

一份不太简短的 L^AT_EX 2_ε 介绍

或 95 分钟了解 L^AT_EX 2_ε

原版作者: Tobias Oetiker

Hubert Partl, Irene Hyna and Elisabeth Schlegl

原版版本: Version 5.05, July 18, 2015

中文翻译: ChinaT_EX 论坛

中文版本: 版本 5.05, 二零一六年二月

版权所有 © 1995 – 2005 Tobias Oetiker 及贡献者。保留所有权利。

本文档是自由的；你可以在自由软件协会颁布的 GNU 通用公共许可证的条款下再次发布或修改本文档。许可证可以是第二版，或者任何（自行选择的）后继版本。

本文档基于使用的目的发布，但并不负责任何担保；亦没有用于商业目的或某一特定目的的任何暗示性的担保。更多的细节请查看 GNU 通用公共许可证。

你应该随本文档同时收到一份 GNU 通用公共许可证的拷贝；如果没有，请发信给自由软件协会，地址：675 Mass Ave, Cambridge, MA 02139, USA。

中文版致谢

5.05 中文版致谢

4.20 中文版致谢

中文 T_EX 学会启动的 lshort-zh-cn 修正计划！本项计划历史八个月，参加的朋友有：

CT _E X 论坛 ID	翻译章节
zpxing	前言、第二章、第五章 1-2.4 & 3、第六章
Frogge	第一章
liwenjun	第三章
lijian605	第四章
gprsnl	第五章 2.5-2.11

haginile 和 Frogge 通读了全篇，并给出了详细的勘误表。blackold 对于第二章亦有所贡献。最后由 zpxing 统筹全书。

.....

3.20 中文版致谢

本文档的翻译工作由 CT_EX 版主“经典问题”倡议，历经近十个月才得以完成。期间参与翻译工作的朋友有：

CT _E X 论坛 ID	翻译章节	源文件名
经典问题	前言	overview.tex
高原之狼	第一章	things.tex
controlong	第二章	typeset.tex
cxterm	第三章	math.tex, lssym.tex
aloft	第四章	spec.tex
ganzhi	第五章	custom.tex

在此特向这些奉献者表示感谢！

英文版致谢

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

Hubert Partl <partl@mail.boku.ac.at>

Zentraler Informatikdienst der Universität für Bodenkultur Wien

Irene Hyna <Irene.Hyna@bmwf.ac.at>

Bundesministerium für Wissenschaft und Forschung Wien

Elisabeth Schlegl <noemail>

in Graz

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

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

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

前言

L^AT_EX [1] 是一个文档准备系统 (Document Preparing System), 它非常适用于生成高印刷质量的科技类和数学类文档。它也能够生成所有其他种类的文档, 小到简单的信件, 大到完整的书籍。L^AT_EX 使用 T_EX [5] 作为它的排版引擎。

这份短小的手册描述了 L^AT_EX 2_ε 的使用, 对 L^AT_EX 的大多数应用来说应该是足够了。参考文献 [1, 2] 对 L^AT_EX 系统提供了完整的描述。

本手册共有八章和两篇附录:

第一章 讲述 L^AT_EX 的来源, 源代码的基本结构, 以及如何编译源代码生成文档。

第二章 讲述在 L^AT_EX 中如何书写文字, 包括中文。

第三章 讲述文档排版的基本元素——标题、目录、列表、图片、表格等等。结合前一章的内容, 你应当能够制作内容较为丰富的文档了。

第四章 L^AT_EX 排版公式的能力是众人皆知的。本章的内容涉及了一些常见的公式形式和符号。章节末尾提供了 L^AT_EX 常见的数学符号。

第五章 介绍了如何修改文档的一些基本样式, 包括字体、段落、页面尺寸、页眉页脚等。

第六章 介绍了 L^AT_EX 的一些扩展功能: 排版参考文献、排版索引、排版带有颜色和超链接的电子文档。

第七章 介绍了如何在 L^AT_EX 里使用 TikZ 绘图代码。作为入门手册, 这一部分点到为止。

第八章 当你相当熟悉前面几章的内容, 需要自己编写命令和宏包扩展 L^AT_EX 的功能时, 本章介绍了一些基本的命令满足你的需求。

附录 A 介绍了如何安装 T_EX 发行版和更新宏包。

附录 B 当新手遇到错误和需要寻求更多帮助时, 本章提供了一些基本的参考。

这些章节是循序渐进的, 建议刚刚熟悉 L^AT_EX 的读者按顺序阅读。一定要认真阅读例子的源代码, 它们贯穿全篇手册, 包含了很多的信息。

如果你已经对 L^AT_EX 较为熟练, 本手册的资源已不足够解决你的问题时, 请访问“Comprehensive T_EX Archive Network” (CTAN) 站点, 主页是 www.ctan.org。所有的宏包也可以从 mirrors.ctan.org 和遍布全球的各个镜像站点中获得。

在本书中你会找到其他引用 CTAN 的地方, 形式为 CTAN:// 和之后的树状结构。引用本身是一个超链接, 点击后将打开内容在 CTAN 上相应位置的页面。

要在自己的电脑上安装 T_EX 发行版, 请参考附录 A 中的内容。各个操作系统下的 T_EX 发行版位于 CTAN://systems。

如果你有意在这份文档中增加、删除或者改变一些内容，请通知作者。作者对 L^AT_EX 初学者的反馈特别感兴趣，尤其是关于这份介绍哪些内容很容易理解，哪些内容可能需要更好地解释，而哪些内容由于太过难以理解、非常不常用而不适宜放在本手册。

鲁尚文

<louisstuart96@gmail.com>

lshort 的的最新中文版本位于 CTAN://info/lshort/chinese。如果用户对其他语言的版本感兴趣，请浏览 CTAN://info/lshort。

目录

中文版致谢	i
英文版致谢	iii
前言	v
第一章 L ^A T _E X 须知	1
1.1 概述	1
1.1.1 T _E X	1
1.1.2 L ^A T _E X	1
1.1.3 L ^A T _E X 的优缺点	1
1.2 L ^A T _E X 命令和代码结构	2
1.2.1 L ^A T _E X 命令和环境	2
1.2.2 L ^A T _E X 源代码结构	3
1.3 用命令行操作 L ^A T _E X	3
1.3.1 引擎、格式和命令	4
1.3.2 latex 命令	4
1.3.3 pdf _l atex 和 xelatex 命令	5
1.4 宏包和文档类	5
1.4.1 文档类	5
1.4.2 宏包	6
1.5 L ^A T _E X 用到的文件一览	7
1.6 文件的组织方式	7
第二章 用 L ^A T _E X 排版文字	9
2.1 语言文字和编码	9
2.1.1 ASCII 编码	9
2.1.2 扩展编码	9
2.1.3 UTF-8 编码	10
2.2 排版中文	10
2.2.1 xeCJK 宏包	10
2.2.2 ctex 宏包和文档类	10
2.3 L ^A T _E X 中的符号	11
2.3.1 空格和分段	11
2.3.2 注释	11
2.3.3 特殊字符	11
2.4 文字强调	12

2.5	段落	12
2.6	单词之间的空格	12
2.7	断行和断页	12
2.7.1	手动断行和断页	12
2.7.2	连字符	12
2.8	预定义的字符串	12
第三章	文档元素	13
3.1	章节和目录	13
3.1.1	章节标题	13
3.1.2	目录	13
3.1.3	文档结构的划分	14
3.2	标题页	15
3.3	交叉引用	15
3.4	脚注	16
3.5	特殊环境	16
3.5.1	列表	16
3.5.2	对齐环境	17
3.5.3	引用环境	18
3.5.4	摘要环境	18
3.5.5	代码环境	18
3.6	表格	19
3.6.1	列格式	20
3.6.2	列宽	21
3.6.3	横线	21
3.6.4	合并单元格	22
3.6.5	行距控制	22
3.7	图片	23
3.8	盒子	24
3.8.1	水平盒子	24
3.8.2	带框的水平盒子	24
3.8.3	垂直盒子	25
3.8.4	标尺盒子	25
3.9	浮动体	26
3.9.1	浮动体的标题	26
3.9.2	并排和子图表	27
第四章	排版数学公式	29
4.1	\LaTeX 宏集	29
4.2	公式排版基础	29
4.2.1	数学模式和文本	30
4.3	数学符号	31
4.3.1	一般符号	31
4.3.2	指数、上下标和导数	31
4.3.3	分式和根式	32

4.3.4	关系符	32
4.3.5	算符	32
4.3.6	巨算符	33
4.3.7	数学重音和上下括号	34
4.3.8	箭头	35
4.3.9	括号和定界符	35
4.4	多行公式	35
4.4.1	长公式折行	35
4.4.2	多行公式	36
4.4.3	公用编号的多行公式	37
4.5	数组和矩阵	37
4.6	公式中的间距	38
4.7	数学符号的字体控制	39
4.7.1	数学字母字体	39
4.7.2	数学符号的尺寸	39
4.7.3	加粗的数学符号	40
4.8	定理环境	41
4.8.1	<code>amsthm</code> 宏包	42
4.8.2	证明环境和证毕符号	42
4.9	符号表	44
4.9.1	L ^A T _E X 普通符号	44
4.9.2	\mathcal{AMS} 符号	48
第五章	排版样式设定	53
5.1	字体和字号	53
5.1.1	字体样式	53
5.1.2	字号	53
5.1.3	选用字体宏包	55
5.1.4	<code>xelatex</code> 命令下使用 <code>fontspec</code> 宏包更改字体	55
5.1.5	使用 <code>xeCJK</code> 宏包更改中文字体	55
5.2	段落格式和间距	57
5.2.1	长度和长度变量	57
5.2.2	行距	57
5.2.3	段落格式	58
5.2.4	水平间距	59
5.2.5	垂直间距	59
5.3	页面和分栏	60
5.3.1	利用 <code>geometry</code> 宏包设置页面参数	60
5.3.2	页面内容的垂直对齐	62
5.3.3	分栏	62
5.4	页眉页脚	62
5.4.1	基本的页眉页脚样式	62
5.4.2	手动更改页眉页脚的内容	63
5.4.3	<code>fancyhdr</code> 宏包	64

第六章 特色工具和功能	65
6.1 参考文献和 BibTeX 工具	65
6.1.1 基本的参考文献和引用	65
6.1.2 BibTeX 数据库	66
6.1.3 使用 BibTeX 排版参考文献	66
6.1.4 natbib 宏包	68
6.2 索引和 makeindex 工具	68
6.2.1 使用 makeindex 工具的方法	68
6.2.2 索引项的写法	69
6.3 使用颜色	69
6.3.1 颜色的表达方式	70
6.3.2 带颜色的文本和盒子	71
6.4 使用超链接	71
6.4.1 hyperref 宏包	71
6.4.2 超链接	72
6.4.3 PDF 书签	72
6.4.4 PDF 文档属性	73
第七章 绘图功能	75
7.1 绘图语言简介	75
7.2 TikZ	75
7.2.1 TikZ 坐标和路径	76
7.2.2 TikZ 绘图命令和参数	77
7.2.3 TikZ 文字结点	77
第八章 自定义 L^AT_EX 命令和功能	81
8.1 自定义命令和环境	81
8.1.1 定义新命令	81
8.1.2 定义环境	82
8.2 编写自己的宏包和文档类	82
8.2.1 编写简单的宏包	82
8.2.2 在宏包中调用其它宏包	83
8.2.3 编写自己的文档类	83
8.3 计数器	83
8.3.1 定义和修改计数器	84
8.3.2 计数器的值	84
8.3.3 L ^A T _E X 中的计数器	84
附录 A 安装 T_EX 发行版	87
A.1 T _E X 发行版简介	87
A.1.1 安装发行版	87
A.2 安装和更新宏包	88
A.2.1 手动安装宏包	89

附录 B 排除错误、寻求帮助	91
B.1 L ^A T _E X 错误	91
B.2 查看帮助文档	93
B.3 常用宏包简介	93
B.3.1 文字、公式和符号	93
B.3.2 排版元素	94
B.3.3 图表和浮动体	94
B.3.4 修改版式	95
参考文献	97
索引	99

源代码示例列表

1.1	L ^A T _E X 的一个最简单的源代码示例。	4
3.1	book 文档类的文档结构示例。	14
5.1	fancyhdr 宏包的使用方法示例。	64
6.1	BIB _T E _X 数据库示例 books.bib。	67
6.2	利用 books.bib 生成参考文献的源代码 demo.tex。	67
7.1	TikZ 绘图示例源代码。	80
8.1	宏包的一个最简示例。	83

第一章 L^AT_EX 须知

欢迎加入 L^AT_EX 使用者的队伍！本章开头用简短的篇幅介绍了 L^AT_EX 的来源，然后介绍了 L^AT_EX 源代码的写法，如何编译 L^AT_EX 源代码生成文档，以及理解接下来的章节所必须的一些知识。

1.1 概述

1.1.1 T_EX

T_EX 是高德纳 (Donald E. Knuth) 开发的、以排版文字和数学公式为目的的一个计算机软件 [5]。高德纳从 1977 年开始开发 T_EX，以发掘当时开始用于出版工业的数字印刷设备的潜力。正在编写著作《计算机程序设计艺术》的高德纳，意图扭转排版质量每况愈下的状况，以免影响他的出书。我们现在使用的 T_EX 排版引擎发布于 1982 年，在 1989 年又稍加改进以更好地支持 8-bit 字符和多语言排版。T_EX 以其卓越的稳定性、跨平台、几乎没有 Bug 而著称。T_EX 的版本号不断趋近于 π ，当前为 3.141592653。

T_EX 读作“Tech”，其中“ch”的发音类似于“h”，与汉字“泰赫”的发音类似。T_EX 的拼写来自希腊词语 τεχνική (technique, 技术) 的开头几个字母。在 ASCII 字符环境，T_EX 写作 TeX。

1.1.2 L^AT_EX

L^AT_EX 为 T_EX 基础上的一套格式，令作者能够使用预定义的专业格式以较高质量排版和印刷他们的作品。L^AT_EX 的最初开发者为 Leslie Lamport 博士 [1]。L^AT_EX 使用 T_EX 程序作为自己的排版引擎。当下 L^AT_EX 主要的维护者为 Frank Mittelbach。

L^AT_EX 读作“Lah-tech”或者“Lay-tech”，近似于汉字“拉泰赫”或“雷泰赫”。L^AT_EX 在 ASCII 字符环境写作 LaTeX。当前的 L^AT_EX 版本为 L^AT_EX 2 _{ϵ} ，意思是超出了第二版，接近但没达到第三版，在 ASCII 字符环境写作 LaTeX2 _{ϵ} 。

1.1.3 L^AT_EX 的优缺点

L^AT_EX 总会拿来和一些“所见即所得”(What You See Is What You Get) 的文字处理和排版工具比较优缺点。笔者认为这种比较并不值得提倡，毕竟所有工具都有自己值得使用的原因。

当然笔者还是会总结一些 L^AT_EX 的优点：

- 专业的排版输出，产生的文档看上去就像“印刷品”一样。
- 方便而强大的数学公式排版能力，无出其右。
- 绝大多数时候，用户只需掌握一些简单易懂的命令来组织文档结构，无需（或很少）操心文档的版面设计。

- 很容易生成复杂的专业排版元素，如脚注、交叉引用、参考文献、目录等。
- 强大的扩展性。世界各地的人开发了数以千计的 L^AT_EX 宏包用于补充和扩展 L^AT_EX 的功能。本手册附录中的 B.3 小节可见一瞥。更多的宏包参考 *The L^AT_EX companion*^[2]。
- L^AT_EX 促使用户写出结构良好的文档——而这也是 L^AT_EX 存在的初衷。
- L^AT_EX 依赖的 T_EX 排版引擎和其它软件是跨平台、免费、开源的。无论用户使用的是 Windows, OS X, GNU/Linux 还是 FreeBSD 等系统，都能轻松获得并使用这一强大的排版工具。

L^AT_EX 的缺点也是显而易见的：

- 入门门槛高。本手册的副标题叫做“95 分钟了解 L^AT_EX 2_ε”，实际上 95 是本手册正文部分的页数。如果你以平均一页一分钟的速度看完了本手册，你只是粗窥门径而已，离学会它还很远。
- 排查错误困难。L^AT_EX 作为一个依靠编写代码工作的排版工具，其使用的宏语言比 C++ 或 Python 等程序设计语言在错误排查方面困难得多。它虽然能够提示错误，但不提供调试的机制，有时错误提示还很难理解。
- 样式定制困难。L^AT_EX 提供了一个基本上良好的样式，为了让用户不去关注样式而专注于文档结构。但如果想要改进 L^AT_EX 生成的文档样式则是十分困难。
- 相比“所见即所得”的模式有一些不便，为了查看生成的文档，用户总要不停地编译。

1.2 L^AT_EX 命令和代码结构

L^AT_EX 的源代码本质上是文本文件。哪怕用 Windows 的记事本或者 Linux 的 gedit 等简单的编辑器，也可以编写一份 L^AT_EX 源代码并编译出文档来。专用于编辑 L^AT_EX 源代码的编辑器如 TeXworks / TeXstudio / WinEdt 等提供了一些语法高亮、命令补全等功能，以及调用排版引擎的一些按钮。

除了文字本身，L^AT_EX 源代码之中还包括大量的命令，用在排版公式、划分文档结构、控制样式等等不同的地方。

1.2.1 L^AT_EX 命令和环境

L^AT_EX 命令以反斜线 \ 开头，为以下两种形式之一：

- 反斜线和后面的一串字母，如 \LaTeX。它们以任意非字母符号（空格、数字、标点等）作为分隔符。
- 反斜线和后面的一个非字母符号，如 \\$。它们无需分隔符。

要注意 L^AT_EX 命令是**对大小写敏感**的，比如输入 \LaTeX 命令可以生成错落有致的 L^AT_EX 字母组合，但输入 \Latex 或者 \LaTex 什么都得不到，还会报错。

字母形式的 L^AT_EX 命令忽略其后的所有空格。如果要人为引入空格，需要在命令后面加一对括号阻止其忽略空格¹：

¹另外也可以在命令后面紧跟一个 _ 命令（反斜线加空格），代表插入一个间距。比如 \TeX_user 的输出效果就是 T_EX user。

```
Shall we call ourselves
\TeX users

or \TeX{} users?
```

```
Shall we call ourselves TEXusers
or TEX users?
```

大多数的 L^AT_EX 命令是带一个或多个参数，每个参数用花括号 { 和 } 包裹。有些命令带一个或多个可选参数，以方括号 [和] 包裹。还有些命令在命令名称后可以带一个星号 *，带星号和不带星号的命令效果有一定差异。

L^AT_EX 的花括号本身也起到**分组**的作用，可以将字体、格式等的更改限制在大括号范围之内。

L^AT_EX 还引入了**环境**的用法，用以令一些效果在局部生效，或是生成特定的文档元素。L^AT_EX 环境的用法为一对命令 \begin 和 \end：

```
\begin{environment name}{arguments}
...
\end{environment name}
```

其中 environment name 为环境名，\begin 和 \end 中填写的环境名应当一致。\\begin 在 environment name 后可以带一个或多个参数，甚至可选参数。环境允许嵌套使用。

1.2.2 L^AT_EX 源代码结构

L^AT_EX 源代码以一个 \documentclass 命令作为开头，它规定了文档使用的**文档类**：

```
\documentclass{...}
```

紧接着我们可以用 \usepackage 命令调用**宏包**：

```
\usepackage{...}
```

再接着，我们需要用以下一对命令来标记正文内容的开始位置和结束位置，而将正文内容写入其中：

```
\begin{document}
\end{document}
```

在 \documentclass 和 \begin{document} 之间的位置称为**导言区**，除了使用 \usepackage 调用宏包之外，一些对文档的全局设置命令也在这里使用。当然也可以什么都不写，一个宏包都不调用。一切视自己需求。

1.3 用命令行操作 L^AT_EX

相信你看到这里已经急不可耐地想要写一个 L^AT_EX 源代码试一试了。我们这就给一个最小的例子，见源代码 1.1。

有相当多的编辑器会提供一个按钮完成编译，不过笔者仍然觉得有必要了解一下其背后的工作原理。L^AT_EX 调用的程序都是基于命令行的，所以建议打开 Windows 命令提示符或者 Linux / OS X 的终端，按照本手册的范例尝试一下调用命令行程序编译。

```
\documentclass{article}
\begin{document}
‘Hello world!’ from \LaTeX.
\end{document}
```

源代码 1.1: L^AT_EX 的一个最简单的源代码示例。

1.3.1 引擎、格式和命令

在开始示例编译过程之前，有必要澄清几个概念：

引擎 全称为排版引擎，是读入源代码并编译生成文档的程序，如 pdfT_EX、X_YT_EX 等。有时也直接称为编译器。

格式 是定义了一组命令的代码集。L^AT_EX 就是最广泛应用的一个格式，高德纳本人还编写了一个简单的 plain T_EX 格式，没有定义诸如 \documentclass 和 \section 等等命令。

命令 是引擎和格式二者的结合体。如下文要用到的 pdflatex 命令是结合 pdfT_EX 引擎和 L^AT_EX 格式的一个命令，用于编译类似源代码 1.1 的代码并输出 PDF。

latex 命令和 L^AT_EX 格式往往容易混淆，在同他人讨论关于 L^AT_EX 的时候需要明确。本手册为避免混淆，文中的 L^AT_EX 一律指的是**格式**，**命令**则用等宽字体 latex 表示。

用一个简单的表格总结一下：

	plain T _E X 格式	L ^A T _E X 格式
T _E X 引擎	tex	N/A
pdfT _E X 引擎	etex	latex ²
	pdftex	pdflatex
X _Y T _E X 引擎	xetex	xelatex

1.3.2 latex 命令

假使你的计算机上已经安装了 L^AT_EX 依赖的程序和工具（安装方法在附录 A 有简单介绍）。我们将源代码 1.1 拷贝到一个文本文件中，保存为 helloworld.tex。然后在命令行输入：

```
latex helloworld.tex
```

（也可以输入不带扩展名的 latex helloworld）。

此时命令行界面会闪过许多信息。如果一切正常，再行检查目录，会发现生成了 helloworld.dvi 以及其它文件。这个扩展名为 DVI 的文件就是编译输出的文档。

Linux 下可以使用命令程序 xdvi 打开这个文件：

```
xdvi helloworld.dvi
```

Windows 下大多预装了 yap 软件 (MikT_EX) 或 dviout 软件 (T_EX Live)，可以双击 helloworld.dvi 打开它。

要进一步生成现今流行的 PDF 文档格式，我们还需要用额外的命令将 DVI 转为 PDF：

```
dvipdfmx helloworld.dvi
```

然后就可以用查看 PDF 的软件 (Adobe Reader / Foxit Reader 等) 打开生成的 helloworld.pdf 查看了。

²现今的发行版中 latex 命令也用 pdfT_EX 引擎结合 L^AT_EX 格式，输出扩展名为 DVI 的文档。

1.3.3 pdf \LaTeX 和 x \LaTeX 命令

这两个命令相比于 \LaTeX 命令更为方便，我们可以直接生成 PDF：

```
pdf $\LaTeX$  helloworld.tex
```

或者

```
x $\LaTeX$  helloworld.tex
```

\LaTeX 命令（有时写作 \TeX ）有着各种新的特性，如能够直接支持使用系统预装的字体、原生支持 UTF-8 编码等。尤其是排版中文， \LaTeX 命令配合适当的宏包是现在最新、最方便的方式（详见 2.2 节）。

1.4 宏包和文档类

我们在 1.2.2 小节描述的 \LaTeX 源代码架构中已经见识了宏包和文档类。本节将仔细解释它们。

1.4.1 文档类

文档类规定了 \LaTeX 源代码所要生成的文档的性质——普通文章、书籍、演示文稿、个人简历等等。 \LaTeX 源代码的开头须用 `\documentclass` 指定文档类：

```
\documentclass[\langle options \rangle]{\langle class-name \rangle}
```

其中 *\langle class-name \rangle* 为文档类的名称，如 \LaTeX 提供的 `article`, `book`, `report`，在其基础上派生的一些文档类如支持中文排版的 `ctexart` / `ctexbook` / `ctexrep`，或者有其它功能的一些文档类，如 `moderncv` / `beamer` 等。 \LaTeX 提供的基础文档类见表 1.1，前三个习惯上称为“标准文档类”。

表 1.1: \LaTeX 提供的基础文档类

article 文章格式的文档类，广泛用于科技论文、报告、说明文档等。

report 长篇报告格式的文档类，具有章节结构，用于综述、长篇论文、简单的书籍等。

book 书籍文档类，包含章节结构和前言、正文、后记等结构。

proc 基于 `article` 文档类的一个简单的学术文档模板。

slides 幻灯格式的文档类，使用无衬线字体。

minimal 一个极其精简的文档类，只设定了纸张大小和字号，用作代码测试的最小工作示例（Minimal Working Example）。

可选参数 *\langle options \rangle* 为文档类指定选项，以全局地影响文档布局的参数，如字号、纸张大小、单双面等等。 \LaTeX 的三个标准文档类可指定的选项见表 1.2。

比如调用 `article` 文档类排版文章，指定纸张为 A4 大小，基本字号为 11pt，双面排版：

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

表 1.2: L^AT_EX 的三个标准文档类可指定的选项

<code>10pt, 11pt, 12pt</code>	指定文档的基本字号。缺省为 10pt。
<code>a4paper, letterpaper, ...</code>	指定纸张大小，默认为美式纸张 <code>letterpaper</code> 。可指定选项还包括 <code>a5paper, b5paper, executivepaper</code> 和 <code>legalpaper</code> 。
<code>fleqn</code>	令行间公式左对齐（缺省为居中）。
<code>leqno</code>	将公式编号放在左边（缺省为右边）。
<code>titlepage, notitlepage</code>	指定标题命令 <code>\maketitle</code> 是否单独成页。 <code>article</code> 缺省为 <code>notitlepage</code> ， <code>report</code> 和 <code>book</code> 缺省为 <code>titlepage</code> 。
<code>onecolumn, twocolumn</code>	指定单栏/双栏排版。
<code>twoside, oneside</code>	指定单面/双面排版。双面排版时，奇偶页的页眉页脚、页边距不同。 <code>article</code> 和 <code>report</code> 缺省为单面排版， <code>book</code> 缺省为双面。
<code>landscape</code>	指定横向排版。缺省为纵向。
<code>openright, openany</code>	指定新的一章 <code>\chapter</code> 是在奇数页（右侧）开头，还是直接紧跟着上一页开头。 <code>report</code> 缺省为 <code>openany</code> ， <code>book</code> 缺省为 <code>openright</code> 。【对 <code>article</code> 无效】

1.4.2 宏包

在编写 L^AT_EX 源代码时，你时常会发现 L^AT_EX 的基础功能不能满足你的需求，比如排版复杂的表格、插入图片、增加颜色甚至超链接等等。这时你需要依赖一些扩展来增强或补充 L^AT_EX 的功能。这些扩展称为**宏包**。调用宏包的方法非常类似调用文档类的方法：

```
\usepackage[options]{package-name}
```

`\usepackage` 的参数里可以使用不止一个宏包，多个宏包用逗号隔开。这种用法一般不要指定选项³：

```
% 一次性载入三个排版表格常用的宏包
\usepackage{tabularx,longtable,multirow}
```

附录 B.3 汇总了常用的一些宏包。我们在手册接下来的章节中，也会穿插介绍一些最常用的宏包的使用方法。

在使用宏包和文档类之前，一定要首先确认它们是否安装在你的计算机中，否则 `\usepackage` 等命令会报错误。详见附录 A.2。

每个宏包（包括前面所说的文档类）都定义了许多命令和环境，或者修改了 L^AT_EX 已有的命令和环境。为了明白它们的用法，需要查阅宏包和文档类的帮助手册。使用方法是在 Windows 命令提示符或者 Linux 终端下输入命令：

```
texdoc pkg-name
```

其中 `pkg-name` 是宏包或者文档类的名称。更多获得帮助的方法见附录 B.2。

³使用多个宏包时指定选项，相当于给每个宏包指定同样的选项。如果有某个宏包不能识别某个选项，则会出错。

1.5 L^AT_EX 用到的文件一览

除了我们需要编写的源代码 `.tex` 文件，我们还可能接触到形形色色的文件。本节简单介绍一下这些文件。

每个宏包和文档类都是带特定扩展名的文件，除此之外也有一些文件出现于 L^AT_EX 模板中：

`.sty` 宏包文件。宏包的名称就是去掉扩展名的文件名。

`.cls` 文档类文件。同样地，文档类名称就是文件名。

`.bib` Bib_TE_X 参考文献数据库文件。

`.bst` Bib_TE_X 用到的参考文献格式模板。详见 6.1.3 小节。

L^AT_EX 在编译过程中生成相当多的辅助文件和日志。一些功能如交叉引用、参考文献、目录、索引等，需要先编译生成辅助文件，然后再次编译时读入辅助文件得到正确的结果，所以复杂的 L^AT_EX 源代码可能要编译多次：

`.log` 排版引擎生成的日志文件，供排查错误使用。

`.aux` L^AT_EX 生成的主辅助文件，记录交叉引用、目录、参考文献的引用等。

`.toc` L^AT_EX 生成的目录记录文件。

`.lof` L^AT_EX 生成的图片目录记录文件。

`.lot` L^AT_EX 生成的表格目录记录文件。

`.bbl` Bib_TE_X 生成的参考文献记录文件。

`.blg` Bib_TE_X 生成的日志文件。

`.idx` L^AT_EX 生成的供 makeindex 处理的索引记录文件。

`.ind` makeindex 处理 `.idx` 生成的格式化索引记录文件。

`.ilg` makeindex 生成的日志文件。

`.out` hyperref 宏包生成的 PDF 书签记录文件。

1.6 文件的组织方式

当编写较大规模的 L^AT_EX 源代码，如书籍、毕业论文等，你有理由将源代码分成若干个文件而不是写到一堆，比如很自然地每章写一个文件，可参考源代码 3.1 的写法。

L^AT_EX 提供了命令 `\include` 用来在源代码里插入文件：

```
\include{<filename>}
```

`<filename>` 为文件名，如果和要编译的主文件不在一个目录中，则要加上相对或绝对路径，例如：

```
\include{chapters/a.tex} % 相对路径
```

```
\include{/home/Bob/file.tex} % Linux 绝对路径
```

```
\include{D:/file.tex} % Windows 绝对路径
```

`<filename>` 可以不带扩展名, 此时默认为 `.tex`; 其它文件必须带扩展名。

值得注意的是 `\include` 在读入 `<filename>` 之前会另起一页。有的时候我们并不需要这样, 而是用 `\input` 命令, 它纯粹是把文件里的内容插入:

```
\input{<filename>}
```

另外 L^AT_EX 提供了一个 `\includeonly` 命令来组织文件, 用于**导言区**, 指定只载入某些文件:

```
\includeonly{<filename1>,<filename2>,...}
```

导言区使用了 `\includeonly` 后, 正文中不在其列表范围的 `\include` 命令不会起效。

最后介绍一个实用的工具宏包 `syntonly`。加载这个宏包后, 在导言区使用 `\syntonly` 命令, 可令 L^AT_EX 编译后不生成 DVI 或者 PDF 文档, 只排查错误, 编译速度会快不少:

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

如果想生成文档, 则将 `\syntonly` 命令那一行用 `%` 注释掉即可。

第二章 用 L^AT_EX 排版文字

文字是排版的基础。在介绍 L^AT_EX 支持的各种文档元素之前，首先应当了解一下如何在 L^AT_EX 源代码中输入并排版文字，尤其是（西文中的）标点符号、连字符、重音等一些细节。

本文的开头首先介绍了中文的支持方式。随着 L^AT_EX 和底层 T_EX 引擎的发展，旧方式（CCT、CJK 宏包等）日渐退出舞台，xelatex 编译命令配合 xeCJK 宏包的方式成为主流。

2.1 语言文字和编码

L^AT_EX 源代码为文本文件，而文本文件的一个至关重要的性质是它的编码。在此用尽量短的篇幅介绍一下。

2.1.1 ASCII 编码

计算机的基本存储单位是字节 (byte)，每个字节为八位 (8-bit)，范围用十六进制写作 0x00-0xFF。ASCII（美国通用信息交换码）使用 0x00-0x7F 对文字编码，也就是 7-bit，能够表示 QWERTY 键盘上能够输入的拉丁字母、数字和符号。

由于 T_EX 最初面向英语文档的排版，ASCII 编码完全够用，而早期版本的 T_EX 也只支持 7-bit 和 ASCII。要排版西欧语系中带重音的字符，就必须用到后文所叙述的重音命令，如 Möbius 必须输入 M\`obius。

2.1.2 扩展编码

在 ASCII 之后，各种语言文字都发展了自己的编码，比如西欧语言的 Latin-1，日本的 Shift-JIS，中国大陆的 GB2312-80 和 GBK 等。它们之中的绝大多数都向下兼容 ASCII，因此无论是在哪种编码下，T_EX 以及 L^AT_EX 的命令和符号都能用。

现在的 T_EX 程序支持 8-bit，能够识别源代码里编码处于 0x80-0xFF 范围的字符。西欧（拉丁字母）、俄语系（西里尔字母）等语言的编码方案刚好能够利用 0x80-0xFF 这个范围，处理起来较为方便。latex 命令及 pdflatex 命令下，这些编码的处理由 inputenc 宏包支持。比如将源代码保存为 Latin-1 编码，并在导言区使用：

```
\usepackage[latin1]{inputenc}
```

接下来，Möbius 就直接可以通过（用适当输入法）输入 Möbius 得到了。其它一些语言也可以使用 inputenc 宏包配合 babel 宏包排版。

GBK 等编码是多字节编码，ASCII 字符为一个字节，汉字等非 ASCII 字符为两个字节，这就需要借助一些宏包进行较为复杂的判断和处理。CJK 宏包就是用于处理中、日、韩等多字节编

码的语言文字的宏包。但 CJK 宏包的使用非常不方便，笔者不再推荐直接使用。后一节将简单介绍现在推荐的中文支持方案。

2.1.3 UTF-8 编码

Unicode 是一个多国字符的集合，覆盖了几近全球范围内的语言文字。UTF-8 是 Unicode 的一套编码方案，一个字符可以由一个到四个（现在用到三个）字节编码，其中单字节字符兼容 ASCII 编码。

latex 命令及 pdf_latex 命令下可以使用 inputenc 宏包支持 UTF-8:

```
\usepackage[utf8]{inputenc}
```

x_el_atex 命令原生支持 UTF-8 编码，而且也不适用 inputenc 宏包。将 .tex 源代码保存为 UTF-8 编码，并借助 fontspec 宏包（见 5.1.4 小节）调用适当的字体，就可以在源代码中输入任意语言文字。但各个语言（印地语、阿拉伯语等）的特殊排版要求需要更多的宏包支持，如 babel、polyglossia 等。

2.2 排版中文

2.2.1 x_eCJK 宏包

用 L^AT_EX 排版中文的一大门槛是中文字体。T_EX 使用的字体格式仅支持不超过 256 个字符，要想排版中文，比如旧式的 CJK 宏包，往往需要复杂的预处理，将中文字体拆分成数百个小字体，非常麻烦¹。

x_el_atex 命令支持直接使用系统安装的 TrueType (.ttf) / OpenType(.otf) 等格式的字体，加上对 UTF-8 编码的原生支持，免去了预处理字体的麻烦。在此基础上的 x_eCJK 宏包更进一步完善了排版中文的一些细节，比如中英文之间插入空隙、中文行尾的回车不引入空格、标点符号不出现在行首，等等。

x_eCJK 宏包支持用简单的命令配置中文字体。举一个在 Windows 下使用 x_eCJK 的例子（设置字体的命令详见 5.1.5 小节），源代码须保存为 UTF-8 编码，并使用 x_el_atex 命令编译：

```
\documentclass{article}
\usepackage{xeCJK}
\setCJKmainfont{SimSun}
\begin{document}
中文LATEX排版。
\end{document}
```

2.2.2 ctex 宏包和文档类

ctex 宏包和文档类是对 CJK 和 x_eCJK 等宏包的进一步封装。ctex 文档类包括 ctexart / ctexrep / ctexbook，是对 L^AT_EX 的三个标准文档类的封装，对 L^AT_EX 的排版样式做了许多调整，以切合中文排版风格。最新版本的 ctex 宏包/文档类甚至支持自动配置字体。比如上述例子可进一步简化为：

¹现在的 pdf_latex 以及 latex + dvipdfmx 命令通过字体映射的方式支持直接使用中文字体，不需要经过预处理，但仍然是不方便新手操作的。

```
\documentclass{ctexart}
\begin{document}
中文LaTeX排版。
\end{document}
```

ctex 宏包/文档类支持源代码保存为 UTF-8 和 GBK 编码²，用 latex + dvipdfmx 命令、pdf_latex 或 xelatex 命令（只支持 UTF-8 编码）都能够编译。笔者建议在使用 ctex 宏包和文档类时总是将源代码保存为 UTF-8 编码，用 xelatex 命令编译。

2.3 L^AT_EX 中的符号

2.3.1 空格和分段

L^AT_EX 源代码中，空格键和 Tab 键输入的空白字符视为“空格”。连续的若干个空白字符视为一个空格。一行开头的空格忽略不计。

行末的回车视为一个空格；但连续两个回车，也就是空行，会将文字分段。多个空行被视为一个空行。也可以在行末使用 \par 命令分段：

```
Several spaces      equal one.
  Front spaces are ignored.

An empty line starts a new
paragraph.\par
A \verb|\par| command also
starts a new line.
```

```
Several spaces equal one. Front spaces are
ignored.

An empty line starts a new paragraph.

A \par command also starts a new line.
```

2.3.2 注释

L^AT_EX 用 % 字符作为注释。在这个字符之后直到行末，所有的字符都被忽略，连同回车引入的空格。我们需要注意以下示例中“回车引入的空格被忽略”的效果：

```
This is an % short comment
% ---
% Long and organized
% comments
% ---
example: Comments do not bre%
ak a word.
```

```
This is an example: Comments do not
break a word.
```

2.3.3 特殊字符

以下字符在 L^AT_EX 里有特殊用途，如 % 表示注释，\$, ^、_ 等用于排版数学公式，& 用于排版表格，等等。直接输入这些字符得不到对应的符号，还往往会出错：

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

如果想要输入以上符号，需要使用以下带反斜线的形式输入：

²使用 GBK 编码时，要为文档类指定 GBK 选项。详见 ctex 宏包手册。

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

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

事实上这些带反斜线的形式就是 L^AT_EX 命令。 \wedge 和 \sim 两个命令是需要带参数的，如果不加一对花括号（空参数），就将后面的字符作为参数，形成重音效果（详见某节）。 \backslash 被直接定义成了手动换行的命令，输入反斜杠就只好用 `\textbackslash`。

2.4 文字强调

2.5 段落

2.6 单词之间的空格

2.7 断行和断页

2.7.1 手动断行和断页

2.7.2 连字符

2.8 预定义的字符串

第三章 文档元素

在知道了如何输入文字后，我们将在本章了解一个结构化的文档所依赖的各种元素——章节、目录、列表、图表、交叉引用、脚注等等。

3.1 章节和目录

3.1.1 章节标题

一篇结构化的、条理清晰文档一定是层次分明的，通过不同的命令分割为章、节、小节。L^AT_EX 的三个标准文档类 `article`、`report` 和 `book`¹ 提供了一系列命令，用以划分章节、生成章节标题并自动编号：

```
\section{<title>}   \subsection{<title>}   \subsubsection{<title>}
\paragraph{<title>} \subparagraph{<title>}
```

`\part` 命令用以将整个文档分割为大的分块，但不影响 `\section` 等的编号：

```
\part{<title>}
```

`book` 和 `report` 提供了章一级的结构：

```
\chapter{<title>}
```

上述命令除了生成带编号的标题之外，还向目录中添加条目，并影响页眉页脚的内容（详见 5.4 节）。每个命令有两种变体：

- 带可选参数的变体：`\section[<short title>]{<title>}`
标题使用 `<title>` 参数，在目录和页眉页脚中使用 `<short title>` 参数；
- 带星号的变体：`\section*{<title>}`
标题不带编号，也不生成目录项和页眉页脚。

3.1.2 目录

在 L^AT_EX 中生成目录非常容易，只需在合适的地方使用命令：

```
\tableofcontents
```

正确生成目录项，一般需要多次编译源代码。

有时我们使用了 `\chapter*` 或 `\section*` 这样不生成目录项的命令，而又想手动生成该章节的目录，可以在标题命令后面使用：

```
\addcontentsline{toc}{<level>}{<title>}
```

其中 `<level>` 为章节层次 `chapter` 或 `section` 等，`<title>` 为需要生成目录项的章节标题。

¹千万注意是**标准文档类**，其它文档类可能没有定义或只定义了一部分，如 `beamer` 或 `moderncv` 等。

3.1.3 文档结构的划分

所有标准文档类都提供了一个 `\appendix` 命令将正文和附录分开²，使用 `\appendix` 后，最高一级章节改为使用拉丁字母编号，从 A 开始。

`book` 文档类还提供了前言、正文、后记结构的划分命令：

`\frontmatter` 前言部分，页码为小写罗马字母格式；其后的章节不编号（但生成页眉页脚和目录项）。

`\mainmatter` 正文部分，页码改回数字格式，并从 1 开始计数；其后的章节正常编号。

`\backmatter` 后记部分，页码继续正常计数，格式不变；其后的章节不编号，类似 `\frontmatter` 命令。

以上三个命令还可和 `\appendix` 命令结合，生成有前言、正文、附录、后记四部分的文档。源代码 3.1 结合 1.6 节的 `\include` 命令和其它一些命令示意了一份完整的文档结构。

```

\documentclass[...]{book}
% 导言区，加载宏包和各项设置
\usepackage{...}
% 此处示意对参考文献和索引的设置
\usepackage{makeidx}
\makeindex
\bibliographystyle{...}

\begin{document}
\frontmatter
\maketitle % 标题页
\include{preface} % 前言章节 preface.tex
\tableofcontents

\mainmatter
\include{chapter1} % 第一章 chapter1.tex
\include{chapter2} % 第二章 chapter2.tex
...
\appendix
\include{appendixA} % 附录 A appendixA.tex
...

\backmatter
\include{prologue} % 后记 prologue.tex
\bibliography{...} % 利用 BibTeX 工具生成参考文献
\printindex % 利用 makeindex 工具生成索引
\end{document}

```

源代码 3.1: `book` 文档类的文档结构示例。

²有的地方可能写作 `\begin{appendix} ... \end{appendix}` 这样，虽然有效，但并不规范，只要使用 `\appendix` 命令就够了。

3.2 标题页

L^AT_EX 支持生成简单的标题页。首先需要给定标题和作者等信息：

```
\title{<title>} \author{<author>} \date{<date>}
```

其中前两个命令是必须的，\date 命令可选。L^AT_EX 还提供了一个 \today 命令自动生成当前日期，可用在 \date 的参数里，或者别的地方。

在 \title、\author 等命令内可以使用 \thanks 命令生成标题页的脚注，如：

```
\author{Mary\thanks{E-mail:*****@***.com}}
```

在信息给定后，就可以使用

```
\maketitle
```

生成一个简单的标题页了。article 文档类的标题默认不单独成页，而 report 和 book 默认单独成页。可在 \documentclass 命令中指定 titlepage / notitlepage 选项修改。

3.3 交叉引用

交叉引用是 L^AT_EX 强大的自动排版功能的体现之一。在能够被交叉引用的地方，如章节、公式、图表、定理等位置使用 \label 命令：

```
\label{<label-name>}
```

之后可以在别处使用 \ref 或 \pageref 命令，分别生成交叉引用的编号和页码：

```
\ref{<label-name>} \pageref{<label-name>}
```

A reference to this subsection
\label{sec:this} looks like:
“see section~\ref{sec:this} on
page~\pageref{sec:this}.”

A reference to this subsection looks like:
“see section 3.3 on page 15.”

为了生成正确的交叉引用，一般也需要多次编译源代码。

\label 命令可用于记录各种类型的交叉引用，使用位置分别为：

章节标题 在章节标题命令 \section 等之后紧接着使用。

行间公式 单行公式在公式内任意位置使用；多行公式在每一行公式的任意位置使用。

有序列表 在 enumerate 环境的每个 \item 命令之后、下一个 \item 命令之前任意位置使用。

图表标题 在图表标题命令 \caption 之后紧接着使用。

定理环境 在定理环境内部任意位置使用。

在使用不记编号的命令形式（\section*、\caption*、带可选参数的 \item 命令等）时不要使用 \label 命令，否则生成的引用编号不正确。

3.4 脚注

使用 `\footnote` 命令可以在页面底部生成一个脚注：

```
\footnote{footnote}
```

假如我们输入以下文字和命令：

“天地玄黄，宇宙洪荒。日月盈昃，辰宿列张。” `\footnote{语出《千字文》。}`

在正文中则为：“天地玄黄，宇宙洪荒。日月盈昃，辰宿列张。”³

有些情况下（比如在表格环境、盒子内）使用 `\footnote` 并不能正确生成脚注。我们可以分两步进行，先使用 `\footnotemark` 为脚注计数，再在合适的位置用 `\footnotetext` 生成脚注。

3.5 特殊环境

3.5.1 列表

L^AT_EX 提供了基本的有序和无序列表环境 `enumerate` 和 `itemize`，两者的用法很类似，都用 `\item` 标明每个列表项。`enumerate` 环境会自动对列表项编号。

```
\begin{enumerate}
\item ...
\end{enumerate}
```

其中 `\item` 可带一个可选参数，将有序列表的计数或者无序列表的符号替换成自定义的符号。列表可以嵌套使用，最多嵌套四层。

```
\begin{enumerate}
\item An item.
\begin{enumerate}
\item A nested item.
\item[*] A starred item.
\item Another item. \label{itref}
\end{enumerate}
\item Go back to upper level.
\item Reference(\ref{itref}).
\end{enumerate}
```

1. An item.
 - (a) A nested item.
 - * A starred item.
 - (b) Another item.
2. Go back to upper level.
3. Reference(1b).

```
\begin{itemize}
\item An item.
\begin{itemize}
\item A nested item.
\item[+] A ‘plus’ item.
\item Another item.
\end{itemize}
\item Go back to upper level.
\end{itemize}
```

- An item.
 - A nested item.
 - + A ‘plus’ item.
 - Another item.
- Go back to upper level.

³语出《千字文》。

关键字环境 `description` 的用法与以上两者类似，不同的是 `\item` 后的可选参数用来写关键字，以粗体显示，一般是必填的：

```
\begin{description}
\item[item title] ...
\end{description}
```

```
\begin{description}
\item[Enumerate] Numbered list.
\item[Itemize] Non-numbered list.
\end{description}
```

Enumerate Numbered list.

Itemize Non-numbered list.

默认的列表间距比较宽， \LaTeX 本身也难以修改，一般要用到 `enumitem` 宏包。本文不详细叙述，有兴趣的可查阅其帮助手册。

3.5.2 对齐环境

`center`、`flushleft` 和 `flushright` 环境分别用于生成居中、左对齐和右对齐的文本环境。

```
\begin{center} ... \end{center}
```

```
\begin{center}
Centered text using a
\verb|center| environment.
\end{center}
\begin{flushleft}
Left-aligned text using a
\verb|flushleft| environment.
\end{flushleft}
\begin{flushright}
Right-aligned text using a
\verb|flushright| environment.
\end{flushright}
```

Centered text using a `center`
environment.

Left-aligned text using a `flushleft`
environment.

Right-aligned text using a `flushright`
environment.

除此之外，还可以用以下命令直接改变文字的对齐方式：

```
\centering \raggedright \raggedleft
```

```
\centering
Centered text paragraph.

\raggedright
Left-aligned text paragraph.

\raggedleft
Right-aligned text paragraph.
```

Centered text paragraph.

Left-aligned text paragraph.

Right-aligned text paragraph.

三个命令和对应的环境经常被误用，有直接用所谓 `\flushleft` 命令或者 `raggedright` 环境的，都是不合理的用法（即使它们可能有效）。有一点可以将两者区分开来：`center` 等环境会在上下文产生一个额外间距，而 `\centering` 等命令不产生，只是改变对齐方式。比如在浮动体环境 `table` 或 `figure` 内实现居中对齐，用 `\centering` 命令即可，没必要再用 `center` 环境。

3.5.3 引用环境

LaTeX 提供了两种引用的环境：`quote` 用于引用较短的文字，首行不缩进；`quotation` 用于引用若干段文字，首行缩进。引用环境较一般文字有额外的左右缩进。

```
Francis Bacon says:
\begin{quote}
Knowledge is power.
\end{quote}
```

```
Francis Bacon says:

Knowledge is power.
```

```
《木兰诗》：
\begin{quotation}
万里赴戎机，关山度若飞。
朔气传金柝，寒光照铁衣。
将军百战死，壮士十年归。

归来见天子，天子坐明堂。
策勋十二转，赏赐百千强。……
\end{quotation}
```

```
《木兰诗》：

万里赴戎机，关山度若飞。
朔气传金柝，寒光照铁衣。将军
百战死，壮士十年归。
归来见天子，天子坐明堂。
策勋十二转，赏赐百千强。……
```

`verse` 用于排版诗歌，与 `quotation` 恰好相反，`verse` 是首行悬挂缩进的。

```
Rabindranath Tagore's short poem:
\begin{verse}
Beauty is truth' s smile
when she beholds her own face in
a perfect mirror.
\end{verse}
```

```
Rabindranath Tagore's short poem:

Beauty is truth' s smile when
she beholds her own face in
a perfect mirror.
```

3.5.4 摘要环境

摘要环境 `abstract` 只在 `article` 和 `report` 文档类可用，一般用于紧跟 `\maketitle` 命令之后介绍文档的摘要。如果文档类给定了 `titlepage` 选项，则单独成页；反之相当于一个小标题加一个 `quotation` 环境（双栏下相当于 `\section*` 定义的一节）。

3.5.5 代码环境

有时我们需要将一段代码原样转义输出，这就要用到代码环境 `verbatim`，它以等宽字体排版代码，回车和空格也分别起到换行和空位的作用；带星号的版本更进一步将空格显示成 `□`。

```
\begin{verbatim}
#include <iostream>
int main()
{
    std::cout << 'Hello, world'
                << std::endl;
    return 0;
}
\end{verbatim}
```

```
#include <iostream>
int main()
{
    std::cout << 'Hello, world'
                << std::endl;
    return 0;
}
```

```
\begin{verbatim*}
for (int i=0; i<4; ++i)
    printf('Number %d', i);
\end{verbatim*}
```

```
for_ (int_ i=0; _i<4; _++i)
    _printf('Number_ %d', i);
```

要排版简短的代码或关键字，可使用 `\verb` 命令：

```
\verb<delim><code><delim>
```

`<delim>` 标明代码的分界位置，不能是字母、空格或星号，前后必须一致，可任意选择使得不与代码本身冲突，习惯上使用 `|` 符号。

同 `verbatim` 环境，`\verb` 后也可以带一个星号，以显示空格：

```
\verb| \LaTeX| \\\
\verb+(a || b)+ \verb*+(a || b)+
```

```
\LaTeX
(a || b) (a_ || _b)
```

`\verb` 命令对符号的处理比较复杂，一般不能用在其它命令的参数里，否则多半会出错。

`verbatim` 宏包优化了 `verbatim` 环境的内部命令，并提供了 `\verbatiminput` 命令用来直接读入文件生成代码环境。`fancyvrb` 宏包提供了可定制格式的 `Verbatim` 环境；`listings` 宏包更进一步，可生成关键字高亮的代码环境，支持各种程序设计语言的语法和关键字。详情请参考各自的帮助手册。

3.6 表格

\LaTeX 里排版表格不如 Word 等所见即所得的工具简便和自由，不过对于不太复杂的表格来讲，完全能够胜任。

排版表格最基本的 `tabular` 环境用法为：

```
\begin{tabular}{<column-spec>}
<item1> & <item2> & ... \\
\hline
<item1> & <item2> & ... \\
\end{tabular}
```

其中 `<column-spec>` 是列格式标记，在接下来的内容将详细介绍；`&` 用来分隔单元格；`\\` 用来换行；`\hline` 用来在行与行之间绘制横线。

直接使用 `tabular` 环境的话，会和周围的文字混排。`tabular` 环境可带一个可选参数控制垂直对齐（默认是垂直居中）：

```
\begin{tabular}{|c|}
center-\\ aligned \\
\end{tabular},
\begin{tabular}[t]{|c|}
top-\\ aligned \\
\end{tabular},
\begin{tabular}[b]{|c|}
bottom-\\ aligned\\
\end{tabular} tabulars.
```

center- aligned	,	top- aligned	,	bottom- aligned	tabulars.
--------------------	---	-----------------	---	--------------------	-----------

但是通常情况下我们不这么用，`tabular` 环境一般会放置在 `table` 浮动体环境中，并用 `\caption` 命令加标题。

3.6.1 列格式

L^AT_EX 表格中基本的列格式如下表：

列格式	说明
<code>l/c/r</code>	单元格内容左对齐/居中/右对齐，不折行
<code>p{⟨width⟩}</code>	单元格宽度固定为 ⟨width⟩，可自动折行
<code> </code>	绘制竖线
<code>@{⟨string⟩}</code>	自定义内容 ⟨string⟩

```
\begin{tabular}{lcr|p{6em}}
\hline
left & center & right
& par box with fixed width\\
L & C & R & P \\
\hline
\end{tabular}
```

left	center	right	par box with fixed width
L	C	R	P

@ 格式可在单元格前后插入任意的文本，但同时它也消除了单元格前后额外添加的间距。@ 格式可以适当使用以充当“竖线”。特别地，@{} 可直接用来消除单元格前后的间距：

```
\begin{tabular}{@{} r@{:}lr @{}}
\hline
1 & 1 & one \\
11 & 3 & eleven \\
\hline
\end{tabular}
```

1:1	one
11:3	eleven

另外 L^AT_EX 还提供了简便的将格式参数重复的写法 `*{⟨n⟩}{⟨column-spec⟩}`，比如以下两种写法是等效的：

```
\begin{tabular}{|c|c|c|c|c|c|p{4em}|p{4em}|}
\begin{tabular}{|*{5}{c}|*{2}{p{4em}}|}
```

有时需要为整列修饰格式，比如整列改变为粗体，如果每个单元格都加上 `\bfseries` 命令会比较麻烦。array 宏包提供了辅助格式 `>` 和 `<`，用于给列格式前后加上修饰命令：

```
\begin{tabular}{>{\itshape}r<{*}l}
\hline
italic & normal \\
column & column \\
\hline
\end{tabular}
```

<i>italic</i> *	normal
<i>column</i> *	column

辅助格式甚至支持插入 `\centering` 等命令改变 p 列格式的对齐方式，一般还要加额外的命令 `\arraybackslash` 以免出错⁴：

```
\begin{tabular}
{>{\centering\arraybackslash}p{9em}}
\hline
Some center-aligned long text. \\
\hline
\end{tabular}
```

Some center-aligned long text.

⁴`\centering` 等对齐命令会破坏表格环境里 `\arraybackslash` 换行命令的定义，`\arraybackslash` 用来恢复之。

3.6.2 列宽

在控制列宽方面， \LaTeX 表格有着明显的不足：`l/c/r` 格式的列宽是由文字内容的自然宽度决定的，而 `p` 格式给定了列宽却不好控制对齐（需要 `array` 宏包辅助），更何况列与列之间通常还有间距，所以直接生成给定总宽度的表格并不容易。

\LaTeX 本身提供了 `tabular*` 环境用来排版定宽表格，但是不太方便使用，比如要用到 `@` 格式插入额外命令，令单元格之间的间距为 `\fill`，但即使这样仍然有瑕疵：

```
\begin{tabular*}{14em}%
{@{\extracolsep{\fill}}|c|c|c|c|}
\hline
A & B & C & D \\ \hline
a & b & c & d \\ \hline
\end{tabular*}
```

A	B	C	D
a	b	c	d

`tabularx` 宏包为我们提供了方便的解决方案。它引入了一个 `X` 格式，类似 `p` 格式，不过会根据表格宽度自动计算列宽，多个 `X` 格式平均分配列宽。`X` 格式也可以用 `array` 里的辅助格式修饰对齐方式：

```
\begin{tabularx}{14em}%
{!{*4}{>{\centering\arraybackslash}X|}}
\hline
A & B & C & D \\ \hline
a & b & c & d \\ \hline
\end{tabularx}
```

A	B	C	D
a	b	c	d

3.6.3 横线

我们已经在之前的例子见过许多次绘制表格线的 `\hline` 命令。另外 `\cline{<i>-<j>}` 用来绘制跨越部分单元格的横线：

```
\begin{tabular}{|c|c|c|}
\hline
4 & 9 & 2 \\ \cline{2-3}
3 & 5 & 7 \\ \cline{1-1}
8 & 1 & 6 \\ \hline
\end{tabular}
```

4	9	2
3	5	7
8	1	6

在科技论文排版中广泛应用的表格形式是三线表，形式干净简明。三线表由 `booktabs` 宏包支持，它提供了 `\toprule`、`\midrule` 和 `\bottomrule` 命令用以排版三线表的三条线，除此之外，最好不要用其它横线以及竖线：

```
\begin{tabular}{cccc}
\toprule
& 1 & 2 & 3 \\
\midrule
Alphabet & A & B & C \\
Roman & I & II & III \\
\bottomrule
\end{tabular}
```

	1	2	3
Alphabet	A	B	C
Roman	I	II	III

3.6.4 合并单元格

L^AT_EX 是一行一行排版表格的，横向合并单元格较为容易，由 `\multicolumn` 命令实现：

```
\multicolumn{⟨n⟩}{⟨column-spec⟩}{⟨item⟩}
```

其中 $\langle n \rangle$ 为要合并的列数， $\langle column-spec \rangle$ 为合并单元格后的列格式，只允许出现一个 `l/c/r` 或 `p` 格式。如果合并前的单元格前后带表格线 `|`，合并后的列格式也要带 `|` 以使得表格的竖线一致。

```
\begin{tabular}{|c|c|c|}
\hline
1 & 2 & Center \\ \hline
\multicolumn{2}{|c|}{3} & \\ \hline
\multicolumn{1}{r|}{4} & \multicolumn{2}{C} \\ \hline
\end{tabular}
```

1	2	Center
3		Right
4	C	

上面的例子还体现了，形如 `\multicolumn{1}{⟨column-spec⟩}{⟨item⟩}` 的命令可以用来修改某一个单元格的列格式。

纵向合并单元格需要用到 `multirow` 宏包提供的 `\multirow` 命令：

```
\multirow{⟨n⟩}{⟨width⟩}{⟨item⟩}
```

$\langle width \rangle$ 为合并后单元格的宽度，可以填 `*` 以使用自然宽度。

我们看一个结合 `\cline`、`\multicolumn` 和 `\multirow` 命令的例子：

```
\begin{tabular}{ccc}
\hline
\multirow{2}{*}{Item} & \multicolumn{2}{c}{Value} \\ \cline{2-3}
& First & Second \\ \hline
A & 1 & 2 \\ \hline
\end{tabular}
```

Item	Value	
	First	Second
A	1	2

3.6.5 行距控制

L^AT_EX 生成的表格看起来通常比较紧凑。修改参数 `\arraystretch` 可以得到行距更加宽松的表格（相关命令参考 8.1 节）：

```
\renewcommand\arraystretch{1.8}
\begin{tabular}{|c|}
\hline
Really loose \\ \hline
tabular rows. \\ \hline
\end{tabular}
```

Really loose
tabular rows.

另一种增加间距的办法是给换行命令 `\\` 添加可选参数，在这一行下面加额外的间距，适用于在行间不加横线的表格：

<pre>\begin{tabular}{c} \hline Head lines \\[6pt] tabular lines \\ tabular lines \\ \hline \end{tabular}</pre>	
--	--

但是这种换行方式的存在导致了一个缺陷——表格的首个单元格不能直接使用中括号 `[]`，否则 `\\` 往往会将下一行的中括号当作自己的可选参数，因而出错。如果要使用中括号，应当放在花括号 `{}` 里面。或者也可以选择将换行命令写成 `\\[0pt]`。

3.7 图片

\LaTeX 本身不支持插图功能，需要由 `graphicx` 宏包辅助支持。

使用 `latex + dvipdfmx` 编译命令时，调用 `graphicx` 宏包时要给定 `dvipdfmx` 选项⁵；而使用 `pdflatex` 或 `xelatex` 命令编译时不需要。

读者可能听说过“ \LaTeX 只能插入 `.eps` 格式的图片，需要把 `.jpg` 转成 `.eps` 格式”的观点。 \LaTeX 发展到今天，这个观点早已过时。事实上不同的编译命令支持的图片格式范围各异，见表 3.1。这个表格也能解答诸如“为什么 `.eps` 格式图片在 `pdflatex` 编译命令下出错”之类的问题。本表格也再一次说明，使用 `xelatex` 命令是笔者最推荐的方式。

表 3.1: 各种编译方式支持的主流图片格式。

格式	矢量图	位图
<code>latex + dvipdfmx</code>	<code>.eps</code>	n/a
└ (调用 <code>bmpsize</code> 宏包)	<code>.eps .pdf</code>	<code>.jpg .png .bmp</code>
<code>pdflatex</code>	<code>.pdf</code>	<code>.jpg .png</code>
└ (调用 <code>epstopdf</code> 宏包)	<code>.pdf .eps</code>	<code>.jpg .png</code>
<code>xelatex</code>	<code>.pdf .eps</code>	<code>.jpg .png .bmp</code>

注：在较新的 \TeX 发行版中，`latex + dvipdfmx` 和 `pdflatex` 命令可不依赖宏包，支持原来需要宏包扩展的图片格式（但 `pdflatex` 命令仍不支持除 `.jpg` 和 `.png` 以外的位图）。

在调用了 `graphicx` 宏包以后，就可以使用 `\includegraphics` 命令加载图片了：

```
\includegraphics[\langle options \rangle]{\langle filename \rangle}
```

其中 `\langle filename \rangle` 为图片文件名，与使用 `\include` 命令类似，文件名有时需要使用相对路径或绝对路径（见 1.6 节）。图片文件的扩展名可写可不写。

另外 `graphicx` 宏包还提供了 `\graphicspath` 命令，用于声明一个或多个图片文件存放的目录，使用这些目录里的图片时可不用写路径：

% 假设主要的图片放在 `figures` 子目录下，标志放在 `logo` 子目录下
`\graphicspath{{figures/}{logo/}}`

⁵早期常使用 `latex + dvips` 组合命令，后者将 `.dvi` 文件转为 `.ps` 文件 (PostScript)，可进一步通过 `ps2pdf` 工具生成 PDF。`dvips` 和 `dvipdfmx` 在图形、颜色、超链接等功能的实现上有差别，而 \LaTeX 无法识别用户是用 `dvips` 还是 `dvipdfmx`，所以要给定选项（缺省为 `dvips`）。6.4 节中的 `hyperref` 宏包同理。

`\includegraphics` 命令的可选参数 $\langle options \rangle$ 支持 $\langle key \rangle = \langle value \rangle$ 形式赋值, 常用的选项如下:

选项	含义
<code>width=\langle width \rangle</code>	将图片缩放到宽度为 $\langle width \rangle$
<code>height=\langle height \rangle</code>	将图片缩放到高度为 $\langle height \rangle$
<code>scale=\langle scale \rangle</code>	将图片相对于原尺寸缩放 $\langle scale \rangle$ 倍
<code>angle=\langle angle \rangle</code>	令图片逆时针旋转 $\langle angle \rangle$ 度

3.8 盒子

盒子是 L^AT_EX 排版的基础单元, 虽然解释上去有些抽象: 每一行是一个盒子, 里面的文字从左到右依次排列; 每一页也是一个盒子, 各行文字从上到下依次排布……颇有一些活字印刷术的味道。

不管如何, L^AT_EX 提供了一些实用的命令让我们生成一些有用的盒子。

3.8.1 水平盒子

生成水平盒子的命令如下:

```
\mbox{...}
\makebox[\langle width \rangle][\langle align \rangle]{...}
```

`\mbox` 生成一个基本的水平盒子, 内容只有一行 (除非嵌套下文的垂直盒子), 不允许分段。外表看上去, `\mbox` 的内容与正常的文本无二, 不过断行时文字不会从盒子里断开。

`\makebox` 更进一步, 可以加上可选参数用于控制盒子的宽度 $\langle width \rangle$, 以及内容的对齐方式 $\langle align \rangle$, 可选居中 `c` (默认值)、左对齐 `l`、右对齐 `r` 和分散对齐 `s`⁶。

```
| \mbox{Test some words.} | \
| \makebox[10em]{Test some words.} | \
| \makebox[10em][l]{Test some words.} | \
| \makebox[10em][r]{Test some words.} | \
| \makebox[10em][s]{Test some words.} |
```

```
| Test some words. |
|   Test some words.   |
| Test some words.   |
|       Test some words. |
| Test   some   words. |
```

3.8.2 带框的水平盒子

`\fbox` 和 `\framebox` 让我们可以为水平盒子添加边框。使用的语法与 `\mbox` 和 `\makebox` 一模一样:

```
\fbox{...}
\framebox[\langle width \rangle][\langle align \rangle]{...}
```

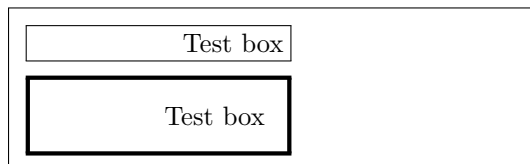
```
\fbox{Test some words.} \
\framebox[10em][r]{Test some words.}
```

```
Test some words.
Test some words.
```

可以通过 `\setlength` 命令 (见 5.2.1 小节) 调节边框的宽度 `\fboxrule` 和内边距 `\fboxsep`:

⁶分散对齐方式强行拉开单词的间距, 往往会报 `Underfull \hbox` 的警告。


```
\framebox[10em][r]{Test box}\[1ex]
\setlength{\fboxrule}{1.6pt}
\setlength{\fboxsep}{1em}
\framebox[10em][r]{Test box}
```



3.8.3 垂直盒子

如果需要排版一个文字可以换行的盒子，L^AT_EX 提供了两种方式：

```
\parbox[⟨align⟩][⟨height⟩][⟨inner-align⟩]{⟨width⟩}{...}

\begin{minipage}[⟨align⟩][⟨height⟩][⟨inner-align⟩]{⟨width⟩}
...
\end{minipage}
```

其中 $\langle align \rangle$ 为盒子和周围文字的对齐情况（类似 `tabular` 环境）； $\langle height \rangle$ 和 $\langle inner-align \rangle$ 设置盒子的高度和内容的对齐方式，类似水平盒子 `\makebox` 的设置，不过 $\langle inner-align \rangle$ 接受的选项是顶部 `t`、底部 `b`、居中 `c` 和分散对齐 `s`。

```
三字经：\parbox[t]{3em}%
{人之初 性本善 性相近 习相远}
\quad
千字文：
\begin{minipage}[b][8ex][t]{4em}
天地玄黄 宇宙洪荒
\end{minipage}
```

天地玄黄
宇宙洪荒

三字经：人之初 千字文：
性本善
性相近
习相远

如果在 `minipage` 里使用 `\footnote` 命令，生成的脚注会出现在盒子底部，编号是独立的，并且使用小写字母编号。这也是 `minipage` 环境之被称为“迷你页”（Mini-page）的原因。而在 `\parbox` 里无法正常使用 `\footnote` 命令，只能在盒子里使用 `\footnotemark`，在盒子外使用 `\footnotetext`。

```
\fbox{\begin{minipage}{10em}%
这是一个垂直盒子的测试。
\footnote{脚注来自 minipage。}
\end{minipage}}
```

这是一个垂直盒子的测试。^a

^a脚注来自 minipage。

3.8.4 标尺盒子

`\rule` 命令用来画一个实心的矩形盒子，也可适当调整以用来画线（标尺）：



```
\rule[⟨raise⟩]{⟨width⟩}{⟨height⟩}
```

Black `\rule{12pt}{4pt}` box.

Upper `\rule[4pt]{6pt}{8pt}` and
lower `\rule[-4pt]{6pt}{8pt}` box.

A `\rule[-.4pt]{3em}{.4pt}` line.

Black  box.

Upper  and lower  box.

A  line.

3.9 浮动体

内容丰富的文章或者书籍往往包含丰富的图片和表格等内容。这些内容的尺寸往往太大，导致分页困难。L^AT_EX 为此引入了浮动体的机制，令大块的内容可以脱离上下文，放置在合适的位置。

L^AT_EX 预定义了两类浮动体环境 `figure` 和 `table`。习惯上 `figure` 里放图片，`table` 里放表格，但并没有严格限制，可以在任何一个浮动体里放置表格、文字、列表等等任意内容。

以 `table` 环境的用法举例，`figure` 同理：

```
\begin{table}[\langle placement \rangle]
...
\end{table}
```

$\langle placement \rangle$ 参数提供了一些符号用来表示浮动体允许排版的位置，如 `[hbp]` 允许浮动体排版在当前位置、底部或者单独成页。`table` 和 `figure` 浮动体的默认设置为 `[tbp]`。

h	当前位置（代码所处的上下文）
t	顶部
b	底部
p	单独成页
!	在决定位置时忽视限制

注 1：排版位置的选取与参数里符号的顺序无关，L^AT_EX 总是以 `h-t-b-p` 的优先级顺序选取位置。也就是说 `[!htp]` 和 `[ph!t]` 没有区别。

注 2：限制包括浮动体个数（除单独成页外，默认每页不超过 3 个浮动体，其中顶部不超过 2 个，底部不超过 1 个）以及浮动体空间占页面的百分比（默认顶部不超过 70%，底部不超过 30%）。

双栏排版环境下，L^AT_EX 提供了 `table*` 和 `figure*` 环境用来排版跨栏的浮动体。它们的用法与 `table` 和 `figure` 一样，不同之处为双栏的 $\langle placement \rangle$ 参数只能用 `tp` 两个位置。

浮动体的位置选取受到先后顺序的限制。如果某个浮动体由于参数限制、空间限制等原因在当前页无法放置，就要推迟到后一页，并使得之后的同类浮动体一并推迟。`\clearpage` 命令会在另起一页之前，先将所有推迟放置的浮动体排版成页。

`float` 宏包为浮动体提供了 `H` 选项，不与 `htbp` 及 `!` 混用。使用 `H` 选项时，会取消浮动机制，将浮动体视为一般的盒子插入当前位置。这在一些特殊情况下（如使用 `multicol` 宏包生成分栏）的时候很有用，但尺寸过大的浮动体可能影响到分页。

3.9.1 浮动体的标题

图表等浮动体提供了 `\caption` 命令加标题，并且自动给浮动体编号：

```
\caption{...}
```

`\caption` 的用法非常类似于 `\section` 等命令，可以用带星号的命令 `\caption*` 生成不计数的标题，也可以使用带可选参数的形式 `\caption[...]{...}`，使得在目录里使用短标题。

`\caption` 之后还可以紧跟 `\label` 命令标记交叉引用。

`table` 和 `figure` 两种浮动体分别有各自的生成目录的命令：

```
\listoftables
\listoffigures
```

3.9.2 并排和子图表

我们时常有在一个浮动体里面放置多张图的用法。最简单的用法就是直接并排放置，也可以通过分段或者换行命令 `\\` 排版多行多列的图片。以下为示意代码，效果大致如图 3.1 所示。

```
\begin{figure}[htbp]
  \centering
  \includegraphics[width=...]{...}
  \quad
  \includegraphics[width=...]{...} \\[.pt]
  \includegraphics[width=...]{...}
  \caption{...}
\end{figure}
```

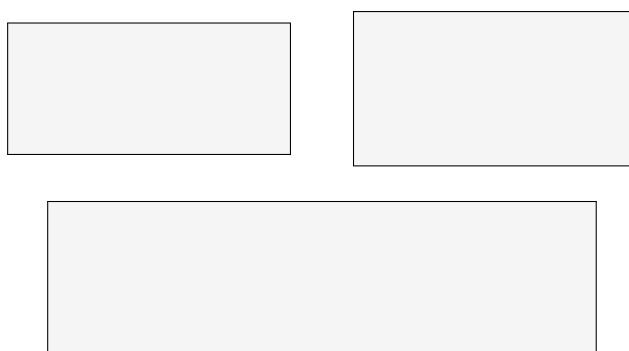


图 3.1: 并排放置图片的示意。

由于标题是横跨一行的，用 `\caption` 命令为每个图片单独生成标题就需要借助前文提到的 `\parbox` 或者 `minipage` 环境，将标题限制在盒子内。效果见图 3.2 和图 3.3。

```
\begin{figure}[htbp]
  \centering
  \begin{minipage}
    \centering
    \includegraphics[width=...]{...}
    \caption{...}
  \end{minipage}
  \quad
  \begin{minipage}
    \centering
    \includegraphics[width=...]{...}
    \caption{...}
  \end{minipage}
\end{figure}
```

当我们需要更进一步，给每个图片定义小标题时，就要用到 `subfig` 宏包的功能了。这里仅举一例，效果见图 3.4。更详细的用法请参考 `subfig` 宏包的帮助文档。

```
\begin{figure}[htbp]
```



图 3.2: 并排图 1。



图 3.3: 并排图 2。

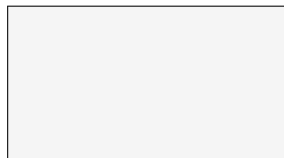
```

\centering
\subfloat[...]{\label{sub-fig-1}}% 为子图加交叉引用
\begin{minipage}
  \centering
  \includegraphics[width=...]{...}
\end{minipage}
}
\quad
\subfloat[...]{%
\begin{minipage}
  \centering
  \includegraphics[width=...]{...}
\end{minipage}
}
\caption{...}
\end{figure}

```



(a) 并排子图 1



(b) 并排子图 2

图 3.4: 使用 subfig 宏包的 \subfloat 命令排版子图。

第四章 排版数学公式

准备好了！本章将见识到 L^AT_EX 闻名的强项——排版数学公式。当然你得注意了，本章的内容只是一点皮毛，虽然对大多数人来说已经够用了，但是如果不能解决你的问题的话也不要大惊小怪，求助于搜索引擎或者有经验的人不失为一个好办法。

4.1 $\mathcal{A}\mathcal{M}\mathcal{S}$ 宏集

在介绍数学公式排版之前，简单介绍一下 $\mathcal{A}\mathcal{M}\mathcal{S}$ 宏集。 $\mathcal{A}\mathcal{M}\mathcal{S}$ 宏集合是美国数学学会 (American Mathematical Society) 提供的对 L^AT_EX 原生的数学公式排版的扩展，其核心是 `amsmath` 宏包，对多行公式的排版提供了有力的支持。此外，`amsfonts` 宏包以及基于它的 `amssymb` 宏包提供了丰富的数学符号；`amsthm` 宏包扩展了 L^AT_EX 定理证明格式。

本章介绍的许多命令和环境依赖于 `amsmath` 宏包。这些命令和环境将以蓝色示意。

4.2 公式排版基础

数学公式有两种排版方式：其一是与文字混排，称为**行内公式**；其二是单独列为一行排版，称为**行间公式**。

行内公式由一对 `$` 符号包裹：

```
Add $a$ squared and $b$ squared  
to get $c$ squared. Or, using  
a more mathematical approach:  
 $a^2 + b^2 = c^2$ 
```

```
Add  $a$  squared and  $b$  squared to get  $c$   
squared. Or, using a more mathematical  
approach:  $a^2 + b^2 = c^2$ 
```

单独成行的**行间公式**在 L^AT_EX 里由 `equation` 环境包裹。`equation` 环境为公式自动生成一个编号，这个编号可以用 `\label` 和 `\ref` 生成交叉引用，`amsmath` 的 `\eqref` 命令甚至为引用自动加上圆括号；还可以用 `\tag` 命令手动修改公式的编号，或者用 `\notag` 命令取消为公式编号（与之基本等效的命令是 `\nonumber`）。

```
Add $a$ squared and $b$ squared  
to get $c$ squared  
\begin{equation}  
a^2 + b^2 = c^2  
\end{equation}  
Einstein says  
\begin{equation}  
E = mc^2 \label{clever}  
\end{equation}  
This is a reference to  
\eqref{clever}.
```

```
Add  $a$  squared and  $b$  squared to get  $c$   
squared  

$$a^2 + b^2 = c^2 \tag{4.1}$$
  
Einstein says  

$$E = mc^2 \tag{4.2}$$
  
This is a reference to \(4.2\).
```

```

It's wrong to say
\begin{equation}
1 + 1 = 3 \tag{dumb}
\end{equation}
or
\begin{equation}
1 + 1 = 4 \notag
\end{equation}

```

It's wrong to say

$$1 + 1 = 3 \quad (\text{dumb})$$

or

$$1 + 1 = 4$$

当然你不会愿意为每个公式都手动取消编号。L^AT_EX 提供了一对命令 `\[` 和 `\]` 用于生成不带编号的行间公式¹，与之等效的是 `displaymath` 环境。有的人更喜欢 `equation*` 环境，体现了带星号和不带星号的环境之间的区别。

```

Again\ldots
\begin{equation*}
a^2 + b^2 = c^2
\end{equation*}
or you can type less for the
same effect:
\[ a^2 + b^2 = c^2 \]
or if you like the long one:
\begin{displaymath}
a^2 + b^2 = c^2
\end{displaymath}

```

Again...

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

or you can type less for the same effect:

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

or if you like the long one:

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

我们通过一个例子展示行内公式和行间公式的对比。为了与文字相适应，行内公式在排版大的公式元素（分式、巨算符等）时显得很“局促”：

```

In text:
$\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2}$
= $\frac{\pi^2}{6}$.

In display:
\[
\lim_{n \to \infty} \sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}
\]

```

In text: $\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$.

In display:

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n \frac{1}{k^2} = \frac{\pi^2}{6}$$

行间公式的对齐、编号位置等性质由文档类选项控制，文档类的 `fleqn` 选项令行间公式左对齐；`leqno` 选项令编号放在公式左边。

4.2.1 数学模式和文本

当你使用 `$` 开启行内公式输入，或是使用 `\[` 命令、`equation` 环境时，你就进入了所谓的**数学模式**。数学模式相比于文本模式有以下特点：

¹L^AT_EX 原生排版行间公式的方法是用一对 `$$` 符号包裹，不过无法通过指定 `fleqn` 选项控制左对齐，与上下文之间的间距也不好调整，故不太推荐使用。

1. 数学模式中输入的空格全部被忽略。数学符号的间隙默认完全由符号的性质（关系符号、运算符等）决定。需要人为引入空隙时，使用 `\quad` 和 `\qquad` 等命令。详见 4.6 节。
2. 不允许有空行（分段）。每个公式（每组多行公式）自成一个段落。
3. 所有的字母被当作数学公式中的变量处理，字母间距与文本模式不一致，也无法生成单词之间的空格。如果想在数学公式中输入正体的文本，简单情况下可用 4.7.1 小节中提供的 `\mathrm` 命令。`amsmath` 提供了更加方便的 `\text` 命令²。

```
$x^{2} \geq 0 \quad \quad \quad
\text{for } \textbf{all } x \in \mathbb{R}$
x \in \mathbb{R}$
```

$$x^2 \geq 0 \quad \text{for all } x \in \mathbb{R}$$

4.3 数学符号

本节我们将接触到形形色色的数学符号，它们是 L^AT_EX 卓越的数学公式排版能力的基础。L^AT_EX 默认提供了常用的数学符号，`amssymb` 宏包提供了一些次常用的符号。大多数常用的数学符号都能在本章末尾的 4.9 节列出的表格里查到。

4.3.1 一般符号

希腊字母符号的名称就是其英文名称，如 α (`\alpha`)、 β (`\beta`) 等等。大写的希腊字母为首字母大写的命令，如 Γ (`\Gamma`)、 Δ (`\Delta`) 等等。无穷大符号为 ∞ (`\infty`)。

省略号有 \dots (`\dots`) 和 \cdots (`\cdots`) 两种形式。它们有各自合适的用途：

```
$a_1, a_2, \dots, a_n$ \\\
$a_1 + a_2 + \cdots + a_n$
```

$$a_1, a_2, \dots, a_n$$

$$a_1 + a_2 + \cdots + a_n$$

除此之外在矩阵中会用到竖排的 \vdots (`\vdots`) 和斜排的 \ddots (`\ddots`)。

4.3.2 指数、上下标和导数

在 L^AT_EX 中用 `^` 和 `_` 标明上下标。注意上下标的内容（子公式）一般需要用花括号包裹，否则上下标只对后面的一个符号起作用。

```
$p^{3_{ij}} \quad \quad \quad
m_{\mathrm{Knuth}} \quad \quad \quad
\sum_{k=1}^3 k \quad \quad \quad
a^{x+y} \neq a^{x+y} \quad \quad \quad
e^{x^2} \neq {e^x}^2$
```

$$p_{ij}^3 \quad m_{\mathrm{Knuth}} \quad \sum_{k=1}^3 k$$

$$a^x + y \neq a^{x+y} \quad e^{x^2} \neq e^{x^2}$$

导数符号 $'$ (`'`) 是一类特殊的上标，可以适当连用表示多阶导数，也可以在其后连用上标：

```
$f(x) = x^2 \quad \quad \quad f'(x)
= 2x \quad \quad \quad f''(x) = 4$
```

$$f(x) = x^2 \quad f'(x) = 2x \quad f''(x) = 4$$

²`\text` 命令仅适合在公式中穿插少量文字。如果你的情况正好相反，需要在许多文字中穿插使用公式，则应该像正常的行内公式那样，而不是滥用 `\text` 命令。

4.3.3 分式和根式

分式使用 `\frac{分子}{分母}` 来书写。分式的大小在行间公式中是正常大小，而在行内被极度压缩。`amsmath` 提供了方便的命令 `\dfrac` 和 `\tfrac`，令用户能够在行内使用正常大小的行间公式，或是反过来。

In display style:
`\[`
`3/8 \quad \frac{3}{8}`
`\quad \tfrac{3}{8}`
`\]`

In display style:

$$3/8 \quad \frac{3}{8} \quad \tfrac{3}{8}$$

In text style:
`$\frac{1}{2}$-hours \quad`
`$\dfrac{1}{2}$-hours`

In text style: $1\frac{1}{2}$ hours $1\frac{1}{2}$ hours

一般的根式使用 `\sqrt{...}`；表示 n 次方根时写成 `\sqrt[n]{...}`。

`\sqrt{x} \Leftrightarrow x^{1/2}`
`\quad \sqrt[3]{2}`
`\quad \sqrt{x^2 + \sqrt{y}}`

$$\sqrt{x} \Leftrightarrow x^{1/2} \quad \sqrt[3]{2} \quad \sqrt{x^2 + \sqrt{y}}$$

特殊的分式形式，如二项式结构，由 `amsmath` 宏包的 `\binom` 命令生成：

Pascal' s rule is
`\[`
`\binom{n}{k} = \binom{n-1}{k}`
`+ \binom{n-1}{k-1}`
`\]`

Pascal' s rule is

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

4.3.4 关系符

\LaTeX 常见的关系符号除了可以直接输入的 $=$, $>$, $<$, 其它符号用命令输入，常用的有不等号 \neq (`\neq`)、大于等于号 \geq (`\geq`)、小于等于号 \leq (`\leq`)、约等号 \approx (`\approx`)、等价 \equiv (`\equiv`)、正比 \propto (`\propto`)、相似 \sim (`\sim`)³ 等等。更多符号命令可参考表 4.6 以及表 4.18。

\LaTeX 还提供了自定义二元关系符的命令 `\stackrel`，用于将一个符号叠加在原有的二元关系符之上：

`\[`
`f_n(x) \stackrel{*}{\approx} 1`
`\]`

$$f_n(x) \overset{*}{\approx} 1$$

4.3.5 算符

\LaTeX 中的算符大多数是二元算符，除了直接用键盘可以输入的 $+$ 、 $-$ 、 $*$ 、 $/$ ，其它符号用命令输入，常用的有乘号 \times (`\times`)、除号 \div (`\div`)、点乘 \cdot (`\cdot`)、加减号 \pm (`\pm`) / 干 (`\mp`) 等等。更多符号命令可参考表 4.7 以及表 4.17。

∇ (`\nabla`) 和 ∂ (`\partial`) 也是常用的算符，虽然它们不属于二元算符。

\LaTeX 将数学函数的名称作为一个算符排版，字体为直立字体。其中有一部分符号在上下位置可以书写一些内容作为条件，类似于后文所叙述的巨算符。

³`\sim` 经常用来替代 `\sim` 作为波浪号。

表 4.1: L^AT_EX 作为算符的函数名称一览。

不带上下限的算符				
<code>\sin</code>	<code>\arcsin</code>	<code>\sinh</code>	<code>\exp</code>	<code>\dim</code>
<code>\cos</code>	<code>\arccos</code>	<code>\cosh</code>	<code>\log</code>	<code>\ker</code>
<code>\tan</code>	<code>\arctan</code>	<code>\tanh</code>	<code>\lg</code>	<code>\hom</code>
<code>\cot</code>	<code>\arg</code>	<code>\coth</code>	<code>\ln</code>	<code>\deg</code>
<code>\sec</code>	<code>\csc</code>			
带上下限的算符				
<code>\lim</code>	<code>\limsup</code>	<code>\liminf</code>	<code>\sup</code>	<code>\inf</code>
<code>\min</code>	<code>\max</code>	<code>\det</code>	<code>\Pr</code>	<code>\gcd</code>

```
\[
  \lim_{x \rightarrow 0}
  \frac{\sin x}{x}=1
\]
```

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1$$

对于求模表达式, L^AT_EX 提供了 `\pmod` 和 `\bmod` 命令, 前者相当于一个二元运算符, 后者作为同余表达式的后缀:

```
$a\bmod b \\\
x\equiv a \pmod{b}$
```

$$a \bmod b \\ x \equiv a \pmod{b}$$

如果表 4.1 中的算符不够用的话, `amsmath` 允许用户用 `\DeclareMathOperator` 定义自己的算符, 其中带星号的命令定义带上下限的算符:

```
\DeclareMathOperator{\argh}{argh}
\DeclareMathOperator*{\nut}{Nut}
```

```
\[\argh 3 = \nut_{x=1} 4x\]
```

$$\argh 3 = \Nut_{x=1} 4x$$

4.3.6 巨算符

积分号 \int (`\int`)、求和号 \sum (`\sum`) 等符号称为**巨算符**。巨算符在行内公式和行间公式的大小和形状有区别。

```
In text:
$\sum_{i=1}^n \quad
\int_0^{\frac{\pi}{2}} \quad
\oint_0^{\frac{\pi}{2}} \quad
\prod_{\epsilon} \$ \\\
In display:
\[\sum_{i=1}^n \quad
\int_0^{\frac{\pi}{2}} \quad
\oint_0^{\frac{\pi}{2}} \quad
\prod_{\epsilon} \]
```

In text: $\sum_{i=1}^n \quad \int_0^{\frac{\pi}{2}} \quad \oint_0^{\frac{\pi}{2}} \quad \prod_{\epsilon}$
In display:

$$\sum_{i=1}^n \quad \int_0^{\frac{\pi}{2}} \quad \oint_0^{\frac{\pi}{2}} \quad \prod_{\epsilon}$$

巨算符的上下标用作其上下限。行间公式中，积分号默认将上下限放在右上角和右下角，求和号默认在上下方；行内公式一律默认在右上角和右下角。可以在巨算符后使用 `\limits` 手动令上下限显示在上下方，`\nolimits` 则相反。

```
In text:
 $\sum\limits_{i=1}^n \quad \int\limits_0^{\frac{\pi}{2}} \quad \prod\limits_{\epsilon} \$ \quad$ 
In display:

$$\sum\limits_{i=1}^n \quad \int\limits_0^{\frac{\pi}{2}} \quad \prod\limits_{\epsilon}$$

```

In text: $\sum_{i=1}^n \int_0^{\frac{\pi}{2}} \prod_{\epsilon}$
 In display:
$$\sum_{i=1}^n \int_0^{\frac{\pi}{2}} \prod_{\epsilon}$$

`amsmath` 宏包还提供了 `\substack`，能够在下限位置书写多行表达式；`subarray` 环境更进一步，令多行表达式可选择居中 (c) 或左对齐 (l)：

```
\[
\sum_{\substack{0 \leq i \leq n \\ j \in \mathbb{R}}}
P(i,j) = Q(n)
\]
\[
\sum_{\begin{subarray}{l} 0 \leq i \leq n \\ j \in \mathbb{R} \end{subarray}}
P(i,j) = Q(n)
\]
```

$$\sum_{\substack{0 \leq i \leq n \\ j \in \mathbb{R}}} P(i,j) = Q(n)$$

$$\sum_{\begin{subarray}{l} 0 \leq i \leq n \\ j \in \mathbb{R} \end{subarray}} P(i,j) = Q(n)$$

4.3.7 数学重音和上下括号

数学符号可以像文字一样加重音，比如对时间求导的符号 \dot{r} (`\dot{r}`)、 \ddot{r} (`\ddot{r}`)、表示向量的箭头 \vec{r} (`\vec{r}`)、表示欧式空间单位向量的 \hat{e} (`\hat{\mathbf{e}}`) 等，详见表 4.9。使用时要注意重音符号的作用区域，一般应当对某个符号而不是“符号加下标”使用重音：

```
 $\bar{x}_0 \quad \bar{x}_0\$ \quad [5pt]$ 
 $\vec{x}_0 \quad \vec{x}_0\$ \quad [5pt]$ 
 $\hat{\mathbf{e}}_x \quad \hat{\mathbf{e}}_x$ 
```

$\bar{x}_0 \quad \bar{x}_0$
 $\vec{x}_0 \quad \vec{x}_0$
 $\hat{e}_x \quad \hat{e}_x$

L^AT_EX 也能为多个字符加重音，包括直接画线的 `\overline` 和 `\underline` 命令（可叠加使用）、宽重音符号 `\widehat`、表示向量的箭头 `\overrightarrow` 等。后两者详见表 4.9 和 4.11 等。

```
 $\overline{\overline{0.3}} =$ 
 $\underline{\underline{\underline{0.3}}}$ 
 $\widehat{XY}$ 
 $\overrightarrow{AB}$ 
```

$0.\overline{3} = \underline{\underline{1/3}}$
 \widehat{XY}
 \overrightarrow{AB}

`\overbrace` 和 `\underbrace` 命令用来生成上/下括号，各自可带一个上/下标公式。

```
$\underbrace{\overbrace{a+b+c}^6
\cdot \overbrace{d+e+f}^7}
_{\text{meaning of life}} = 42$
```

$$\overbrace{a+b+c}^6 \cdot \overbrace{d+e+f}^7 = 42$$

meaning of life

4.3.8 箭头

除了作为上下标之外，箭头还用于表示过程。amsmath 的 `\xleftarrow` 和 `\xrightarrow` 命令可以为箭头增加上下标：

```
\[ a\xleftarrow{x+y+z} b \]
\[ c\xrightarrow[x<y]{a*b*c} d \]
```

$$a \xleftarrow{x+y+z} b$$

$$c \xrightarrow[x<y]{a*b*c} d$$

4.3.9 括号和定界符

L^AT_EX 提供了多种括号和定界符表示公式块的边界。除小括号 ()、中括号 [] 之外，其余都是 L^AT_EX 命令，包括大括号 \{ \}。表 4.12 和 4.13 给出了更多的括号/定界符命令。

```
$\{a,b,c\} \neq \{a,b,c\}$
```

$$a, b, c \neq \{a, b, c\}$$

使用 `\left` 和 `\right` 命令可令括号（定界符）的大小可变，在行间公式中常用。L^AT_EX 会自动根据括号内的公式大小决定定界符大小。`\left` 和 `\right` 必须成对使用。需要使用单个定界符时，另一个定界符写成 `\left.` 或 `\right.`。

```
\[ 1 + \left(\frac{1}{1-x^2}\right)^3 \quad
\left.\frac{\partial f}{\partial t}\right|_{t=0} \]
```

$$1 + \left(\frac{1}{1-x^2}\right)^3 \quad \left.\frac{\partial f}{\partial t}\right|_{t=0}$$

有时我们不满意于 L^AT_EX 为我们自动调节的定界符大小。这时我们还可以用 `\big`、`\bigg` 等命令生成固定大小的定界符。更常用的形式是类似 `\left` 的 `\bigl`、`\bigr` 等，以及类似 `\right` 的 `\bigr`、`\biggr` 等（`\bigl` 和 `\biggr` 不必成对出现）。

```
$\Big((x+1)(x-1)\Big)^2$\\
$\bigr( \Bigl( \biggl( \Biggl(
\quad
\biggr) \Bigr) \biggr) \Biggr)
\quad
\bigl| \Bigl| \biggl| \Biggl|
\quad
\big\Downarrow \Big\Downarrow
\bigg\Downarrow \Bigg\Downarrow$
```

$$\left((x+1)(x-1)\right)^2$$

$$\left(\left(\left(\left(\right)\right)\right)\right) \quad \bigl| \Bigl| \biggl| \Biggl| \quad \big\Downarrow \Big\Downarrow \bigg\Downarrow \Bigg\Downarrow$$

4.4 多行公式

4.4.1 长公式折行

通常来讲应当避免写出需要折行的长公式。如果一定要折行的话，优先在等号之前折行，其次在加号、减号之前，再次在乘号、除号之前。其它位置应当避免折行。

amsmath 宏包的 `multline` 环境提供了书写折行长公式的方便环境。它允许用 `\\` 折行，将公式编号放在最后一行。多行公式的首行左对齐，末行右对齐，其余行居中。

```
\begin{multline}
a + b + c + d + e + f
+ g + h + i \\
= j + k + l + m + n \\
= o + p + q + r + s \\
= t + u + v + x + z
\end{multline}
```

$$\begin{aligned}
 a + b + c + d + e + f + g + h + i \\
 &= j + k + l + m + n \\
 &= o + p + q + r + s \\
 &= t + u + v + x + z \quad (4.3)
 \end{aligned}$$

与表格不同的是，公式的最后一行不写 `\\`，如果写了，反倒会产生一个多余的空行。

类似 `equation*`，`multline*` 环境排版不带编号的折行长公式。

4.4.2 多行公式

更多的情况是，我们需要罗列一系列公式，并令其按照等号对齐。

读者可能阅读过其它手册或者资料，知道 L^AT_EX 提供了 `eqnarray` 环境。它按照等号左边——等号——等号右边呈三列对齐，但等号周围的空隙过大，加上公式编号等一些 bug，目前已不推荐使用⁴。

目前最常用的是 `align` 环境，它将公式用 `&` 隔为两部分并对齐。分隔符通常放在等号左边：

```
\begin{align}
a &= b + c \\
&= d + e
\end{align}
```

$$\begin{aligned}
 a &= b + c \quad (4.4) \\
 &= d + e \quad (4.5)
 \end{aligned}$$

`align` 环境会给每行公式都编号。我们仍然可以用 `\notag` 去掉某行的编号。在以下的例子，为了对齐加号，我们将分隔符放在等号右边，这时需要给等号后添加一对括号 `{}` 以产生正常的间距：

```
\begin{align}
a &= {} & b + c \\
&= {} & d + e + f + g + h + i \\
& & + j + k + l \notag \\
& & & + m + n + o \\
&= {} & p + q + r + s
\end{align}
```

$$\begin{aligned}
 a &= b + c \quad (4.6) \\
 &= d + e + f + g + h + i + j + k + l \\
 & \quad + m + n + o \quad (4.7) \\
 &= p + q + r + s \quad (4.8)
 \end{aligned}$$

`align` 还能够对齐多组公式，除等号前的 `&` 之外，公式之间也用 `&` 分隔：

```
\begin{align}
a &= 1 & b &= 2 & c &= 3 \\
d &= -1 & e &= -2 & f &= -5
\end{align}
```

$$\begin{aligned}
 a = 1 \quad b = 2 \quad c = 3 \quad (4.9) \\
 d = -1 \quad e = -2 \quad f = -5 \quad (4.10)
 \end{aligned}$$

如果我们不需要按等号对齐，只需罗列数个公式，`gather` 将是一个很好用的环境：

⁴<http://tug.org/pracjourn/2006-4/madsen/madsen.pdf>

```
\begin{gather}
a = b + c \\
d = e + f + g \\
h + i = j + k \notag \\
l + m = n
\end{gather}
```

$$a = b + c \quad (4.11)$$

$$d = e + f + g \quad (4.12)$$

$$h + i = j + k$$

$$l + m = n \quad (4.13)$$

`align` 和 `gather` 有对应的不带编号的版本 `align*` 和 `gather*`。

4.4.3 公用编号的多行公式

另一个常见的需求是将多个公式组在一起公用一个编号，编号位于公式的居中位置。为此，`amsmath` 宏包提供了诸如 `aligned`、`gathered` 等环境，与 `equation` 环境套用。以 `-ed` 结尾的环境用法与前一节不以 `-ed` 结尾的环境用法一一对应。我们仅以 `aligned` 举例：

```
\begin{equation}
\begin{aligned}
a &= b + c \\
d &= e + f + g \\
h + i &= j + k \\
l + m &= n
\end{aligned}
\end{equation}
```

$$\begin{aligned} a &= b + c \\ d &= e + f + g \\ h + i &= j + k \\ l + m &= n \end{aligned} \quad (4.14)$$

`split` 环境和 `aligned` 环境用法类似，也用于和 `equation` 环境套用，区别是 `split` 只能将每行的一个公式分两栏，`aligned` 允许每行多个公式多栏。

4.5 数组和矩阵

为了排版二维数组，`LaTeX` 提供了 `array` 环境，用法与 `tabular` 环境极为类似，也需要定义列格式，并用 `\\` 换行。数组可作为一个子公式，在外套用 `\left`、`\right` 等定界符：

```
\[
\mathbf{X} = \left(
\begin{array}{ccc}
x_1 & x_2 & \ldots \\
x_3 & x_4 & \ldots \\
\vdots & \vdots & \ddots
\end{array}
\right)
\]
```

$$\mathbf{X} = \begin{pmatrix} x_1 & x_2 & \dots \\ x_3 & x_4 & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

值得注意的是，上一节末尾介绍的 `aligned` 等环境也可以用定界符包裹。我们还可以利用空的定界符排版出这样的效果：

```
\[
|x| = \left\{
\begin{array}{rl}
-x & \text{if } x < 0, \\
0 & \text{if } x = 0, \\
x & \text{if } x > 0.
\end{array}
\right.
\]
```

$$|x| = \begin{cases} -x & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ x & \text{if } x > 0. \end{cases}$$

不过上述例子可以用 `amsmath` 提供的 `cases` 环境更轻松地完成：

```
\[
|x| =
\begin{cases}
-x & \text{if } x < 0, \\
0 & \text{if } x = 0, \\
x & \text{if } x > 0.
\end{cases}
\]
```

$$|x| = \begin{cases} -x & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ x & \text{if } x > 0. \end{cases}$$

我们当然也可以用 `array` 环境排版各种矩阵。`amsmath` 宏包还直接提供了多种排版矩阵的环境，包括不带定界符的 `matrix`，以及带各种定界符的矩阵 `pmatrix`、`bmatrix`、`Bmatrix`、`vmatrix`、`Vmatrix`。使用这些环境时，无需给定列格式：

```
\[
\begin{matrix}
1 & 2 \\
3 & 4
\end{matrix} \quad \quad \quad
\begin{bmatrix}
p_{11} & p_{12} & \ldots & p_{1n} \\
p_{21} & p_{22} & \ldots & p_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
p_{m1} & p_{m2} & \ldots & p_{mn}
\end{bmatrix}
\]
```

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \quad \quad \quad \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix}$$

4.6 公式中的间距

前文提到过，绝大部分时候，数学公式中各元素的间距是根据符号类型自动生成的，需要我们手动调整的情况极少。我们已经认识了两个生成间距的命令 `\quad` 和 `\qqquad`。在公式中我们还能用到的间距包括 `\,`、`\:`、`\;`；以及负间距 `\!`，其中 `\quad`、`\qqquad` 和 `\,` 在文本和数学环境中可用，后三个命令只用于数学环境。文本中的 `_` 也能使用在数学公式中。

无额外间距	aa	<code>\,</code>	$a \, a$
<code>\quad</code>	$a \quad a$	<code>\:</code>	$a : a$
<code>\qqquad</code>	$a \quad \quad a$	<code>\;</code>	$a ; a$
<code>_</code>	$a _ a$	<code>\!</code>	$a ! a$

一个常见的用途是修正积分的被积函数 $f(x)$ 和微元 dx 之间的距离。注意微元里的 d 用的是直立体：

```
\[
\int_a^b f(x)\mathrm{d}x
\quad
\int_a^b f(x)\,,\mathrm{d}x
\]
```

$$\int_a^b f(x)dx \quad \int_a^b f(x) \, dx$$

另一个用途是生成多重积分号。如果我们直接连写两个 `\int`，之间的间距将会过宽，此时可以使用负间距 `\!` 修正之。不过 `amsmath` 提供了更方便的多重积分号，如二重积分 `\iint`、三重积分 `\iiint` 等。

```
\newcommand\diff{\,\mathrm{d}}
\begin{gather*}
\int\int f(x)g(y)
\diff x \diff y \!
\int\!\!\!\!\!\int f(x)g(y)
\diff x \diff y \!
\iint f(x)g(y) \diff x \diff y \!
\iint\quad\iiint\quad\idotsint
\end{gather*}
```

$$\begin{aligned} & \int \int f(x)g(y) \, dx \, dy \\ & \iint f(x)g(y) \, dx \, dy \\ & \iint f(x)g(y) \, dx \, dy \\ & \iint \quad \iiint \quad \int \dots \int \end{aligned}$$

4.7 数学符号的字体控制

4.7.1 数学字母字体

L^AT_EX 允许一部分数学符号切换字体，主要是拉丁字母、数字等等。表 4.2 给出了切换字体的命令。某一些命令需要字体宏包的支持。

```
$\mathcal{R} \quad \mathfrak{R} \quad \mathbb{R}
\quad \mathbb{R}$
\[\mathcal{L}
= -\frac{1}{4}F_{\mu\nu}F^{\mu\nu}\]
$\mathfrak{su}(2)$ and
$\mathfrak{so}(3)$ Lie algebra
```

\mathcal{R} \mathfrak{R} \mathbb{R}

$$\mathcal{L} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu}$$

$\mathfrak{su}(2)$ and $\mathfrak{so}(3)$ Lie algebra

4.7.2 数学符号的尺寸

数学符号按照符号排版的位置规定尺寸，从大到小包括行间公式尺寸、行内公式尺寸、上下标尺寸、次级上下标尺寸。除了字号有别之外，行间和行内公式尺寸下的巨算符也使用不一样的大小。L^AT_EX 为每个数学尺寸指定了一个切换的命令。

例如，分式的分子和分母使用比当前公式小一号的尺寸，比如巨算符将使用行内公式下的小尺寸。你可以使用合适的数学尺寸命令调整：

表 4.2: 数学字母字体。

例子	命令	依赖的宏包
$ABCDEabcde1234$	<code>\mathnormal{...}</code>	
$ABCD\!Eabcde1234$	<code>\mathrm{...}</code>	
$ABCDE\!abcde1234$	<code>\mathit{...}</code>	
$\mathbf{ABCDEabcde1234}$	<code>\mathbf{...}</code>	
$ABCD\!Eabcde1234$	<code>\mathsf{...}</code>	
$ABCD\!Eabcde1234$	<code>\mathtt{...}</code>	
$ABCDE$	<code>\mathcal{...}</code>	只大写字母
$ABCD\!E$	<code>\mathcal{...}</code>	eucal, 只大写字母
$\mathscr{A}\mathscr{B}\mathscr{C}\mathscr{D}\mathscr{E}$	<code>\mathscr{...}</code>	mathrsfs, 只大写字母
$\mathbb{A}\mathbb{B}\mathbb{C}\mathbb{D}\mathbb{E}abcde1234$	<code>\mathfrak{...}</code>	amssymb 或 eufrak
\mathbf{ABCDE}	<code>\mathbf{...}</code>	amssymb, 只大写字母

表 4.3: 数学符号尺寸。

<code>\displaystyle</code>	行间公式尺寸	$\sum a$
<code>\textstyle</code>	行内公式尺寸	$\sum a$
<code>\scriptstyle</code>	上下标尺寸	a
<code>\scriptscriptstyle</code>	次级上下标尺寸	a

```
\[
P = \frac{\displaystyle
\sum_{i=1}^n (x_i - x)
(y_i - y) }
{\displaystyle \left[
\sum_{i=1}^n (x_i - x)^2
\sum_{i=1}^n (y_i - y)^2
\right]^{1/2} }
\]
```

$$P = \frac{\sum_{i=1}^n (x_i - x)(y_i - y)}{\left[\sum_{i=1}^n (x_i - x)^2 \sum_{i=1}^n (y_i - y)^2 \right]^{1/2}}$$

4.7.3 加粗的数学符号

在 L^AT_EX 中为符号切换数学字体并不十分自由，只能通过 `\mathbf` 等有限的命令切换字体。比如想得到粗斜体的符号，就没有现成的命令⁵；再比如 `\mathbf` 只能改变拉丁字母，希腊字母就没有用。

L^AT_EX 提供了一个命令 `\boldmath` 令用户可以将整套数学字体切换为粗体版本。但这个命令只能在公式外使用：

```
$\mu, M \quad
\mathbf{\mu}, \mathbf{M}
\quad \{\boldmath$\mu, M$\}
```

$\mu, M \quad \mu, \mathbf{M} \quad \boldsymbol{\mu}, \mathbf{M}$

⁵国内可能还有使用粗斜体表示向量符号的习惯，但这并不是正确的习惯。

amsmath 提供了一个 `\boldsymbol` 命令 (由调用的 `amsbsy` 宏包提供), 用于打破 `\boldmath` 的限制, 在公式内部将一部分符号切换为粗体。

```
$\mu, M \quad
```

```
\boldsymbol{\mu}, \boldsymbol{M}
```

$$\mu, M \quad \boldsymbol{\mu}, \boldsymbol{M}$$

然而定界符、巨算符等一些符号本身没有粗体版本, `\boldsymbol` 也得不到粗体。L^AT_EX 工具宏集之一的 `bm` 宏包可以用 `\bm` 命令生成“伪粗体”, 一定程度上解决了不带粗体版本的符号的问题。这里不做过多介绍, 详情请参考 `bm` 宏包的帮助文档。

4.8 定理环境

使用 L^AT_EX 排版数学和其他科技文档时, 会接触到大量的定理、证明等内容。L^AT_EX 提供了一个基本的命令 `\newtheorem` 提供定理环境的定义:

```
\newtheorem{<type>}{<title>}[<section-name>]
```

```
\newtheorem{<type>}[<counter>]{<title>}
```

`<type>` 为定理类型的名称, 作为一个环境来使用。定理环境都需要定义, L^AT_EX 里没有现成的 `theorem` 环境, 直接使用很可能会出错。`<title>` 是定理类型的标签 (“定理”, “公理” 等), 排版在序号之前。

定理的序号由两个可选参数之一决定, 它们不能同时使用:

- `<section name>` 为章节名称, 这使定理序号成为章节的下一级序号;
- `<counter>` 为用 `\newcounter` 自定义的计数器名称 (详见 8.3 节), 定理序号由这个计数器管理。

如果两个可选参数都不用的话, 则使用一个默认的计数器。

例如, 我们用以下代码定义了一个 `mythm` 环境:

```
\newtheorem{mythm}{My Theorem}[section]
```

于是我们可以使用 `mythm` 环境排版定理。定理带一个可选参数, 用于注明定理的名称, 如“法拉第定律”等。在环境内还可以用 `\label` 声明引用:

```
\newtheorem{mythm}{My Theorem}[section]
\begin{mythm}\label{thm:light}
The light speed in vaccum
is $299,792,458\,\mathrm{m/s}$.
\end{mythm}
\begin{mythm}[Energy]
The relationship of energy,
momentum and mass is

$$E^2 = m_0^2 c^4 + p^2 c^2$$

where  $c$  is the light speed
described in theorem \ref{thm:light}.
\end{mythm}
```

My Theorem 4.8.1. *The light speed in vaccum is 299,792,458 m/s.*

My Theorem 4.8.2 (Energy). *The relationship of energy, momentum and mass is*

$$E^2 = m_0^2 c^4 + p^2 c^2$$

where c is the light speed described in theorem 4.8.1.

4.8.1 amsthm 宏包

L^AT_EX 只给了原始的证明环境格式（粗体标签、斜体正文、定理名用小括号包裹）。如果需要修改格式，则要依赖其它的宏包，如 amsthm、ntheorem 等等。本小节简单介绍一下 amsthm 的用法。

amsthm 提供了 \theoremstyle 命令支持定理格式的切换，在用 \newtheorem 命令定义定理环境之前使用。amsthm 预定义了三种格式用于 \theoremstyle: plain 和 L^AT_EX 原始的格式一致；definition 使用粗体标签、正体内容；remark 使用斜体标签、正体内容。

另外 amsthm 还支持用带星号的 \newtheorem* 定义不带序号的定理环境：

```
\theoremstyle{definition} \newtheorem{law}{Law}
\theoremstyle{plain} \newtheorem{jury}[law]{Jury}
\theoremstyle{remark} \newtheorem*{mar}{Margaret}
```

以上例子定义的 jury 环境与 law 环境共用编号，mar 环境不编号：

```
\begin{law} \label{law:box}
Don' t hide in the witness box
\end{law}
\begin{jury}[The Twelve]
It could be you! So beware and
see law~\ref{law:box}.\end{jury}
\begin{jury}
You will disregard the last
statement.\end{jury}
\begin{mar}No, No, No\end{mar}
\begin{mar}Denis!\end{mar}
```

Law 1. Don' t hide in the witness box

Jury 2 (The Twelve). *It could be you! So beware and see law 1.*

Jury 3. *You will disregard the last statement.*

Margaret. No, No, No

Margaret. Denis!

amsthm 还支持使用 \newtheoremstyle 命令自定义定理格式，更为方便使用的是 ntheorem 宏包。感兴趣的读者可参阅它们的帮助手册。

4.8.2 证明环境和证毕符号

amsthm 还提供了一个 proof 环境用于排版定理的证明过程。proof 环境末尾自动加上一个 □ 证毕符号：

```
\begin{proof}
For simplicity, we use
\[
E=mc^2
\]
That's it.
\end{proof}
```

Proof. For simplicity, we use

$$E = mc^2$$

That's it. □

如果行末是一个不带编号的公式，□ 符号会另起一行，这时可使用 \qedhere 命令将 □ 符号放在公式末尾：

```
\begin{proof}
For simplicity, we use
\[
E=mc^2 \quad \text{\qedhere}
\]
\end{proof}
```

Proof. For simplicity, we use

$$E = mc^2 \quad \square$$

`\qedhere` 对于 `align*` 等命令也有效：

```
\begin{proof}
Assuming  $\gamma$ 
 $= 1/\sqrt{1-v^2/c^2}$ , then
\begin{align*}
E &= \gamma m_0 c^2 \\
p &= \gamma m_0 v \quad \text{\qedhere}
\end{align*}
\end{proof}
```

Proof. Assuming $\gamma = 1/\sqrt{1 - v^2/c^2}$, then

$$\begin{aligned} E &= \gamma m_0 c^2 \\ p &= \gamma m_0 v \end{aligned} \quad \square$$

在使用带编号的公式时，建议最好**不要使用** `\qedhere` 命令，而是让 `proof` 环境自动生成。对带编号的公式使用 `\qedhere` 命令会使 \square 符号放在一个难看的位置，紧贴着公式：

```
\begin{proof}
For simplicity, we use
\begin{equation}
E=mc^2.\text{\qedhere}
\end{equation}
\end{proof}
```

Proof. For simplicity, we use

$$E = mc^2. \quad (4.15) \quad \square$$

在 `align` 等环境中使用 `\qedhere` 命令会使 \square 盖掉公式的编号；使用 `equation` 嵌套 `aligned` 等环境时，`\qedhere` 命令会将 \square 直接放在公式后。这些位置都不太正常。

4.9 符号表

有几个注意事项：

1. 蓝色的命令依赖 `amsmath` 宏包（非 `amssymb` 宏包）；
2. 带有角标^ℓ的符号命令依赖 `latexsym` 宏包。

4.9.1 L^AT_EX 普通符号

表 4.4: 文本/数学模式通用符号

这些符号可用于文本和数学模式。

{	\{	}	\}	\$	\\$	%	\%
†	\dag	§	\S	©	\copyright	...	\dots
‡	\ddag	¶	\P	£	\pounds		

表 4.5: 希腊字母。

`\Alpha`, `\Beta` 等希腊字母符号不存在，因为它们和拉丁字母 A,B 等一模一样；小写字母里也不存在 `\omicron`，直接用 `o` 代替。

α	<code>\alpha</code>	θ	<code>\thetaeta</code>	o	<code>o</code>	v	<code>\upsilon</code>
β	<code>\betaeta</code>	ϑ	<code>\varthetaeta</code>	π	<code>\pi</code>	ϕ	<code>\phi</code>
γ	<code>\gamma</code>	ι	<code>\iota</code>	ϖ	<code>\varpi</code>	φ	<code>\varphi</code>
δ	<code>\delta</code>	κ	<code>\kappa</code>	ρ	<code>\rho</code>	χ	<code>\chi</code>
ϵ	<code>\epsilon</code>	λ	<code>\lambda</code>	ϱ	<code>\varrho</code>	ψ	<code>\psi</code>
ε	<code>\varepsilon</code>	μ	<code>\mu</code>	σ	<code>\sigma</code>	ω	<code>\omega</code>
ζ	<code>\zeta</code>	ν	<code>\nu</code>	ς	<code>\varsigma</code>		
η	<code>\eta</code>	ξ	<code>\xi</code>	τ	<code>\tau</code>		
Γ	<code>\Gamma</code>	Λ	<code>\Lambda</code>	Σ	<code>\Sigma</code>	Ψ	<code>\Psi</code>
Δ	<code>\Delta</code>	Ξ	<code>\Xi</code>	Υ	<code>\Upsilon</code>	Ω	<code>\Omega</code>
Θ	<code>\Theta</code>	Π	<code>\Pi</code>	Φ	<code>\Phi</code>		
Γ	<code>\varGamma</code>	Λ	<code>\varLambda</code>	Σ	<code>\varSigma</code>	Ψ	<code>\varPsi</code>
Δ	<code>\varDelta</code>	Ξ	<code>\varXi</code>	Υ	<code>\varUpsilon</code>	Ω	<code>\varOmega</code>
Θ	<code>\varTheta</code>	Π	<code>\varPi</code>	Φ	<code>\varPhi</code>		

表 4.6: 二元关系符。

所有的二元关系符都可以加 `\not` 前缀得到相反意义的关系符，例如 `\not=` 就得到不等号（同 `\ne`）。

$<$	<code><</code>	$>$	<code>></code>	$=$	<code>=</code>
\leq	<code>\leq</code> or <code>\le</code>	\geq	<code>\geq</code> or <code>\ge</code>	\equiv	<code>\equiv</code>
\ll	<code>\ll</code>	\gg	<code>\gg</code>	$\dot{=}$	<code>\doteq</code>
\prec	<code>\prec</code>	\succ	<code>\succ</code>	\sim	<code>\sim</code>
\preceq	<code>\preceq</code>	\succeq	<code>\succeq</code>	\simeq	<code>\simeq</code>
\subset	<code>\subset</code>	\supset	<code>\supset</code>	\approx	<code>\approx</code>
\subseteq	<code>\subseteq</code>	\supseteq	<code>\supseteq</code>	\cong	<code>\cong</code>
\sqsubset^ℓ	<code>\sqsubset^\ell</code>	\sqsupset^ℓ	<code>\sqsupset^\ell</code>	\bowtie	<code>\Join^\ell</code>
\sqsubseteq	<code>\sqsubseteq</code>	\sqsupseteq	<code>\sqsupseteq</code>	\bowtie	<code>\bowtie</code>
\in	<code>\in</code>	\ni , \owns	<code>\ni</code> , <code>\owns</code>	\propto	<code>\propto</code>
\vdash	<code>\vdash</code>	\dashv	<code>\dashv</code>	\models	<code>\models</code>
\mid	<code>\mid</code>	\parallel	<code>\parallel</code>	\perp	<code>\perp</code>
\smile	<code>\smile</code>	\frown	<code>\frown</code>	\asymp	<code>\asymp</code>
$:$	<code>:</code>	\notin	<code>\notin</code>	\neq	<code>\neq</code> or <code>\ne</code>

表 4.7: 二元运算符。

$+$	<code>+</code>	$-$	<code>-</code>	
\pm	<code>\pm</code>	\mp	<code>\mp</code>	\triangleleft <code>\triangleleft</code>
\cdot	<code>\cdot</code>	\div	<code>\div</code>	\triangleright <code>\triangleright</code>
\times	<code>\times</code>	\setminus	<code>\setminus</code>	\star <code>\star</code>
\cup	<code>\cup</code>	\cap	<code>\cap</code>	$*$ <code>\ast</code>
\sqcup	<code>\sqcup</code>	\sqcap	<code>\sqcap</code>	\circ <code>\circ</code>
\vee , \lor	<code>\vee</code> , <code>\lor</code>	\wedge , \land	<code>\wedge</code> , <code>\land</code>	\bullet <code>\bullet</code>
\oplus	<code>\oplus</code>	\ominus	<code>\ominus</code>	\diamond <code>\diamond</code>
\odot	<code>\odot</code>	\oslash	<code>\oslash</code>	\uplus <code>\uplus</code>
\otimes	<code>\otimes</code>	\bigcirc	<code>\bigcirc</code>	\amalg <code>\amalg</code>
\bigtriangleup	<code>\bigtriangleup</code>	\bigtriangledown	<code>\bigtriangledown</code>	\dagger <code>\dagger</code>
\lhd^ℓ	<code>\lhd^\ell</code>	\rhd^ℓ	<code>\rhd^\ell</code>	\ddagger <code>\ddagger</code>
\unlhd^ℓ	<code>\unlhd^\ell</code>	\unrhd^ℓ	<code>\unrhd^\ell</code>	\wr <code>\wr</code>

表 4.8: 巨算符。

Σ	\sum	<code>\sum</code>	\cup	\bigcup	<code>\bigcup</code>	\vee	\bigvee	<code>\bigvee</code>
\prod	\prod	<code>\prod</code>	\cap	\bigcap	<code>\bigcap</code>	\wedge	\bigwedge	<code>\bigwedge</code>
\coprod	\coprod	<code>\coprod</code>	\sqcup	\bigsqcup	<code>\bigsqcup</code>	\oplus	\bigoplus	<code>\bigoplus</code>
\int	\int	<code>\int</code>	\oint	\oint	<code>\oint</code>	\odot	\bigodot	<code>\bigodot</code>
\oplus	\oplus	<code>\bigoplus</code>	\otimes	\bigotimes	<code>\bigotimes</code>			
\iint	\iint	<code>\iint</code>	\iiint	\iiint	<code>\iiint</code>	\iiint	\iiint	<code>\iiint</code>
$\int \cdots \int$	$\int \cdots \int$	<code>\dotsint</code>						

表 4.9: 数学重音符号。

最后一个 `\wideparen` 依赖 `yhmath` 宏包。

\hat{a}	<code>\hat{a}</code>	\check{a}	<code>\check{a}</code>	\tilde{a}	<code>\tilde{a}</code>
\acute{a}	<code>\acute{a}</code>	\grave{a}	<code>\grave{a}</code>	\breve{a}	<code>\breve{a}</code>
\bar{a}	<code>\bar{a}</code>	\vec{a}	<code>\vec{a}</code>	\mathring{a}	<code>\mathring{a}</code>
\dot{a}	<code>\dot{a}</code>	\ddot{a}	<code>\ddot{a}</code>	\dddot{a}	<code>\dddot{a}</code>
\ddot{a}	<code>\dddot{a}</code>				
\widehat{AAA}	<code>\widehat{AAA}</code>	\widetilde{AAA}	<code>\widetilde{AAA}</code>	\wideparen{AAA}	<code>\wideparen{AAA}</code>

表 4.10: 箭头。

\leftarrow	<code>\leftarrow</code> or <code>\gets</code>	\longleftarrow	<code>\longleftarrow</code>
\rightarrow	<code>\rightarrow</code> or <code>\to</code>	\longrightarrow	<code>\longrightarrow</code>
\leftrightarrow	<code>\leftrightarrow</code>	\longleftrightarrow	<code>\longleftrightarrow</code>
\Leftarrow	<code>\Leftarrow</code>	\Longleftarrow	<code>\Longleftarrow</code>
\Rightarrow	<code>\Rightarrow</code>	\Longrightarrow	<code>\Longrightarrow</code>
\Leftrightarrow	<code>\Leftrightarrow</code>	\Longleftrightarrow	<code>\Longleftrightarrow</code>
\mapsto	<code>\mapsto</code>	\longmapsto	<code>\longmapsto</code>
\hookrightarrow	<code>\hookrightarrow</code>	\hookrightarrow	<code>\hookrightarrow</code>
\leftharpoonup	<code>\leftharpoonup</code>	\rightharpoonup	<code>\rightharpoonup</code>
\leftharpoondown	<code>\leftharpoondown</code>	\rightharpoondown	<code>\rightharpoondown</code>
\rightleftharpoons	<code>\rightleftharpoons</code>	\iff	<code>\iff</code>
\uparrow	<code>\uparrow</code>	\downarrow	<code>\downarrow</code>
\updownarrow	<code>\updownarrow</code>	\Uparrow	<code>\Uparrow</code>
\Downarrow	<code>\Downarrow</code>	\Updownarrow	<code>\Updownarrow</code>
\nearrow	<code>\nearrow</code>	\searrow	<code>\searrow</code>
\swarrow	<code>\swarrow</code>	\nwarrow	<code>\nwarrow</code>
\leadsto^ℓ	<code>\leadsto^\ell</code>		

表 4.11: 作为重音的箭头符号。

\overrightarrow{AB}	<code>\overrightarrow{AB}</code>	\overleftarrow{AB}	<code>\overleftarrow{AB}</code>	\overleftrightarrow{AB}	<code>\overleftrightarrow{AB}</code>
\overleftarrow{AB}	<code>\overleftarrow{AB}</code>	\overrightarrow{AB}	<code>\overrightarrow{AB}</code>	\overleftrightarrow{AB}	<code>\overleftrightarrow{AB}</code>
\overleftrightarrow{AB}	<code>\overleftrightarrow{AB}</code>	\overleftarrow{AB}	<code>\overleftarrow{AB}</code>	\overrightarrow{AB}	<code>\overrightarrow{AB}</code>

表 4.12: 定界符。

(())	↑	<code>\uparrow</code>
[[or <code>\lbrack</code>]] or <code>\rbrack</code>	↓	<code>\downarrow</code>
{	<code>\{</code> or <code>\lbrace</code>	}	<code>\}</code> or <code>\rbrace</code>	↕	<code>\updownarrow</code>
⟨	<code>\langle</code>	⟩	<code>\rangle</code>	↗	<code>\Uparrow</code>
	or <code>\vert</code>		or <code>\Vert</code>	⇓	<code>\Downarrow</code>
/	/	\	<code>\backslash</code>	⇕	<code>\Updownarrow</code>
⌊	<code>\lfloor</code>	⌋	<code>\rfloor</code>		
⌈	<code>\lceil</code>	⌉	<code>\rceil</code>		

表 4.13: 用于行间公式的大定界符。

(())	\lgroup	\rgroup	\lmoustache	\rmoustache
				\arrowvert	\Arrowvert	\bracevert	

表 4.14: 其他符号。

...	<code>\dots</code>	...	<code>\cdots</code>	⋮	<code>\vdots</code>	⋱	<code>\ddots</code>
ℏ	<code>\hbar</code>	ℐ	<code>\imath</code>	ℐ	<code>\jmath</code>	ℓ	<code>\ell</code>
ℜ	<code>\Re</code>	ℑ	<code>\Im</code>	ℵ	<code>\aleph</code>	wp	<code>\wp</code>
∀	<code>\forall</code>	∃	<code>\exists</code>	ℳ	<code>\mho</code> ^ℓ	∂	<code>\partial</code>
'	'	'	<code>\prime</code>	∅	<code>\emptyset</code>	∞	<code>\infty</code>
∇	<code>\nabla</code>	△	<code>\triangle</code>	□	<code>\Box</code> ^ℓ	◇	<code>\Diamond</code> ^ℓ
⊥	<code>\bot</code>	⊤	<code>\top</code>	∠	<code>\angle</code>	√	<code>\surd</code>
♦	<code>\diamondsuit</code>	♥	<code>\heartsuit</code>	♣	<code>\clubsuit</code>	♠	<code>\spadesuit</code>
¬	<code>\neg</code> or <code>\not</code>	♭	<code>\flat</code>	♮	<code>\natural</code>	♯	<code>\sharp</code>

4.9.2 \mathcal{AMS} 符号

本小节所有符号依赖 `amssymb` 宏包。

表 4.15: \mathcal{AMS} 定界符

\lrcorner	<code>\ulcorner</code>	\urcorner	<code>\urcorner</code>	\llcorner	<code>\llcorner</code>	\lrcorner	<code>\lrcorner</code>
\lvert	<code>\lvert</code>	\rvert	<code>\rvert</code>	\lVert	<code>\lVert</code>	\rVert	<code>\rVert</code>

表 4.16: \mathcal{AMS} 希腊字母和希伯来字母。

\digamma	<code>\digamma</code>	\varkappa	<code>\varkappa</code>	\beth	<code>\beth</code>	\gimel	<code>\gimel</code>	\daleth	<code>\daleth</code>
------------	-----------------------	-------------	------------------------	---------	--------------------	----------	---------------------	-----------	----------------------

表 4.17: \mathcal{AMS} 二元运算符。

$\dot{+}$	<code>\dotplus</code>	\cdot	<code>\centerdot</code>		
\ltimes	<code>\ltimes</code>	\rtimes	<code>\rtimes</code>	\div	<code>\divideontimes</code>
\mathcal{U}	<code>\doublecup</code>	\mathcal{M}	<code>\doublecap</code>	\smallsetminus	<code>\smallsetminus</code>
\veebar	<code>\veebar</code>	$\bar{\wedge}$	<code>\barwedge</code>	$\overline{\wedge}$	<code>\doublebarwedge</code>
\boxplus	<code>\boxplus</code>	\boxminus	<code>\boxminus</code>	\ominus	<code>\circleddash</code>
\boxtimes	<code>\boxtimes</code>	\boxdot	<code>\boxdot</code>	\odot	<code>\circledcirc</code>
\intercal	<code>\intercal</code>	\circledast	<code>\circledast</code>	\ltimes	<code>\rightthreetimes</code>
\curlyvee	<code>\curlyvee</code>	\curlywedge	<code>\curlywedge</code>	\leftthreetimes	<code>\leftthreetimes</code>

表 4.18: \mathcal{AMS} 二元关系符。

\lessdot	<code>\lessdot</code>	\gtrdot	<code>\gtrdot</code>	\doteqdot	<code>\doteqdot</code>
\leqslant	<code>\leqslant</code>	\geqslant	<code>\geqslant</code>	\risingdotseq	<code>\risingdotseq</code>
\leqslantless	<code>\leqslantless</code>	\geqslantgtr	<code>\geqslantgtr</code>	\fallingdotseq	<code>\fallingdotseq</code>
\leqq	<code>\leqq</code>	\geqq	<code>\geqq</code>	\eqcirc	<code>\eqcirc</code>
\lll or \llless	<code>\lll</code> or <code>\llless</code>	\ggg	<code>\ggg</code>	\circeq	<code>\circeq</code>
\lesssim	<code>\lesssim</code>	\gtrsim	<code>\gtrsim</code>	\triangleq	<code>\triangleq</code>
\lessapprox	<code>\lessapprox</code>	\gtrapprox	<code>\gtrapprox</code>	\bumpeq	<code>\bumpeq</code>
\lessgtr	<code>\lessgtr</code>	\gtrless	<code>\gtrless</code>	\Bumpeq	<code>\Bumpeq</code>
\lesseqgtr	<code>\lesseqgtr</code>	\gtreqless	<code>\gtreqless</code>	\thicksim	<code>\thicksim</code>
\lesseqqgtr	<code>\lesseqqgtr</code>	\gtreqqless	<code>\gtreqqless</code>	\thickapprox	<code>\thickapprox</code>
\preccurlyeq	<code>\preccurlyeq</code>	\succcurlyeq	<code>\succcurlyeq</code>	\approxeq	<code>\approxeq</code>
\curlyeqprec	<code>\curlyeqprec</code>	\curlyeqsucc	<code>\curlyeqsucc</code>	\backsim	<code>\backsim</code>
\precsim	<code>\precsim</code>	\succsim	<code>\succsim</code>	\backsimeq	<code>\backsimeq</code>
\precapprox	<code>\precapprox</code>	\succapprox	<code>\succapprox</code>	\vDash	<code>\vDash</code>
\subseteq	<code>\subseteq</code>	\supseteq	<code>\supseteq</code>	\Vdash	<code>\Vdash</code>
\shortparallel	<code>\shortparallel</code>	\Supset	<code>\Supset</code>	\Vvdash	<code>\Vvdash</code>
\blacktriangleleft	<code>\blacktriangleleft</code>	\sqsupset	<code>\sqsupset</code>	\backepsilon	<code>\backepsilon</code>
\vartriangleright	<code>\vartriangleright</code>	\because	<code>\because</code>	\varpropto	<code>\varpropto</code>
\blacktriangleright	<code>\blacktriangleright</code>	\Subset	<code>\Subset</code>	\between	<code>\between</code>
\trianglerighteq	<code>\trianglerighteq</code>	\smallfrown	<code>\smallfrown</code>	\pitchfork	<code>\pitchfork</code>
\vartriangleleft	<code>\vartriangleleft</code>	\shortmid	<code>\shortmid</code>	\smallsmile	<code>\smallsmile</code>
\trianglelefteq	<code>\trianglelefteq</code>	\therefore	<code>\therefore</code>	\sqsubset	<code>\sqsubset</code>

表 4.19: \mathcal{AMS} 箭头。

\dashleftarrow	<code>\dashleftarrow</code>	\dashrightarrow	<code>\dashrightarrow</code>
\leftrightsquigarrow	<code>\leftrightsquigarrow</code>	\rightleftarrows	<code>\rightleftarrows</code>
\leftrightsquigarrow	<code>\leftrightsquigarrow</code>	\rightleftarrows	<code>\rightleftarrows</code>
\Lleftarrow	<code>\Lleftarrow</code>	\Rrightarrow	<code>\Rrightarrow</code>
\twoheadleftarrow	<code>\twoheadleftarrow</code>	\twoheadrightarrow	<code>\twoheadrightarrow</code>
\leftarrowtail	<code>\leftarrowtail</code>	\rightarrowtail	<code>\rightarrowtail</code>
\leftrightharpoons	<code>\leftrightharpoons</code>	\rightleftharpoons	<code>\rightleftharpoons</code>
\Lsh	<code>\Lsh</code>	\Rsh	<code>\Rsh</code>
\looparrowleft	<code>\looparrowleft</code>	\looparrowright	<code>\looparrowright</code>
\curvearrowleft	<code>\curvearrowleft</code>	\curvearrowright	<code>\curvearrowright</code>
\circlearrowleft	<code>\circlearrowleft</code>	\circlearrowright	<code>\circlearrowright</code>
\multimap	<code>\multimap</code>	\Uparrow	<code>\Uparrow</code>
\downdownarrows	<code>\downdownarrows</code>	\Uparrow	<code>\Uparrow</code>
\upharpoonright	<code>\upharpoonright</code>	\downharpoonright	<code>\downharpoonright</code>
\rightsquigarrow	<code>\rightsquigarrow</code>	\leftrightsquigarrow	<code>\leftrightsquigarrow</code>

表 4.20: \mathcal{AMS} 反义二元关系符和箭头。

\nless	<code>\nless</code>	\ngtr	<code>\ngtr</code>	\varsubsetneqq	<code>\varsubsetneqq</code>
\lneq	<code>\lneq</code>	\gneq	<code>\gneq</code>	\varsupsetneqq	<code>\varsupsetneqq</code>
\nleq	<code>\nleq</code>	\ngeq	<code>\ngeq</code>	\nsubseteqeq	<code>\nsubseteqeq</code>
\nleqslant	<code>\nleqslant</code>	\ngeqslant	<code>\ngeqslant</code>	\nsupseteqeq	<code>\nsupseteqeq</code>
\lneqq	<code>\lneqq</code>	\gneqq	<code>\gneqq</code>	\nmid	<code>\nmid</code>
\lvertneqq	<code>\lvertneqq</code>	\gvertneqq	<code>\gvertneqq</code>	\nparallel	<code>\nparallel</code>
\nleqq	<code>\nleqq</code>	\ngeqq	<code>\ngeqq</code>	\nshortmid	<code>\nshortmid</code>
\lnsim	<code>\lnsim</code>	\gnsim	<code>\gnsim</code>	\nshortparallel	<code>\nshortparallel</code>
\lnapprox	<code>\lnapprox</code>	\gnapprox	<code>\gnapprox</code>	\nsim	<code>\nsim</code>
\nprec	<code>\nprec</code>	\nsucc	<code>\nsucc</code>	\ncong	<code>\ncong</code>
\npreceq	<code>\npreceq</code>	\nsucceq	<code>\nsucceq</code>	\nvdash	<code>\nvdash</code>
\precneqq	<code>\precneqq</code>	\succneqq	<code>\succneqq</code>	\nvDash	<code>\nvDash</code>
\precnsim	<code>\precnsim</code>	\succnsim	<code>\succnsim</code>	\nVdash	<code>\nVdash</code>
\precnapprox	<code>\precnapprox</code>	\succnapprox	<code>\succnapprox</code>	\nVDash	<code>\nVDash</code>
\subsetneq	<code>\subsetneq</code>	\supsetneq	<code>\supsetneq</code>	\ntriangleleft	<code>\ntriangleleft</code>
\varsubsetneq	<code>\varsubsetneq</code>	\varsupsetneq	<code>\varsupsetneq</code>	\ntriangleright	<code>\ntriangleright</code>
\nsubseteq	<code>\nsubseteq</code>	\nsupseteq	<code>\nsupseteq</code>	\ntrianglelefteq	<code>\ntrianglelefteq</code>
\subsetneqq	<code>\subsetneqq</code>	\supsetneqq	<code>\supsetneqq</code>	\ntrianglerighteq	<code>\ntrianglerighteq</code>
\nleftarrow	<code>\nleftarrow</code>	\nrightarrow	<code>\nrightarrow</code>	\nleftrightharpoonup	<code>\nleftrightharpoonup</code>
\nLeftarrow	<code>\nLeftarrow</code>	\nRightarrow	<code>\nRightarrow</code>	\nLeftrightarrow	<code>\nLeftrightarrow</code>

表 4.21: \mathcal{AMS} 其它符号。

\hbar	<code>\hbar</code>	\hslash	<code>\hslash</code>	\Bbbk	<code>\Bbbk</code>
\square	<code>\square</code>	\blacksquare	<code>\blacksquare</code>	\textcircled{S}	<code>\circledS</code>
\triangle	<code>\vartriangle</code>	\blacktriangle	<code>\blacktriangle</code>	\complement	<code>\complement</code>
∇	<code>\triangledown</code>	\blacktriangledown	<code>\blacktriangledown</code>	\Game	<code>\Game</code>
\lozenge	<code>\lozenge</code>	\blacklozenge	<code>\blacklozenge</code>	\bigstar	<code>\bigstar</code>
\angle	<code>\angle</code>	\measuredangle	<code>\measuredangle</code>		
$/$	<code>\diagup</code>	\diagdown	<code>\diagdown</code>	\backprime	<code>\backprime</code>
\nexists	<code>\nexists</code>	\Finv	<code>\Finv</code>	\varnothing	<code>\varnothing</code>
\eth	<code>\eth</code>	\sphericalangle	<code>\sphericalangle</code>	\mho	<code>\mho</code>

第五章 排版样式设定

至此你已经基本学会排版内容丰富的文档，标题、目录、章节、公式、列表、图片、表格等等应有尽有。但是你可能已经有点不甘心了，因为似乎你排版出来的文档是千篇一律的模样—— \LaTeX 默认的字体、单调的页眉页脚、不太令你满意的页边距，等等。本章的内容将带你一览如何修改 \LaTeX 的排版样式。

5.1 字体和字号

\LaTeX 根据文档的逻辑结构（章节、脚注等）来选择默认的字体样式以及字号。需要更改字体样式或字号的话，可以使用表 5.1 和表 5.2 中列出的命令。

```
{\small The small and  
\textbf{bold} Romans ruled}  
\Large all of great big  
\itshape Italy}.
```

The small and **bold** Romans ruled all of
great big *Italy*.

\LaTeX 2_ε（相比于早期的 \LaTeX 2.09 版本）的一个重要特征是：字体的各种属性是相互独立的，这意味着用户可以改变字体的大小，而仍然保留字体原有的粗体或者斜体的特性。

5.1.1 字体样式

\LaTeX 提供了两组修改字体的命令，见表 5.1。其中诸如 `\bfseries` 形式的命令将会影响之后所有的字符，如果想要让它在局部生效，需要用花括号**分组**，也就是写成 `{\bfseries <some text>}` 这样的形式；对应的 `\textbf` 形式带一个参数，只改变参数内部的字体，更为常用。

在公式中，直接使用 `\textbf` 等命令不会起效，甚至报错。 \LaTeX 已有修改数学字体的命令，详见 4.7.1 小节。

5.1.2 字号

字号命令实际大小依赖于所使用的文档类及其选项。表 5.3 列出了这些命令在标准文档类中的绝对大小，单位为 pt。

使用字号命令的时候，通常也需要用花括号进行分组，如同 `\rmfamily` 那样。

```
He likes {\LARGE large and  
\small small} letters}.
```

He likes large and small letters.

\LaTeX 还提供了一个基础的命令 `\fontsize` 用于设定任意大小的字号：

```
\fontsize{<size>}{<base line-skip>}
```

表 5.1: 字体命令。

<code>\rmfamily</code>	<code>\textrm{...}</code>	roman	衬线字体 (罗马体)
<code>\sffamily</code>	<code>\textsf{...}</code>	sans serif	无衬线字体
<code>\ttfamily</code>	<code>\texttt{...}</code>	typewriter	等宽字体
<code>\mdseries</code>	<code>\textmd{...}</code>	medium	正常粗细 (中等)
<code>\bfseries</code>	<code>\textbf{...}</code>	bold face	粗体
<code>\upshape</code>	<code>\textup{...}</code>	upright	直立体
<code>\itshape</code>	<code>\textit{...}</code>	<i>italic</i>	意大利斜体
<code>\slshape</code>	<code>\textsl{...}</code>	<i>slanted</i>	倾斜体
<code>\scshape</code>	<code>\textsc{...}</code>	SMALL CAPS	小字母大写
<code>\em</code>	<code>\emph{...}</code>	<i>emphasized</i>	强调, 默认斜体
<code>\normalfont</code>	<code>\textnormal{...}</code>	normal font	默认字体

表 5.2: 字号。

<code>\tiny</code>	tiny font	<code>\Large</code>	larger font
<code>\scriptsize</code>	very small font	<code>\LARGE</code>	very large font
<code>\footnotesize</code>	quite small font	<code>\huge</code>	huge
<code>\small</code>	small font	<code>\Huge</code>	largest
<code>\normalsize</code>	normal font		
<code>\large</code>	large font		

表 5.3: 标准文档类中的字号大小。

字号	10pt 选项 (默认)	11pt 选项	12pt 选项
<code>\tiny</code>	5pt	6pt	6pt
<code>\scriptsize</code>	7pt	8pt	8pt
<code>\footnotesize</code>	8pt	9pt	10pt
<code>\small</code>	9pt	10pt	10.95pt
<code>\normalsize</code>	10pt	10.95pt	12pt
<code>\large</code>	12pt	12pt	14.4pt
<code>\Large</code>	14.4pt	14.4pt	17.28pt
<code>\LARGE</code>	17.28pt	17.28pt	20.74pt
<code>\huge</code>	20.74pt	20.74pt	24.88pt
<code>\Huge</code>	24.88pt	24.88pt	24.88pt

`\fontsize` 用到两个参数, $\langle size \rangle$ 为字号, $\langle base\ line\ skip \rangle$ 为基础行距。表 5.3 中的命令也都各自设定了与字号对应的基础行距, 大小为字号的 1.2 倍。如果不是在导言区, `\fontsize` 的设定需要 `\selectfont` 命令才能立即生效, 而表 5.2 的字号设定都是立即生效的。

任意设置字号有个问题: L^AT_EX 排版用到的一些老式字体是固定字号的¹, 它们往往无法自动调整成任意的字号大小。如果可能的话, 应当尽量使用表 5.2 中的命令设置字号。

5.1.3 选用字体宏包

尽管到了这里你知道了如何切换粗体、斜体等等, 以及如何改变字号, 但你依然用着 L^AT_EX 默认的那套、由 T_EX 程序的开发者高德纳亲自制作的 Computer Modern 字体。有的人可能很喜欢 Times 或者 Palatino, 或者更好看的字体。这些字体样式的自由设置在 L^AT_EX 里还不太容易。

幸好大部分时候, 许多字体宏包为我们完成了整套配置, 我们可以在调用宏包之后, 照常使用 `\bfseries` 或 `\ttfamily` 等我们熟悉的命令。表 5.4 列出了较为常用的字体宏包, 其中相当多的宏包还配置了数学字体, 或者文本、数学字体兼而有之。更多的字体配置参考 [17, 18]。

5.1.4 xelatex 命令下使用 fontspec 宏包更改字体

xelatex 编译命令能够支持直接调用系统安装的 .ttf 或 .otf 格式字体²。相比于上一小节, 我们有了更多修改字体的余地。

xelatex 命令下支持用户调用字体的宏包是 fontspec。宏包提供了几个设置全局字体的命令, 设置 `\rmfamily` 等对应命令的默认字体³:

```
\setmainfont[ $\langle font\ features \rangle$ ]{ $\langle font\ name \rangle$ }
\setsansfont[ $\langle font\ features \rangle$ ]{ $\langle font\ name \rangle$ }
\setmonofont[ $\langle font\ features \rangle$ ]{ $\langle font\ name \rangle$ }
```

其中 $\langle font\ name \rangle$ 使用字体的文件名 (带扩展名) 或者字体的英文名称。 $\langle font\ features \rangle$ 用来手动配置对应的粗体或斜体, 比如为 Windows 下的无衬线字体 Arial 配置粗体和斜体 (通常情况下自动检测并设置对应的粗体和斜体, 无需手动指定):

```
\setsansfont[BoldFont={Arial Bold}, ItalicFont={Arial Italic}]{Arial}
```

$\langle font\ features \rangle$ 还能配置字体本身的各种特性, 这里不再赘述, 感兴趣的读者请参考 fontspec 宏包的帮助文档。

需要注意的是: fontspec 宏包会覆盖数学字体设置。需要调用表 5.4 中列出的一些数学字体宏包时, 应当在调用 fontspec 宏包时指定 `no-math` 选项。fontspec 宏包可能被其它宏包或文档类 (如 xeCJK、ctex 文档类) 自动调用时, 则在文档开头的 `\documentclass` 命令里指定 `no-math` 选项。

5.1.5 使用 xeCJK 宏包更改中文字体

前文已经介绍过的 xeCJK 宏包使用了和 fontspec 宏包非常类似的语法设置中文字体:

¹不难看出表 5.3 中的许多字号大致呈等比数列, 比例为 1.2; 而 10.95pt 实际上是 $10 \times \sqrt{1.2}$ 。

²Linux 下的 T_EX Live 为了支持系统安装的字体, 需要额外的配置。详见附录 A。

³新版本 fontspec 的命令支持把必选参数 $\langle font\ name \rangle$ 放在可选参数 $\langle font\ features \rangle$ 的前面。

表 5.4: 常见的 L^AT_EX 字体宏包。

lmodern	Latin Modern 字体, 对 Computer Modern 字体的扩展
txfonts	Times 风格的字体宏包
pxfonts	Palation 风格的字体宏包
stix	Times 风格的字体宏包
newtxtext,newtxmath	txfonts 的改进版本, 分别设置文本和数学字体
newpxtext,newpxmath	pxfonts 的改进版本, 分别设置文本和数学字体
mathptmx	psnfss 组件之一, Times 风格, 较为陈旧, 不推荐使用
mathpazo	psnfss 组件之一, Palatino 风格, 较为陈旧, 不推荐使用
ccfonts	Concrete 风格字体
euler	Euler 风格数学字体, 也出自于高德纳之手
fourier	fourier 风格数学字体
arev	Arev 风格的无衬线字体
cmbright	仿 Computer Modern 风格的无衬线字体
libertine	Linux Libertine 衬线字体
droid	Droid Serif/Droid Sans 等
inconsolata	Inconsolata 一款不错的开源等宽字体
sourcesanspro	Source Sans Pro 开源无衬线字体
sourcecodepro	Source Code Pro 开源等宽字体
mathabx	数学符号宏包之一
MnSymbol	数学符号宏包之一
fdsymbol	数学符号宏包之一
mathdesign	配合 Charter/Garamond/Utopia 正文字体的数学字体宏包 (Garamond 字体可能需要单独安装)


```
\setCJKmainfont[⟨font features⟩]{⟨font name⟩}
\setCJKsansfont[⟨font features⟩]{⟨font name⟩}
\setCJKmonofont[⟨font features⟩]{⟨font name⟩}
```

由于中文字体少有对应的粗体或斜体，⟨font features⟩ 里多用其他字体来配置，比如习惯上将宋体的 `BoldFont` 配置为黑体，而 `ItalicFont` 配置为楷体。

5.2 段落格式和间距

5.2.1 长度和长度变量

在前面的一些章节，我们已经见到一些长度和长度变量的用法。本节首先统一介绍长度和长度变量。

长度的数值 ⟨length⟩ 由数字和单位组成。常用的单位如下：

pt	点阵宽度，1/72.27in
bp	点阵宽度，1/72in
in	英寸
cm	厘米
mm	毫米
em	当前字号下大写字母 M 的宽度， 常用于水平距离的设定
ex	当前字号下小写字母 x 的高度， 常用于垂直距离的设定

在一些情况下还会用到可伸缩的“弹性长度”，如 `12pt plus 2pt minus 3pt` 表示基础长度为 `12pt`，可以伸展到 `14pt`，也可以收缩到 `9pt`。也可只定义 `plus` 或者 `minus` 的部分，如 `0pt plus 5pt`。

长度的数值还可以用长度变量的倍数来表达，如 `2.5\parindent` 等。

L^AT_EX 预定义了大量的长度变量用于控制版面格式。如页面宽度和高度、首行缩进、段落间距等。如果需要自定义长度变量，需使用如下命令：

```
\newlength{⟨length command⟩}
```

长度变量可以用 `\setlength` 赋值，或用 `\addtolength` 增加长度：

```
\setlength{⟨length command⟩}{⟨length⟩}
\addtolength{⟨length command⟩}{⟨length⟩}
```

5.2.2 行距

前文中我们提到过 `\fontsize` 命令可以为字号设定对应的行距，但我们很少那么用。更常用的办法是在导言区使用 `\linespread` 命令，

```
\linespread{⟨factor⟩}
```

`\linespread` 命令的参数 ⟨factor⟩ 还并不是最终的行距。前文所叙述的 `\fontsize` 命令为字号设定了对应的基本行距。`\small`、`\large` 等命令也自动设置了基本行距为字号的 1.2 倍。而 `\linespread` 设定的行距相当于在 1.2 倍的基础上乘以 ⟨factor⟩。所以要想设定 1.5 倍行距的话，应当将命令写作 `\linespread{1.25}`。

如果不是在导言区全局修改，而想要局部地改变某个段落的行距，需要用 `\selectfont` 命令使 `\linespread` 命令的改动立即生效：

```
{\linespread{1.67}\selectfont
The baseline skip is set to twice
the font size ( $1.67 \times 1.2 = 2$ ).
Pay attention to the \verb|\par|
command at the end. \par}

In comparison, after the
curly brace has been closed,
everything is back to normal.
```

The baseline skip is set to twice the font size ($1.67 \times 1.2 = 2$). Pay attention to the `\par` command at the end.

In comparison, after the curly brace has been closed, everything is back to normal.

字号的改变是即时生效的，而行距的改变直到文字分段时才生效。如果你想改变某一部分文字的行距，那么不能简单地将文字包含在花括号内。注意下面两个例子中 `\par` 命令的位置（`\par` 相当于分段，见 2.3.1 小节）。

```
{\Large Don't read this!
It is not true.
You can believe me!\par}
```

Don't read this! It is not true.
You can believe me!

```
{\Large This is not true either.
But remember I am a liar.}\par
```

This is not true either. But remember I am a liar.

5.2.3 段落格式

以下长度分别为段落的左缩进、右缩进和首行缩进：

```
\setlength{\leftskip}{20pt}
\setlength{\rightskip}{20pt}
\setlength{\parindent}{2em}
```

它们和设置字号的命令一样，在分段时生效。

L^AT_EX 默认在段落开始时缩进，长度为你用上述命令设置的 `\parindent`。如果你在某一段不想使用缩进，可使用某一段开头使用

```
\noindent
```

命令。相反地，

```
\indent
```

命令强制开启一段首行缩进的段落。多个 `\indent` 命令的效果可以累加。

L^AT_EX 还默认在 `\chapter`、`\section` 等章节命令之后的第一段不缩进⁴。如果如果不习惯这种设定，可以调用 `indentfirst` 宏包：

```
\usepackage{indentfirst}
```

段与段之间的垂直间距为 `\parskip`，如设置段落间距为弹性长度，可在 $0.8ex$ 到 $1.5ex$ 变动：

```
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

⁴ctex 宏包默认按照中文习惯保持第一段的首行缩进。

5.2.4 水平间距

L^AT_EX 默认为单词之间增添了水平间距。我们可以用之前提到的 `\quad` 和 `\qqquad` 命令制造一个额外的间距。但是如果想要得到任意长度的间距，需要用到如下命令：

```
\hspace{<length>}
```

```
This\hspace{1.5cm}is a space  
of 1.5 cm.
```

```
This          is a space of 1.5 cm.
```

`\hspace` 命令生成的间距如果位于一行的开头或末尾，则有可能被“吞掉”。这时可以使用 `\hspace*` 代替 `\hspace` 命令得到不会因断行而消失的水平间距。

命令 `\stretch{<n>}` 生成一个特殊弹性长度，参数 $\langle n \rangle$ 为权重。它的基础长度为零，但可以一直延伸，直到一行内可用的空间都被填满。如果同一行内出现多个 `\stretch{<n>}`，这一行的所有可用空间将以每个 `\stretch` 设置的权重 $\langle n \rangle$ 进行分配。命令 `\fill` 相当于 `\stretch{1}`。

```
x\hspace{\stretch{1}}  
x\hspace{\stretch{3}}  
x\hspace{\fill}x
```

```
x          x          x          x
```

在正文中用 `\hspace` 调节水平间距时，往往使用 `em` 作为单位，它会随字号大小而变：

```
{\Large big\hspace{1em}y}\\  
{\tiny tin\hspace{1em}y}
```

```
big  y  
tin  y
```

5.2.5 垂直间距

在页面中，段落、章节标题、行间公式、列表、浮动体等元素之间的间距是 L^AT_EX 预设的。比如 `\parskip`，默认设置为 0pt plus 1pt。

如果我们想要人为地增加段落之间的垂直间距，可以在**两个段落之间**的位置使用如下命令：

```
\vspace{<length>}
```

`\vspace` 的间距在一页的顶端或底端可能被“吞掉”，类似 `\hspace` 在一行的开头和末尾那样。对应地，`\vspace*` 命令产生不会因断页而消失的垂直间距。`\vspace` 也可用 `\stretch` 设置无限延伸的垂直长度。

在段落内的两行之间增加垂直间距，一般通过给换行命令 `\` 加可选参数，如 `\\[6pt]`。

另外 L^AT_EX 还提供了 `\bigskip`、`\medskip`、`\smallskip` 来增加预定义长度的垂直间距。

```
\parbox[t]{3em}{%  
  文字\par 文字}  
\parbox[t]{3em}{%  
  文字\par\smallskip 文字}  
\parbox[t]{3em}{%  
  文字\par\medskip 文字}  
\parbox[t]{3em}{%  
  文字\par\bigskip 文字}
```

```
文字  文字  文字  文字  
文字  文字  文字  文字
```

5.3 页面和分栏

我们不妨回顾一下第一章介绍的文档类属性。L^AT_EX 允许你通过文档类选项控制纸张的大小 (见表 1.2), 包括 `a4paper`、`letterpaper` 等等, 并配合字号设置了适合的页边距。

控制页边距的参数由图 5.1 里给出的各种长度变量控制 (水平和垂直虚线显示的纸张宽度和高度对应的变量为, `\paperheight` 和 `\paperwidth`)。可以用 `\setlength` 命令修改这些长度变量, 以达到调节页面尺寸和边距的作用; 反之也可以利用这些长度变量来决定排版内容的尺寸, 如设置图片或表格的宽度为 `0.8\textwidth`。

但是, 如果你想要直接设置页边距等参数, 着实是一件麻烦事。我们根据图 5.1 将各个方向的页边距计算公式给出 (以奇数页为例):

$$\begin{aligned}\langle left-margin \rangle &= 1\text{in} + \text{\hoffset} + \text{\oddsidemargin} \\ \langle right-margin \rangle &= \text{\paperwidth} - \langle left-margin \rangle - \text{\textwidth} \\ \langle top-margin \rangle &= 1\text{in} + \text{\voffset} + \text{\topmargin} + \text{\headheight} + \text{\headsep} \\ \langle bottom-margin \rangle &= \text{\paperheight} - \langle top-margin \rangle - \text{\textheight}\end{aligned}$$

如果我们想设置合适的 $\langle left-margin \rangle$ 和 $\langle right-margin \rangle$, 就要靠上述方程组把 `\oddsidemargin` 和 `\textwidth` 等参数解出来!

幸好 `geometry` 宏包能够帮我们完成背后繁杂的计算, 让我们能够用简便的方法设置页面参数。

5.3.1 利用 `geometry` 宏包设置页面参数

`geometry` 宏包的调用方式类似于 `graphicx`, 在 `latex + dvipdfmx` 命令下需要指定选项 `dvipdfm` (注意这里不是 `dvipdfmx`); `pdflatex` 和 `xelatex` 编译命令下不需要。

你既可以调用 `geometry` 宏包然后用其提供的 `\geometry` 命令设置页面参数:

```
\usepackage{geometry}
\geometry{<geometry-settings>}
```

也可以将参数指定为宏包的选项:

```
\usepackage[<geometry-settings>]{geometry}
```

其中 $\langle geometry-settings \rangle$ 多以 $\langle key \rangle = \langle value \rangle$ 的形式组织。

比如, 符合 Microsoft Word 习惯的页面设定是 A4 纸张, 上下边距 1 英寸, 左右边距 1.25 英寸, 于是我们可以通过如下两种等效的方式之一设定页边距:

```
\usepackage[left=1.25in,right=1.25in,%
top=1in,bottom=1in]{geometry}
% or like this:
\usepackage[hmargin=1.25in,vmargin=1in]{geometry}
```

又比如, 需要设定周围的边距一致为 1.25 英寸, 可以用更简单的语法:

```
\usepackage[margin=1.25in]{geometry}
```

对于书籍等双面文档, 习惯上奇数页右边、偶数页左边留出较多的页边距, 而书脊一侧的奇数页左边、偶数页右边页边距较少。我们可以这样设定:

```
\usepackage[inner=1in,outer=1.25in]{geometry}
```

`geometry` 宏包本身也能够修改纸张大小、页眉页脚高度、边注宽度等等参数。更详细的用法不再赘述, 感兴趣的用户可查阅 `geometry` 宏包的帮助文档。

The circle is at 1 inch from the top and left of the page. Dashed lines represent $(\backslash\text{hoffset} + 1\text{ inch})$ and $(\backslash\text{voffset} + 1\text{ inch})$ from the top and left of the page.

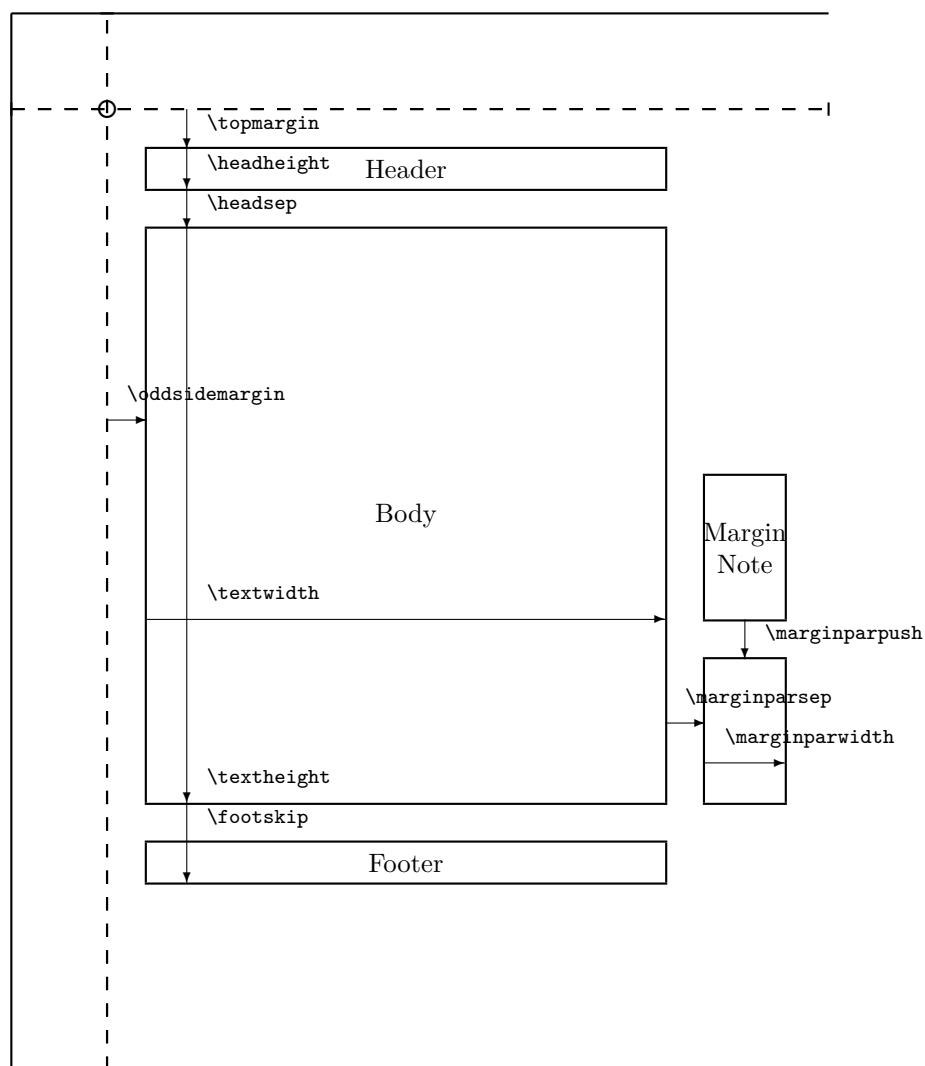


图 5.1: 控制页面的各种参数示意图。

5.3.2 页面内容的垂直对齐

L^AT_EX 默认将页面内容在垂直方向分散对齐。对于有大量图表的文档，许多时候想要做到排版匀称的页面很困难，垂直对齐会造成某些页面的垂直间距过宽。

L^AT_EX 还提供了另一种策略，即将页面内容向顶部对齐，给底部留出高度不一的空白。在导言区或者适合的位置使用以下命令开启顶部对齐的效果：

```
\raggedbottom
```

相反地，`\flushbottom` 命令用于设置页面分散对齐。

5.3.3 分栏

L^AT_EX 支持简单的单栏或双栏排版。标准文档类的全局选项 `onecolumn`、`twocolumn` 可控制全文分单栏或双栏排版。L^AT_EX 也提供了切换单/双栏排版的命令：

```
\onecolumn
\twocolumn[<one-column top material>]
```

`\twocolumn` 支持带一个可选参数，用于排版双栏之上的一部分单栏内容。

切换单/双栏排版时总是会另起一页 (`\clearpage`)。在双栏模式下使用 `\newpage` 会换栏而不是换页；`\clearpage` 则能够换页。

一个比较好用的分栏解决方案是 `multicol`，它提供了简单的 `multicols` 环境（注意不要写成 `multicol` 环境）自动产生分栏，如以下环境将内容分为 3 栏：

```
\begin{multicols}{3}
...
\end{multicols}
```

`multicol` 宏包能够在一页之中切换单栏/多栏，也能处理跨页的分栏，且各栏的高度分布平衡。但代价是在 `multicols` 环境中无法正常使用 `table` 和 `figure` 等浮动体环境，它会直接让浮动体丢失。浮动体只能用跨栏的 `table*` 和 `figure*`，或者用 `float` 宏包提供的 H 选项固定浮动体的位置。

直接在 `multicols` 环境里套用 `\tableofcontents` 等命令生成目录时，无疑会将目录的标题也放进分栏里。有的人可能喜欢跨栏的标题，解决的方案是直接调用 `multitoc` 宏包，它能够正确地生成单栏的标题和多栏的内容，可自行调整分栏数量（默认双栏）。

5.4 页眉页脚

5.4.1 基本的页眉页脚样式

L^AT_EX 中提供了命令 `\pagestyle` 来修改页眉页脚的样式：

```
\pagestyle{<page-style>}
```

另外一个命令只影响当页的页眉页脚样式：

```
\thispagestyle{<page-style>}
```

`<page-style>` 参数为样式的名称，在 L^AT_EX 里预定义了四类样式，见表 5.5。

其中 `headings` 的情况较为复杂：

表 5.5: L^AT_EX 预定义的页眉页脚样式

empty	页眉页脚为空
plain	页眉为空，页脚为页码。(article 和 report 文档类默认；book 文档类的每章第一页也为 plain 格式)
headings	页眉为章节标题和页码，页脚为空。(book 文档类默认)
myheadings	页眉为页码及 \markboth 和 \markright 命令手动指定的内容，页脚为空。

- twoside 选项的 article 文档类，偶数页页眉为页码和节标题，奇数页页眉为小节标题和页码；
- twoside 选项的 book/report 文档类，偶数页页眉为页码和章标题，奇数页页眉为节标题和页码；
- oneseide 选项的 article 文档类，页眉为节标题和页码；
- oneseide 选项的 book/report 文档类，页眉为章标题和页码。

5.4.2 手动更改页眉页脚的内容

对于 headings 或者 myheadings 样式，L^AT_EX 允许用户使用命令手动修改页眉上面的内容，特别是因为使用了 \chapter* 等命令而无法自动生成页眉页脚的情况：

```
\markright{⟨right-mark⟩}
\markboth{⟨left-mark⟩}{⟨right-mark⟩}
```

在（双面排版的）默认情况下，⟨left-mark⟩ 和 ⟨right-mark⟩ 的内容分别预期出现在左页（偶数页）和右页（奇数页）。

事实上 \chapter、\section 等命令内部也使用 \markboth 或者 \markright 写页眉。L^AT_EX 默认在写页眉的时候强制将英文字母变为大写。如果你不喜欢这样，可以尝试以下代码（相关命令的用法参照 8.1 节）⁵：

```
\renewcommand\chaptermark[1]{%
  \markboth{Chapter \thechapter\quad #1}{}}
\renewcommand\sectionmark[1]{%
  \markright{\thesection\quad #1}}
```

其中 \thechapter、\thesection 等命令为章节计数器的数值（详见 8.3 节）。以上代码适用于 report/book 文档类。对于 article 文档类，与两个页眉相关的命令分别为 \sectionmark 和 \subsectionmark。

⁵但是这不能改变页眉页脚的斜体样式，斜体是定义在 headings 样式里的。如果不喜欢斜体，办法之一是在 \markboth 等命令的参数里使用 \normalfont，再使用想要的字体样式命令，或直接尝试使用接下来介绍的 fancyhdr 宏包。

5.4.3 fancyhdr 宏包

fancyhdr 宏包改善了页眉页脚样式的定义方式，允许我们将内容自由安置在页眉和页脚的左、中、右三个位置，还为页眉和页脚各加了一条横线。

fancyhdr 自定义了样式名称 fancy。使用 fancyhdr 宏包定义页眉页脚之前，通常先用 `\pagestyle{fancy}` 调用这个样式。在 fancyhdr 中定义页眉页脚的命令为：

```
\fancyhead[⟨position⟩]{...}
\fancyfoot[⟨position⟩]{...}
```

其中 `⟨position⟩` 为 L（左）/C（中）/R（右）以及与 O（奇数页）/E（偶数页）字母的组合。

我们用一个示例说明 fancyhdr 的用法，这段代码可以用于导言区，它的效果是将章节标题放在和 headings 一致的位置，但使用加粗格式；页码都放在页脚正中；并修改 fancy 格式引入的横线宽度，“去掉”页脚的横线。更多的用法请读者参考 fancyhdr 宏包的帮助文档。

% 导言区部分

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]{\markboth{#1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ #1}}
\fancyhf{} % 清空当前的页眉页脚
\fancyfoot[C]{\bfseries\thepage}
\fancyhead[LO]{\bfseries\rightmark}
\fancyhead[RE]{\bfseries\leftmark}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0pt}
```

源代码 5.1: fancyhdr 宏包的使用方法示例。

第六章 特色工具和功能

本章介绍一些特色的 L^AT_EX 辅助功能。前两个功能 BibT_EX 和 makeindex 依靠一些辅助程序自动生成参考文献、索引等；之后的使用颜色、超链接等则令我们生成美观易用的电子文档。

6.1 参考文献和 BibT_EX 工具

6.1.1 基本的参考文献和引用

L^AT_EX 提供的参考文献和引用方式比较原始，需要用户自行书写参考文献列表。不同学术论文对参考文献列表的格式要求不一样，自行书写是一件极其头疼的事情。相关的命令我们只作最简单的介绍。

L^AT_EX 提供了最基本的 `\cite` 命令用于在正文中引用参考文献：

```
\cite{<citation>}
```

`\cite` 带一个可选参数，为引用的编号后加上额外的内容，如 `\cite[page 22]{<pa>}` 可能得到形如 [13, page 22] 这样的引用。

参考文献由 `thebibliography` 环境包裹。每条参考文献由 `\bibitem` 开头，其后是参考文献本身的内容：

```
\bibitem[<item number>]{<citation>}
```

其中 `<citation>` 是 `\cite` 用以引用的一个标签，类似交叉引用里的 `\label`。`<item number>` 自定义参考文献的序号，如果省略，则按自然排序给定序号。

`thebibliography` 环境带一个参数，用以设定 `\cite` 命令生成的引用编号的最大宽度，如 99 意味着不超过两位数字。通常设定为与参考文献的数目一致。`thebibliography` 环境自动生成不带编号的一章（report / book 文档类）或一节（article 文档类）。

以下为一个使用 `thebibliography` 排版参考文献的例子：

```
\documentclass{article}
\begin{document}
\section{Introduction}
Part1~\cite{pa} has proposed that \ldots

\begin{thebibliography}{99}
\bibitem{pa} H.~Part1: \emph{German \TeX},
  TUGboat Volume~9, Issue~1 (1988)
\end{thebibliography}
\end{document}
```

6.1.2 BibTeX 数据库

BibTeX 是最为流行的参考文献数据组织格式之一。它的出现让我们摆脱手写参考文献条目的麻烦。我们还可以通过参考文献格式的支持，让同一份 BibTeX 数据库生成不同格式的参考文献列表。我们首先简单介绍 BibTeX 数据库，之后介绍如何将数据库利用在 L^AT_EX 中。

BibTeX 数据库以 .bib 作为扩展名，其内容是若干个文献条目，每个条目的格式为：

```
@<type>{<citation>,
  <key1> = {<value1>},
  <key2> = {<value2>},
  ...
}
```

其中 <type> 为参考文献的类型，如 article 为学术论文，book 为书籍，incollection 为论文集集中的某一篇，等等。<citation> 为 \cite 命令所用的参考文献的标签。在 <citation> 之后为条目里的各个数据项，以 <key> = {<value>} 的形式组织。

我们在此简单列举学术论文里使用较多的 BibTeX 文献条目格式。所有条目格式参考 [CTAN: //biblio/bibtex/base/btxdoc.pdf](http://biblio/bibtex/base/btxdoc.pdf)。

article 学术论文，必需数据项有 author, title, journal, year; 可选数据项包括 volume, issue, number, pages, doi 等；

book 书籍，必需数据项有 author/editor, title, publisher, year; 可选数据项包括 volume/number, series, address 等；

incollection 论文集集中的一篇，必需数据项有 author, title, booktitle, publisher, year; 可选数据项包括 editor, volume/number, chapter, pages, address 等；

inbook 书中的一章，必需数据项有 author/editor, title, chapter/pages, publisher, year; 可选数据项包括 volume/number, series, address 等。

多数时候，我们无需自己手写 BibTeX 文献条目。从 Google Scholar 或者期刊/数据库的网站上都能够导出 BibTeX 文献条目，老牌的文献管理软件 EndNote 也支持生成 BibTeX 格式的数据库。开源软件 JabRef 甚至支持 BibTeX 文献条目的导入、导出和管理。

6.1.3 使用 BibTeX 排版参考文献

现在我们来看如何利用 BibTeX 数据库生成参考文献和引用。

第一步：我们当然需要一份 BibTeX 数据库，假设起名为 books.bib，和我们的 L^AT_EX 源代码一般位于同一个目录下。

第二步：在源代码中添加必要的命令。假设源代码的名称为 demo.tex。首先需要的是在导言区设定参考文献的格式：

```
\bibliographystyle{<bst-name>}
```

其中 <bst-name> 为格式文件的名称。BibTeX 提供了几个预定义的格式，如 plain, unsrt, alpha 等。一些学术论文的模板会提供自有的格式文件，以 .bst 作为扩展名，同样和 L^AT_EX 源代码放在同一个目录下。

其次就是在正文中引用参考文献了。BibTeX 程序在生成参考文献列表的时候，通常只列出你用 \cite 命令引用的那些。如果需要列出没有被引用的文献，则需要 \nocite{<citation>} 命令；而 \nocite{*} 则让所有未引用的参考文献都列出。

```
@book{Lamport1994,
  title = {{\LaTeX} A Document Preparation System},
  author = {Leslie Lamport},
  publisher = {Addison-Wesley},
  address = {Reading, Massachusetts},
  year = {1994},
  edition = {2nd}
}

@book{Mittelbach2004,
  title = {The {\LaTeX} Companion},
  author = {Frank Mittelbach and Michel Goossens and
    Johannes Braams and David Carlisle and Chris Rowley},
  publisher = {Addison-Wesley},
  address = {Reading, Massachusetts},
  year = {2004},
  edition = {2nd}
}
```

源代码 6.1: BibTeX 数据库示例 books.bib。

再次，在你需要列出参考文献的位置，使用 `\bibliography` 命令代替 `thebibliography` 环境：

```
\bibliography{<bib-name>}
```

其中 `<bib-name>` 是 BibTeX 数据库的文件名，**不要带 .bib 扩展名**。

```
\documentclass{article}
\bibliographystyle{plain}
\begin{document}
\section{Some words}
Some excellent books, for example, \cite{Lamport1994}
and \cite{Mittelbach2004} \ldots

\bibliography{books}
\end{document}
```

源代码 6.2: 利用 books.bib 生成参考文献的源代码 demo.tex。

注意：`\bibliographystyle` 和 `\bibliography` 命令缺一不可，没有这两个命令，使用 BibTeX 生成参考文献列表的时候会报错。

第三步：写好了以上两个文件之后，我们就可以开始编译了。

1. 首先使用 `pdflatex` 或 `xelatex` 等命令编译 L^AT_EX 源代码 `demo.tex`；
2. 接下来用 `bibtex` 命令处理 `demo.aux` 文件记录的参考文献格式、引用条目等信息。`bibtex` 命令处理完毕后会生成 `demo.bbl` 文件，内容就是一个 `thebibliography` 环境；
3. 再使用 `pdflatex` 或 `xelatex` 等命令把源代码 `demo.tex` 编译**两遍**，读入参考文献并正确生成引用。

整个过程使用的命令如下（可以略去扩展名）：

```
pdflatex demo
bibtex demo
pdflatex demo
pdflatex demo
```

使用 `latex + dvipdfmx` 命令编译时，则 `dvipdfmx` 命令放在最后，相当于先后使用 `latex`, `bibtex`, `latex`, `latex`, `dvipdfmx`。

6.1.4 natbib 宏包

时下许多学术期刊比较喜欢使用人名——年份的引用方式，形如 (*Alice et al.*, 2005)。natbib 宏包提供了对这种“自然”引用方式的处理。

除了 `\cite` 之外，natbib 宏包在正文中支持两种引用方式：

```
\citep{<citation>}
\citet{<citation>}
```

它们分别生成形如 (*Alice et al.*, 2005) 和 *Alice et al.*(2005) 的人名——年份引用。

natbib 宏包同样也支持数字引用，并且支持将引用的序号压缩，例如：

```
\usepackage[numbers,sort&compress]{natbib}
```

调用 natbib 宏包时指定以上选项后，连续引用多篇文献时，会生成形如 (3-7) 的引用而不是 (3, 4, 5, 6, 7)。

natbib 宏包还有更多选项和用法，比如默认的引用是用小括号包裹的，可以为宏包添加 `square` 选项改为中括号；再比如 `\citep` 命令也支持可选参数，为引用前后都添加额外内容。这里不再赘述，请参考 natbib 宏包的帮助文档。

6.2 索引和 makeindex 工具

书籍和大文档通常用索引来归纳关键词，方便用户查阅。L^AT_EX 借助配套的 makeindex 程序完成对索引的排版。

6.2.1 使用 makeindex 工具的方法

要使用索引，须经过这么几个步骤（仍设源代码名为 `demo.tex`）：

第一步，在 L^AT_EX 源代码的导言区调用 `makeidx` 宏包，并使用 `\makeindex` 命令开启索引的收集：

```
\usepackage{makeidx}
\makeindex
```

第二步，在正文中需要索引的地方使用 `\index` 命令。`\index` 命令的参数写法详见下一小节；并在需要输出索引的地方（如所有章节之后）使用 `\printindex` 命令。

第三步，编译过程：

1. 首先用 `pdflatex` 等命令编译源代码 `demo.tex`。编译过程中产生索引记录文件 `demo.idx`；
2. 用 `makeindex` 程序处理 `demo.idx`，生成用于排版的索引列表文件 `demo.ind`；
3. 再次编译源代码 `demo.tex`，正确生成索引列表。

表 6.1: 索引项的写法列表。

举例	索引项	备注
普通索引		
hello	hello, 1	普通索引
分级索引，以 ! 分隔，最多支持三级		
hello	hello, 1	一级索引
hello!Peter	Peter, 3	二级索引
hello!Peter!Jack	Jack, 3	三级索引
格式化索引，形式为 $\langle\alpha\rangle@{\langle format\rangle}$		
$\langle\alpha\rangle$ 为纯字母，用来排序		
$\langle format\rangle$ 为索引的格式，可以包括 L ^A T _E X 代码和简单的公式		
Möbius@M{"obius	Möbius, 2	输出重音
alpha@\${\alpha}\$	α , 7	输出公式
bold@{\textbf{bold}}	bold , 12	输出粗体
页码范围		
morning (morning, 6-7	范围索引的开头
morning)		范围索引的结尾
格式化索引页码		
Jenny textbf	Jenny, 3	调用 \textbf 加粗页码
Joe see{Jenny}	Joe, <i>see</i> Jenny	调用 \see 生成特殊形式
Joe seealso{Jenny}	Joe, <i>see also</i> Jenny	调用 \seealso 生成特殊形式

6.2.2 索引项的写法

添加索引项的命令为：

```
\index{<index entry>}
```

其中 $\langle index\ entry\rangle$ 为索引项，写法由表 6.1 汇总。其中 !、@ 和 | 为特殊符号，如果要向索引项直接输出这些符号，需要加前缀 "；而 " 需要输入两个引号 "" 才能输出到索引项。

读者可以钻研一下下面给出的一个较为复杂的，结合多级索引、索引格式、页码格式等的示例。但在自己使用时，最好还是遵循“简单的就是最好的”原则，尽量使用表 6.1 中的写法。

```
Test index.
\index{Test@{\textsf{"Test"}}|(\textbf{
\index{Test@{\textsf{"Test"}}!sub@"|sub"|see{Test}}
\newpage
Test index.
\index{Test@{\textsf{"Test"}}|)\textbf{
```

6.3 使用颜色

L^AT_EX 原生不支持颜色，它依赖 color 宏包或者 xcolor 宏包，在编译过程中给 PDF 输出生成颜色的特殊指令。

6.3.1 颜色的表达方式

调用 `color` 或 `xcolor` 宏包后，我们就可以用如下命令切换颜色：

```
\color[⟨color-mode⟩]{⟨code⟩}
\color{⟨color-name⟩}
```

颜色的表达方式有两种，其一是使用色彩模型和色彩代码，代码用 0 ~ 1 的数字代表成分的比例。`color` 宏包支持 `rgb`、`cmymk` 和 `gray` 模型，`xcolor` 支持更多的模型如 `hsb` 等。

```
\large\sffamily
{\color[gray]{0.6}
  60\% 灰色} \\\
{\color[rgb]{0,1,1}
  青色}
```

60% 灰色

青色

其二是直接使用名称代表颜色，前提是已经定义了颜色名称（没定义的话会报错）：

```
\large\sffamily
{\color{red} 红色} \\\
{\color{blue} 蓝色}
```

红色

蓝色

`color` 宏包仅定义了 8 种颜色名称，`xcolor` 补充了一些，总共有 19 种，见表 6.2。

表 6.2: `color` 和 `xcolor` 宏包可用的颜色名称。

基本的 8 种颜色名称：			
black 	red 	green 	blue 
white 	cyan 	magenta 	yellow 
<code>xcolor</code> 额外可用的颜色名称：			
darkgray 	gray 	lightgray 	
brown 	olive 	orange 	lime 
purple 	teal 	violet 	pink 

`xcolor` 还支持将颜色通过表达式混合或互补：

```
\large\sffamily
{\color{red!40} 40\% 红色} \\\
{\color{blue} 蓝色}
\color{blue!50!black} 蓝黑
\color{black} 黑色} \\\
{\color{-red} 红色的互补色}
```

40% 红色

蓝色 蓝黑 黑色

红色的互补色

我们还可以通过命令自定义颜色名称，注意这里的 `⟨color-mode⟩` 是必选参数：

```
\definecolor{⟨color-name⟩}{⟨color-mode⟩}{⟨code⟩}
```

如果调用 `color` 或 `xcolor` 宏包时使用 `dvipsnames` 选项，就有额外的 68 种颜色名称可用。`xcolor` 宏包还支持通过其它选项载入更多颜色名称。限于篇幅不展开介绍，详情请参考 `xcolor` 宏包的手册。

6.3.2 带颜色的文本和盒子

原始的 `\color` 命令类似于字体命令 `\bfseries`，它使之后排版的内容全部变成指定的颜色，所以直接使用时通常要加花括号分组。`color` / `xcolor` 宏包都定义了一些方便用户使用的带颜色元素。

输入带颜色的文本可以用类似 `\textbf` 的命令：

```
\textcolor[⟨color-mode⟩]{⟨code⟩}{⟨text⟩}
\textcolor{⟨color-name⟩}{⟨text⟩}
```

以下命令构造一个带背景色的盒子，`⟨material⟩` 为盒子中的内容：

```
\colorbox[⟨color-mode⟩]{⟨code⟩}{⟨material⟩}
\colorbox{⟨color-name⟩}{⟨material⟩}
```

以下命令构造一个带背景色和有色边框的盒子，`⟨fcode⟩` 或 `⟨fcolor-name⟩` 用于设置边框颜色：

```
\fcolorbox[⟨color-mode⟩]{⟨fcode⟩}{⟨code⟩}{⟨material⟩}
\fcolorbox{⟨fcolor-name⟩}{⟨color-name⟩}{⟨material⟩}
```

```
\sffamily
文字用\textcolor{red}{红色}强调\
\colorbox[gray]{0.95}{浅灰色背景} \
\fcolorbox{blue}{yellow}{%
\textcolor{blue}{蓝色边框+文字，%
黄色背景}
}
```

文字用红色强调
浅灰色背景
蓝色边框 + 文字，黄色背景

`\fcolorbox` 也可以像 3.8.2 小节里的 `\fbox` 那样调节 `\fboxrule` 和 `\fboxsep`。这里不再示例。

6.4 使用超链接

PDF 文档格式是现今最流行的电子文档格式，而电子文档最实用的需求之一就是链接功能。 \LaTeX 中实现这一功能的是 `hyperref` 宏包。

6.4.1 `hyperref` 宏包

`hyperref` 宏包涉及到的链接遍布 \LaTeX 的每一个角落——目录、引用、脚注、索引、参考文献等等都被封装成链接。但这也使得它与其它宏包的冲突机会大大增加，虽然宏包已经尽力解决各方面的兼容性，但仍不能面面俱到。为减少冲突的可能性，习惯上将 `hyperref` 宏包放在其它宏包之后调用。

与 `graphicx` 宏包类似，`latex + dvipdfmx` 命令下调用 `hyperref` 宏包时，需要指定选项 `dvipdfmx`；而 `pdflatex` 或 `xelatex` 命令下不需要。

`hyperref` 宏包提供了命令 `\hypersetup` 配置各种选项，或者也可以作为宏包选项，在调用宏包时使用：

```
\hypersetup{⟨option1⟩,⟨option2⟩={value},...}
\usepackage[⟨option1⟩,⟨option2⟩={value},...]{hyperref}
```


6.4.2 超链接

hyperref 宏包提供了直接书写超链接的命令，用于在 PDF 中生成 URL：

```
\url{<url>}
\nolinkurl{<url>}
```

\url 和 \nolinkurl 都生成可以点击的 URL，区别是前者有彩色，后者没有。在 \url 命令中作为参数的 URL 里，可直接输入如 %、& 这样的特殊符号。

我们也可以像网页一样，把一段文字赋予其“超链接”的作用：

```
\href{<url>}{<text>}
```

```
\url{http://wikipedia.org} \\\nolinkurl{http://wikipedia.org} \\\href{http://wikipedia.org}{Wiki}
```

```
http://wikipedia.org
http://wikipedia.org
Wiki
```

使用 hyperref 宏包后，文档中所有的引用、参考文献、索引等等都转换为超链接。用户也可对某个 \label 命令定义的标签 <label> 作超链接（注意这里的 <label> 虽然用了方括号，但是是必填的¹）：

```
\hyperref[<label>]{<text>}
```

默认的超链接在文字外边加上一个带颜色的方框（在打印 PDF 时边框不会打印），可使用 colorlinks 选项修改为将文字本身加上颜色，或修改 pdfborder 选项的参数；hidelinks 则令超链接既不变色也不加框。

```
\hypersetup{hidelinks}
% or:
\hypersetup{pdfborder={0 0 0}}
```

6.4.3 PDF 书签

hyperref 宏包另一个强大的功能是为 PDF 生成书签。对于章节命令 \chapter、\section 等，默认情况下会为 PDF 自动生成书签。和交叉引用、索引等类似，生成书签也需要多次编译源代码，第一次编译将书签记录写入 .out 文件，第二次编译才正确生成书签。

书签的一些属性见表 6.3。在 latex + dvipdfmx 或 pdflatex 命令下使用 ctex 宏包或 CJK 宏包时，为了正确生成中文书签而不出现乱码，需要额外的设置，甚至繁琐的工序（这也是我们推荐使用 xelatex 命令处理中文的原因）。

hyperref 还提供了手动生成书签的命令：

```
\pdfbookmark[<level>]{<bookmark>}{<anchor>}
```

<bookmark> 为书签名称，<anchor> 为书签项使用的锚点（类似交叉引用的标签）。可选参数 <level> 为书签的层级，默认为 0。

章节命令里往往有 L^AT_EX 命令甚至数学公式，而 PDF 书签是纯文本，对命令和公式的处理很困难，有出错的风险。hyperref 宏包已经为我们处理了许多常见命令，如 \LaTeX 和字体命令 \textbf 等，其它一些情况就要使用在 \section 等命令中使用如下命令，分别提供 L^AT_EX 代码和 PDF 书签可用的纯文本：

¹不带方括号的 \hyperref 命令带四个必选参数，前三个组成特殊格式的超链接，最后一个同 <text>。


```
\texorpdfstring{\LaTeX code}\{PDF bookmark text\}
```

比如在章节名称里使用公式 $E = mc^2$ ，而书签使用字符 E=mc^2:

```
\section{质能公式 \texorpdfstring{$E=mc^2$}{E=mc\textasciicircum 2}}
```

6.4.4 PDF 文档属性

hyperref 宏包还提供了一些选项用于改变 PDF 文档的属性，部分见表 6.3。

表 6.3: hyperref 宏包提供的设置。

选项	默认值	含义
colorlinks= $\langle true false \rangle$	false	设置为 true 为链接文字带颜色，反之加上带颜色的边框
hidelinks		取消链接的颜色和边框
pdfborder= $\{\langle n \rangle \langle n \rangle \langle n \rangle\}$	0 0 1	超链接边框设置，设为 0 0 0 可取消边框
bookmarks= $\langle true false \rangle$	true	是否生成书签
bookmarksopen= $\langle true false \rangle$	false	是否展开书签
bookmarksnumbered= $\langle true false \rangle$	false	书签是否带章节编号
CJKbookmarks= $\langle true false \rangle$	false	使用 CJK 宏包/ GBK 编码时必须指定的选项，在第一次编译后需要将生成的 .out 文件用工具处理编码
unicode		使用 CJKutf8 宏包/ UTF-8 编码时必须指定的选项
pdftitle= $\langle string \rangle$	空	标题
pdfauthor= $\langle string \rangle$	空	作者
pdfsubject= $\langle string \rangle$	空	主题
pdfkeywords= $\langle string \rangle$	空	关键词
pdfstartview= $\langle Fit FitH FitV \rangle$	Fit	设置 PDF 页面以适合页面/适合宽度/适合高度等方式显示，默认为适合页面

第七章 绘图功能

除了排版文字， \LaTeX 也支持用代码表示图形。不同的扩展已经极大地丰富了 \LaTeX 的图形功能，`TikZ` 就是其中之一。本章将带你了解一些基本的绘图功能。

一些特殊的绘图，如交换图、树状图甚至分子式和电路图也能够通过代码绘制，不过其复杂程度已经超出本手册范围，有兴趣的读者可以查阅一些帮助手册，或者在互联网寻求帮助。

7.1 绘图语言简介

\LaTeX 提供了原始的 `picture` 环境，能够绘制一些基本的图形如点、线、矩形、圆、Bézier 曲线等等，不过受制于 \LaTeX 本身，它的绘图功能极为有限，效果也往往不够美观。

现在流行的绘图代码有以下几种：

- **PSTricks**

基于 PostScript 绘图方式的宏包，具有优秀的绘图能力。它对老式的 `latex + dvips` 编译命令支持最好，而现在的几种编译命令下使用起来都不够方便。

- **TikZ & pgf**

德国的 Till Tantau 在开发著名的 \LaTeX 幻灯片文档类 `beamer` 时一并开发了绘图宏包 `pgf`，以令绘图语言能够在不同的编译方式下使用，如 `pdflatex` 和 `xelatex` 等。`TikZ` 是在 `pgf` 基础上的一个封装，提供了方便的绘图语言，绘图能力不输 `PSTricks`。

- **METAPOST & Asymptote**

`METAPOST` 脱胎于高德纳为 \TeX 配套开发的字体生成程序 `METAFONT`，具有优秀的绘图能力，并能够调用 \TeX 引擎向图片中插入文字和公式。`Asymptote` 在 `METAPOST` 的基础上更进一步，具有一定的类似 C 语言的编程能力，支持三维图形的绘制。

它们往往需要把代码写在单独的文件里，用特定的工具去编译，也可以借助特殊的宏包在 \LaTeX 代码里直接使用。

本手册将介绍 `TikZ` 绘图语言里最基本的部分。`TikZ` 还支持各种自定义的扩展，基于 `TikZ` 的专门用途的绘图宏包也不胜枚举，其复杂程度已远远超出入门手册的范围（`TikZ` 的帮助文档有上千页之厚）。对此感兴趣的读者需要自行查阅帮助手册，或者到互联网上参考现成的范例。

7.2 TikZ

在导言区调用 `tikz` 宏包，就可以用以下命令和环境使用 `TikZ` 的绘图功能了¹：

¹`latex + dvipdfmx` 编译方式要在 `tikz` 宏包之前加载 `graphicx` 宏包并指定 `dvipdfmx` 选项。

```

\tikz[...] <tikz code>;

\begin{tikzpicture}[...]
<tikz code 1>;
<tikz code 2>;
...
\end{tikzpicture}

```

前一种用法为 `\tikz` 带单条绘图命令，以分号结束，一般用于在文字之间插入简单的图形；后一种用法较为常见，使用多条绘图命令，可以在 `figure` 等浮动体中使用。

7.2.1 TikZ 坐标和路径

TikZ 用直角坐标系或者极坐标系描述点的位置。

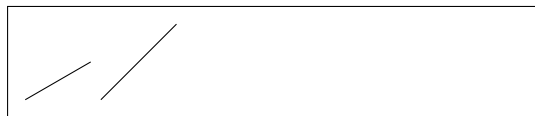
- 直角坐标下，点的位置写作 $(\langle x \rangle, \langle y \rangle)$ ，坐标 $\langle x \rangle$ 和 $\langle y \rangle$ 可以用 L^AT_EX 支持的任意单位表示，缺省为 `cm`；
- 极坐标下，点的位置写作 $(\langle \theta \rangle : \langle r \rangle)$ 。 θ 为极角，单位是度。

我们还可以为某个点命名：`\coordinate (A) at (<coordinate>)` 然后就可以使用 `(A)` 作为点的位置了。

```

\begin{tikzpicture}
\draw (0,0) -- (30:1);
\draw (1,0) -- (2,1);
\end{tikzpicture}

```



TikZ 最基本的路径为两点之间连线，如 $(\langle x_1 \rangle, \langle y_1 \rangle) -- (\langle x_2 \rangle, \langle y_2 \rangle)$ ，可以连用表示多个连线（折线）。连续使用连线时，可以使用 `cycle` 作为最后一个点，生成闭合的路径。

```

\begin{tikzpicture}
\draw (0,0) -- (1,1)
      -- (2,0) -- cycle;
\end{tikzpicture}

```



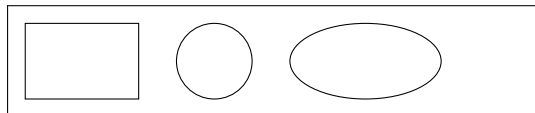
其它常用的路径还包括：

- 矩形、圆和椭圆：

```

\begin{tikzpicture}
\draw (0,0) rectangle (1.5,1);
\draw (2.5,0.5) circle [radius=0.5];
\draw (4.5,0.5) ellipse
[x radius=1,y radius=0.5];
\end{tikzpicture}

```

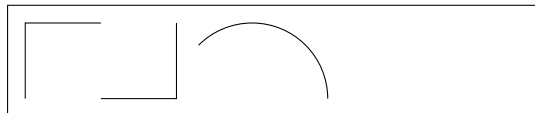


- 直角、圆弧：

```

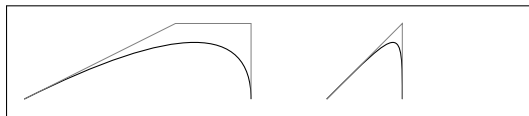
\begin{tikzpicture}
\draw (0,0) |- (1,1);
\draw (1,0) -| (2,1);
\draw (4,0) arc (0:135:1);
\end{tikzpicture}

```



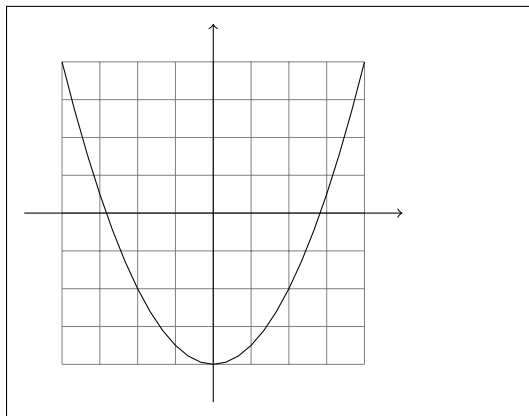
- 二次和三次 Bézier 曲线，分别使用一个和两个控制点：

```
\begin{tikzpicture}
\draw (0,0) .. controls
  (2,1) and (3,1) .. (3,0);
\draw (4,0) .. controls
  (5,1) .. (5,0);
\draw[help lines] (0,0)
  -- (2,1) -- (3,1) -- (3,0)
  (4,0) -- (5,1) -- (5,0);
\end{tikzpicture}
```



- 网格、函数图像，网格可用 `step` 选项控制网格大小，函数图像用 `domain` 选项控制定义域：

```
\begin{tikzpicture}
\draw[help lines,step=0.5]
  (-2,-2) grid (2,2);
\draw[->] (-2.5,0) -- (2.5,0);
\draw[->] (0,-2.5) -- (0,2.5);
\draw[domain=-2:2]
  plot(\x,{\x*\x-2});
\end{tikzpicture}
```



7.2.2 TikZ 绘图命令和参数

除了 `\draw` 命令之外，TikZ 还提供了 `\fill` 命令用来填充图形，`\filldraw` 命令则同时填充和描边。除了矩形、圆等现成的闭合图形外，`\fill` 和 `\filldraw` 命令也能够填充人为构造的闭合路径。

```
\draw[...] <path>; \fill[...] <path>;
\filldraw[...] <path>;
```

绘图参数可作为可选参数用在 `tikzpicture` 环境或 `\tikz` 命令时，参数会影响到所有具体的绘图命令；用在单个绘图命令 `\draw`、`\filldraw` 等时，只对这个命令起效。

TikZ 有数不清的绘图参数，这些参数令 TikZ 能够绘制丰富多彩的图像，同时也令 TikZ 易学难精。本手册仅总结常用的一些绘图参数，见表 7.1。

7.2.3 TikZ 文字结点

TikZ 用 `\node` 命令绘制文字结点：

```
\node[<options>] (<name>) at (<coordinate>) {\text};
```

```
\begin{tikzpicture}
\node (A) at (0,0) {A};
\node (B) at (1,0) {B};
\node (C) at (60:1) {C};
\draw (A) -- (B) -- (C) -- (A);
\end{tikzpicture}
```

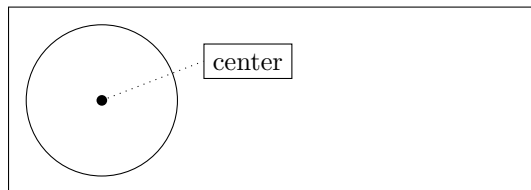


表 7.1: TikZ 常用的一些绘图参数。

<ul style="list-style-type: none"> • <code>color=<color></code> 为线条 (<code>\draw</code>) 或填充 (<code>\fill</code>) 指定颜色, <code><color></code> 使用颜色名或是 <code>xcolor</code> 的混合颜色语法。往往可以不写 <code>color=</code> 直接写颜色名称。 • <code>fill=<color> / draw=<color></code> 分别给 <code>\filldraw</code> 指定填充和描边的颜色。也可分别给 <code>\fill</code> 和 <code>\draw</code> 命令使用。不带参数直接使用 <code>fill</code> 和 <code>draw</code>, 相当于用默认颜色。
<ul style="list-style-type: none"> • <code>line width=<length></code> 指定线条粗细为 <code><width></code>。默认为普通线宽 0.4pt。 • <code>thin / semithick / thick / ...</code> 指定线条粗细为预定义的某个类型, 默认为 <code>thin</code>。总共有七种预定义的类型: <code>ultra thin</code>, <code>very thin</code>, <code>thin</code>, <code>semithick</code>, <code>thick</code>, <code>very thick</code>, <code>ultra thick</code>。 • <code>help lines</code> 指定线条为辅助线, 相当于 <code>line width=0.2pt,gray</code>。 • <code>solid / dashed / dotted / dash dot / dash dot dot / ...</code> 指定线条类型 (实线、虚线等)。 • <code>rounded corners</code> 将路径转向处绘制成圆角。可写成 <code>rounded corners=<radius></code> 使用给定的半径。
<ul style="list-style-type: none"> • <code>-> / -< / -to / -latex / -stealth / ...</code> 指定路径终点的箭头种类。 • <code><- / >- / to- / latex- / -stealth / ...</code> 指定路径起点的箭头种类。起点和终点的箭头可以搭配, 如 <code><-></code> 或者 <code>latex-to</code> 等。
<ul style="list-style-type: none"> • <code>scale=<scale></code> 指定整个图像或某个路径的缩放比例。 • <code>xshift=<length> / yshift=<length></code> 指定整个图像或某个路径相对于原位置的水平/垂直位移。 • <code>rotate=<angle></code> 指定整个图像或某个路径旋转一定角度。

`\node` 命令不仅为文字结点的位置命名（类似 `\coordinate` 命令），在 `\draw` 等命令中还可以使用某个结点的相对位置，以“东南西北”的方式命名：

```
\begin{tikzpicture}
\draw (0,0) circle[radius=1];
\fill (0,0) circle[radius=2pt];
\node[draw] (P) at (15:2) {center};
\draw[dotted] (0,0) -- (P.west);
\end{tikzpicture}
```



另一种用法是在 `\draw` 等命令的路径中使用 `node`，不仅可以对某个位置标记节点，还能够对线标记：

```
\begin{tikzpicture}
\draw (2,1.5) node[above] {$A$}
-- node[above left] {$c$}
(0,0) node[below left] {$B$}
-- node[below] {$a$}
(2.5,0) node[below right] {$C$}
-- node[above right] {$b$}
cycle;
\end{tikzpicture}
```

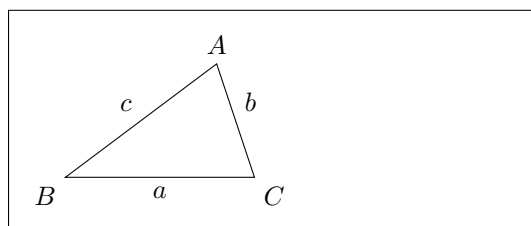


表 7.1 中的参数可用于 `\node` 命令的配置。除此之外，`\node` 还有一些特定的参数，见表 7.2。

表 7.2: TikZ 结点使用的一些绘图参数。

-
- `anchor=<position>`
指定结点的某个角落 `<position>` 位于给定的位置 `<coordinate>`。
 - `above / below / left / right / above left / ...`
`anchor=<position>` 的等效写法，`above` 相当于 `anchor=south`，以此类推。
 - `shape=<shape>`
结点的形状，默认可用 `rectangle` 和 `circle`，可省略 `shape=` 直接写。在导言区使用命令 `\usetikzlibrary{shapes.geometric}` 可用更多的形状。
 - `inner sep=<length> / outer sep=<length>`
结点边界向外和向内的额外距离。
 - `minimum size=<length> / minimum height=<length> / minimum width=<length>`
结点的最小大小或最小高度/宽度。
 - `text=<color>`
结点文字的颜色。
 - `node font=`
结点文字的字体，形如 `\bfseries` 或 `\itshape` 等。
-

除了 `\node` 命令之外，`\coordinate` 也可以通过参数为某个位置添加文字。我们最后举一个较为复杂的例子，综合前面介绍过的各种路径、形状、文字结点和参数设置。

```

\begin{tikzpicture}
\draw[-stealth,line width=0.2pt] (-0.5,0) -- (4.5,0);
\draw[-stealth,line width=0.2pt] (0,-0.5) -- (0,2.5);
\coordinate (a) at (0.5,1.9);
\coordinate (b) at (4,1.2);
\coordinate[label=below:$a$] (a0) at (0.5,0);
\coordinate[label=below:$b$] (b0) at (4,0);
\filldraw[fill=gray!20,draw,thick]
  (a0) -- (a) .. controls (1,2.8) and (2.7,0.4) .. (b) -- (b0) -- cycle;
\node[above right,outer sep=0.2cm, rounded corners,
  fill=green!20,draw=gray,text=blue!60!black,scale=0.6]
  at (b) {$\displaystyle \int_a^b f(x)\,\mathrm{d}x = F(b) - F(a)$};
\end{tikzpicture}

```

源代码 7.1: TikZ 绘图示例源代码。

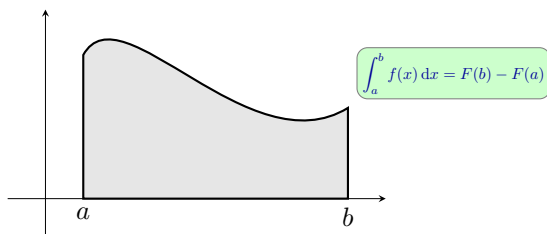


图 7.1: TikZ 绘图示例。

第八章 自定义 L^AT_EX 命令和功能

读到这一章之前，如果你确保掌握了前几章的知识并熟练运用，你已经能制作出内容和形式都相当丰富的文档了。但你可能还不会满足：我要如何制作一个简单但像样的毕业论文/书籍/简历模板，每次可以直接套用，而不是再在导言区写一堆代码？

本章的内容将有助于你实现这一个目标，让你能编写可重复利用的模块——宏包和文档类，并在其中自己定义命令和环境。不过作为入门手册，这些知识仍是不全面的。如果你不满足于此，需要参考更多资料，比如 [2, 9]。

8.1 自定义命令和环境

你也许已经意识到了，在本手册中介绍的所有命令都被包含在一个带颜色的矩形背景框中。笔者并没有直接使用基础的 L^AT_EX 命令来实现这个效果，而是创建了一个**宏包**，并在其中定义了笔者所需要的命令和环境。现在我只需写成这样简单的形式：

```
\begin{command}  
\cmd{dum}  
\end{command}
```

```
\dum
```

这个例子中使用了一个新的环境 `command`。这个环境负责给命令代码加上一个带颜色的矩形背景框。同时还使用了一个命令：`\cmd`，这个命令负责输出命令的名字，包括前面的反斜杠。

一旦笔者想要修改命令代码的央视，比如更换颜色、加边框等等，可以通过改变 `command` 环境的定义来很容易地创建新的外观，而不是挨个修改每个命令示例。

8.1.1 定义新命令

使用如下命令可以定义你自己的命令：

```
\newcommand{\<name>}[<num>]{<definition>}
```

基本上，这个命令有两个参数，第一个 `<name>` 是你想要建立的命令的名称（带反斜杠），第二个 `<definition>` 是命令的定义。方括号里的参数 `<num>` 是可选的，用于指定新命令所需的参数数目（最多 9 个）。如果缺省可选参数，默认就是 0，也就是新建的命令不带任何参数。

接下来的两个例子有助你的理解。第一个例子定义了一个新的命令：`\tnss`。这个命令是手册英文名称 “The Not So Short Introduction to L^AT_EX 2_ε” 的简写。如果你需要在文档中多次使用本手册的名称，那么使用这个命令是一个非常方便的办法。

```
\newcommand{\tnss}{The not  
so Short Introduction to  
\LaTeXe}  
This is ‘‘\tnss’’ \ldots{}  
‘‘\tnss’’
```

This is “The not so Short Introduction to L^AT_EX 2_ε” ... “The not so Short Introduction to L^AT_EX 2_ε”

第二个例子演示了如何定义一个带参数的命令。在命令的定义中，标记 #1 代表指定的参数。如果想使用多个参数，可以依次使用 #2、……、#9 等标记。

```
\newcommand{\txsit}[1]
{This is the \emph{#1} Short
  Introduction to \LaTeXe}
% in the document body:
\begin{itemize}
\item \txsit{not so}
\item \txsit{very}
\end{itemize}
```

- This is the *not so* Short Introduction to L^AT_EX 2_ε
- This is the *very* Short Introduction to L^AT_EX 2_ε

L^AT_EX 不允许你定义一个与现有命令重名的命令。如果需要修改命令定义的话，使用 \renewcommand 命令。它使用与命令 \newcommand 相同的语法。

在某些情况之下，你可能会希望使用 \providecommand 命令。在命令不存在时，它相当于 \newcommand；在命令已经存在时，仍沿用存在的定义。

8.1.2 定义环境

与 \newcommand 命令类似，有一个命令用于定义新的环境。这个命令是 \newenvironment，它的语法如下所示：

```
\newenvironment{<name>}[<num>]{<before>}{<after>}
```

同样地，\newenvironment 命令有一个可选的参数。在 <before> 中的内容将在此环境包含的文本之前处理，而在 <after> 中的内容将在遇到 \end{<name>} 命令时处理。

下面的例子演示了 \newenvironment 命令的用法：

```
\newenvironment{king}
{\rule{1ex}{1ex}%
  \hspace{\stretch{1}}}
{\hspace{\stretch{1}}%
  \rule{1ex}{1ex}}

\begin{king}
My humble subjects \ldots
\end{king}
```

■ My humble subjects ... ■

参数 <num> 的使用方式与 \newcommand 命令相同。L^AT_EX 还同样保证你不会不小心新建重名的环境。如果你确实希望改变一个现有的环境，你可以使用命令 \renewenvironment，它使用和命令 \newenvironment 相同的语法。

8.2 编写自己的宏包和文档类

8.2.1 编写简单的宏包

如果你定义了很多新的环境和命令，你的文档的导言区将变得相当长，在这种情况下，好的方式是建立一个新的 L^AT_EX 宏包来存放所有你自己定义的命令和环境，然后在你的文档中使用 \usepackage 命令来调用自定义的宏包。

写一个宏包的基本工作就是将原本在你的文档导言区里很长的内容拷贝到另一个文件中，这个文件需要以 .sty 作扩展名。你还需要加入一个宏包专用的命令：

```
% Demo Package by Tobias Oetiker
\ProvidesPackage{demopack}
\newcommand{\tnss}{The not so Short Introduction
                to \LaTeXe}
\newcommand{\txsit}[1]{The \emph{#1} Short
                      Introduction to \LaTeXe}
\newenvironment{king}{\begin{quote}}{\end{quote}}
```

源代码 8.1: 宏包的一个最简示例。

```
\ProvidesPackage{<package name>}
```

这个命令应该放在你的宏包的最前面，并且一定要注意：<package name> 需要和宏包的文件名一致。`\ProvidesPackage` 让 L^AT_EX 记录宏包的名称，从而在你尝试再次调用同一个宏包的时候忽略后面的引入¹。源代码 8.1 给出了一个小的宏包示例，其中包含了我们之前定义的一些命令。

8.2.2 在宏包中调用其它宏包

如果你想进一步把各种宏包的功能汇总到一个文件里，而不是在文档的导言区罗列一大堆宏包的话，L^AT_EX 允许你在自己编写的宏包中调用其它宏包，命令为 `\RequirePackage`，用法和 `\usepackage` 一致：

```
\RequirePackage[<option1>,<option2>]{<package name>}
```

8.2.3 编写自己的文档类

当你更进一步，需要编写自己的文档类，如论文模板等，问题就稍稍麻烦了一些。首先，自己的文档类以 `.cls` 作扩展名，开头使用 `\ProvidesClass` 命令：

```
\ProvidesClass{<class name>}
```

当然了，{<class name>} 也需要和文档类的文件名一致。

但是有了上述命令和和你之前学到的 `\newcommand` 等，还并不能完成一个文档类的编写，因为诸如 `\chapter`、`\section` 等等许多常用的命令都是在文档类中定义的。事实上，许多时候我们只需要像调用宏包那样调用一个基本的文档类，省去许多不必要的麻烦。在你的文档类中调用其它文档类的命令是 `\LoadClass`，用法和 `\documentclass` 十分相像：

```
\LoadClass[<option1>,<option2>]{<package name>}
```

8.3 计数器

我们早就见识到了 L^AT_EX 对文档元素自动计数的能力：章节符号、列表、图表……它们都是依靠 L^AT_EX 提供的“计数器”完成的。

¹但如果你以不同的选项多次引入宏包，则有可能会引起错误，见附录 B.1。

8.3.1 定义和修改计数器

定义一个计数器的方法为：

```
\newcounter{<counter name>}
\newcounter{<counter name>}[<parent counter name>]
```

<counter name> 为计数器的名称，可选参数 *<parent counter name>* 定义为 *<counter name>* 的上级计数器。

以下命令修改计数器的数值，`\setcounter` 将数值设为 *<number>*；`\addtocounter` 将数值加上 *<number>*；`\stepcounter` 将数值加一，并将所有下级计数器归零。

```
\setcounter{<counter name>}{<number>}
\addtocounter{<counter name>}{<number>}
\stepcounter{<counter name>}
```

8.3.2 计数器的值

计数器 *<counter>* 的数值由 `\the<counter>` 表示，如我们在 5.4 一节见过的 `\thechapter` 等。这个值默认以阿拉伯数字形式输出，如果想改成其它形式，需要重定义 `\the<counter>`，如定义为大写字母：

```
\renewcommand{\the<counter>}{\Alph{<counter>}}
```

命令 `\Alph` 控制计数器 *<counter>* 的值以大写字母形式显示。下表列出所有可用于修改计数器格式的命令。注意：这些命令只能用于计数器，不能直接用于数字，如 `\roman{1}` 这样的命令会出错。

<code>\arabic</code>	阿拉伯数字（默认）
<code>\alph</code>	小写字母
<code>\Alph</code>	大写字母
<code>\roman</code>	小写罗马数字
<code>\Roman</code>	大写字母
<code>\fnsymbol</code>	一系列符号，用于标题页生成的脚注中

计数器的值还可以利用其它字符，甚至其它计数器的值与之组合。如标准文档类里对 `\subsection` 相关的计数器的值给出的定义相当于：

```
\renewcommand\thesubsection{\thesection.\arabic{subsection}}
```

8.3.3 L^AT_EX 中的计数器

- 我们当然早就意识到了，所有章节命令 `\chapter`、`\section` 等分别对应计数器 `chapter`、`section` 等等，而且有上下级的关系。而计数器 `part` 是独立的。
- 有序列表 `enumerate` 的各级计数器为 `enumi`、`enumii`、`enumiii`、`enumiv`，也是有上下级的关系。
- 图表浮动体的计数器就是 `table` 和 `figure`；公式的计数器为 `equation`。这些计数器在 `article` 文档类中是独立的，而在 `book` 和 `report` 中以 `chapter` 为上级计数器。
- 页码、脚注的计数器分别是 `page` 和 `footnote`。

我们可以利用前面介绍过的命令，修改计数器的样式以达到想要的效果，比如把页码修改成大写罗马字母，或是给脚注加上方括号。修改页码的样式有个更方便的命令 `\pagenumbering`：

```
\pagenumbering{Roman}  
\renewcommand\thefootnote{[\arabic{footnote}]}
```

最后介绍两个有用的计数器：

- `secnumdepth`

L^AT_EX 内部对章节规定了层级：

- 在 `article` 文档类里 `part` 为 0，`section` 为 1，依此类推；
- 在 `report/book` 文档类里 `part` 为 -1，`chapter` 为 0，`section` 为 1，等等。

`secnumdepth` 计数器控制章节编号的深度，如果章节的层级大于 `secnumdepth`，那么章节的标题、在目录和页眉页脚的标题都不编号（照常生成目录和页眉页脚），章节计数器也不计数。

可以用 `\setcounter` 命令设置 `secnumdepth` 为较大的数使得层级比较深的章节也编号，如设置为 4 令 `\paragraph` 也编号；或者设置一个较小的数以取消编号，如设置为 -1 令 `\chapter` 不编号。后者是生成不编号的章节的一个妙招，免去了手动使用 `\addcontentsline` 和 `\markboth` 的麻烦。

`secnumdepth` 计数器在 `article` 文档类里默认为 3（`subsubsection` 一级）；在 `report` 和 `book` 文档类里默认为 2（`subsection` 一级）。

- `tocdepth`

`tocdepth` 计数器控制目录的深度，如果章节的层级大于 `tocdepth`，那么章节将不会自动写入目录项。默认值同 `secnumdepth`。

附录 A 安装 T_EX 发行版

高德纳的 T_EX 程序开发于 20 世纪 80 年代，那时候电子计算机的运算能力有限，T_EX 还是大型服务器上的玩物。而如今个人计算机完全能够胜任排版的工作，并催生了用于个人计算机的工具集合——T_EX 发行版的发展。

本章会简单介绍如何安装 T_EX 发行版，以及保持发行版的内容紧跟最新。后者非常重要，因为 L^AT_EX 宏包是不断更新换代的。

A.1 T_EX 发行版简介

一个 **T_EX 发行版** 是 T_EX 排版引擎、支持排版的文件（基本格式、L^AT_EX 宏包、字体等）以及一些辅助工具的集合。各式各样的 T_EX 发行版经过十多年的发展，大浪淘沙，最终形成两个成熟的分支：

- **T_EX Live**

T_EX Live 由类 Unix 系统上的 teT_EX 发展并取而代之，最终成为跨平台的 T_EX 发行版。T_EX Live 自 2011 年起以年份作为发行版的版本号，保持了一年一更的频率。

MacT_EX 是 OS X 系统下的一个定制化的 T_EX Live 版本，与 T_EX Live 同步更新。

- **MikT_EX**

MikT_EX 是主要用于 Windows 平台的一个稳定发展的 T_EX 发行版。中国的 L^AT_EX 用户应该对“CT_EX 套装”比较熟悉，它是一个经过本地化配置的 MikT_EX。

T_EX Live 和 MikT_EX 都集成了一个简单的 L^AT_EX 源代码编辑器 T_EXworks（MacT_EX 则集成了类似的 T_EXshop）。用户在完成发行版的安装后，可直接打开编辑器开始编写 L^AT_EX 源代码。

A.1.1 安装发行版

T_EX Live

T_EX Live 在 <http://www.tug.org/texlive/> 上提供 ISO 光盘镜像¹。下载镜像到本地，挂载到虚拟光驱，或者用压缩工具解压后，在其根目录有几个用于安装的脚本：

- 用于 Windows 的批处理文件：
 - `install-tl-windows.bat` 双击启动图形界面安装程序（简单安装）；
 - `install-tl-advanced.bat` 双击启动图形界面安装程序（定制安装）；
 - 在命令提示符中输入 `install-tl-windows.bat -no-gui` 启动文本界面安装程序。

¹Linux 发行版的软件源也提供 T_EX Live 的安装，不过不够完整，更新也不是很及时。建议直接从镜像安装。

- 用于 Linux 的 Perl 脚本 `install-tl` :
 - `install-tl` 启动文本界面安装程序;
 - `install-tl -gui=wizard` 启动图形界面安装程序 (简单安装);
 - `install-tl -gui=peritk` 启动图形界面安装程序 (定制安装)。

Linux 下 T_EX Live 安装完毕后, 还需要在 root 权限下进行以下操作, 使得 `xelatex` 命令能正确通过 `fontspec` 等宏包使用字体²:

1. 将 `texlive-fontconfig.conf` 文件复制到 `/etc/fonts/conf.d/09-texlive.conf`。
2. 运行 `fc-cache -fsv`。

MikT_EX

从 MikT_EX 官网 <http://www.miktex.org/> 下载名为 `basic-miktex-***.exe` 的 Windows 安装程序包。下载后直接双击打开, 按照程序的提示进行安装即可。

A.2 安装和更新宏包

T_EX Live 提供了图形界面的宏包管理器 T_EX Live Manager 用于安装和更新宏包, 而 MikT_EX 也提供了管理器 MikT_EX Package Manager。用户可直接打开程序, 进行宏包的安装和更新 (MikT_EX Package Manager 有普通权限和管理员权限的版本, 建议总是打开管理员权限的程序)。

两者也可以通过各自的命令行工具安装更新宏包:

```
% TeX Live 命令行工具 tlmgr 的使用示例
tlmgr install <package-name> % 安装某个宏包
tlmgr remove <package-name> % 卸载某个宏包
tlmgr update --all --self      % 更新所有宏包 (包括 tlmgr 本身)
tlmgr update --list           % 列出所有可更新的宏包
tlmgr repository set http://.../CTAN/systems/texlive/tlnet
                             % 指定更新源 (CTAN) 地址
tlmgr info <package-name>     % 查看宏包信息
                             % 加 --list 选项可列出宏包的所有文件
```

```
% MikTeX 命令行工具 mpm 的使用示例
% 建议始终加 --admin 选项使用
mpm --admin --install <package-name> % 安装某个宏包
mpm --admin --uninstall <package-name> % 卸载某个宏包
mpm --admin --update                  % 更新所有宏包
mpm --admin --set-repository=http://.../CTAN/systems/win32/miktex/tm/packages
                                     % 指定更新源 (CTAN) 地址
mpm --admin --print-package-info <package-name>
                                     % 查看宏包信息
```

²<http://www.tug.org/texlive/doc/texlive-zh-cn/texlive-zh-cn.pdf>, 可用 `texdoc texlive-zh-cn` 在本地打开。

T_EX Live 默认安装所有宏包，而 MikT_EX 的安装程序只包含了若干用于 L^AT_EX 的基本宏包。从 T_EX Live 的光盘镜像和 MikT_EX 的安装包体积可见一斑。默认情况下，编译过程中如果遇到宏包未安装而报错的情况下，MikT_EX 会弹出一个对话框，让用户可以选择临时安装宏包，安装成功后继续编译。

A.2.1 手动安装宏包

如非万不得已，尽量不要手动安装宏包。绝大多数宏包都已打包到 T_EX Live 和 MikT_EX 两大发行版的安装源，可用宏包管理器安装。如果你知道某个宏包的名称，但不确定是否在发行版中已打包，可在 CTAN 中搜索。

如果确实有手动安装宏包的需要，本小节的内容将有所帮助。在手动安装之前，有必要了解一下 T_EX 目录结构 (T_EX Directory Structure, TDS)。它是 T_EX 发行版中宏包、字体、帮助文档等文件的组织结构。TDS 有时也称为 TEXMF 树，取 T_EX+METAFONT 之意。

以 T_EX Live 为例，系统的 TEXMF 树根目录为 C:\texlive\2015\texmf-dist，其下有很多子目录，仅举几例：

tex/latex L^AT_EX 宏包。

doc/latex L^AT_EX 宏包的帮助文档。

source/latex L^AT_EX 宏包的源代码。

fonts/tfm T_EX 使用的字体文件，TFM 格式。

fonts/type1 PostScript 字体文件 (Type1)，PFB 格式。

fonts/opentype OpenType 格式的字体文件。

需要手动安装的宏包，一般已经按照上述目录结构打包完成。手动安装时，尽量不要拷贝到系统的 TEXMF 树，而是拷贝到发行版提供的用户 TEXMF 树，如 T_EX Live 的 C:\texlive\texmf-local。安装完成后，还需**刷新 T_EX 发行版的文件名数据库**，令新安装的宏包文件能够被系统找到。T_EX Live 用户须在 Windows 命令行或者 Linux 终端执行命令：

```
mktexlsr
```

MikT_EX 用户的命令为：

```
initexmf --update-fndb
```


附录 B 排除错误、寻求帮助

L^AT_EX 入门用户总会为两大问题头疼：我写的代码到底哪里出错了？如果想要实现某种用法该怎么办？本章首先总结了常见的 L^AT_EX 错误及应对的办法。

B.1 L^AT_EX 错误

当我们用排版引擎编译 L^AT_EX 代码时，命令行的窗口（终端）会显示大量信息（T_EXworks 等编辑器会有一个区域显示这些信息）。当编译过程中出现错误时，信息将会停止在出错的地方，等待我们接下来的操作。

比如说我们有一个明显出错的例子：

```
\documentclass{article}
\begin{document}
Test \LaTeX{} and it's friends.
\end{document}
```

编译过程中遇到这个错误将会停顿下来，提示错误，并等待用户输入指令：

```
! Undefined control sequence.
1.3 Test \Latex
    {} and it's friends.
```

这种错误信息分两部分，前一部分（一行或多行）提示了错误的原因，后一部分指出了错误发生的行号，以及通过错落的文字告知发生错误的命令所在位置。如上错误显示 `\Latex` 位置发生了错误，错误提示是“未定义的控制序列”，意思是 `\LaTeX` 是 T_EX 程序无法识别的一个命令，很显然是我们把 `\LaTeX` 的大小写写错了。

处理方式

出现错误时，编译过程将暂停，等待用户输入命令。用户可以直接敲回车跳过当前的错误，继续编译，相当于丢掉了写错的命令，将“Test and it's friends.”编译出来。但这个例子过于简单，有些复杂的代码中，有可能会由于一个小问题导致一连串的错误。此时可以选择按 **S/R/Q** 选择跳过接下来的所有错误，或者按 **X** 直接退出编译。

常见的 L^AT_EX 错误信息

笔者在此总结一些经常发生、问题比较明确的 L^AT_EX 错误：

- **! Undefined control sequences.**

使用了未定义的命令。拼写错误是原因之一，如把 `\LaTeX` 写作 `\Latex` 这样。也有可能是没有调用某个宏包，但用了该宏包定义的命令。

- `! Missing $ inserted.`

缺少数学环境的符号 `$`。多由于将数学符号用在公式之外而导致。

- `Runaway argument?`

`! Paragraph ended before ... was complete.`

- `! File ended while scanning use of`

两个错误都是主要由于漏写了包裹命令参数的花括号，导致识别参数时出现错误。

- `! Extra alignment tab has been changed to \cr.`

- `! Misplaced \noalign.`

两个错误都与表格有关。

– 前者的字面意义是“一行中使用的列分隔符 `&` 太多”，有的时候可能确实是 `&` 的个数和列格式不匹配，但多数情况是漏了行尾的 `\\`。

– 后者常出现于漏掉了行尾的 `\\` 而接着使用 `\hline` 画横线的时候。

- `! LaTeX Error: Lonely \item-perhaps a missing list environment.`

- `! LaTeX Error: Something's wrong-perhaps a missing \item.`

两个错误信息都与列表环境和 `\item` 命令有关。前者意味着在没有使用列表环境的情况下用了 `\item`；后者则相反，是在列表环境中漏了 `\item`。

- `! I can't find file '...'.`

- `! LaTeX Error: File '...' not found.`

两个错误都意味着缺少文件。

– 如果使用 `\input` 或者 `\include` 命令添加文件，文件不存在，文件名或路径不正确，将会出现上述错误；

– 如果错误提示里的文件名带 `.cls` 或者 `.sty` 扩展名，那么很显然，是因为**没有安装所需的宏包或文档类**。

- `! LaTeX Error: Missing \begin{document}.`

字面上是缺少 `\begin{document}`，实际上往往是由于在 `\begin{document}` 之前（导言区）输入了文字。

- `! LaTeX Error: Can be used only in preamble.`

与上一条相反，由于将必须用于导言区的命令放到了 `\begin{document}` 之后使用而产生。

- `! LaTeX Error: \begin{...} on input line ...ended by \end{...}.`

环境首尾不匹配。比如 `\begin{enumerate}` 用了 `\end{itemize}` 结尾。往往是由于漏写了 `\begin` 或者 `\end` 命令。

- `! LaTeX Error: Option clash for package '...'.`

以**不同选项**重复调用宏包造成冲突。有可能是因为其它宏包内部事先调用了这个宏包，用户再次带选项调用而导致冲突。可尝试去掉重复调用的宏包避免冲突。如果宏包允许的话，尽量使用其定义的命令改变设置，避免将设置作为宏包选项。

- **! LaTeX Error: Unknown option ‘...’ for package ‘...’.**

调用宏包时使用了不能被其识别的选项。此时应该查找宏包的帮助文档来解决问题。

- **! Package ‘...’ error: ...**

宏包或文档类自定义的错误，由于不正确地使用宏包里的命令而导致。此时应该查找宏包的帮助文档来解决问题。

B.2 查看帮助文档

无论是 $\text{T}_\text{E}\text{X}$ Live 还是 $\text{MikT}_\text{E}\text{X}$ ，提供了一个命令行模式的程序 `texdoc`。比如对 5.4.3 小节的 `fancyhdr` 宏包感兴趣，这时在 Windows 命令提示符或者 Linux 终端输入以下命令，则会弹出宏包的帮助手册 `fancyhdr.pdf`：

```
texdoc fancyhdr
```

除了宏包的帮助手册外， $\text{T}_\text{E}\text{X}$ 发行版还包括了各类有用的文档，有一部分在参考文献中给出。

如果不熟悉命令行工具的话， $\text{T}_\text{E}\text{X}$ Live 提供了一个图形界面的程序 `TeXdoc GUI`。打开后，可以看到程序里的许多按钮，分别代表某一类的帮助文档。除此之外，点击 `File Search` 弹出搜索框，输入想要搜索的宏包和文件并按回车键，`TeXdoc GUI` 会弹出它搜索到的所有结果，可点击任意一项来打开文档。

当然对于初学者，有一个现实而棘手的问题：某个命令到底是 $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$ 自有的，还是哪个宏包提供的？很遗憾地说，除了通过慢慢积累、熟悉较多宏包之外，没有很方便的办法解决这个问题，因为 $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$ 的宏包扩展实在太多了。本手册末尾的索引给出了所有在本手册见到的命令和环境，其中哪些命令和环境需要调用哪个宏包才能使用，一目了然。但是这个索引远远不够。

解决这个问题有几点可行的办法：

1. 查询一些扩展资料，如总结所有 $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$ 自带命令的文档 [11]、 $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$ 符号大全 [14] 等；
2. 在互联网上搜索自己不清楚的命令；
3. 在论坛上提问求助有经验的人。

B.3 常用宏包简介

此处不包含设置字体或数学符号的宏包，它们已经在表 5.4 中列出。

B.3.1 文字、公式和符号

`amsmath` $\mathcal{A}\mathcal{M}\mathcal{S}$ 数学公式扩展。

`mathtools` 数学公式扩展宏包，提供了公式编号定制和更多的符号、矩阵等。

`amsfonts` $\mathcal{A}\mathcal{M}\mathcal{S}$ 扩展符号的基础字体支持。

`amssymb` 在 `amsfonts` 基础上将 $\mathcal{A}\mathcal{M}\mathcal{S}$ 扩展符号定义成命令。

`bm` 提供将符号加粗的命令 `\bm`。

`siunitx` 以国际单位规范排版物理量的单位。

mhchem 排版化学公式。

tipa 排版国际音标。

B.3.2 排版元素

ulem 提供排版可断行下划线的命令 `\uline` 以及其它装饰文字的命令。

endnote 排版尾注。

multicol 提供将内容自由分栏的 `multicols` 环境。

multitoc 生成多栏排版的目录。

minitoc 为章节生成独立的小目录。

glossaries 生成词汇表。

verbatim 对原始的 `verbatim` 环境的改善。提供了命令 `\verbatiminput` 调用源文件。

fancyvrb 提供了代码排版环境 `Verbatim` 以及对版式的自定义。

listings 提供了排版关键字高亮的代码环境 `lstlisting` 以及对版式的自定义。类似宏包有 `minted`。

algorithm2e 排版算法。

ntheorem 定制定理环境。类似宏包包括 `theorem`、`thmtools`、`amsthm` 等。

B.3.3 图表和浮动体

booktabs 制作三线表。

array 对表格列格式的扩展。

tabularx 提供 `tabularx` 环境排版定宽表格，支持自动计算宽度的 X 列格式。

colortbl 支持修改表格的行、列、单元格的顏色。

multirow 支持合并多行单元格。

makecell 支持在单元格里排版多行内容（嵌套一个单列的小表格）。

diagbox 制作斜线表头。

longtable 提供排版跨页长表格的 `longtable` 环境。

ltxtable 跨页长表格可使用 `tabularx` 的 X 列格式。

tabu 提供排版复杂格式表格的 `tabu` 环境。与 `longtable` 一同调用时，提供排版复杂格式跨页长表格的 `longtabu` 环境。

graphicx 支持插图。

bmpsize `latex + dvipdfmx` 命令下支持 BMP/JPG/PNG 等格式的位图。

epstopdf `pdflatex` 命令下支持 EPS 格式的矢量图。

wrapfig	支持简单的文字在图片周围的绕排。
subfig	提供子图表和子标题的排版。类似宏包有 subfigure 和 subcaption 等。
caption	控制图表标题的格式。
float	为浮动体提供不浮动的 H 模式；提供自定义的浮动体结构。

B.3.4 修改版式

geometry	修改页面尺寸、页边距、页眉页脚等参数。
fancyhdr	修改页眉页脚格式，令页眉页脚可以左对齐、居中、右对齐。
titlesec	修改章节标题 \chapter、\section 等的格式。
titletoc	修改目录中各条目的格式。
tocloft	类似 titletoc 的修改目录条目格式的宏包。
tocbibind	支持将目录、参考文献、索引本身写入目录项。
footmisc	修改脚注 \footnote 的格式。
indentfirst	令章节标题后的第一段首行缩进。
enumitem	修改列表环境 enumerate 和 itemize 等的格式。

参考文献

§ L^AT_EX 经典书籍：

- [1] Leslie Lamport. *L^AT_EX: A Document Preparation System*, 2nd edition. Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-52983-1.
- [2] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley. *The L^AT_EX Companion*, 2nd edition. Addison-Wesley, Reading, Massachusetts, 2004, ISBN 0-201-36299-6.
- [3] Michel Goossens, Sebastian Rahtz and Frank Mittelbach. *The L^AT_EX Graphics Companion*, 2nd edition. Addison-Wesley, Reading, Massachusetts, 1997, ISBN 0-301-50892-0.
- [4] 刘海洋. *L^AT_EX 入门*. 电子工业出版社, 北京, 2013, ISBN 978-7-121-20208-7

§ T_EX 经典书籍，介绍底层的命令和排版方式，有一些书籍已经开放网络资源：

- [5] Donald E. Knuth. *The T_EXbook*, Volume A of *Computers and Typesetting*, 2nd edition. Addison-Wesley, Reading, Massachusetts, 1984, ISBN 0-201-13448-9.
- [6] Paul Abrahams, Kathryn A. Hargreaves, Karl Berry. *T_EX for the impatient*, 2nd edition. Addison-Wesley, Reading, Massachusetts, 1984, ISBN 0-201-51375-7.
[CTAN://info/impatient/book.pdf](http://ctan.org/info/impatient/book.pdf) (texdoc impatient)
- [7] Victor Eijkhout. *T_EX by Topic, A T_EXnician's Reference*. Addison-Wesley, Reading, Massachusetts, 1992, ISBN 0-201-56882-9.
[CTAN://info/texbotopic/TeXbyTopic.pdf](http://ctan.org/info/texbotopic/TeXbyTopic.pdf) (texdoc texbytopic)

§ T_EX 发行版内置的资源（除宏包帮助手册之外的一些文档），可以在本地使用 `texdoc` 命令查找，也可以在 CTAN 上找到：

- [8] L^AT_EX3 Project Team. *L^AT_EX 2_ε for authors*.
[CTAN://macros/latex/doc/usrguide.pdf](http://ctan.org/macros/latex/doc/usrguide.pdf) (texdoc usrguide)
- [9] L^AT_EX3 Project Team. *L^AT_EX 2_ε for class and package writers*.
[CTAN://macros/latex/doc/clsguide.pdf](http://ctan.org/macros/latex/doc/clsguide.pdf) (texdoc clsguide)
- [10] L^AT_EX3 Project Team. *L^AT_EX 2_ε font selection*.
[CTAN://macros/latex/doc/fntguide.pdf](http://ctan.org/macros/latex/doc/fntguide.pdf) (texdoc fntguide)
- [11] Karl Berry, Jim Hefferon, Vincent Belaïche. *L^AT_EX 2_ε: An unofficial reference manual*.
[CTAN://info/latex2e-help-texinfo/latex2e.pdf](http://ctan.org/info/latex2e-help-texinfo/latex2e.pdf) (texdoc latex2e)

- [12] The UK TeX users group (TUG). *The UK TeX FAQ*. HTML version on <http://www.tex.ac.uk/faq/>.
CTAN://help/uk-tex-faq/newfaq.pdf (texdoc faq)
- [13] Jürgen Fenn. *An essential guide to L^AT_EX 2_ε usage: Obsolete commands and packages* (English translation).
CTAN://info/l2tabu/english/l2tabuen.pdf (texdoc l2tabuen)
- [14] Scott Pakin. *The Comprehensive L^AT_EX Symbol List*.
CTAN://info/symbols/comprehensive/symbols-a4.pdf (texdoc symbols-a4)
- [15] Winston Chang. *L^AT_EX 2_ε cheatsheet*.
CTAN://info/latexcheat/latexcheat/latexsheet.pdf (texdoc latexcheat)

§ CTAN 上的其它网络资源:

- [16] Graham Williams. *The T_EX Catalogue (Preface CTAN Edition)*.
CTAN://help/Catalogue/catalogue.html
- [17] Stephen G. Hartke. *A Survey of Free Math Fonts for T_EX and L^AT_EX*.
CTAN://info/Free_Math_Font_Survey/survey.pdf

§ 其它网络渠道可获得的资源:

- [18] Palle Jørgensen. *The L^AT_EX font catalogue*, a font catalogue of L^AT_EX font packages.
<http://www.tug.dk/FontCatalogue/>
- [19] Indian T_EX user group. *L^AT_EX Tutorials: A primer*.
<http://www.tug.org/twg/mactex/tutorials/ltxprimer-1.0.pdf>
- [20] 黄新刚. 雷太赫排版系统简介 (*L^AT_EX Notes*), 第二版.
<http://dralpha.altervista.org/zh/tech/lnotes2.pdf>
(旧版可用 texdoc latex-notes-zh-cn)

索引

符号

\! (数学命令), 38
\$ (数学模式), 29
% (注释), 11
& (单元格/对齐), 19, 36, 37
\,, 38
\: (数学命令), 38
\; (数学命令), 38
\[, 30
\\ (换行), 19, 22, 59
\], 30
^ (数学上标), 31
_ (数学下标), 31

A

abstract 环境, 18
\addcontentsline, 13
\addtocounter, 84
\addtolength, 57
align 环境 (amsmath), 36
align* 环境 (amsmath), 37
aligned 环境 (amsmath), 37
\Alph, 84
\alph, 84
amsfonts 宏包, 29
amsmath 宏包, 29
amssymb 宏包, 29, 31
amsthm 宏包, 29, 42
\appendix, 14
\approx (数学符号 \approx), 32
\arabic, 84
array 环境, 37
array 宏包, 20
\arraystretch, 22

article 文档类, 5, 62
\author, 15

B

\backmatter, 14
\begin, 3
\bfseries, 54
编码, 9
ASCII, 9
GBK, 9
Latin-1, 9
UTF-8, 10
表格, 19-23
标题页, 15
\bibliography, 66
\bibliographystyle, 66
BibTeX 工具, 66
BibTeX 数据库, 66
\Big (数学命令), 35
\big (数学命令), 35
\Bigg (数学命令), 35
\bigg (数学命令), 35
\Bigl (数学命令), 35
\bigl (数学命令), 35
\Bigr (数学命令), 35
\bigr (数学命令), 35
\biggr (数学命令), 35
\Bigl (数学命令), 35
\bigl (数学命令), 35
\Bigr (数学命令), 35
\bigr (数学命令), 35
\bigskip, 59
\binom (amsmath), 32
\bm (bm), 41
bm 宏包, 41
Bmatrix 环境 (amsmath), 38

`bmatrix` 环境 (amsmath), 38
`\bmod` (数学命令), 33
`\boldmath`, 40
`\boldsymbol` (amsmath), 40
 book 文档类, 5, 13, 14, 62
 booktabs 宏包, 21
`\bottomrule` (booktabs), 21
 bp (长度单位), 57

C

参考文献, 65
 参数, 3
`\caption`, 26
`cases` 环境 (amsmath), 38
`\cdot` (数学符号 \cdot), 32
`\cdots` (数学符号 \cdots), 31
`center` 环境, 17
`\centering`, 17
 超链接, 71
`\chapter`, 13
`\cite`, 65, 66
`\citep` (natbib), 68
`\citet` (natbib), 68
`\clearpage`, 26, 62
`\cline`, 21
 cm (长度单位), 57
`\color` (color/xcolor), 70
 color 宏包, 70
`\colorbox` (color/xcolor), 71
`\coordinate` (tikz), 76
 ctex 宏包, 10
 ctexart 文档类, 5, 10
 ctexbook 文档类, 5, 10
 ctexrep 文档类, 5, 10

D

导数符号' (a'), 31
 导言区, 3
`\date`, 15
`\ddot` (数学重音 \ddot{a}), 34
`\DeclareMathOperator` (amsmath), 33
`description` 环境, 17
`\dfrac` (amsmath), 32
`displaymath` 环境, 30

`\displaystyle` (数学命令), 39
`\div` (数学符号 \div), 32
`document` 环境, 3
`\documentclass`, 3
`\dot` (数学重音 \dot{a}), 34
`\dots` (数学符号 \dots), 31
`\draw` (tikz), 77
 多行公式, 36–37
`dvipdfmx` 命令, 4

E

`\em`, 54
 em (长度单位), 57
`\emph`, 54
`\end`, 3
`enumerate` 环境, 16
`\eqref` (amsmath), 29
`equation` 环境, 29
`equation*` 环境 (amsmath), 30
`\equiv` (数学符号 \equiv), 32
 ex (长度单位), 57

F

`fancyhdr` 宏包, 64
`\fbox`, 24
`\fboxrule`, 24, 71
`\fboxsep`, 24, 71
`\fcolorbox` (color/xcolor), 71
 分组, 3, 53
`figure` 环境, 26, 62
`figure*` 环境, 26
`\fill`, 59
`\fill` (tikz), 77
`\filldraw` (tikz), 77
`float` 宏包, 26, 62
`\flushbottom`, 62
`flushleft` 环境, 17
`flushright` 环境, 17
`\fnsymbol`, 84
`\fontsize`, 53
`fontspec` 宏包, 55
`\footnote`, 16, 25
`\footnotemark`, 16, 25
`\footnotesize`, 54

\footnotetext, 16, 25

\frac (数学命令), 32

\framebox, 24

\frontmatter, 14

浮动体, 62

G

gather 环境 (amsmath), 36

gather* 环境 (amsmath), 37

gathered 环境 (amsmath), 37

\ge (数学符号 \geq), 32

\geometry (geometry), 60

geometry 宏包, 60

\graphicspath (graphicx), 23

graphicx 宏包, 23

H

行距, 57

行内公式, 29

\hat (数学重音 $\hat{}$), 34

\hline, 19, 21

宏包, 3, 6

\href (hyperref), 72

\hspace, 59

\hspace*, 59

\Huge, 54

\huge, 54

\hyperref (hyperref), 72

hyperref 宏包, 71

\hypersetup (hyperref), 71

I

in (长度单位), 57

\include, 7, 14

\includegraphics (graphicx), 23

\includeonly, 8

\indent, 58

indentfirst 宏包, 58

\index, 68, 69

\infty (数学符号 ∞), 31

\input, 8

inputenc 宏包, 9, 10

\int (数学符号 \int), 33

\item, 16

itemize 环境, 16

\itshape, 54

J

计数器, 83

K

Knuth, Donald E. (高德纳), 1

L

\label, 15, 26, 29, 41

\LARGE, 54

\Large, 54

\large, 54

L^AT_EX, 1

L^AT_EX 环境, 3

L^AT_EX 命令, 2

L^AT_EX 2_ε, 1

latexsym 宏包, 44

\le (数学符号 \leq), 32

\left (数学命令), 35, 37

\limits (数学命令), 34

\linespread, 57

\listoffigures, 26

\listoftables, 26

\LoadClass, 83

M

\mainmatter, 14

\makebox, 24

makeidx 宏包, 68

\makeindex (makeidx), 68

makeindex 工具, 68

\maketitle, 15

\markboth, 63

\markright, 63

\mathbb (amssymb), 39

\mathbf (数学命令), 39

\mathcal (数学命令), 39

\mathfrak (amssymb), 39

\mathit (数学命令), 39

\mathnormal (数学命令), 39

\mathrm (数学命令), 39

\mathscr (mathrsfs), 39

`\mathsf` (数学命令), 39
`\mathtt` (数学命令), 39
`matrix` 环境 (amsmath), 38
`\mbox`, 24
`\mdseries`, 54
`\medskip`, 59
`\midrule` (booktabs), 21
`minipage` 环境, 25
`mm` (长度单位), 57
`\mp` (数学符号 \mp), 32
`multicol` 宏包, 62
`multicols` 环境 (multicol), 62
`\multirow` (multirow), 22
`multirow` 宏包, 22
`multitoc` 宏包, 62
`multiline` 环境 (amsmath), 35
`multiline*` 环境 (amsmath), 36

N

`\nabla` (数学符号 ∇), 32
`natbib` 宏包, 68
`\ne` (数学符号 \neq), 32
`\newcommand`, 81
`\newcounter`, 84
`\newenvironment`, 82
`\newlength`, 57
`\newpage`, 62
`\newtheorem`, 41
`\nocite`, 66
`\node` (tikz), 77
`\noindent`, 58
`\nolimits` (数学命令), 34
`\nolinkurl` (hyperref), 72
`\nonumber`, 29
`\normalfont`, 54
`\normalsize`, 54
`\notag` (amsmath), 29, 36

O

`\oint` (数学符号 \oint), 33
`\onecolumn`, 62
`\overbrace` (数学重音 \overbrace{AB}), 34
`\overline` (数学重音 \overline{AB}), 34
`\overrightarrow` (数学重音 \overrightarrow{AB}), 34

P

`\pagenumbering`, 85
`\pageref`, 15
`\pagestyle`, 62
排版引擎, 4
`\par`, 11, 58
`\paragraph`, 13
`\parbox`, 25
`\parskip`, 58
`\part`, 13
`\partial` (数学符号 ∂), 32
`\pdfbookmark` (hyperref), 72
`pdflatex` 命令, 5
PDF 书签, 72
`pdfTeX`, 4
`\pm` (数学符号 \pm), 32
`pmatrix` 环境 (amsmath), 38
`\pmod` (数学命令), 33
`\printindex` (makeidx), 68
`\prod` (数学符号 \prod), 33
`proof` 环境 (amsthm), 42
`\propto` (数学符号 \propto), 32
`\providecommand`, 82
`\ProvidesClass`, 83
`\ProvidesPackage`, 82
`pt` (长度单位), 57

Q

`\qedhere` (amsthm), 42
`\qqquad`, 38
`\quad`, 38
`quotation` 环境, 18
`quote` 环境, 18

R

`\raggedbottom`, 62
`\raggedleft`, 17
`\raggedright`, 17
`\ref`, 15, 29
`\renewcommand`, 82
`\renewenvironment`, 82
`report` 文档类, 5, 13, 62
`\RequirePackage`, 83
`\right` (数学命令), 35, 37

`\rmfamily`, 54
`\Roman`, 84
`\roman`, 84
`\rule`, 25

S

`\scriptscriptstyle` (数学命令), 39
`\scriptsize`, 54
`\scriptstyle` (数学命令), 39
`\scshape`, 54
`secnumdepth` (计数器), 85
`\section`, 13
`\selectfont`, 55, 57
`\setCJKmainfont` (xeCJK), 55
`\setCJKmonofont` (xeCJK), 55
`\setCJKsansfont` (xeCJK), 55
`\setcounter`, 84
`\setlength`, 24, 57
`\setmainfont` (fontspec), 55
`\setmonofont` (fontspec), 55
`\setsansfont` (fontspec), 55
`\sffamily`, 54
 数学函数, 32
 数学模式, 30
`\sim` (数学符号 \sim), 32
`\slshape`, 54
`\small`, 54
`\smallskip`, 59
`split` 环境 (amsmath), 37
`\sqrt` (数学命令), 32
`\stackrel` (数学命令), 32
`\stepcounter`, 84
`\stretch`, 59
`subarray` 环境 (amsmath), 34
`subfig` 宏包, 27
`\subfloat` (subfig), 27
`\subparagraph`, 13
`\subsection`, 13
`\substack` (amsmath), 34
`\subsubsection`, 13
`\sum` (数学符号 \sum), 33
 索引, 68
`syntonly` 宏包, 8

`table` 环境, 26, 62
`table*` 环境, 26
`\tableofcontents`, 13
`tabular` 环境, 19
`tabular*` 环境, 21
`tabularx` 环境 (tabularx), 21
`tabularx` 宏包, 21
`\tag` (amsmath), 29
`TEX`, 1
`texdoc` 工具, 93
`\texorpdfstring` (hyperref), 72
`\text` (amsmath), 30
`\textbf`, 54
`\textcolor` (color/xcolor), 71
`\textheight`, 60
`\textit`, 54
`\textmd`, 54
`\textnormal`, 54
`\textrm`, 54
`\textsc`, 54
`\textsf`, 54
`\textsl`, 54
`\textstyle` (数学命令), 39
`\texttt`, 54
`\textup`, 54
`\textwidth`, 60
`\tfrac` (amsmath), 32
`\thanks`, 15
`thebibliography` 环境, 65
`\theoremstyle` (amsthm), 42
`\thispagestyle`, 62
`TikZ`, 75
`\tikz` (tikz), 75
`tikz` 宏包, 75
`tikzpicture` 环境 (tikz), 75
`\times` (数学符号 \times), 32
`\tiny`, 54
`\title`, 15
`tocdepth` (计数器), 85
`\today`, 15
`\toprule` (booktabs), 21
`\ttfamily`, 54
`\twocolumn`, 62

T

U

`\underbrace` (数学重音 \underbrace{AB}), 34
`\underline` (数学重音 \underline{AB}), 34
`\upshape`, 54
`\url` (hyperref), 72
`\usepackage`, 3, 6, 82

V

`\vec` (数学重音 \vec{a}), 34
`\verb`, 19
`verbatim` 环境, 18
`verse` 环境, 18
`Vmatrix` 环境 (amsmath), 38
`vmatrix` 环境 (amsmath), 38
`\vspace`, 59
`\vspace*`, 59

W

文档类, 3, 5
`\widehat` (数学重音 \widehat{AB}), 34

X

`xcolor` 宏包, 70
`xeCJK` 宏包, 10, 55
`xelatex` 命令, 5, 10, 55
`XqTeX`, 4
希腊字母 (数学符号), 31
`\xleftarrow` (amsmath), 35
`\xrightarrow` (amsmath), 35
选项 (宏包/文档类), 5, 6, 60, 71

Y

颜色, 69
`black`, 70
`blue`, 70
`brown` (xcolor), 70
`cyan`, 70
`darkgray` (xcolor), 70
`gray` (xcolor), 70
`green`, 70
`lightgray` (xcolor), 70
`lime` (xcolor), 70
`magenta`, 70
`olive` (xcolor), 70
`orange` (xcolor), 70
`pink` (xcolor), 70
`purple` (xcolor), 70
`red`, 70
`teal` (xcolor), 70
`violet` (xcolor), 70
`white`, 70
`yellow`, 70

页脚, 62
页眉, 62

Z

字号, 53
字体, 53