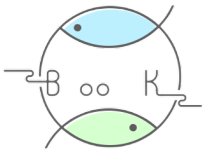
LaTeX 入门

作者：刘海洋

项目：GitHub lihui588211/BookRewriting

时间：2023/10/2

书籍模板：ElegantBook



目录

第一章 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、注重细节和品味。而 LaTeX 则为了减轻这种写作、排版一肩挑的负担，把大片排版的格式隐藏在若干样式之后，以内容的逻辑结构统帅纷繁的格式，使之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。[1]

LaTeX 的读音和写法

TeX 一名源自 technology 的希腊词根 $\tau\varepsilon\chi$，TeX 之父高德纳教授近乎固执地要求它的发音必须是（按国际音标）[tɛx]，尽管英语中它常被读作 [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”。

LaTeX 这个名字则是把 LaTeX 之父 Lamport 博士的姓和 TeX 混合得到的。所以 LaTeX 大约应该读成“拉泰赫”。不过人们仍然按照自己的理解和拼写发音习惯去读它：[’la:tɛk]、[’lei:tɛk] 或是 [la:’tɛk]，甚至不怎么合理的 [’leitɛks] [2]。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。

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 字母错位的排版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 2ε，意思是超出了第 2 版，接近却没有达到第 3 版，因此写成LaTeX 2ε。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，虽然比较烦琐，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，或者在读本书之前就已经迫不及待地做了不少尝试的话，可以直接跳到开始一个实际的例子。

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 发行版就是把这样的部件都集合起来，打包发布的软件。

[1] 不过现代人好像都追求快，沉不下心来。这也是我通过此项目强迫自己学习的理由

[2] 额。我的读法...

尽管内容复杂，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 的 CTeX 套装，二是1.1.1.2 的 TeX Live。前者是 Windows 系统下的软件，后者则可以用在各种常用的桌面操作系统上。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[3]。

1.1.1.1 C TeX 套装

已过时，跳过。

1.1.1.2 TeX Live

TeX Live 是由 TUG （TeXUser Group ， TeX 用户组）发布的一个发行版。TeX Live 可以在类 Unix/Linux 、macOS 和 Windows 系统等不同操作系统下安装使用，并且提供相当可靠的工作环境。TeX Live 可以安装到硬盘上运行，也可以经过便携（portable ）安装刻录在光盘上直接运行（故有“Live ”之称）。

不同操作系统下安装设置 TeX Live 的方式基本一样，这里以 Windows 操作系统为例进行演示。[4]

TeX Live 一般以安装镜像的方式在互联网上发布。光盘镜像文件可以从官网上下载[5]。载入镜像后，执行 `install-tl-windows.bat` 进行安装。只要选好安装的位置，不断单击“下一步”就可以安装 TeX Live 了，如图所示。

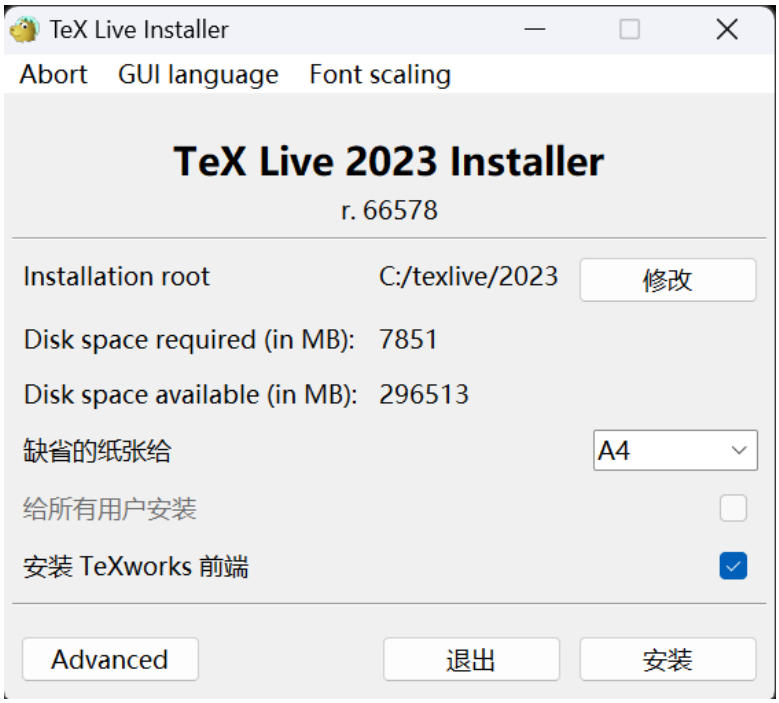


图 1.1: 在 Windows 11 上安装 TeX Live2023

程序安装好了之后，会在开始菜单栏增加 TeX Live 文件夹图标。其中包含的内容比较简单，它包含以下项目：

- **TeXworks editor ：** 这是 TeX Live 预装的一个 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。我们很少用到它。

[3]并不是。由于 CTeX 早已停止维护及历史原因，现在 TeX Live 是更好的选择。于是我选择跳过1.1.1.1

[4]原书中还包含部分 Linux 系统上的操作讲解，这里删去。

[5]https://mirrors.tuna.tsinghua.edu.cn/CTAN/systems/texlive/Images/

- **TeX Live command-line**：它打开 Windows 的命令提示符，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 4000 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 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。
- **TLShell TeX Live Manager**：这是 TeX Live 管理工具的图形界面，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。

1.1.2 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，会使工作方便很多。

LaTeX 代码编辑器有很多，大致可以分为两类：一种主要为 LaTeX/TeX 代码编辑而专门设计的编辑器，二是可以为 LaTeX/TeX 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一些，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 的界面非常简洁（见图1.2）：它分为两个部分，左侧是 TeX 源文件的编辑窗口，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栏，接着是工具栏，中间最大的编辑区，最下面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。

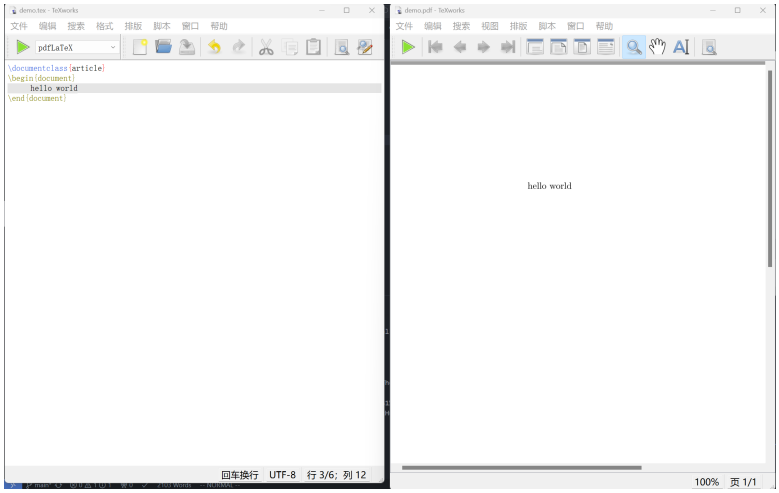


图 1.2: TeXworks 界面

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中用来选择排版时所使用的命令，通常对应一条单一的命令，但也可以配置为好几条命令的复合。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。

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 也很简单：

1. 在编辑区输入 TeX 源文件
2. 单击“保存”按钮，给源文件起名并保存在指定位置
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”面板中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 结果。

在文本编辑区或 PDF 预览区用鼠标左键单击可以从源文件跳转到 PDF 文件中的对应位置；或反过来从 PDF 跳转到源文件中的对应位置。这个功能称为 TeX 文档的**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 输出程序的共同参与，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} *
```

光标在环境中央等待输入，再次按下 Ctrl + Tab 组合键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。

下面来看看 TeXworks 中一些常见的配置。

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，给编辑工作带来诸多不便。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。要使字体和语法高亮的设置对所有文档生效，则应该修改 TeXworks 的默认选项。单击 TeXworks “编辑”菜单的最后一项“首选项”，将弹出 TeXworks 首选项窗口（见图1.3）。在“编辑器”选项卡中，可以设置编辑器默认的字体和字号，下面则有语法高亮、自动缩进等格式。

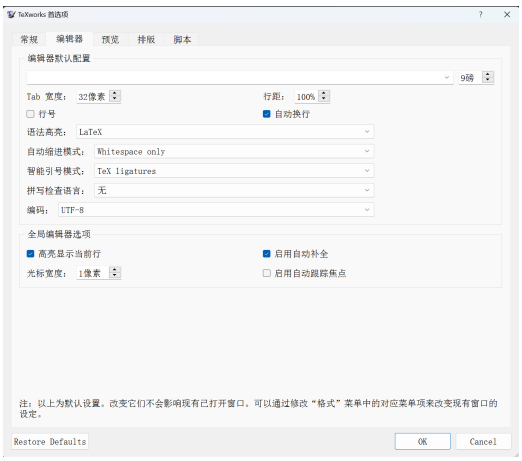


图 1.3: TeXworks 编辑器首选项设置

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，可以在首选项设置窗口的“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，一般应选择 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 首选项窗口的“排版”选项卡可以用来设置 TeXworks 的“排版”按钮所执行的命令。选择对应的处理工具，单击“编辑...”按钮，就可以在弹出的窗口中设置对应的命令及参数。参数中使用的变量，可参见

TeXworks 的帮助文档。

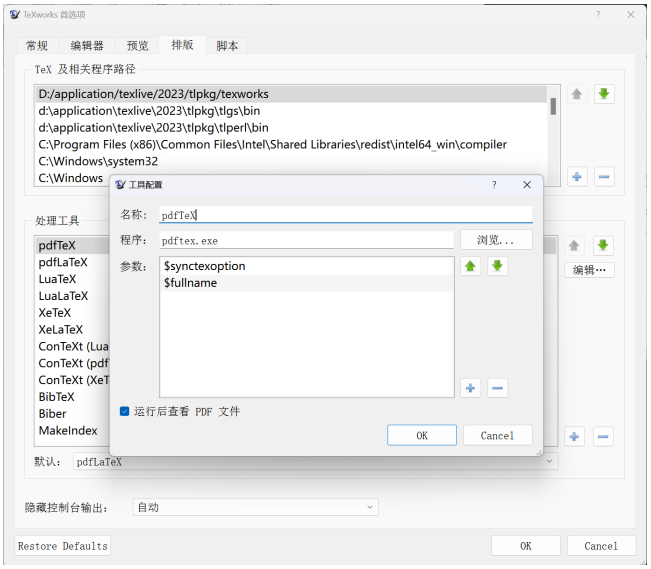


图 1.4: TeXworks 排版首选项设置

文本编码与 Unicode

在使用 TEX 编辑器时，必须要注意文档保存的文字编码。如果编码使用错误，轻则遇到“乱码”，重则导致程序运行错误。我们提到的“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、储存和运输的，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一种编码编一个号，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。

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 110181 个字符，包括世界上 100 种文字，而且还在不断修订拓充之中。[6]Unicode 已逐渐成为字符编码的新方向，包括 GB 18030 也可以看成是 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，所有拓展 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其实是一种黑客手段，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于起步阶段，本书将着重介绍基于 XeTeX 的方式。

[6] 目前最新版本的 Unicode 13.0 中，包含的字符总数已达 143859 个。

1.1.2.2 PDF 阅读器

TeX Live 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。现在使用的 TeX 系统基本上最终都输出 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后效果。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 的效果和 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。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 PDF 文件到源文件的正反向索引。

要满足第二个要求，我们建议使用 Adobe Reader 最新的版本[7]（Linux 系统中通常称为 arcoread ）。Adobe Reader 是官方免费提供的 PDF 格式阅读器，通常它的显示结果最好，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 制作的幻灯片中使用到。）其他一些常见的 PDF 阅读器，如 Foxit Reader ，则可能在一些功能上有欠缺。

如果你还没有安装 Adobe Reader ，可以直接从 Adobe 的官方，或几乎任何的下载站点中得到并安装。不必抱怨它占用上百兆的安装空间：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 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 。于是，PostScript 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。就连国内最大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门语言的特点：它是一种基于后缀表达式和栈操作的解释型计算机语言。例如，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）到坐标（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 代码也都是由其他计算机程序自动生成的。不过，富有经验的老手可以就凭借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将要讲到的 `PSTricks`宏包中。

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，因而出现了所谓封装的 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的图形格式。TeX 的许多输出引擎都支持这种图形格式。

由于在电子出版领域的地位，PostScript 一度成为 TeX 最重要的输出格式，至今也能在网络上见到大量 TeX 系统产生的 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 和 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

[7]全名为 Adobe Acrobat Reader

方面则比 PostScript 大为削减，并增强文档格式结构化，可以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。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，TeX Live 还给出了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做一个了解。

命令行是以文字形式与计算机交互的方式，与图形方式相对。在 Windows 系统中，命令行通常由命令行解释程序 `cmd.exe` 处理；在 Linux 及其他类 UNIX 操作系统中，命令行解释程序通常称为 Shell ，最常见的 Shell 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` 可以在“开始”菜单搜索框中输入 `cmd` 查找到“命令提示符”进入。如果使用频繁，可以在桌面或任务栏中创建快捷方式，或设定快捷键随时使用。

使用 TeX 经常需要在特定文件所在的目录进行命令行操作，这时只需要在资源管理器的地址栏内输入 `cmd` 回车。

打开命令行窗口之后，会显示命令提示符，默认的命令提示符由当前盘符、目录和一个大于号 `>` 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，输入一行命令按下回车键即开始执行。

使用最频繁的命令是文件列表命令 `dir` （directory 的缩写），直接输入 `dir` 后按下回车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。`?` 可以替代任意一个字符，`*` 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` 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）和当前目录的上一级目录。

`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冒号：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中，保存为后缀为 .bat 或 .cmd 的文件，就得到一个批处理文件（又称为批处理脚本）。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、文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。

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 TeX Live 附带安装了一份简化版本的 GhostScript，程序名为 rungs。一般还是最好单独下载安装完全版本的 GhostScript[8]，因为一些 LaTeX 输出引擎有时仍会调用它。

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 ERS 图片时自动调用 GhostScript 。

GhostScript 为一些常用的转换提供简单的命令，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 。类似的命令还包括 pdf2ps 和 ps2eps等。

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 gswin64.exe，在 Linux 等系统下通常叫做 gs，也可以使用 TeX Live 的 rungs，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.ps
```

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 128dpi 的分辨率，裁剪到适当大小，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 https://imagemagick.org/script/download.php进行下载安装。

在 Windows 下安装 ImageMagick 可以在开始菜单栏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 的联机帮助文档。

[8] https://www.ghostscript.com/releases/gsdnld.html

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.jpg
```

则是把 EPS 格式的图片 `foo.eps` 转换为 JPG 格式的图片 `bar.jpg` 。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”菜单栏进入联机帮助，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储存，如 GhostScript 和 ImageMagick 在 Windows 下都提供网页形式的帮助文档，可以在“开始”菜单栏中找到。也有很多程序提供 CHM 、PDF 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 命令行的基本命令后加 `/?` 参数获得帮助。如输入 `dir /?`可以在屏幕上获取到 `dir` 命令的帮助信息。来自类 UNIX 系统的程序命令行选项以 `-`开头，命令行帮助通常可以在命令后加上 `--help` 获得。如输入`convert --help` 将会在屏幕上得到关于 ImageMagick 的所有命令行选项的说明。

类 UNIX 系统在命令行下还有一个 `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 系统下的 `dir` 命令）的详细帮助，类似的文档程序还有 `info` 。在 Windows 中，用 `help` 可以达到类似的效果，不过效果与使用 `/?` 选项差不多。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。

1.1.3 “Happy TeXing 与“特可爱排版”

在做完所有的准备工作后，我们一起来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：

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

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 或者 XeLaTeX 命令，点击编辑器上对应的排版按钮。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 !

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。虽然我们之前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 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：

```
\documentclass[UTF8]{ctexart}

\begin{document}
\section{汉字}
特可爱排版
\section{数学}
\[
    a^2 + b^2 = c^2
\]
\end{document}
```

1 汉字

特可爱排版

2 数学

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

这段代码其实也不难看懂：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；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；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，你一定早就听说过它了。

上面两个简单的例子给了我们一个 LaTeX 的直观印象，而且正确运行它们也许能够增强你学习 LaTeX 的信心。粗略地看，LaTeX 是一种标记式排版语言，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，命令一般以英文单词命名，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。

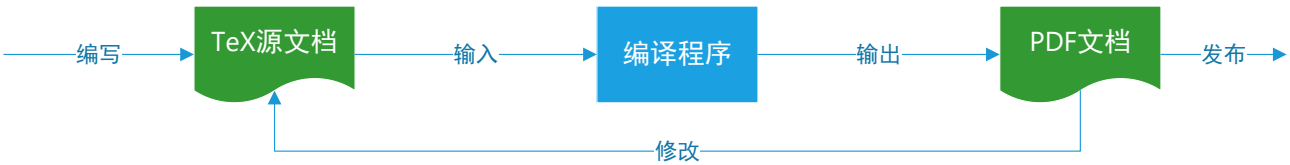


图 1.5: LaTeX 文档的写作流程

LaTeX 的写作流程见图1.5。通常这个流程都是自动化完成的，编写 TeX 源文件通常是在专门的 TeX 编辑器中进行，例如 TeXworks，而后按下一个按钮，源文件就会被送给 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。

编译程序

TeXworks 等编辑器里面给出了许多编译程序的按钮，往往让人有种应接不暇的感觉，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？

高纳德设计的 TEX 原本只是一个相对简单的程序，命令 `tex` 可以调用最基本的 TEX 程序。它使用高纳德定义的一个相对简单的格式 Plain TEX 进行排版。`tex` 读入 TEX 源文件，输出一种“与设备无关”（DeVive Independent ）的格式，即 DVI 文件，DVI 文件曾经是 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，但在 PostScript 和 PDF 流行之后，DVI 格式就主要成为一种到 PS 或 PDF 的中间格式了。

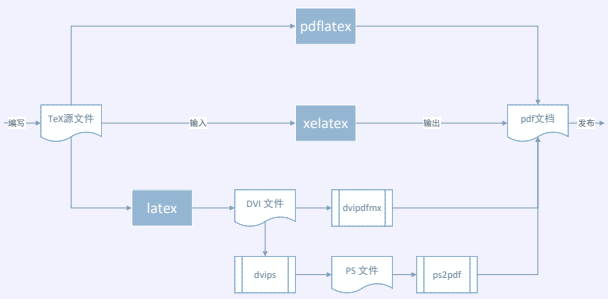
程序 Dvips 将 DVI 文件转换为 PostScript 文件，可以直接拿到支持 PostScript 的 ps2pdf 或 Adobe Acrobat 提供的 Distiller 等程序再从 PostScript 转换到 PDF 文件。PDF 流行之后，又有了能直接把 DVI 文件直接转换成 PDF 的 dvipdf 程序，后来又出现了更为先进的 dvipdfm 和 dvipdfmx ，可以支持更丰富的 PDF 功能和东亚字体等，现在新的发行版中主要还在使用的是 dvipdfmx （常常写作 DVIPDFMx ）。这类把 DVI 文件转换为其他实用格式的程序常被称为 TEX 输出的驱动（driver ）。

除了最初的 TEX 程序，后来有很多人对 TEX 进行了拓展。先是有了 ε– TEX，后来在 ε– TEX 的基础上，Hàn Thế Thành 设计了能够直接输出 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 的后续是 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 ，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 ASCII 编码改为 UTF-8 编码，并且可以更方便地使用各种字体。TEX 程序连同这些扩展被称为不同的 TEX 引擎（engine ）。

不同的引擎都可以编译 Plain TEX、LATEX 或是 ConTEXt 等不同格式的文档，不同的组合就使用不同的命令，如下表所示，我们主要关注 PDFTeX 和 XeTeX 引擎使用 LATEX 格式的命令。

命令 / 格式 / 引擎	Plain TEX	LATEX	ConTEXt	
TEX/ε-TEX	tex / etex			输出 DVI
pdfTEX	tex	**latex**		
	pdftex	**pdflatex**	texexec	输出 PDF
XƎTEX	xetex	**xelatex**	特殊参数	
LuaTEX	luatex	lualatex	context	

使用 LATEX 格式的排版得到 PDF 文件的方式也有好几种。其中使用 `latex + dvips` 的方式最为古老，不便于中文文档的排版。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，一些宏包的兼容性会更好一些。不过本书的大部分内容并不限于任何一种模式，只是在处理中文时，将主要讨论 `xelatex`。



1.2 从一个例子说起

这一节将研究一个相对实际的例子。在这个简单的例子中，我们将看到在真正的写作排版工作中常遇到的一些模式、问题的解决思路。有一些代码或许一时难以理解，不用担心，我们将在后面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的很小，四页拼成一页，完成的样子如图所示。如果你以前已经对 LaTeX 有一些基础，不妨自己动手排一下这个小例子（不偷看本章后面的说明），看看你能否准确高效地完成；即使你对 LaTeX 的实际了解有限，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：

```
1  % -*- coding: UTF-8 -*-
2  % gougu.tex
3  % 勾股定理
4  \documentclass[UTF8]{ctexart}
5
6  \title{杂谈勾股定理}
7  \author{张三}
8  \date{\today}
9
10 \bibliographystyle{plain}
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% 开头的是**注释**。在 TeX 中，源文件一行中% 后的内容都会被忽略。这里有三行注释，第 1 行表明了这个文档的编码是 UTF-8 ，这对中文文档往往非常有用；第 2 行是源文件的文件名gougu.tex；第三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，源文件往往分成几个文件，这类说明性文字就十分重要了。

- 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。
- 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。
- 第 10 行的\bibliographystyle声明**参考文献**的格式。

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，或者自定义一些命令。

- 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。
- 第 15 行的\tableofcontents命令输出目录。
- 第 16 至 17 行的两个\section开始新的一节。
- 最后第 18 行的\bibliography{math} 则是提示 TEX 从文献数据库 math中提取文献信息，打印参考文献列表。

为了格式上的清晰，源文件中适当加入了一些空行作为分隔。在正文外的部分，空行不代表任何意义。

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。注意这里生成目录至少要编译两次，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␣世纪的毕达哥拉斯学派。该学派得到了一个法则，
2 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得《
3 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正方形之和。”证明是用面积做的。
4
5 我国《周髀算经》载商高（约公元前␣12␣世纪）答周工问……
```

填写正文的部分看起来比较简单，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。空白行，也就是至多有空白的行，会使文字另起一段。空行只是起分段的作用，使用很多空行并不起增大段间距的作用。
- **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，而其它符号后面的空格则会保留**，因而用*左␣右*| 就得到连续的“左右”，但left␣right就会输出有空格的“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，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它符号之间加上一个空格，这会使代码更加清晰。

换行与空格的使用，正是在 LATEX 中文字排版中最基本的部分，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。