tiny 则使所有标题字体与正文大小相同，medium 和 small 介于最大和最小之间。选项

```
raggedright  center  raggedleft
```

用来设置标题的对齐方式，例如：

```
\usepackage[sf,bf,it,centering]{titlesec}
```

2-3-7

将设置章节标题的字体为 \sffamily\bfseries\itshape，居中。

宏包选项 compact 会使章节标题前后的垂直间距比较紧。

\titlelabel{⟨代码⟩} 命令用来设置编号标签的格式，⟨代码⟩ 中可以使用 \thetitle 指代 \thesection、\thesubsection 等命令。对标准文档类，默认设置是：

```
\titlelabel{\thetitle\quad}
```

可以修改为需要的格式，例如：

```
\titlelabel{\S~\thetitle\quad}
```

2-3-8

将在所有的章节编号前加符号 §。

带星号的 \titleformat* 命令则可以作为宏包选项的补充，用来设置某一级标题的格式，例如：

```
\titleformat*{\section}{\Large\itshape\centering}
```

2-3-9

将使节标题使用 \Large 字号的意大利体并居中。这里的设置将会覆盖默认的和在宏包选项中出现的全局设置，因此一般对字体、字号、对齐方式等都要进行设置。

titlesec 的其他功能包括使用不带星号的 \titleformat 命令对章节格式做更详细的设置，使用 \titlespace 命令设置章节标题的间距，增添横线，控制页面版式（可代替 fancyhdr，参见 2.4.3 节），控制章节分页，增加新的章节层次等功能，篇幅所限，这里就不再一一介绍了。

练 习

2.6 试试看，下面代码的作用是什么？

```
\CTEXsetup[name={\S,}, number={\arabic{section}}, afternname={---},
  format={\raggedright}, nameformat={\Large\bfseries},
  titleformat={\LARGE\sffamily}, indent={1pc}]{section}
```

使用 titlesec 宏包的功能重新实现上面的设置（可以只考虑 \section）。

2.7 本书的章节格式就是用 titlesec 宏包设置的。查阅文档，试试你能否仿做出本书的章节格式。

2.4 文档类与整体格式设计

这一节，我们主要将注意力集中在导言区的代码，讨论文档的一些全局设置，特别是版面设计的相关内容。

2.4.1 基本文档类和 ctex 文档类

文档类（document class）是 LaTeX 2ε 中基本的格式组织方式。文件类通常都是针对某一类格式相近的文档设计的，从适用范围广泛的“文章”、“书籍”到非常专门的“某大学博士论文模板”，不一而足。选定了一个文档类，就相当于选定了很大的一集 LaTeX 格式，可以在一个规范的框架下进行文档编写。

本书的大部分内容都是基于 LaTeX 2ε 的基本文档类和 ctex 文档类叙述的。ctex 文档类是由 CTeX 中文社区[①]组织编写的 LaTeX 2ε 基本文档类的中文对应物，它是在 LaTeX 2ε 基本文档类的基础上编写的。LaTeX 2ε 基本文档类和 ctex 文档类主要提供了文档的整体框架（如 \maketitle, \section 等命令）、基本设置（如默认字号、默认的中文字体）、基本工具（如 enumerate 等环境）等，它们提供了最基本的通用文档格式，但并不对文档的适用性做过多限制。下面就对这两组适用最广的文档类做一简单介绍。

LaTeX 2ε 基本文档类主要有三个：article, report 和 book。这三个基本文档类分别设计用来编写小篇幅的文章、中篇幅的报告和长篇幅的书籍。三个文档类的格式都很简单，提供的命令也相差不多，只有少数区别，如 article 没有 \chapter，\mainmatter 只有 book 类才有等。基本文档类的许多格式是可以通过选项调整的，例如：

2-4-1
```
\documentclass[a4paper,titlepage]{article}
```

将设置文档为 A4 纸大小，标题单独占一页。

基本文档类预定义的所有选项总结见表 2.14。

单面文档的奇偶数页面是相同的，双面文档则不同。对双面印制的文档，通常在页面左侧装订，翻开后奇数页一般在右边，偶数页在左边，因而在页面边距、页眉页脚、边注位置等方面都与单面文档有所区别，大多取左右对称的设置。如果选定 twoside

① http://bbs.ctex.org/

表 2.14　LaTeX 2ε 标准文档类的选项

类型	选项	说明
纸张大小	a4paper	21.0 cm × 29.7 cm
	a5paper	14.8 cm × 21.0 cm
	b5paper	17.6 cm × 25.0 cm
	letterpaper	8.5 in × 11 in　（默认值）
	leagalpaper	8.5 in × 14 in
	executivepaper	7.25 in × 10.5 in
纸张方向	landscape	横向，即长宽交换　（默认无，即为纵向页面）
单双面	oneside	单面　（article, report 默认值）
	twoside	双面，奇偶页面版式不同，左右对称　（book 默认值）
字号大小	10pt	正文字号为 10 pt，\large 等命令与之相衬（默认值）
	11pt	正文字号为 11 pt
	12pt	正文字号为 12 pt
分栏	onecolumn	单栏　（默认值）
	twocolumn	双栏
标题格式	titlepage	标题独自成页　（report, book 默认值）
	notitlepage	标题不独自成页　（article 默认值）
章格式	openright	每章只从奇数页开始　（book 默认值）
	openany	每章可从任意页开始　（article, report 默认值）
公式编号	leqno	公式编号在左边　（默认无，即公式编号在右边）
公式位置	fleqn	公式左对齐，固定缩进　（默认无，即公式居中）
草稿设置	draft	草稿，会把行溢出的盒子着重显示为黑块
	final	终稿　（默认值）
参考文献	openbib	每条文献分多段输出　（默认无）

模式，可以使用 openright 使每个 \part 和 \chapter 都只出现在右边的页面（奇数页），之前不足的页面用空白补足。

纸张大小、方向的设置可进一步参见 2.4.2 节，字号参见 2.1.4 节，分栏参见 2.4.4 节，标题参见 2.3.1 节 titlepage 环境的说明，titlepage 选项也会使摘要独自成页，公式设置参见 4.5 节，参考文献参见 3.3.5 节。

基本文档类的默认选项总结如下：

article	letterpaper,10pt,oneside,onecolumn,notitlepage,final
report	letterpaper,10pt,oneside,onecolumn,titlepage,openany,final
book	letterpaper,10pt,twoside,onecolumn,titlepage,openright,final

LaTeX 在文档类中的选项是全局设定的，不仅会影响文档类的代码，也会影响文档所使用的宏包。例如文档类中的纸张选项也会影响 geometry 宏包，草稿设置选项也会影响插图的 graphicx 宏包（参见 5.2.1 节）等。

ctex 宏包[58] 提供了三个文档类：ctexart, ctexrep 和 ctexbook，分别与三个标准文档类对应，用来编写中文短文、中文报告和中文书籍。除了三个文档类，ctex 宏包还包括 ctex.sty 和 ctexcap.sty 两个格式文件，以及上述文档类和格式文件的 UTF8 编码变体。

ctex.sty 提供基本的中文输出支持，ctexcap.sty 则提供 ctex.sty 的全部功能和英文标题的汉化，几个文档类则在 ctexcap 的基础上进一步设置默认的字号大小为中文字号。在大多数情况下，我们都直接使用 ctex 提供的文档类，但在一些非标准的文档类中，则可以按需要使用 ctexcap 或 ctex 格式。例如，在编写宏包文档的 ltxdoc 文档类（基于标准文档类）中，可以使用 ctexcap 支持中文[①]：

```
\documentclass{ltxdoc}
\usepackage{ctexcap}
\zihao{5}
\begin{document}
.....
\end{document}
```

而在已经失去标准文档类原有结构的个人简历文档类 moderncv 中，就可以使用 ctex 支持中文：

① 在更新版本中的 ctex 宏包将简化不同格式的使用，ctex 格式将默认带有 ctexcap 及文档类的汉化、字号设置等功能，此时 ctexcap 就不再需要了。

```
\documentclass{moderncv}
\usepackage{ctex}
\zihao{-4}
\begin{document}
.....
\end{document}
```

2-4-2

表 2.14 中列出的标准文档类的选项也可在 ctex 文档类中使用。ctex 宏包及文档类的其他选项（这里去掉了专用于旧的中文处理方式 CJK 和 CCT 的选项）见表 2.15。

表 2.15　ctex 宏包及文档类的选项

类型	选项	说明
字号大小	**c5size**	正文五号字　（仅用于文档类，默认值）
	cs4size	正文四号字　（仅用于文档类）
章节标题	**sub3section**	使 \paragraph 标题单独占一行　（仅用于 ctexcap 及文档类，默认无）
	sub4section	使 \paragraph 和 \subparagraph 标题都单独占一行　（仅用于 ctexcap 及文档类，默认无）
中文编码	**GBK**	使用 GBK 编码，但对 XeTeX 无效
	UTF8	使用 UTF8 编码
XeTeX 中文字库	**nofonts**	不设定中文字体，需要在文中自己定义
	winfonts	设定 Windows 操作系统预装的中文字体　（默认值）
	adobefonts	设定 Adobe 中文字体
排版风格	**cap**	使用中文标题、编号、日期等　（默认值）
	nocap	保留英文标题、编号、日期等
	punct	启用标点压缩　（默认值）
	nopunct	关闭标点压缩
	indent	标题后首行缩进　（默认值）
	noindent	标题后首行不缩进
宏包兼容	**fancyhdr**	调用 fancyhdr 并与之兼容　（默认无）
	hyperref	调用 hyperref 并自动设置防止标签乱码选项　（默认无）

续表

类型	选项	说明
	fntef	调用 CJKfntef 并定义等价的 \CTEX 开头的命令（默认无）

\paragraph 和 \subparagraph 的标题在标准文档类中与其后的正文段在同一行，使用 sub3section 和 sub4section 选项可以使它们的标题单独占用一行。

ctex 宏包有选择编码的选项 GBK 和 UTF8，但它们主要是为旧方案中使用 CJK 宏包处理中文所准备的；在 XeLaTeX 编译的文档中，总是相当于使用了 UTF8 编码选项。如果需要使用 XeLaTeX 处理 GBK 编码的中文文档，可以在每个文件的开头使用 XeTeX 原始命令 \XeTeXinputencoding 选择编码：

2-4-3
```
\XeTeXinputencoding "GBK"
\documentclass{ctexart}
\begin{document}
GBK 编码的中文文档。
\end{document}
```

ctex 宏包以前对 UTF-8 编码的支持是由另外单独的文件而不是选项实现的，现在为了兼容也保留了 ctexutf8, ctexcaputf8 两个格式和 ctexartutf8, ctexreputf8, ctexbookutf8 三个文档类，相当于加了 UTF8 选项。

关于中文字库的选项请参见 2.1.3.2 节，标点压缩请参见 2.1.1.2 节，fancyhdr 选项请参见 2.4.3 节，fntef 选项请参见 2.1.3.3 节。关于 ctex 宏包提供的其他功能可进一步参考宏包的文档 ctex.org [58]，这里不再赘述。

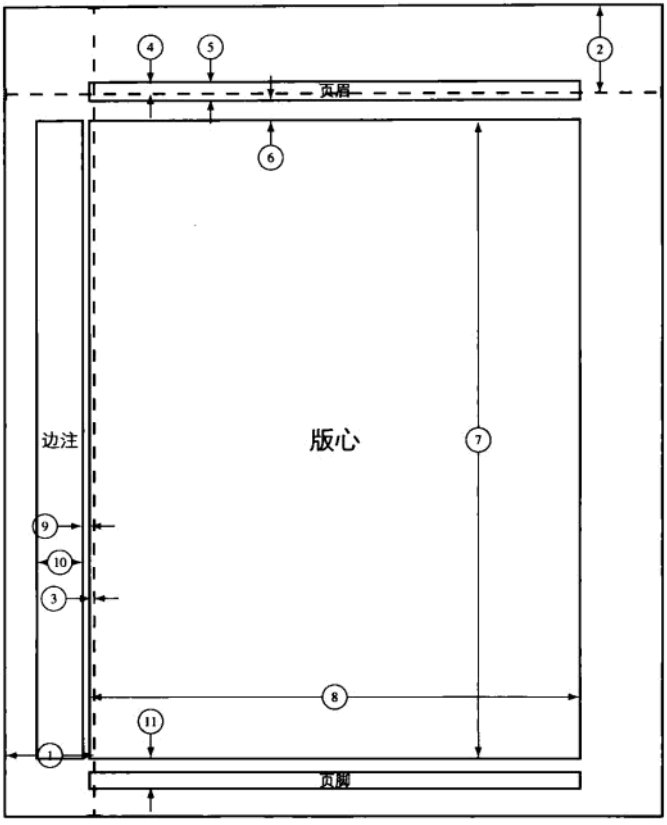
2.4.2 页面尺寸与 geometry

在文档类的选项中，可以选择几种常见的排版所用的纸张页面大小。不过，实际的排版往往要求设置更为复杂的页面尺寸参数，下面我们就来考量有关页面尺寸的设置。

如图 2.4 所示[①]，LaTeX 的页面尺寸布局是由一系列长度变量控制的。其中 \paperwidth 和 \paperheight 是纸张的宽和高；\hoffset 和 \voffset 是减去一英寸

*本节内容初次阅读可略过。

① 这个页面示意图是利用 layout 宏包[157] 绘制的。



1	一英寸 + \hoffset	2	一英寸 + \voffset
3	\evensidemargin = -3.98 pt	4	\topmargin = -9.67 pt
5	\headheight = 14.05 pt	6	\headsep = 18.07 pt
7	\textheight = 525.87 pt	8	\textwidth = 398.34 pt
9	\marginparsep = 7 pt	10	\marginparwidth = 36.14 pt
11	\footskip = 25.29 pt		\marginparpush = 5 pt（未显示）
	\hoffset = 45.52 pt		\voffset = 42.68 pt
	\paperwidth = 534.91 pt		\paperheight = 668.64 pt

图 2.4　本书的页面尺寸设置示意图（偶数页）

的页面整体偏移量[①]；\textwidth 和 \textheight 是版心的宽和高；\topmargin 是额外的上边距，\oddsidemargin 和 \evensidemargin 在双面模式下分别是奇数页和偶数页的额外左边距（对单面模式，页偶数页都是 \oddsidemargin）；\headheight 是页眉高，\headsep 是页眉与版心间距；\marginparwidth 是边注宽，\marginparsep 是边注与版心间距，\marginparpush 是相邻边注的最小间距；\footskip 是页脚基线与正文最后一行基线的间距。

可以直接设置图 2.4 中的长度变量来控制页面尺寸，不过设置这些参数时必须非常小心，比如当我们想要设置页面的右边距（不含边注）为 3 cm 时，就必须按下面的关系式做一番计算：

$$1\,\text{in} + \texttt{\textbackslash hoffset} + \texttt{\textbackslash oddsidemargin} + \texttt{\textbackslash textwidth} + \text{右边距} = \texttt{\textbackslash paperwidth}$$

从中反解出需要调整的参数值来。在涉及双面文档时这些计算和调整的工作尤为麻烦。

geometry 宏包把我们从 LaTeX 众多相互影响的页面参数中解救出来，提供了一个设置页面参数相对简单直观的用户界面。例如，设置右边距的工作就可以简化为一句 \geometry{right=3cm} 的命令。

geometry 主要提供两种设置页面的方式，一是作为宏包的选项，例如设置纸张大小和左右边距：

2-4-4
```
\usepackage[a4paper,left=3cm,right=3cm]{geometry}
```

二是使用 \geometry 命令，内容和宏包选项一样：

2-4-5
```
\usepackage{geometry}
\geometry{a4paper,left=3cm,right=3cm}
```

geometry 宏包支持方便的 ⟨参数⟩ = ⟨值⟩ 的语法，并且可以根据命令给出的页面距离值和其他参数的默认值，自动计算原始 LaTeX 的页面参数进行设置，十分方便。

图 2.5 来自 geometry 的文档 Umeki [272]，它展示了 geometry 宏包最常用的距离参数名称。这些参数与前面给出的 LaTeX 标准的页面长度变量名字相同或更为简化。同时，geometry 宏包也支持 a4paper，landscape 这样的纸张参数，并且可以在输出 PS/PDF 时也对页面进行剪裁。

使用 geometry 宏包不需要给出完全的参数设置，有时甚至可以相当模糊，比如可以用 centering 选项设置版心居中，使用 scale = ⟨比例⟩ 选项设置版心占页面长度的比例，使用 ratio = ⟨比例⟩ 选项设置版面边距占页面长度的比例，使用 lines = ⟨行数⟩ 设置版心高度在默认字体和行距下能容纳的文本行等。例如，可以不使用任何长度就设置好文档的页面参数：

① 这里的一英寸是通常打印机驱动默认设置的页面与纸张边距。

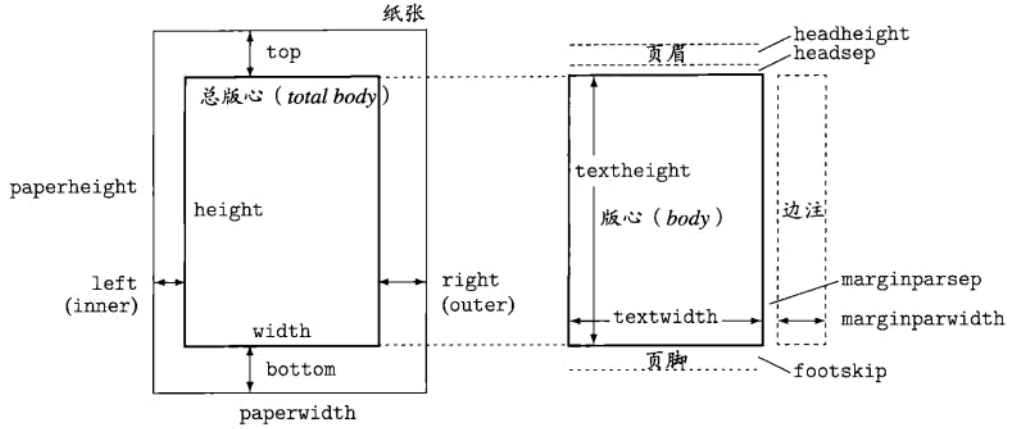


图 2.5　geometry 宏包的距离参数名称。默认有 width = textwidth 及 height = textheight。left, right, top 和 bottom 是左、右、上、下 4 个边距。使用 twoside（双面）选项后，left 和 right 的方向交换，即分别表示（靠近装订线的）内部和外部，此时可以分别用 inner 和 outer 表示

```
\geometry{b5paper,scale=0.8,centering}
```

除了上面所介绍的，geometry 提供的参数选项还有很多，如用来调试的 showframe 选项、设置编译引擎的 pdftex, xetex 选项等，读者可进一步参考宏包文档 Umeki [272]，这里不再赘述。

2.4.3　页面格式与 fancyhdr

2.4.2 节中已经对页面的大小尺寸有了一个整体的设置，本节来进入页面，看看页面的格式设置——也就是页码、页眉、页脚等地方的设计。

页码的计数器是 page，它会随着文档自动计数。LaTeX 提供了一个简单的命令 \pagenumbering{⟨格式⟩} 来设置页码的编号方式，这个命令会令页码重新从 1 开始按 ⟨格式⟩ 参数进行计数，例如：

```
\pagenumbering{roman}
```

2-4-6

*本节内容初次阅读可略过。

相当于设置 \thepage 为 \roman{page}，并设置计数器值为 1。book 类中的 \frontmatter 等命令就有控制页码编号的作用，而在 report 和 article 类中就必须用 \pagenumbering 了。

LaTeX 提供了多种预定义的页面风格（page style），它们控制页眉页脚的整体风格设置：

empty	没有页眉页脚；
plain	没有页眉，页脚是居中的页码；
headings	没有页脚，页眉是章节名称和页码；
myheadings	没有页脚，页眉是页码和用户自定义的内容。

可以用 \pagestyle{⟨风格⟩} 整体设置页面风格，也可以用 \thispagestyle{⟨风格⟩} 单独设置当前页的风格。标准文档类中，book 类默认使用 headings 风格，report 和 article 默认使用 plain 风格；中文的几个 ctex 文档类则都默认使用 headings 风格[①]，例如，可以在 article 类中设置带章节名称的页眉：

2-4-7
```
\documentclass{article}
\pagestyle{headings}
```

又如，可以在插入大幅图片的页面使用 plain 风格，取消复杂的页眉：

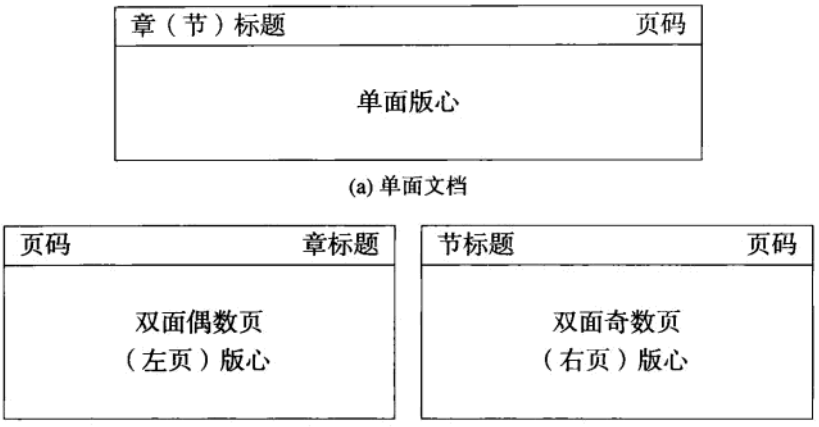
2-4-8
```
\begin{figure}[p]
  \thispagestyle{plain}
  ..........
\end{figure}
```

LaTeX 已经对一些必要的地方自动设置好了页面风格。例如在标题页（包括手工或自动由 \maketitle 生成的 titlepage 环境），会使用 empty 风格禁用所有页眉页脚；而在不单独成页的 \maketitle，单独成页的 \part，以及 \chapter 命令所在的一页，则使用 plain 风格只显示页码：这些都是排版中的一些定式。

headings 和 myheadings 两种风格是由标准文档类定义的，它们的表现其实基本相同，都是在页眉显示页码和一些文字。不同的是，headings 风格的页眉内容不能改变，它是由 \chapter、\section 等命令自动生成的（如图 2.6 所示），而 myheadings 风格的页眉可以由用户自己使用 \markright 和 \markboth 命令设置：

单面文档（**oneside**）	**\markright{**⟨页眉文字⟩**}**
双面文档（**twoside**）	**\markboth{**⟨左面页眉⟩**}{**⟨右面页眉⟩**}**

① 如果使用了 fancyhdr 选项，则为 fancy 风格。



(a) 单面文档

(b) 双面文档

图 2.6　标准文档类的 headings 页面风格

这里“右面”指奇数页，“左面”指偶数页，因为当双面文档在侧面装订好后，翻开的页面正好是右奇左偶，单面文档的所有页面都认为是右面。\markright 和 \markboth 命令实际会修改 \leftmark 和 \rightmark 两个宏的内容，并在页眉处输出。例如，如果想让文章在每页的页眉显示作者的名字，就可以使用：

```
\documentclass{ctexart}
\pagestyle{headings}
\markright{张三}
```

2-4-9

LaTeX 和标准文档类提供的页面风格非常朴素，而且不能做进一步的格式修改。为此，fancyhdr 宏包提供了新的页面风格 fancy，以及一系列设置的命令，用作标准 LaTeX 的 myheadings 及 \markboth, \markright 机制的扩展。

fancyhdr 的 fancy 页面风格把页面的页眉和页脚都分成左、中、右 3 个部分，因而一个页面就有 6 个部分。对于双面文档，则还分奇数页和偶数页，即有 12 个部分（见图 2.7）。

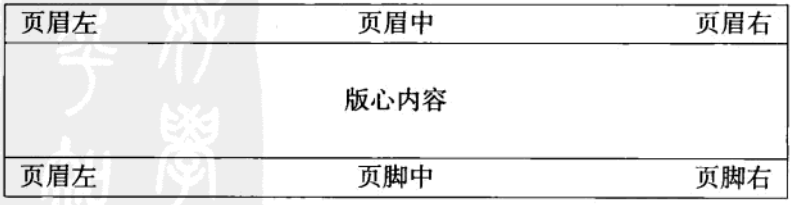


图 2.7　fancyhdr 宏包的 fancy 页面风格

图 2.7 中的各个部分可以用下列命令进行设置修改：

`\lhead{⟨内容⟩}`	设置页眉左
`\chead{⟨内容⟩}`	设置页眉中
`\rhead{⟨内容⟩}`	设置页眉右
`\lfoot{⟨内容⟩}`	设置页脚左
`\cfoot{⟨内容⟩}`	设置页脚中
`\rfoot{⟨内容⟩}`	设置页脚右
`\fancyhead[⟨位置⟩]{⟨内容⟩}`	设置页眉，位置可以是 E、O 与 L、C、R 的组合
`\fancyfoot[⟨位置⟩]{⟨内容⟩}`	设置页脚，位置可以是 E、O 与 L、C、R 的组合
`\fancyhf[⟨位置⟩]{⟨内容⟩}`	设置页眉及页脚，位置可以是 H、F 与 E、O 与 L、C、R 的组合

这里，\fancyhead、\fancyfoot 和 \fancyhf 命令可以带表示位置的可选参数，其中 H、F 分别表示页眉（header）和页脚（footer）；E、O 分别表示双面文档的偶数页（even page）和奇数页（odd page），单面文档仅奇数页有效；L、C、R 分别表示左（left）、中（center）、右（right）。位置参数可以任意组合，多个参数用逗号分隔。如果省略位置参数，则表示所有的页眉、页脚。例如：

```
\documentclass[twoside]{ctexrep}
\usepackage{fancyhdr}
\pagestyle{fancy}          % 使用 fancy 风格
\fancyhf{}                 % 清除所有页眉页脚
\cfoot{\thepage}           % 页脚居中页码
\fancyhead[CO]{张三}       % 奇数页居中页眉作者名
\fancyhead[CE]{论语言}     % 偶数页居中页眉文章题目
\fancyfoot[RO,LE]{$\heartsuit$}
    % 奇数页脚右，偶数页脚左（即外侧）装饰符号
```

2-4-10

在 fancy 页面风格的设置中，可以在页眉页脚的内容中使用 \leftmark 和 \rightmark 命令，它们的意义与 headings 风格中的页眉相同，即为文档的章节标题内容：article 只有 \rightmark 是节标题；report 和 book 的 \leftmark 是章标题，\rightmark 是节标题。事实上，fancy 风格的默认设置就是：

```
\fancyhead[LE,RO]{\slshape \rightmark}
\fancyhead[LO,RE]{\slshape \leftmark}
\fancyfoot[C]{\thepage}
```

在 ctex 宏包提供的文档类中，可以使用 fancyhdr 选项，表示使用 fancyhdr 宏包及 fancy 页面风格，因而例 2-4-10 中的前几行也可以简化为：

```
\documentclass[twoside,fancyhdr]{ctexrep}
\fancyhf{}
% ......
```

2-4-11

除了页眉页脚的内容，fancy 页面风格还会给页眉和页脚加一条横线。可以重定义宏 \headrulewidth 和 \footrulewidth 来修改页眉线和页脚线的宽度，如果宽度为零就是没有页眉页脚线，注意它们只是文本宏而不是长度变量，如：

```
\renewcommand\headrulewidth{0.6pt}  % 默认为 0.4pt
\renewcommand\footrulewidth{0.6pt}  % 默认为 0pt
```

2-4-12

使用 fancyhdr 还可以使用 \fancypagestyle 命令重定义原有的页面风格，通常可以用它来重定义 plain 风格，这样在每章的第 1 页等位置也可以使用特殊的页面风格，如：

```
\fancypagestyle{plain}{%
  \fancyhf{}
  \cfoot{--\textit{\thepage}--}      % 改变页码形状
  \renewcommand\headrulewidth{0pt}  % 无页眉线
  \renewcommand\footrulewidth{0pt}  % 无页脚线
}
```

2-4-13

fancyhdr 宏包的功能大致就是这么多，有关页面设置的更多命令和技巧，可参见 fancyhdr 的文档 van Oostrum [190]。

titlesec 宏包使用 pagestyles 选项时，也提供了与 fancyhdr 类似的命令，如 \sethead, \setfoot, \renewpagestyle 等，可以完全代替 fancyhdr 的功能，可参见文档 Bezos [27, § 5]，这里不再赘述。

2.4.4　分栏控制与 multicol

给文档类加 twocolumn 选项就可以使文档双栏排版。双栏排版的文档通常比较节省纸张，较短的行在阅读时也比较省力。书籍的索引默认就是双栏排版的，许多期刊都使用双栏排版，参考文献等也常常分栏。例如：

```
\documentclass[twocolumn]{article}
% ...
```

分栏也可以在正文中使用命令切换。\twocolumn 进入双栏模式，\onecolumn 进入单栏模式，两个命令都会先使用 \clearpage 换页，并不产生一页之内单双栏混合的效果。例如，可以在书籍的某一章：

```
\twocolumn
\chapter{双栏的一章}
\onecolumn
```

\twocolumn 命令可以带一个可选参数，在新开始的双栏页面顶部插入一部分单栏的内容。这个功能特别适合有通栏标题的双栏文章，例如：

```
% 文档类选项并不使用双栏
\documentclass{article}
\title{Languages}
\author{someone}
\begin{document}
% 通栏标题
\twocolumn[\maketitle]
% 双栏的正文
blah blah blah...
\end{document}
```

在双栏模式下，\newpage 和 \pagebreak 只表示分栏，不表示分页，此时可以使用 \clearpage 和 \cleardoublepage 完成分页或进入双面奇数页的功能。

栏与栏的间距是由长度变量 \columnsep 控制的，可以使用 \setlength 等命令自行修改。栏宽的变量是 \columnwidth，对双栏文档，它的大小等于 (\textwidth − \columnsep)/2，可以把它用在设置其他长度的地方，但不要手工修改这个值。

栏与栏之间有一条竖线分隔，竖线的宽度由长度变量 \columnseprule 控制。\columnseprule 默认值为 0 pt，也就是没有竖线，可以设置一个大于 0 的宽度增加分栏线，通常的线宽是 0.4 pt：

```
\setlength{\columnseprule}{0.4pt}
```

双栏文档的一页内容如果没有填满，默认是左右不平衡的，内容会先填满左边一栏，再填充右边一栏（见图 2.8）。使用 balance 宏包[61] 提供的 \balance 命令可以让页面左右平衡，\nobalance 命令则恢复不平衡的双栏页面，如：

```
\documentclass[twocolumn]{article}
\usepackage{balance}
\balance
% ...
```

multicol 宏包[162] 提供了更为强大的分栏功能，它可以在不另起一页的情况下把页面的后半部分分成多栏，也可以随时停止分栏或改变栏数。multicol 宏包提供的分栏也是平衡的。宏包提供的 multicols 环境完成所有的工作，注意在 multicols 宏包中不能使用浮动体和边注。宏包不再使用 \newpage 强制分栏，而使用 \columnbreak。例如：

```
% 导言区
% \usepackage{multicol}
% 正文
\begin{multicols}{3}
```

```
分成三栏的内容……
\end{multicols}
```

flowfram 宏包[247] 在这方面走得更远，它可以把文档的文本分成好几个文本流，每个文本流是一个固定位置的块，让正文按顺序通过各个块。报纸和时尚杂志常常把版面拆分成许多“豆腐块”，然后把文章排列在各个“豆腐块”中，这种复杂的版面就可以用 flowfram 来实现，有兴趣的读者可以参考宏包的文档，这里不再详细介绍。

grid 宏包[208] 提供了一组选项和环境，可以用来设置页面的高度和行数，控制各类环境的高度，以保证在双栏排版中让左右两栏的文字能逐行对齐，这种对于某些双栏期刊是非常有用的功能。本节没有使用这种机制，所以左右两列的文字行可能是参差不齐的。

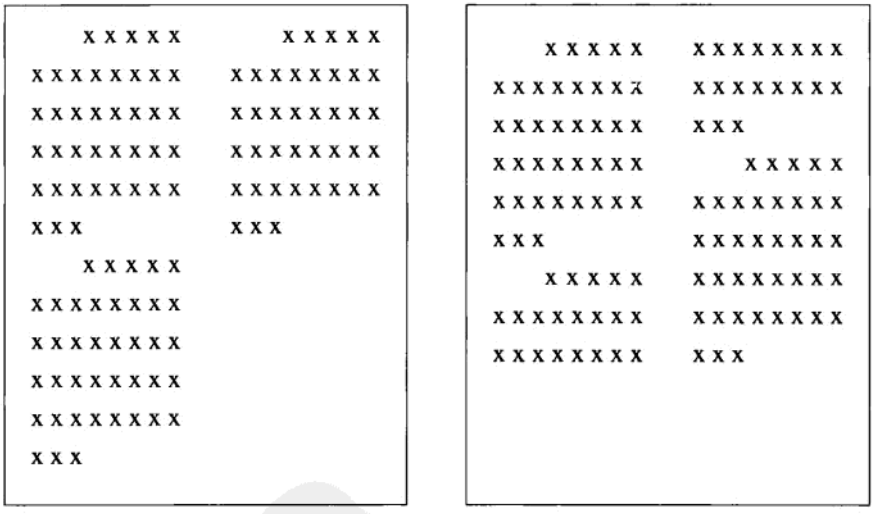


图 2.8　不平衡的双栏页面和平衡的双栏页面

2.4.5　定义命令与环境

在 1.2.4 节中我们初识了命令与环境，经过本章前面的内容，已经见识了许多有用的 LaTeX 命令和环境，1.2.8 节甚至还自己定义了一些命令与环境，命令的重定义也在

*本节内容初次阅读可略过。

前面的格式设置中屡屡出现。现在，我们来总结一下这些零散的内容，进一步讨论在LaTeX中的宏定义功能。

一个TeX宏（macro）是以反斜线\开头，后面紧接着一串字母的字符串，它在TeX中通常用来代替另外一个字符串，也有时表示其他一些特殊的含义。传统上将这种字符串代替的机制称为宏，TeX就是一种复杂的宏语言。

在LaTeX中，宏通常被分为两类：一般的宏被形象地叫做“命令”（command），而以命令\begin{⟨环境名⟩}和\end{⟨环境名⟩}包围的结构则称为“环境”（environment）。正同1.2.4节所说的，命令和环境都可以带有若干可选的和必须的参数，表示不同的含义，命令和环境是LaTeX中使用格式相对固定两类宏。

可以使用\newcommand来新定义一个命令，其语法格式如下：

\newcommand⟨命令⟩[⟨参数个数⟩][⟨首参数默认值⟩]{⟨具体定义⟩}

命令名只能由字母组成，并且不能以\end开头[①]。

\newcommand最简单的用法就是定义没有参数的命令（此时不需要⟨参数个数⟩和⟨首参数默认值⟩），其功能就是简单的字符串替换，例如：

2-4-14
```
\newcommand\PRC{People's Republic of \emph{China}}
```

定义了一个\PRC命令，在文中任何地方使用\PRC就相当于使用了“People's Republic of \emph{China}”这么一串内容。这正是宏的本来含义。注意在定义宏时，⟨具体定义⟩的外面必须要有花括号界定，即使定义的内容只有一个字符。在使用宏时，命令产生的效果则没有这对花括号。

如果指定了参数的个数（从1到9），则可以在宏的定义中使用参数。在⟨具体定义⟩中，宏的参数依次编号，用#1, #2, ..., #9表示，使用宏时它们会替换为参数的字符串，例如：

2-4-15
```
\newcommand\loves[2]{#1喜欢#2}
\newcommand\hatedby[2]{#2不受#1喜欢}
```

那么使用\loves{猫儿}{鱼}就相当于写猫儿喜欢鱼，而\hatedby{猫儿}{萝卜}则相当于萝卜不受猫儿喜欢。

如果在指定参数个数的同时还指定了首个参数的默认值，那么这个命令的第一个参数就成为可选的参数（要使用中括号指定）。例如，我们可以加强前面的例子：

2-4-16
```
\newcommand\loves[3][喜欢]{{#2#1#3}}
```

① \end开头的命令名保留用于系统实现生成环境，因而不能作为普通命令的开头。

此时 \loves{猫儿}{鱼} 的作用与前面无异，还表示 猫儿喜欢鱼，同时可以用可选参数覆盖默认值，如使用 \loves[最爱]{猫儿}{鱼} 表示 猫儿 最爱鱼。

重定义一个 LaTeX 命令使用：

\renewcommand(命令)[(参数个数)][(首参数默认值)]{(具体定义)}

它的意义和用法与 \newcommand 完全相同，只是 \renewcommand 用于改变已有命令的定义，而 \newcommand 只能用于定义新命令，例如：

```
\renewcommand\abstractname{内容简介}
```

这与 2.2.2 节中使用 \CTEXoptions 命令的效果相同。

如果用 \newcommand 重定义已有命令或用 \renewcommand 定义新命令，都会产生一个编译时的错误。

命令可以嵌套定义。如果要在一个命令的定义中定义或重定义命令，则里面一层的命令参数就要多加一个 #，例如：

```
\newcommand\Emph[1]{\emph{#1}}
\newcommand\setEmph[1]{%
  \renewcommand\Emph[1]{%
    #1{##1}}}
```

2-4-17

这里首先把自定义一个表示强调的命令 \Emph，初始与 \emph 相同；\setEmph 则提供了对 \Emph 的修改。如 \setEmph\textbf 会将 \Emph 重定义为 \textbf，而 \setEmph\textsc 则将 \Emph 重定义为 \textsc。

\providecommand 的语法与 \newcommand、\renewcommand 相同，它也可以用来定义命令。不同的是，它检查命令是否已经定义后，对已定义的命令不会报错，也不重定义，而是保留旧定义忽略新定义。这种方式可以用来保证一个命令的存在性，特别是可以用它来准备一个“备用”的定义，作为标准定义的补充。例如，url 宏包和 hyperref 宏包都提供有排版 URL 网址的 \url 命令，如果在文档中（特别是在设计通用的宏包和模板时）并不知道是否已经定义了 \url 命令，就可以提供一个质量比较差的版本：

```
\providecommand\url[1]{\texttt{#1}}
```

这样就可以在后面放心使用 \url 命令，保证这个命令可用。\providecommand 也可以与 \renewcommand 连用，表示无论之前是否定义过这个命令，都改用 \renewcommand 的新定义。

LaTeX 环境与命令类似，事实上，一个 LaTeX 环境就相当于一个分组，在分组内部的最前面和最后面各有一条命令。例如 quote 环境 \begin{quote} ⟨环境内容⟩ \end{quote} 就大致等于 {\quote ⟨环境内容⟩ \endquote}。环境可以有参数，就相当于是环境前面命令的参数。

定义新环境和重定义环境分别使用下面的命令：

\newenvironment{⟨环境名⟩}[⟨参数个数⟩][⟨首参数默认值⟩]
{⟨环境前定义⟩}{⟨环境后定义⟩}
\renewenvironment{⟨环境名⟩}[⟨参数个数⟩][⟨首参数默认值⟩]
{⟨环境前定义⟩}{⟨环境后定义⟩}

例如，book 类中没有显示摘要的 abstract 环境，我们可以仿照 article 类中的格式利用 quotation 环境定义一个（同时增加了改变标题的可选参数）：

2-4-18
```
\newenvironment{myabstract}[1][摘要]%
  {\small
   \begin{center}\bfseries #1\end{center}%
   \begin{quotation}}%
  {\end{quotation}}
```

需要注意的是，在定义有参数的环境时，只有 ⟨环境前定义⟩ 中可以使用参数，在 ⟨环境后定义⟩ 中不能再使用环境参数。如果有这种需要，可以先把前面得到的参数保存在一个命令中，在后面使用。例如可以定义一个带出处的引用环境：

2-4-19
```
1 \newenvironment{Quotation}[1]%
2   {\newcommand\quotesource{#1}%
3    \begin{quotation}}%
4    {\par\hfill —— 《\textit{\quotesource}》%
5    \end{quotation}}
6
7 \begin{Quotation}{易 · 乾}
8 初九，潜龙勿用。
9 \end{Quotation}
```

初九，潜龙勿用。

——《易 · 乾》

定义命令和环境是进行 LaTeX 格式定制、达成内容与格式分离目标的利器。使用自定义的命令和环境把字体、字号、缩进、对齐、间距等各种琐细的内容包装起来，赋以

一个有意义的名字，可以使文档结构清晰、代码整洁、易于维护。在使用宏定义的功能时，要综合利用各种已有的命令、环境、变量等功能，事实上，前面所介绍的长度变量与盒子（参见 2.1.5、2.2.8 节），字体字号（参见 2.1.3、2.1.4 节）等内容，大多并不直接出现在文档正文中，而主要都是用在实现各种结构化的宏定义里。

有关 LaTeX 宏的内容暂且介绍这些，还有一些更深入的内容，可参见 8.1 节和其他文献。

练 习

2.8 输出标志“LaTeX 2_ε”的命令是 `\LaTeXe`，为什么不是看起来更合理的 `\LaTeX2e` 呢？

2.9 使用 2.2.3.3 节中的 `list` 环境和 ctex 宏包提供的 `\chinese` 命令（参见 2.2.3.2 节），自定义计数器，定义一个 `clist` 环境，产生一个用中文数字编号的列表。

本章注记

符号和字体是 LaTeX 以至各种排版软件的中心问题之一。Pakin [192] 是一份全面的符号列表，可用的字体目录见 Hartke [100]、Jørgensen [118]。LaTeX 字体命令和 NFSS 机制的详解可见 LaTeX3 Project Team [142]、陈志杰等 [317]，在 Mittelbach and Goossens [166, Chapter 7] 中则有更为详尽的论述。TeX 的原始字体机制可参见 Knuth [126]。关于 PostScript、TrueType 和 OpenType 字体的背景知识与安装使用，可以参考 Goossens [83]。

有关过去旧的中文处理方式，如 CCT、CJK、天元系统等，现在已逐渐让步于以 XeLaTeX 和 ctex 宏包为核心的方式，故本书没有涉及旧的中文处理方式。如果使用的 TeX 系统较为陈旧，需要处理历史文档，可参考其他文档。CCT 参见 吴凌云 [308]、张林波 [311, 312]；CJK 参见 Lemberg [147]、吴凌云 [308]。CCT 和 CJK 也可以使用 ctex 宏包及文档类[58]。

基于 CJK 的中文支持方式目前仍然使用广泛，较新的进展是 zhmetrics 包和 zhmCJK 包[307]。zhmetrics 把汉字都看成是相同的方块尺寸，从而简化了中文字体安装配置。而本书作者编写的 zhmCJK 宏包则在 zhmetrics 机制的基础上，动态配置中文字体，免除了 CJK 中文字体的安装工作。zhmCJK 同时提供了类似 xeCJK 的字体设置语法。

XeLaTeX 的使用属于较新的内容，在较早出版的书籍中都少有论述。除了 fontspec 和 xeCJK 的文档 [212、310] 外，相关使用还可以参见文档 Goossens and Rahtz [86]、kmc [123]。

LuaTEX 的东亚语言支持目前还没有完全成熟。由 LuaTEX-ja 团队开发的 luatexja 包[150] 是目前基于 LuaTEX 和 fontspec 机制的一种相对完备的日文支持方案。LuaTEX-ja 的机制部分源自日本原有的 pTEX 系统，对日文排版有很多针对性的设计，马起园为 LuaTEX-ja 完成了一些关于中文排版的配置，使之也可以用来处理中文文档。

本书描述的 ctex 宏包是在 TEX Live 2012 与 CTEX 套装 2.9 中默认安装的版本，在更新版本的 ctex 宏包中，会采用更现代的 ⟨*key*⟩=⟨*value*⟩ 格式的宏包选项，增强 ctex.sty 格式的功能，并对 zhmCJK、LuaTEX-ja 等新的中文处理方式也有支持，不过大部分功能和用法是一致的（或向后兼容的），使用新版本的用户可以参考相应的宏包手册使用。

LATEX 在段落方面提供的新命令较少，TEX 的高级段落功能仍参见 Abrahams et al. [1]、Knuth [126] 等书籍。

LATEX 的全部基本功能都可以在源代码 Braams et al. [33] 中找到，基本文档类的全部功能则可以参见源代码 Lamport et al. [139]。

memoir 是一个面向专业排版的文档类，它的文档 Wilson [294] 也是很好的排版实践参考书，对标准文档类的设计也有借鉴意义；KOMA-Script 是另一套类似的面向专业排版的文档类[133]，它与 memoir 一样提供方便定制的排版环境。排版理论的书籍如 Bringhurst [35] 也可作为参考。

第3章 自动化工具

使用 LaTeX 编写文档，有很多内容是不需要手工排版的，可以由计算机计算或提取需要的信息，按预定的格式自动化输出。最常见的自动化工具是编号，比如页码、章节编号等，我们已经在前面见到了。更重要的自动化排版包括目录、交叉引用、参考文献和词汇索引表，其中有的可以直接使用 TeX 排版引擎完成工作，有的还需要其他外部程序的帮助，这些自动化工具也是 LaTeX 特别为人津津乐道的地方。

3.1 目录

3.1.1 目录和图表目录

目录是最基本的自动化工具。LaTeX 会自动收集章节命令所定义的各章节标题，用 \tableofcontents 命令输出，例如：

```
\documentclass{article}
\begin{document}
\tableofcontents
\section{Foo}
\subsection{blah}
\section{Bar}
\end{document}
```

3-1-1

上面的文档连续编译两遍以后，会在两节内容之前产生一个章节目录，大致效果如下：

Contents

其中“Contents”是目录的名称，后面是目录项的列表，包括每节的编号、标题和页码。

在 article 中，目录标题的格式相当于由 \section* 开始的一节，而在 report 和 book 类中，目录标题的格式相当于由 \chapter* 开始的一章，因此，可以按照 2.3.4 节的方法，控制目录标题的格式。

类似地，命令 \listoffigures 和 \listoftables 则会收集在 figure 环境和 table 环境中 \caption 命令的图表标题，产生图表的目录，其格式与章节目录类似。

需要特别注意的是，要产生正确的章节目录和图表目录，必须在不修改内容的情况下编译 .tex 源文件至少两遍。前面的编译 LaTeX 做好目录项的收集工作，才能确保最后一遍编译时正确输出。如果文档从未被编译过，那么第一遍编译是没有目录内容输出的。

3.1.2 控制目录内容

在继续深入说明有关 LaTeX 目录的工具之前，首先说说在 LaTeX 中生成目录的原理。与很多“高级”的排版工具通过特殊的样式和搜索提取功能生成目录不同，LaTeX 的目录是通过一个简单的辅助文件实现的（见图 3.1）。在使用 \chapter, \section 等章节命令时，LaTeX 引擎同时把章节的编号和标题写进一个扩展名为 .toc 的目录文件中，而在遇到 \tableofcontents 命令时，LaTeX 会读入目录文件（如果存在的话），生成目录。

与之类似，图目录是通过扩展名为 .lof 的目录文件，表目录是通过扩展名为 .lot 的目录文件实现目录的。

例如，编译例 3-1-1 将生成的下面的目录文件[①]：

```
\contentsline {section}{\numberline {1}Foo}{1}
\contentsline {subsection}{\numberline {1.1}blah}{1}
\contentsline {section}{\numberline {2}Bar}{1}
```

*本节内容初次阅读可略过。

① 如果文档使用了其他宏包，如 hyperref，目录文件的格式与里面命令的语法可能会有所变化。

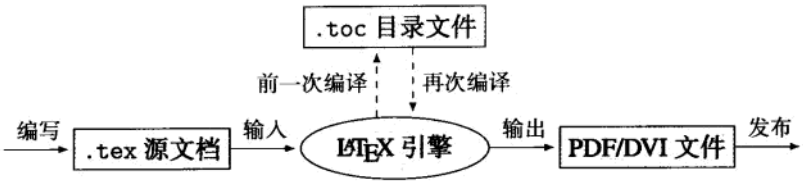


图 3.1　LaTeX 章节目录生成示意图

这里的三个 \contentsline 命令定义了各个目录项，三个参数分别是目录层次、编号标题和页码，它们正是由例 3-1-1 的两个 \section 命令产生的。里面 \numberline 命令专用于在 \contentsline 中输出章节编号。

目录文件的生成通常不需要人工干预。除 \part 外，标准文档类的默认输出 3 级目录：article 是 \section、\subsection、\subsubsection；而 report 和 book 是 \chapter、\section、\subsection 3 级。可以通过修改计数器 tocdepth 的值来控制输出目录的深度。tocdepth 通常与 secnumdepth 一起修改（参见 2.3.2 节）。例如，可以将其改为 4 而使 articie 中的 \paragraph 一级标题也进入目录：

```
\setcounter{secnumdepth}{4} % 增加编号深度
\setcounter{tocdepth}{4}    % 增加目录深度
```

3-1-2

也可以用下面的命令直接在正文中手工写入一条目录项：

\addcontentsline{⟨文件⟩}{⟨类型⟩}{⟨标题文字⟩}

这里 ⟨文件⟩ 指目录文件的扩展名，章节目录使用 toc，图表目录分别用 lof 和 lot。⟨类型⟩ 参数指章节或图表命令名，即 section, figure 等。标题文字是目录项的标题。生成目录项的页码会自动使用当前页码。例如，可以在 \maketitle 前使用 \addcontentsline 把文章题目也加进目录：

```
\addcontentsline{toc}{section}{Title}
\maketitle
\tableofcontents
```

3-1-3

这样，在生成的 .toc 目录文件中，就会有下面一行额外的项目产生：

```
\contentsline {section}{Title}{1}
```

最终生成的目录里面就会有一条“Title”的项目，与 \section 产生的项目格式相同。\chapter*、\section* 等命令生成的章节不编号也不进入目录，也可以用这种方法把它们加入目录。

目录本身、参考文献、索引表等的标题在标准文档类中都是使用 \section* 或 \chapter* 实现的，因而也可以用 \addcontentsline 加入目录，如：

```
% book 类，把目录本身加入目录
\addcontentsline{toc}{chapter}{\contentsname}
\tableofcontents
```

这里 \contentsname 是目录名，标准文档类中默认为 “Contents”，ctex 文档类中是 “目录”。不过，正如 1.2.8 节所介绍的，可以直接使用 tocbibind 宏包[292] 把这些项目加入章节目录。tocbibind 默认会把章节目录、图表目录、文献、索引等都加入章节目录，下面的宏包选项可以做进一步控制：

- notbib 不加入参考文献；
- notindex 不加入索引；
- nottoc 不加入章节目录；
- notlot 不加入表目录；
- notlof 不加入图目录；
- none 以上项目都不加入（如同没有用宏包一样）；
- chapter 使用 \chapter 样式（对 book, report 类）；
- section 使用 \section 样式；
- numbib 对参考文献按章节编号（默认无）；
- numindex 对索引按章节编号（默认无）。

因而，把目录、参考文献等条目加入目录，就可以简单地在导言区增加一行来实现：

3-1-4
```
\usepackage{tocbibind}
```

更为底层的命令是 \addtocontents{(文件)}{(内容)}，使用它可以直接把任意代码写入目录文件，而不只是 \contentsline。如可以要求目录在某一部分前分页：

3-1-5
```
\addtocontents{toc}{\newpage}
\part{Foo}
```

titletoc 宏包[27] 的部分目录（Partial TOC）或者 minitoc 宏包[68] 的迷你目录功能，可以在文档的每一章节前面各自添加一个目录。这一功能对于长篇的书籍会特别有用，读者可以参考这两个宏包的文档了解具体的用法。

3.1.3 定制目录格式

LaTeX 的标准文档类并没有就目录格式的修改给出一个良好的界面，使用第三方的宏包来修改目录格式更为方便。最为常见的是 tocloft 和 titletoc 宏包，其中 tocloft 宏包[293] 的语法和功能较为简单，titletoc 宏包[27] 则是与 titlesec 一起发布的更为复杂的目录格式定制宏包。

tocloft 的 `\tocloftpagestyle{⟨风格⟩}` 可以修改目录项的页面风格（参见 2.4.3 节），默认为 `plain`。

原本 LaTeX 目录标题是按不编号的章节输出的，tocloft 会修改目录标题的输出，进行进一步的格式控制。这里目录是章节目录（`toc`）、图目录（`lof`）、表目录（`lot`）等，每种目录都有一套类似的命令，表 3.1 是相关的长度参数和命令。

表 3.1 tocloft 宏包提供的控制目录标题格式的参数和命令

格式	章节目录	图目录	表目录
标题[①]	`\contentsname`	`\listfigurename`	`\listtablename`
标题前间距	`\cftbeforetoctitleskip`	`\cftbeforeloftitleskip`	`\cftbeforelottitleskip`
标题后间距	`\cftaftertoctitleskip`	`\cftafterloftitleskip`	`\cftafterlottitleskip`
标题字体	`\cfttoctitlefont`	`\cftloftitlefont`	`\cftlottitlefont`
标题后代码	`\cftaftertoctitle`	`\cftafterloftitle`	`\cftafterlottitle`

由这些命令的设置，排版输出目录标题的实际代码大约相当于：

```
{\cfttoctitlefont \contentsname}{\cftaftertoctitle}\par
```

可以通过修改标题前后间距的长度变量，或重定义其他几个宏，来对目录标题的输出格式进行修改。也可以给 tocloft 加 `titles` 选项，禁用 tocloft 对目录标题排版格式的特殊控制。下面举一个例子，让章节目录无衬线粗体放大居中，前后间距 2ex：

```
% \usepackage{tocloft}
\renewcommand\cfttoctitlefont{\hfill\Large\sffamily\bfseries}
\renewcommand\cftaftertoctitle{\hfill}
\setlength\cftbeforetoctitleskip{2ex}
```

*本节内容初次阅读可略过。

① 这几个命令是标准文档类提供的，不依赖 tocloft。

3-1-6
```
\setlength\cftaftertoctitleskip{2ex}
```

下面来看目录中的项目，例如这是一个 \subsection 的项目：

tocloft 可以控制目录项的页码前引导的点、页码的输出位置、前后的间距、缩进、编号宽度、字体等项目。相关命令和变量如表 3.2 所示。

表 3.2 tocloft 宏包提供的目录项格式设置命令与变量

项目	命令	说明
引导点	\cftdot	页码前引导的点的符号（默认是英文句号）
	\cftdotsep	默认的页码前引导的点之间的距离，它是一个数字宏，单位是数学间距 mu，也可用于下面 \cft某dotsep 的定义
	\cftnodots	常数，意义类似 \cftdotsep，表示没有点，多用于下面的 \cft某dotsep 的定义
	\cftdotfill{⟨数⟩}	按 ⟨数⟩ 的间距和 \cftdot 的符号画出引导线的命令，如 \cftdotfill{\cftdotsep} 表示默认点间距的引导线，\cftdotfill{\cftnodots} 表示没有点
页码	\cftsetpnumwidth{⟨宽度⟩}	设置页码所占最大宽度
	\cftsetmarg{⟨宽度⟩}	设置标题与引导点右端与行右边界距离
段间距	\cftparskip	长度变量，目录项中的段间距（默认为零）
条目设置	\cftbefore某skip	长度变量，条目前垂直间距
	\cft某indent	长度变量，条目前缩进
	\cft某numwidth	长度变量，条目编号占用宽度
	\cftsetindents {⟨某⟩}{⟨缩进⟩}{⟨宽度⟩}	同时设置 \cft某indent 和 \cft某numwidth 的命令
	\cft某font	条目字体
	\cft某presnum	条目编号前内容（编号盒子内）
	\cft某aftersnum	条目编号后内容（编号盒子内）
	\cft某aftersnumb	条目编号后的内容（编号盒子外）
	\cft某leader	条目使用的引导线，通常定义为 \cftdotfill{\cft某dotsep}

续表

项目	命令	说明
	\cft某dotsep	条目引导线中两点间距，它是一个宏，通常就直接定义为 \cftdotsep
	\cft某pagefont	条目页码字体
	\cft某afterpnum	条目页码后代码

其中条目设置是针对每个目录层次的具体设置，带“某”的命令表示一组相似的命令，“某”根据要修改的目录层次实际替换为：

- part，当设置 \part 的目录项时；
- chap，当设置 \chapter 的目录项时；
- sec，当设置 \section 的目录项时；
- subsec，当设置 \subsection 的目录项时；
- subsubsec，当设置 \subsubsection 的目录项时；
- para，当设置 \paragraph 的目录项时；
- subpara，当设置 \subparagraph 的目录项时；
- fig，当设置 figure 环境的 \caption 的目录项时；
- tab，当设置 table 环境的 \caption 的目录项时。

表 3.2 中有关长度变量的意义，可参见图 3.2。

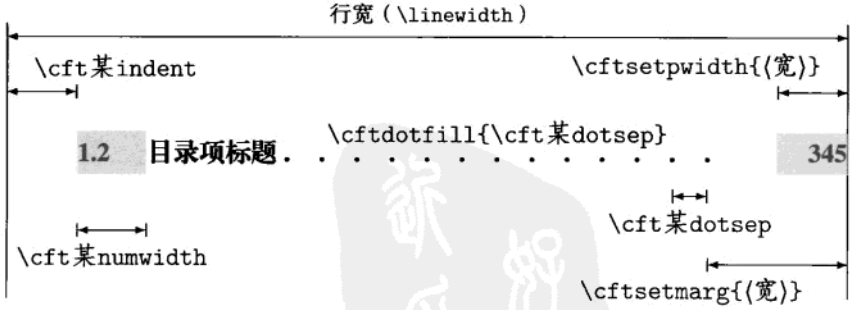


图 3.2　tocloft 的各种长度变量

由上述设置，排版目录的一个 \section 的条目的逻辑大约相当于（略去距离设置的代码等）：

```
% 有编号
```

```
{\cftsecfont {\cftsecpresnum 编号\cftsecaftersnum\hfil}
  \cftsecaftersnumb 条目标题}%
{\cftsecleader}{\cftsecpagefont 页码}\cftsecafterpnum\par
% 无编号
{\cftsecfont 条目标题}
{\cftsecleader}{\cftsecpagefont 页码}\cftsecafterpnum\par
```

举个例子，目录项前面引导的点默认是西文的句号，它与基线平齐。中文排版中经常需要修改为一串紧密排列的中文省略号“…”，可以用如下命令：

3-1-7

```
% \usepackage{tocloft}
\renewcommand\cftdot{…}
\renewcommand{\cftdotsep}{0}
```

中文编号往往非常宽（如“第十一章”），不适合使用固定的编号宽度，因而 ctex 的文档类及 ctexcap 包会略微修改目录编号的命令 \numberline 使编号宽度可变。但 tocloft 宏包也会修改 \numberline 使 ctex 文档类的修改失效，所以使用 tocloft 时需要对中文编号的章节层次设置较大的 \cft某numwidth，如：

3-1-8

```
\settowidth{\cftchapternumwidth}{第几十几章}        % 最宽的可能编号
\renewcommand\cftchapteraftersnumb{\hspace{0.5em}}  % 额外间距
```

也可以临时使用下面的方式综合两个宏包的修改，使编号成为可变长的。或许 ctex 宏包未来的版本会加入这个补丁，不过目前在中文文档中使用 tocloft 时必须要小心：

3-1-9

```
\documentclass{ctexbook} % 或 ctexrep, ctexart
\usepackage{tocloft}
\makeatletter
\renewcommand{\numberline}[1]{%
  \settowidth\@tempdimb{#1\hspace{0.5em}}%
  \ifdim\@tempdima<\@tempdimb%
    \@tempdima=\@tempdimb%
  \fi%
  \hb@xt@\@tempdima{\@cftbsnum #1\@cftasnum\hfil}\@cftasnumb}
\makeatother
```

tocloft 宏包的基本功能大致就这么多，还有一些不大常用的功能未做介绍，可进一步参见 Wilson [293]。使用 titletoc 宏包可以排版出格式更为复杂多变的目录格式，限于篇幅，这里不多做介绍，读者可参考宏包文档 Bezos [27]。

3.2　交叉引用

交叉引用（cross reference）是 LaTeX 中另一种常用的自动化工具，它可以通过一个符号标签引用文档中某个对象的编号、页码或标题等信息，而不必知道这个对象具体在什么地方。例如，这一章的标题在第 157 页，编号第 3 章，标题是“自动化工具”，但实际输出这段话的时候使用的代码却是：

```
这一章的标题在第~\pageref{chap:autotool} 页，编号
第~\ref{chap:autotool} 章，标题是“\nameref{chap:autotool}”……
```

这样无论后来的修改怎样改变了文档的页面、章节的编号，或者干脆连这一章的标题都变化了，上面这段隐藏在正文中的文字都不用再做修改，这种性质对于编写大型文档是至关重要的。

下面就来详细说明交叉引用的使用。

3.2.1　标签与引用

交叉引用的使用可以分成两个部分：定义标签和引用标签。定义标签是在适当的位置给一个（带有编号的）对象加一个标签（label），也就是赋给对象一个标识；引用标签则是在文档的另一个地方，利用已有的标签获得对象的编号、页码等信息。

给一个对象加标签，可以在这个对象里面或后面使用 \index{⟨标签⟩} 命令，⟨标签⟩可以是任意字符串，但不要包含特殊字符，而且最好起一个简洁有意义的名字。

例如，给一篇文章的“Lanuages”一节加标签，可以用：

```
\section{Lanuages}
\label{sec:lang}
```

3-2-1

这里标签 sec:lang 是“Section: Lanuages”的缩写，当然也可以改用 sec-lang、language 或其他什么好记的名称。但最好不要使用类似 sec1.1、1.1 这种标签，像 1.1 这种标签的意义很含混，更重要的是今天的 1.1 节到明天修改时可能会变成 2.3 节，旧的标签名 sec1.1 只会把作者（也就是我们自己）搞糊涂。标签的命名并没有一定的规则，一种非常流行的标签命名习惯就是像例 3-2-1 中那样使用 类型缩写:内容缩写 的形式，其中类型缩写可以是，例如：

- part：部分（part）
- chap：章（chapter）
- sec：节（section）
- subsec：小节（subsection）
- subsubsec：小小节（subsubsection）
- para：段（paragraph）
- subpara：小段（subparagraph）
- fig：图（figure）
- tab：表（table）
- eq：公式（equation）
- fn：脚注（footnote）
- item：项目（item）
- thm：定理（theorem）
- algo：算法（algorithm）

这些与 3.1.3 节 tocloft 宏包所使用的缩写一致。当然，也可以使用其他缩写，只要保持条理和一致性。

前面的列表大致圈定了标签的使用范围，即各种自动编号的对象，大部分是 LaTeX 预定义的命令和环境所产生的，也有一些可以是自定义的编号环境，比如定理和算法（参见 2.2.4 节、2.2.6 节）。如果对象是用一条命令产生的（如 \section, \item），标签通常直接放在命令的后面；如果对象是由环境产生的（如 table 环境，equation 环境），则放在环境里面。不过这里有个别例外：脚注的标签应该放在脚注内容里面，即 \footnote{\label{fn:foo}}，否则会被看做是外面一层对象的标签；个别特殊环境的标签则是使用其他方法定义的，如 listings 宏包提供的 lstlisting 环境，使用环境的可选参数 label 定义[174]。在多行编号的数学公式中，标签则应该加在每行公式的后面（\\ 命令之前），例如：

3-2-2

```
\begin{align}
  c^2 &= a^2 + b^2
    \label{eq:gougu-formula} \\
  5^2 &= 3^2 + 4^2
    \label{eq:gougu-example}
\end{align}
```

$$c^2 = a^2 + b^2 \tag{3.1}$$
$$5^2 = 3^2 + 4^2 \tag{3.2}$$

有了标签，引用便水到渠成了。LaTeX 提供了两个引用的命令：\ref 和 \pageref，它们分别产生被引用对象的编号和所在页码，例如：

3-2-3

```
勾股定理公式（\ref{eq:gougu-formula}）
出现在第~\pageref{eq:gougu-formula} 页。
```

勾股定理公式 (3.1) 出现在第 166 页。

与目录一样，交叉引用功能也是通过在编译过程中读写扩展名为 .aux 的辅助文件（auxiliary file）实现的（见图 3.3）。因此，要产生正确的交叉引用的文档，必须至

少编译两次源文件。如果在编译前辅助文件中尚没有交叉引用的信息，或者 \ref 命令使用的标签拼写错误，被引用的内容会使用几个问号代替，如“?? 节”，同时在编译的提示信息和日志文件中会显示相关的警告。

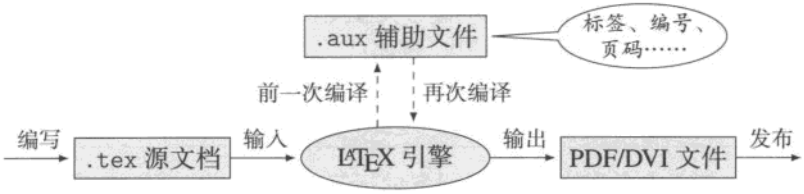


图 3.3　LaTeX 交叉引用生成示意图

3.2.2　更多交叉引用

除了 LaTeX 标准的 \label 和 \ref, \pageref，很多宏包也对交叉引用作了各种功能的扩展。扩展的方面包括引用的格式、内容、范围等等，下面选择一些做一简要的介绍。

首先是公式的引用。公式编号在圆括号之中，但引用的计数器值并不包括圆括号。因而如果要引用第 166 页的公式 (3.1)，就需要用代码 (\ref{eq:gougu-formula})，把 \ref 命令放进圆括号之中。amsmath 提供的 \eqref 命令可以自动为引用加上括号，即使用 \eqref{eq:gougu-formula} 来得到引用 (3.1)。\eqref 在功能上大致相当于下面的定义：

```
% amsmath 中 \eqref 的一个简易实现
\newcommand\eqref[1]{(\ref{#1})}
```

不过，\eqref 的实际定义比上面的定义要复杂一些，即使修改了公式的编号方式（如不使用括号），\eqref 仍然可以得到正确的引用。

类似上面 \eqref 的简易实现，我们也可以自己定义需要的引用格式。例如，可以定义一个 \thmref 专门用来引用自定义的定理环境：

```
\newcommand\thmref[1]{定理~\ref{#1}}
```

3-2-4

*本节内容初次阅读可略过。

可以按这种方法定义其他命令，引用章、节、图、表等编号，一些宏包也会这样做，例如 ntheorem 宏包就提供了 [thref] 选项和 \thref 命令，同时定义了扩展的 \label 命令语法，用来引用定理类环境。但这样重复的工作实在令人厌烦，而且有违自动化的初衷，能不能让 LaTeX 自己辨认引用的编号类型呢？确实可以，hyperref 宏包就提供了自动识别编号类型的 \autoref 命令[201]，例如：

3-2-5

```
See \autoref{fig:xref}.
```

See figure 3.3.

这里 fig:xref 就是本书为前面图 3.3 设置的标签。LaTeX 会根据标签所引用的计数器 figure 自动选取引用的前缀名称“Figure”。对计数器“某”，hyperref 首先会检查是否有 \某autorefname 这个宏，如果没有的话会使用 \某name 宏。对常用的计数器，宏 \某autorefname 已经有了合适的英文定义，可以重定义它来修改前缀的名称为汉字，例如：

3-2-6

```
\renewcommand\figureautorefname{图}
参见\autoref{fig:xref}。
```

参见图 3.3。

不过，\autoref 只能对引用的数字加前缀名称，并不能产生“第 3 章”这样的引用效果，此时就只能自己手写名称或另行定义单独的命令了。

LaTeX 默认的引用只能得到编号和页码，既然 \autoref 可以得到编号的类型，那么能否得到更进一步的信息，得到被引用的章节或图表的标题内容呢？确实可以，nameref 宏包就完成了这个功能[200]：

3-2-7

```
% 导言区 \usepackage{nameref}
见“\nameref{fig:xref}”。
```

见“LaTeX 交叉引用生成示意图”。

\nameref 命令可以用来引用文章的章节标题和图表的标题，如果用它引用其他没有标题的计数器，会产生错误的结果。

排版中有时需要知道整个文档的总页数，这可以引用文档最后一页的页码得到。不过自己直接在文档末尾写 \label{LastPage} 往往并不能正确引用，因为在这之后可能还要输出一些浮动体、尾注等。为此，lastpage 宏包[82] 提供了一个特殊的标签 LastPage，它是保证在文档末尾的标签，可以得到文档最后一页的页码，例如期刊页眉的页码往往可以写成下面的样子：

3-2-8

```
page \thepage\ of \pageref{LastPage}
```

page 168 of 566

交叉引用是利用辅助文件保存的信息实现的，因此从原理上交叉引用并不限定引用的标签是否来自当前文档前一次编译的结果。使用 \include 命令引入的所有子文档，都会生成单独的 .aux 辅助文件，里面分别保存有这个文件中影响的计数器编号、交叉引用等辅助信息。因此，即使使用 \includeonly 命令只引入部分源文件，也可以利用以前生成的辅助文件完整地进行整个文档的交叉引用。类似的功能甚至可以扩展到不相干的文档，只要有引用所需的 .aux 辅助文件，就可以在一个文档中对另一个文档的内容进行交叉引用。xr 宏包[41] 就实现了对外部文件进行引用的功能，它实现了一个命令来声明外部辅助文件：

\externaldocument[⟨前缀⟩]{⟨文件名⟩}

这里 ⟨文件名⟩ 是没有 .aux 后缀的外部辅助文件名。然后就可以像引用本地文档中的编号一样引用外部文档中的标签了。使用 \ref 等命令引用时，为了避免标签冲突，可以在外部文档的标签前面添加可选的 ⟨前缀⟩。例如当前文件是 languages.tex，我们又从另一个文档 chinese.tex 中编译生成了 chinese.aux 辅助文档，里面是有关 chinese.tex 文档中的引用信息，那么，就可以把 chinese.aux 复制到 languages.tex 所在的工作目录，然后在 languages.tex 的导言区声明：

```
% languages.tex 导言区
\usepackage{xr}
\externaldocument[ch:]{chinese}
```

3-2-9

于是，就可以使用 \ref{ch:fig:popular} 来引用在 chinese.tex 中以 \label{fig:popular} 声明的图的标签了。

3.2.3　电子文档与超链接

作为排版软件，LaTeX 首要的任务是为输出到纸面作准备的。不过，PDF 格式也是重要的电子文档格式，支持文档标签、超链接、电子表单等许多特殊的功能。通过相关宏包的支持，LaTeX 也可以输出带有这些高级功能的电子文档。

最常用的电子文档功能是文档的目录标签和超链接功能，这可以由 hyperref 宏包完成。在制作软件说明书、幻灯片演示文件等场合，hyperref 提供的电子文档功能起着尤为重要的作用。

hyperref 宏包可算是 LaTeX 最为复杂的宏包之一，它提供了大量的选项和命令，完成各种设置和功能。前面 3.2.2 节介绍的 \autoref 就是 hyperref 众多功能中的一个。这里主要介绍以 PDF 格式输出时，hyperref 有关标签和超链接的一些最基本的功能和设置，

更进一步的说明可参见 hyperref 的手册 Rahtz and Oberdiek [201] 及相关文档 Oberdiek [179] 等。

hyperref 最基本的用法非常简单，就是直接调用此宏包：

```
\usepackage{hyperref}
```

如果是使用 ctex 宏包或文档类，则可以加 hyperref 选项，这样 ctex 宏包会自动根据编码和编译方式选择合适的选项，避免出现乱码：

3-2-10

```
\documentclass[hyperref,UTF8]{ctexart}
```

引入 hyperref 后在编译文档时，会根据章节结构，自动生成目录结构的 PDF 文档标签。同时，正文中的目录和所有交叉引用，都会自动成为超链接，可以用鼠标点击跳转到引用的位置（见图 3.4）。与生成目录类似，要得到正确的 PDF 标签，也应至少编译两遍文档。

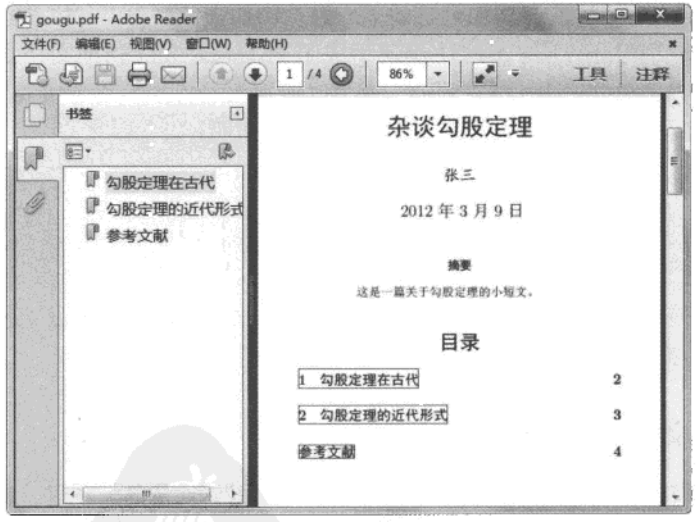


图 3.4 使用 hyperref 的文档。在 Adobe Reader 中左边栏是文档的目录标签，每个目录项外的彩色边框表示超链接

可以给 hyperref 设置许多宏包选项来控制其行为，除了直接加在 \usepackage 命令后，宏包选项也可以使用 \hypersetup 命令单独设置。hyperref 的选项大多使用

*本节后面内容初次阅读可略过。

⟨选项⟩ = ⟨值⟩ 的方式设置，如果是布尔类型的真假值选项（值为 true 或 false），通常可以省略为真的值。常用的一些选项见表 3.3。

表 3.3 hyperref 宏包的常用选项

选项	类型	默认值	说明
colorlinks	⟨布尔⟩	false	超链接用彩色显示。相关的彩色选项包括 linkcolor, anchorcolor, citecolor, filecolor, urlcolor 等，颜色参见 5.4 节。在一些版本中默认值与编译引擎相关
bookmarks	⟨布尔⟩	true	生成 PDF 目录书签
bookmarksopen	⟨布尔⟩	false	在 PDF 阅读器中自动打开书签
bookmarksnumbered	⟨布尔⟩	false	目录书签带编号
pdfborder	⟨数⟩ ⟨数⟩ ⟨数⟩	0 0 1	当 colorlink 为假时，超链接由彩色边框包围（不会被打印）。默认值表示 1 pt 宽的边框，可以设置为 0 0 0 表示没有边框
pdfpagemode	⟨文本⟩		在 PDF 阅读器中的页面显示方式，常用值是 FullScreen，表示全屏显示
pdfstartview	⟨文本⟩	Fit	在 PDF 阅读器中的页面缩放大小。默认值 Fit 表示“适合页面”；常用取值有适合宽度 FitH、适合高度 FitV
pdftitle	⟨文本⟩		文档标题，会在 PDF 文档属性中显示
pdfauthor	⟨文本⟩		文档作者，会在 PDF 文档属性中显示
pdfsubject	⟨文本⟩		文档主题，会在 PDF 文档属性中显示
pdfkeywords	⟨文本⟩		文档关键字，会在 PDF 文档属性中显示

例如，在制作 PDF 格式的电子书时，为了方便在屏幕上阅读，可能需要特殊的页面、彩色超链接、全屏显示，并有合适的 PDF 文档信息，那么，很可能以下面的方式开始导言区：

```
\documentclass[hyperref,UTF8]{ctexbook}
\usepackage{geometry}
\geometry{screen}
\hypersetup{
  colorlinks=true,
```

```
  bookmarks=true,
  bookmarksopen=false,
  pdfpagemode=FullScreen,
  pdfstartview=Fit,
  pdftitle={初等几何教程（电子版）},
  pdfauthor={张三}
}
```

3-2-11

除了自动的超链接功能，hyperref 还提供了许多命令，用来生成书签、超链接或网址等。

\url 命令用来输出 URL 地址，同时也具有超链接的功能。与排版纯文本不同，在 \url 命令的参数中，网址允许使用的合法符号（如 &、% 等）直接输入，并且默认以打字机字体输出，例如：

3-2-12

```
\url{http://bbs.ctex.org/forum.php?mod=viewthread&tid=48244#pid337079}
```

http://bbs.ctex.org/forum.php?mod=viewthread&tid=48244#pid337079

如果 URL 地址不需要超链接的效果，可以改用 \nolinkurl 命令。hyperref 是使用 url 宏包排版 URL 地址的，如果文档是要印制输出，只是需要排版 URL 地址，不需要任何电子文档的超链接功能，也可以直接使用 url 宏包提供的 \url 等命令。修改 URL 的输出格式等更多功能，参见宏包文档 Arseneau and Fairbairns [19]。

\path 命令可以用来排版文件路径，它的格式和 \url 差不多。

\href{⟨*URL*⟩}{⟨文字⟩} 命令可用来使文字产生指向 URL 地址的超链接效果，同样，这里 URL 地址中的 #、~ 等特殊符号一般不需要做特别处理，例如：

3-2-13

```
\href{http://bbs.ctex.org/}{CTeX 论坛}
```

\hyperref[⟨标签⟩]{⟨文字⟩} 命令可用来产生使文字指向标签的超链接效果，这里方括号中的标签与 \ref 使用的标签相同，不能省略，如：

3-2-14

```
\hyperref[eq:gougu-formula]{点击查看勾股定理公式}
```

\hypertarget{⟨名称⟩}{⟨文字⟩} 用来给文字定义带有名称的链接点，在文档的其他地方，则可以使用命令 \hyperlink{⟨名称⟩}{⟨文字⟩} 让另一段文字链接到指定名称的链接点。

\phantomsection 命令自动产生一个超链接点，它通常用在下面的场合，为手工添加的目录项指定正确的链接位置：

```
\phantomsection
\addcontentsline{toc}{section}{习题}
\section*{本章习题}
```
3-2-15

\pdfbookmark[⟨层次⟩]{⟨书签文字⟩}{⟨链接点名称⟩} 命令可用来手工添加 PDF 书签，同时定义一个带有名称的链接点（可用于 \hyperlink 等命令）。这里书签层次与章节层次可对应，大约等同于第 129 页的表 2.13 中的层次（但 article 的 \part 是第 0 级）。例如建立与 \subsection 同级的书签，可以用：

```
\pdfbookmark[2]{勾股定理证明}{gouguproof}
\begin{proof} …… \end{proof}
```
3-2-16

章节标题中，有时会出现一些数学式、特殊符号等命令，在 PDF 书签中可能比较费解，甚至被展开成复杂的内容，效果很难看。此时可以用命令 \texorpdfstring{⟨*TEX 形式*⟩}{⟨*PDF 形式*⟩} 分别定义在 TEX 文档输出的代码和 PDF 书签等处的纯文本形式，例如：

```
\section{\texorpdfstring{$\frac{1}{\pi}$}{1/π} 的计算}
```
3-2-17

它在输出的 PDF 文本中会显示标题为 “$\frac{1}{\pi}$ 的计算”，而在 PDF 书签中则会显示标题为 “1/π 的计算”。

hyperref 使用的是 TEX 的扩展功能，只有与相应的 TEX 引擎或输出驱动正确配合才能使用。dvips、pdftex、dvipdfmx、xetex 等选项就用于选择输出的引擎或驱动，其中 pdftex、xetex 选项对应的 XeTEX、pdfTEX 引擎[①]通常会自动检测到，但使用 DVIPDFMx 驱动时就必须手工加上 dvipdfmx 选项才能正确输出。

hyperref 的目录标签是通过扩展名为 .out 的辅助文件帮助产生的，因此，要生成正确的标签也需要两遍以上编译。利用 CCT、CJK 等旧的中文处理方式时，通常必须选用特定的选项才能正确编译输出。当使用 CJK 方式以非 Unicode 编码处理中文时，必须给 hyperref 加 CJKbookmarks 选项，同时使用外部工具[②]或 DVIPDFMx 驱动的特殊命令[③]，转换标签文档编码；而 CJK 在使用 UTF-8 编码时，则应使用 unicode 选项。这类选项和命令的选择比较复杂，最好使用 ctex 宏包或文档类处理中文。使用 ctex 的 hyperref 选项来引入 hyperref 宏包，ctex 会自动选择尽可能合适的相关选项及编码转换命令。

① pdftex 选项指使用 pdfTEX 引擎且输出 PDF 格式。输出 DVI 格式时，仍应根据情况选用 dvips 或 dvipdfmx 等选项。

② 如 CCT 系统或 CTEX 套装提供的命令行工具 gbk2uni，它将 .out 辅助文件由 GBK 编码转换为 UCS-2 编码。CTEX 套装配置的 WinEdt 编辑器在使用 latex 或 pdflatex 编译时，一般会调用这个程序。

③ 命令是 \AtBeginDvi{\special{pdf:tounicode GBK-EUC-UCS2}}，且需要 TEX 发行版中有对应的转换码表。

3.3 BibTeX 与文献数据库

LaTeX 主要被用来排版学术文档，因而参考文献列表也是 LaTeX 中特别重要的项目。与交叉引用和目录类似，现代文档的参考文献通常也采取自动生成方式，这里需要一个外部工具 BibTeX，它可以根据文章的内容，从一个文献数据库中抽取、整理和排版文献列表。本节的内容主要就是基于 BibTeX 的。

3.3.1 BibTeX 基础

使用 BibTeX 处理参考文献列表，首先需要准备好参考文献数据库。

文献数据库是以 .bib 结尾的文本文件，其内容是许多个文献条目，里面保存着文献的类型、引用标签、标题、作者、年代等各种信息，如：

```
% tex.bib 中的一条
@BOOK{mittelbach2004,
  title = {The {{\LaTeX}} Companion},
  publisher = {Addison-Wesley},
  year = {2004},
  author = {Frank Mittelbach and Michel Goossens},
  series = {Tools and Techniques for Computer Typesetting},
  address = {Boston},
  edition = {Second}
}
```

3-3-1

文献数据库可以手工逐条录入，许多 TeX 相关的编辑器（如 WinEdt）都有 .bib 文件类型的语法高亮和条目模板功能。不过更方便的做法是在网上下载现成的文献数据库，一些电子期刊数据库网站会提供相应的 BibTeX 数据库文件下载或是 BibTeX 条目的导出功能，搜索引擎 Google Scholar 也免费提供这项功能[①]，一些文献管理软件也可以将其他类型的文献数据库转换为 BibTeX 格式。在文献管理软件中用填空的方式录入文献数据库，通常也比直接手工录入容易一些。

在 .tex 源文件中我们则需要做好下面三件事：

1. 使用 \bibliographystyle 命令设定参考文献的格式，这通常在导言区完成。基本的 BibTeX 文献格式包括 plain、unsrt、alpha 和 abbrv[195]（见表 3.4）。前两

① 需要在 Google Scholar 的 Preferences 选项中设置 Bibliography Manager 为 BibTeX。

种使用一般的数字编号文献，plain 格式按作者、日期、标题排序（见图 3.5），unsrt 不排序（保持引用的次序）；alpha 则使用一种三字母缩写的方式编号并按作者排序（见图 3.6）；abbrv 格式与 plain 基本相同，只是定义了一些缩写，如：

```
\bibliographystyle{alpha}
```

表 3.4　几种基本 BibTeX 文献格式的属性比较。表中“月份全称”和“期刊全称”是指对于一些预定义 BibTeX 宏的按月份和期刊名的全称或缩写展开

格式	字母编号	条目排序	不缩写人名	月份全称	期刊全称
plain	✗	✓	✓	✓	✓
unsrt	✗	✗	✓	✓	✓
alpha	✓	✓	✓	✓	✓
abbrv	✗	✓	✗	✗	✗

TeX and LaTeX see [1], [2].

References

[1] Donald Ervin Knuth. *The TeXbook*, volume A of *Computers & Typesetting*. Addison-Wesley, 1986.

[2] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusett, 2nd edition, November 1994.

[3] Frank Mittelbach and Michel Goossens. *The LaTeX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, second edition, 2004.

图 3.5　plain 格式的引用和文献列表

2. 在正文中用标签使用 \cite 命令引用需要的文献，或是使用 \nocite 命令指明不引用但仍需要列出的文献标签。\cite 命令引用的位置会出现文献的编号，同时将提示 LaTeX 列出所引用的文献，如：

```
\TeX{} and \LaTeX{} see \cite{knuthtex1986}, \cite{lamport1994}.
\nocite{mittelbach2004}
```

如果要列出数据库中的所有文献，可以用 \nocite{*} 命令，不过这只适用于专为一篇文章编写的小型文献数据库。

\cite 命令中可以直接使用多个文献的标签，表示同时引用多条文献，也可以带一个可选参数，用来表示附加的说明，例如 \cite[\S~4.3]{lamport1994}

TeX and LaTeX see [Knu86], [Lam94].

References

[Knu86] Donald Ervin Knuth. *The TeXbook*, volume A of *Computers & Typesetting*. Addison-Wesley, 1986.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusett, 2nd edition, November 1994.

[MG04] Frank Mittelbach and Michel Goossens. *The LaTeX Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, second edition, 2004.

图 3.6 alpha 格式的引用和文献列表

可能会得到“[136, § 4.3]”。

3. 使用 \bibliography 命令指明要使用的文献数据库。LaTeX 将在这条命令的位置插入参考文献列表，如：

```
\bibliography{tex}% 表示使用的数据库是 tex.bib
```

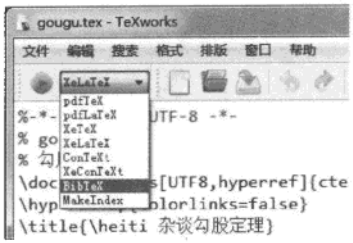
指定数据库文件时不带 .bib 后缀。可以同时从多个文献数据库中提取文献，只要将用到的所有文献数据库文件用逗号分隔开即可，如：

```
\bibliography{springer,ieee}
```

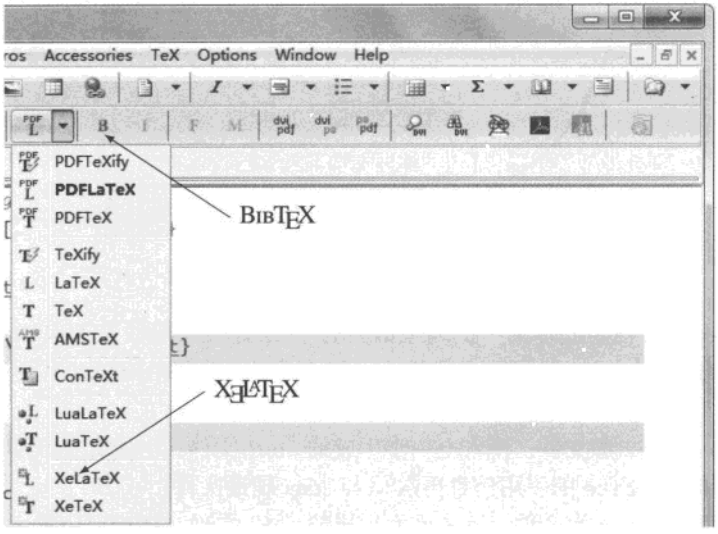
BibTeX 是一个单独的命令行工具，要使用 BibTeX，必须先对 .tex 源文件编译，然后运行 BibTeX 生成文献列表，然后再对 .tex 源文件至少编译两次，才能得到插入文献列表并有正确引用的文档。以 XeLaTeX 为例，编译带有参考文献列表的文件 foo.tex 的命令是：

```
xelatex foo.tex
bibtex foo.aux
xelatex foo.tex
xelatex foo.tex
```

其中文件的扩展名可以省略。当然，使用专门为 TeX 配置的编辑器，上述四条命令就相当于点了四次按钮（见图 3.7）。有的编辑器配置还可能把几条命令合为一个按钮。



(a) TeXworks



(b) WinEdt

图 3.7　TeXworks 与 WinEdt 中的 BibTeX 命令按钮

如图 3.8 所示，BibTeX 的参考文献列表和引用也是通过读写一系列辅助文件完成的，不过这个功能比一般的目录和交叉引用生成还要复杂一些，要得到完整的文献列表和正确的引用信息，至少需要以下四个步骤：

1. 编译 .tex 源文件，生成没有文献列表的 PDF 文件，同时将 \cite 命令产生的引用信息、\bibliography 指定的数据库名、\bibliographystyle 指定的文献格式名都写入 .aux 辅助文件；

*本节后面内容初次阅读可跳过。

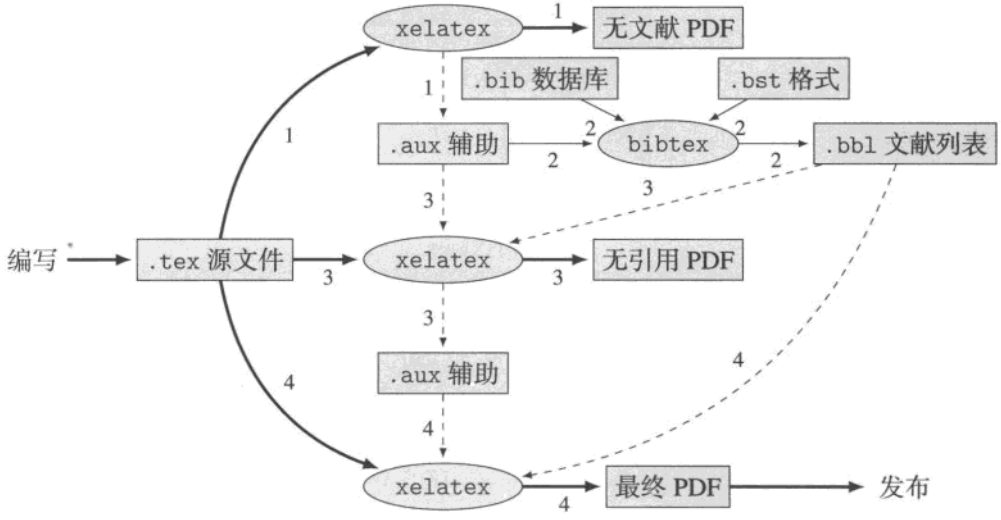


图 3.8 BibTeX 编译处理流程（这里以 XeLaTeX 为例）

2. 使用 bibtex 程序处理第一次编译得到的 .aux 辅助文件，按照其中记录的引用、文献信息，从 .bib 数据库中提取出排版参考文献列表的 LaTeX 代码，写入 .bbl 文件；
3. 再次编译 .tex 源文件，读入上一步生成的 .bbl 文件，生成有文献列表的 PDF 文件，同时将 \cite 的引用信息再次写入 .aux 辅助文件；
4. 第三次编译 .tex 源文件，读入前面生成的 .bbl 文件，在指定位置生成文献列表，读入上一步生成的 .aux 辅助文件，在引用处生成正确的引用编号信息，得到有正确文献列表和引用的 PDF 文件。

上面的过程中，后面生成 \cite 命令引用编号的信息与一般的交叉引用没有区别，只是前面增加了调用 BibTeX 生成文献列表的过程。当然，这一流程是针对最后输出完全正确的文献、引用信息而言的，在文档编写、修改过程中，如果不是为了检查最终的结果，只需要编译一遍就可以了。

下面我们来看 .bib 文献数据库文件的编写。如前所示，一个 BibTeX 数据库条目的结构为：

```
@⟨类型⟩{⟨引用标签⟩,
    ⟨项目⟩ = {⟨项目内容⟩},
    ⟨项目⟩ = {⟨项目内容⟩},
```

```
    ......
}
```

文献数据库的输出方式由 .bst 文献格式文件确定，在基本文献格式中定义的文献类型如下[193]，类型名不区分大小写：

- **article** 在期刊上发表的论文
  必需项目：author, title, journal, year
  可选项目：volume, number, pages, month, note
- **book** 正式出版的书籍
  必需项目：author/editor, title, publisher, year
  可选项目：volume/number, series, address, edition, month, note
- **booklet** 没有正式出版机构的印刷品
  必需项目：title
  可选项目：author, howpublished, address, month, year, note
- **conference** inproceedings 的别名
- **inbook** 书籍的一部分，可以是一章、一节或若干页等
  必需项目：author/editor, title, chapter/pages, publisher, year
  可选项目：volume/number, series, type, address, edition, month, note
- **incollection** 书中有独立标题的一部分，如论文集中的一篇
  必需项目：author, title, booktitle, publisher, year
  可选项目：editor, volume/number, series, type, chapter, pages, address, edition, month, note
- **inproceedings** 会议报告集中的一篇
  必需项目：author, title, booktitle, year
  可选项目：editor, volume/number, series, pages, address, month, organization, publisher, note
- **manual** 技术手册
  必需项目：title
  可选项目：author, organization, address, edition, month, year, note
- **mastersthesis** 硕士学位论文
  必需项目：author, title, school, year
  可选项目：type, address, month, note
- **misc** 其他难以分类的、未定义的类型会被归于此类
  必需项目：无
  可选项目：author, title, howpublished, month, year, note

- **phdthesis** 博士学位论文

  必需项目：author, title, school, year

  可选项目：type, address, month, note

- **proceedings** 会议报告集

  必需项目：title, year

  可选项目：editor, volumeornumber, series, address, month, organization, publisher, note

- **techreport** 学院或研究所出版的报告

  必需项目：author, title, institution, year

  可选项目：type, number, address, month, note

- **unpublished** 未出版的文档

  必需项目：author, title, note

  可选项目：month, year

对每一种文献类型，都可以有许多不同的排版项目，有的项目对于特定文献类型是必需的，有些是可选的。必需项目如果缺失，BibTeX 在处理时会发出警告，并以问号排版缺失的部分，标题、作者、年代是大部分文献类型的必需项目，如果遇到未定义的项目，BibTeX 会忽略它。在基本文献格式中定义的项目如下[193]，不区分大小写：

- **address** publisher（出版社）的地址，对于大的出版社，通常可以省略。
- **author** 作者姓名，不同的作者之间用 and 分隔（无论多少个作者）。这里姓名可以是名␣姓 或 姓,␣名 的格式。如果姓名中有重音符号或其他特殊排版的内容，注意放在分组里面。有些姓名的一些部分多于一个单词，可放在单独的分组中。汉字的中文、日文人名可以不分姓名，统一使用。如：

  ```
  author={D. Hilbert and G{\"o}del, Kurt and John von Neumann and 陈省身}
  ```

实际上，西文姓名被分为四个部分：First, von, Last 以及 Jr，每个部分由一个或多个词或分组构成，如数学家 John von Neumann 的名字，John 是 First 部分，von 是单独一个部分，Neumann 是 Last 部分，Jr 部分为空。BibTeX 支持以下三种姓名书写格式：

- First von Last：前面 John von Neumann 就是这样写的，如果只有两个词就没有 von 的部分，如果只有一个词就只算作 Last 部分（姓）。
- von Last, First：如前面 Kurt Gödel 的名字就写成了 G{\"o}del, Kurt。
- von Last, Jr, First：这里 Jr 部分使用 Junior、Senior 或者罗马数字 III、IV 等表示同名的第几代人。

部分复杂的人名录入时必须小心，更详细的说明可参见 [152, § 11]。

- **booktitle** （所在）书籍的标题。
- **chapter** 章编号，如“2”。
- **crossref** 被这一文献所引用的 key 值。
- **edition** 书籍的出版版次，如“Second”。
- **editor** 编辑的姓名，格式与 author 一致。
- **howpublished** 特殊的出版方式。
- **institution** 技术报告的主办机构。
- **journal** 期刊名。标准文献格式中用宏定义了少量期刊名的简写。
- **key** 用于 crossref 项等。
- **month** 发表或出版的月份。
- **note** 额外的说明。
- **number** 期刊号、丛书号、报告编号等。
- **organization** 主办会议或发布手册的机构。
- **pages** 页码，多使用页码的范围表示所引用的文献位置，书籍类型则用来表示总页码。如 13--20，370 + xii。
- **publisher** 出版社名。
- **school** 学院。
- **series** 丛书名。
- **title** 文献标题。
- **type** 技术报告的类型，如“Research Note”。
- **volume** 文献所在期刊或多卷丛书的卷数。
- **year** 出版年份，或未出版文献的写作年份。

书写文献项目内容时，除非是年份等纯数字，应该把项目内容放在双引号 " " 之间或是放在花括号的分组内。项目中的内容如果包含重音符号等命令，要把它放在分组之内保证 BibTeX 正确处理。由于输入的内容有可能在进入 LaTeX 编译前就被 BibTeX 程序改变大小写，因此在不希望改变大小写的地方（如标题中的人名），就要使用分组来保持原来的大小写，如：

```
title="Harmonic analysis of operators on {Hilbert} space"
```

可以在 .bib 数据库中定义一种特殊的字符串（string）类型，然后可以在其他文献中使用它来代替重复的部分。基本的文献格式中也预定义了一些这样的宏，如用三字母的缩写表示月份，以及一些期刊的名称，这种宏与 .bib 中的字符串具有相同的效果。

下面这条 Knuth 的 Literate Programming 的文章就使用了两个字符串宏，一个是在 .bib 文件中直接定义的期刊名缩写 j-CJ，一个是五月的缩写 may：

```
@String{ j-CJ = "The Computer Journal" }

@Article{Knuth:CJ-2-27-97,
  author =       "Donald E. Knuth",
  title =        "Literate Programming",
  journal =      j-CJ,
  year =         "1984",
  number =       "2",
  volume =       "27",
  pages =        "97--111",
  month =        may,
}
```

3-3-2

同一项目定义里面的不同字符串之间可以用 # 连接，也可以用它连接预定义的字符串或宏，如 January 2（1 月 2 日）可以写成：

```
month = jan # "~2"
```

此外，在 .bib 文件中还有一个导言（preamble）类型，可以在里面写一些在文献中可能使用的 TeX 代码，特别是可以用 \providecommand 命令给出一些定义：

```
@Preamble{
  \providecommand\url{\texttt}% 如果没有定义 \url 命令，用 \texttt 代替
}
```

练习

3.1 打开 CTAN 的 Web 页面或 FTP 网站，查看发布在 CTAN 上面的文献数据库，找到其中关于 TeX/LaTeX 本身的文献数据库，看看 Knuth [126] 在哪个数据库中？（有关 CTAN 可参见 8.3 节“LaTeX 资源寻找”。）

3.2 使用 Google Scholar 或其他数据库检索 Hoare 关于快速排序（quicksort）算法的原始论文，给出其 BibTeX 数据库条目。

3.3 使用 BibTeX 按 plain 格式排版一篇文章，引用几篇你熟悉领域的文献。如果不用 \cite 命令引用文献，会发生什么情况？把 plain 改成其他格式再看看结果如何。

3.3.2　JabRef 与文献数据库管理

尽管 .bib 文献数据库是纯文本文件，可以用一般的文本编辑器编辑，但使用专门的文献管理工具可能更加方便。JabRef 就是一款轻量级的开源文献管理软件，专门用来处理 BibTeX 的文献数据库。

JabRef 是 Java 程序，Windows 平台下使用可能需要首先安装 Java 运行库：

```
http://www.java.com/zh_CN/download/manual.jsp
```

可以在 SourceForge 的站点下载安装 JabRef：

```
http://jabref.sourceforge.net/
```

刚刚安装的 JabRef 是英文界面，不过没有关系，我们可以在 Options 菜单中的 Preferences 项中设置 JabRef 的界面语言和默认编码，见图 3.9。由于一般使用 XeLaTeX 处理中文文档，所以这里将默认的编码设置为 UTF-8，而不是通常中文操作系统默认的 GBK 编码。设置以后重启 JabRef，它的程序界面就会像图 1.17、图 3.10 一样成为中文的了。同样，可以在首选项菜单中设置合适的程序字体和字号。

JabRef 安装好之后，就会与 .bib 类型的文件关联，可以用它直接打开扩展名为 .bib 的数据库文件。

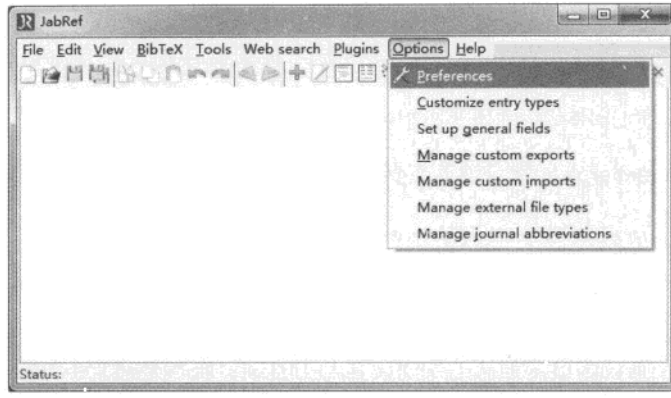
对于现成的数据库文件，如直接从网站上下载的 .bib 文件不需要修改就可以直接使用，此时可以把 JabRef 简单地看做一个文献数据库的查看器，可以用它查看所有的文献信息及引用标签。对于特定的编辑器，如 WinEdt、Vim、Emacs 等，还可以点击 JabRef 上的一个按钮直接把 \cite{(标签)} 命令发送到编辑器中[①]，非常方便（见图 3.10）。

建立新的文献数据库非常简单，在“文件”菜单中选“新建数据库”，或在工具栏点新建的图标，就会得到一个未命名的空数据库文件。直接点击保存命令就可以为数据库进行命名。

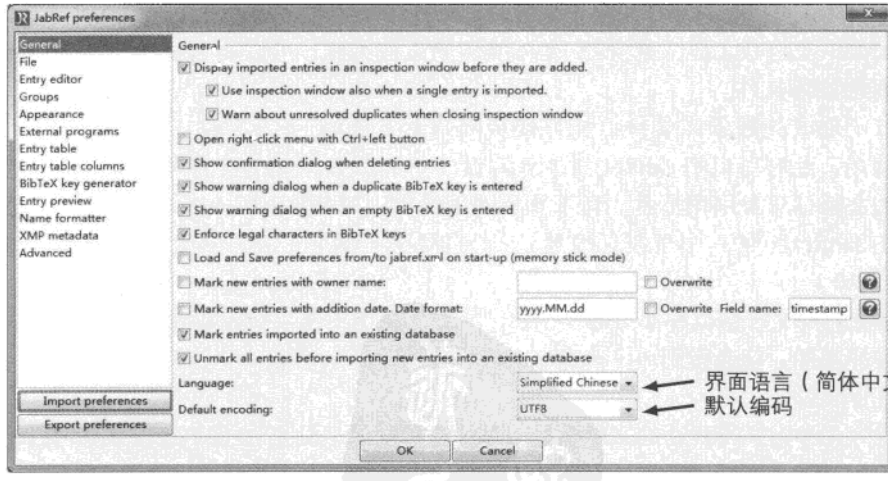
有好几种方式得到新的文献条目，最为自动化的一种是让 JabRef 直接联网进行搜索。点击“Web 搜索”菜单，可以在其中选择一个在线数据库进行搜索抓取（见图 3.11），然后将所有搜索到的文献导入到当前打开的 .bib 文件中。

也可以从其他类型的文献数据库中导入条目。在 JabRef 的“文件”菜单中有“导入到新数据库”和“导入到当前数据库”命令，可以从许多其他文献管理软件（如 Endnote）导入条目。

① 要正确完成推送，可能需要在 JabRef 的选项中进行简单的设置。



(a) JabRef 初安装的界面与选项菜单



(b) JabRef 首选项（Preferences）设置

图 3.9 设置 JabRef 的界面语言和默认编码