

L^AT_EX 之菜鸟入门

司英成

2022 年 1 月 20 日

摘要

本文是根据 github 上的一个 latex-cookbook(<https://github.com/xinychen/latex-cookbook>) 学习 L^AT_EX 的基本使用，本文也使用 L^AT_EX 编写。

目录

第一章 横空出世的 L^AT_EX	9
1.1 Tex 和 L ^A T _E X	9
1.2 引领浪潮的 L ^A T _E X	10
1.2.1 L ^A T _E X 的出现	10
1.2.2 L ^A T _E X 的特点	11
1.2.3 L ^A T _E X 编辑器	13
1.3 应运而生的在线系统	14
1.3.1 L ^A T _E X 在线系统的出现	14
1.3.2 L ^A T _E X 在线系统的特点	15
1.4 L ^A T _E X 问答社区	17
1.4.1 问答社区的介绍	17
1.4.2 高频访问问题	18
1.5 关于 L ^A T _E X 的开源项目	18
1.5.1 开源社区 GitHub	18
1.5.2 学位论文 L ^A T _E X 模板	19
1.5.3 L ^A T _E X 绘制图形	20
1.5.4 L ^A T _E X 制作简历	21
1.5.5 L ^A T _E X 制作幻灯片	21
1.5.6 L ^A T _E X 制作海报	22
1.5.7 推荐指导本书的开源项目	22
1.6 L ^A T _E X 制作中文文档	22
1.6.1 使用 CJKutf8 宏包	23
1.6.2 使用 CTEX 宏包	23
第二章 L^AT_EX 基础介绍	25
2.1 L ^A T _E X 语法规则	25

2.1.1 命令	25
2.1.2 环境	26
2.1.3 宏包	27
2.2 LATEX 代码结构	27
2.2.1 前导代码	28
2.2.2 主体代码	29
2.3 文档类型介绍	30
2.3.1 基本介绍	30
2.4 简单文档的制作	31
2.4.1 制作封面	31
2.4.2 开始创建文档	31
2.5 一些基本命令	33
2.5.1 全局格式设置	33
第三章 文本编辑	37
3.1 创建标题部分、摘要及关键词	37
3.1.1 创建标题部分	37
3.1.2 创建摘要及关键词	38
3.2 创建章节	38
3.3 生成目录	39
3.3.1 调整章节层次深度	39
3.3.2 给目录设置别名	40
3.3.3 给目录设置链接	40
3.4 编辑段落	40
3.4.1 段落首行缩进	40
3.4.2 段落间距调整	40
3.4.3 段落添加文本框	41
3.4.4 段落对齐方式调整	41
3.5 文字编辑	41
3.5.1 调整字体样式	41
3.5.2 调整字体大小	41
3.5.3 调整字体颜色	43
3.5.4 字体设置	43
3.5.5 下划线与删除线	44
3.5.6 特殊字符	45

3.6 创建列表	46
3.6.1 无序列表	46
3.6.2 排序列表	47
3.6.3 阐述性列表	48
3.6.4 自定义列表格式	48
3.7 创建页眉、页脚及脚注	50
第四章 公式编辑	53
4.1 基本介绍	53
4.1.1 数学公式环境	54
4.1.2 基本格式调整	57
4.2 常用数学符号	60
4.2.1 运算符号	60
4.2.2 标记符号	62
4.2.3 各类括号	63
4.2.4 空心符号	64
4.2.5 特殊函数	65
4.3 希腊字母	66
4.4 微积分	68
4.4.1 极限	68
4.4.2 导数	69
4.4.3 积分	70
4.5 线性代数	72
4.5.1 矩阵	72
4.5.2 符号	74
4.5.3 范数	77
第五章 表格制作	81
5.1 基本介绍	81
5.1.1 tabular 环境：创建表格内容	82
5.1.2 table 环境：自动编号与浮动表格	84
5.2 合并单元格	86
5.2.1 合并不同列的单元格	86
5.2.2 合并不同行的单元格	87
5.2.3 合并不同行不同列的单元格	88

5.3 插入斜线与表注	89
5.3.1 插入斜线	89
5.3.2 插入表注	90
5.4 调整表格样式	91
5.4.1 表格尺寸	92
5.4.2 单元格自动对齐与换行	93
5.4.3 小数点对齐	97
5.4.4 行高	99
5.4.5 列宽	100
5.4.6 线宽	100
5.4.7 表格字体大小	101
5.4.8 文字环绕表格	101
5.5 创建色彩表格	102
5.6 创建三线表格	104
5.7 创建跨页表格	105
5.8 旋转表格	106
5.9 导入表格	107
5.9.1 快速创建表格	107
5.9.2 创建三线表	108
5.9.3 设置表格属性	109
第六章 插入图形	113
6.1 插入浮动图片	113
6.2 插入非浮动图片	115
6.3 插入图表目录	117
6.4 定制图表标题样式	118
6.4.1 调整标题头部	118
6.4.2 调整编号	118
6.4.3 调整编号分隔符	120
6.5 插入子图	121
6.5.1 基本介绍	121
6.5.2 调整子图间距	122
6.6 图片排列与布局	126
6.6.1 多图并排	127

第七章 图形绘制	129
7.1 基本介绍	129
7.1.1 使用 tikzpicture 环境创建图形元素	129
7.1.2 绘制直线	131
7.1.3 图形缩放	133
7.1.4 绘制箭头	134
7.1.5 调整线条粗细	135
7.1.6 虚线	136
7.1.7 颜色	137
7.2 节点介绍	137
7.2.1 节点基本介绍	137
7.2.2 节点样式	138
7.2.3 节点命名	139
7.2.4 基于相对位置绘制节点	140
7.2.5 连接节点	140
7.3 高级功能	143
7.3.1 矩形、圆形、曲线	143
7.3.2 平滑过渡曲线	144
7.3.3 根据函数绘制曲线	145
7.3.4 简单图形的区域填充	147
7.3.5 在图形中填写标签	148
7.4 复杂模型实战解析	150
第八章 建立索引及文献引用	155
8.1 公式和图表的索引	155
8.1.1 公式的索引	155
8.1.2 图形的索引	156
8.1.3 表格的索引	156
8.2 创建超链接	157
8.2.1 超链接类型	157
8.2.2 超链接格式	159
8.3 BibTeX 用法	161
8.3.1 创建参考文献	161
8.3.2 使用 BibTeX 文件	162
8.4 文献引用格式	164

8.4.1 几种标准样式	165
第九章 幻灯片制作	167
9.1 基本介绍	168
9.1.1 Beamer 简介	168
9.1.2 创建幻灯片	170
9.1.3 创建章节与生成目录	175
9.1.4 幻灯片内容分栏	182
9.2 添加动画效果	186
9.2.1 pause 命令	187
9.2.2 item<> 命令	189
9.2.3 其它命令	191
9.2.4 自动计数	194
9.3 块与盒子——添加框元素	197
9.3.1 区块环境	199
9.3.2 定理类环境	200
9.3.3 Beamer 中的盒子	202
9.4 设置主题样式	209
9.4.1 基本介绍	210
9.4.2 颜色主题	213
9.4.3 字体主题	213
9.4.4 内部主题	215
9.4.5 外部主题	218
9.4.6 表格字体大小	219
9.4.7 样式调整	220

引言

1977 年，计算机科学家克努斯博士¹开发了一款名为 TeX 的文档排版系统，作为一种计算机程序语言，它能够专门用于制作各类技术文档，并且对制作包含数学公式的技术文档具有良好的适用性。克努斯博士开发 TeX 其实存在一些意外：上世纪 70 年代，克努斯博士在修改自己的著作时，由于当时的排版质量差到让他难以容忍，所以他便转而开始思考能否开发出高质量的文档排版系统。

在使用过程中，TeX 制作文档的方式非常特殊，与今天常用的办公软件 Word 等截然不同，它是完全使用计算机程序语言来制作文档的。由于其对计算机语言的高度依赖，这款系统的使用门槛较高，但也具有很多优点，其中最为人称道的优点是它可以书写大量复杂的数学表达式。基于 TeX，兰波特博士²于 1985 年开发了另一款文档排版系统，名为 LATEX，兰波特博士设计这款系统初衷是让人们从排版样式这些繁琐的细节中解放出来，从而将精力集中在文档结构和文档内容上，这一做法很快便让 LaTeX 取代了 TeX。后来，LaTeX 的众多开发者对 LaTeX 最初版本进行了更新和提升，也就是我们今天一直在用的 LaTeX。

¹Donald E. Knuth，直译名为唐纳德·尔文·克努斯，中文名为高德纳，美国计算机科学家，现代计算机科学的先驱人物，在计算机科学及数学领域著有多部影响深远的著作，于 1974 年获得图灵奖。

²Leslie Lamport，直译名为莱斯利·兰波特，美国计算机科学家，于 2013 年获得图灵奖，他获得图灵奖的原因并非在于开发了 LaTeX，而是源于他在所研究的学术领域做出的突出贡献。

第一章 橫空出世的 LATEX

1.1 Tex 和 LATEX

TeX 是一种专门用于文档排版的计算机程序语言，同时也是一款文档排版系统，它几乎和微软推出的 Office 办公软件同时出现，后来成为人们制作文档的两种最佳工具。TeX 和 Office 制作文档的方式截然不同，Office 的使用门槛并不高，只要掌握一些基本操作就能够制作文档；而 TeX 则需要一定的计算机编程基础，除了一些基本命令，还要掌握 TeX 环境和一些特定的宏包。实际应用中，TeX 以其高质量、高效率的排版输出，特别是数学公式的排版能力而闻名，被科研工作者广泛用于科技文档的制作。

TeX 是怎么出现的呢？有时候，新生事物的出现往往伴随着一定的契机和巧合。在 20 世纪 70 年代末，克努斯博士正准备出版其著作《计算机程序设计艺术》时，他发现出版社提供的排版效果不太理想，当时的计算机排版技术也十分粗糙，这严重影响了他的著作的印刷质量，于是，他计划花费几个月的时间开发出一套更有效的文档排版系统，具体的开发目标是实现高质量的书籍排版。

由于克努斯博士此次在数学公式的排版上下足了功夫，就在他启动这项计划不久后，他收到了美国数学协会（American Mathematical Society, AMS）的邀请，克努斯博士在此次邀请中汇报的内容是“基于 TeX 排版，如何让计算机服务于数学”，这次汇报成功吸引了一大批数学家的目光。由于 TeX 在数学公式排版方面的优秀表现，比如数学公式的自动间距调整，TeX 后来摇身一变成为了书写数学公式的“利器”。

为了提升 TeX 的开发质量，克努斯博士悬赏奖励任何能够在 TeX 中发现程序漏洞的人，也就是我们一般认为的“找 bug”。每一个 bug 的奖励金额从 2.56 美元（16 进制的 100 美分）开始，以后每发现一个 bug，都会翻倍，直到 327.68 美元封顶。然而，克努斯博士从未因此而损失大笔金钱，因为 TeX 中的 bug 极少，而真正发现 bug 的人在获得支票后往往因其纪念价值而不愿兑现。

随着时间的推移，TeX 也派生出了很多优秀的软件，其中最著名的派生软件便是 LaTeX。另外，美国数学学会也发布了 TeX 版本的数学公式宏包，其中，以 ams 命



图 1.1: 克努斯博士, 注: 图片来源为克努斯博士的维基百科

名的宏包就有 *amsfonts*、*amsmath*、*amssymb* 等, 这些宏包都可以在 *LaTeX* 上进行使用, 在 *LaTeX* 上使用这些宏包可以编辑出各种数学公式。

参考资料

TeX 的维基百科介绍: <https://zh.wikipedia.org/wiki/TeX>.

1.2 引领浪潮的 *LATEX*

1.2.1 *LATEX* 的出现

LaTeX 是一款高质量的文档排版系统, *LaTeX* 在读法上一般发作 Lay-tek 或者 Lah-tek 的音, 而不是大家普遍认为的 Lay-teks。*LaTeX* 的历史可以追溯到 1984 年, 兰伯特博士作为早期开发者在这一年发布了 *LaTeX* 的最初版本。事实上, *LaTeX* 完全是兰伯特博士的意外所得, 他当年出于自己写书的需要, 在早先发布的文档排版系统 *TeX* 基础上新增了一些特定的宏包, 为了便于自己日后可以重复使用这些宏包, 他将这些宏包进行规整, 于是, 便有了相应的标准宏包 (standard macro package)。谁曾想, 正是这些不经意间开发出来的宏包, 在经过后续封装和发布使用手册之后, 形成了 *LaTeX* 的雏形。

在很长一段时间里, *LaTeX* 的版本其实没有多少大的更新, 从技术层面来说, *LaTeX* 实在没有什么可供更新的地方了, 它最初的面貌已趋近于完美且深入人心。*LaTeX* 的最初版本是由兰伯特博士于上世纪 80 年代初开发出来的, 目前, 广泛使用

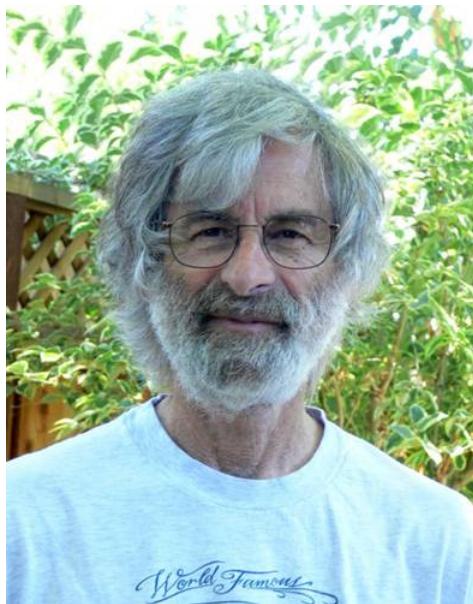


图 1.2: 兰伯特博士, 注: 图片来源为兰伯特博士的维基百科

的版本 LaTeX2e 是在 1994 年发布的, 发布后一直没有大的更新, 甚至发布后的首次更新出现在二十多年后的 2020 年。尽管 LaTeX2e 的后续版本更新工作早在上世纪九十年代初就已经开展了, 但时至今日, 新版的 LaTeX 仍未进入人们的视野。从开发者兰伯特博士的角度来看, 开发 LaTeX 的目的是为了降低 TeX 的使用门槛、更容易地发挥 TeX 强大的排版功能, 提供一款高质量、解释性强的计算机程序语言, 所以 LaTeX 最初的风格就是精简, 这也是为什么 LaTeX 在日后可供提升的地方不是很多的原因。

1.2.2 *LATEX* 的特点

由于种种原因, 时至今日, TeX 几乎淡出了人们的视线, 不过我们现在依旧能看到: 在使用 LaTeX 制作文档时, 通常需要创建一个以.*tex* 为拓展名的文件。对于很多人来说, 日常制作各类文档的首选可能是 Word 等软件, 它简单好用、所写即所见, 但当我们制作几十页甚至上百页的文档时, Word 的劣势就会展露无疑, 因为我们需要投入大量的时间和精力来对文档内容进行排版。反观 LaTeX, 它对文档的排版都是自动完成的, 我们根本不需要像 Word 那样完全手动调整格式, 另外, 使用 LaTeX 插入各种图形、表格、公式、文献时, 相应的索引出错的可能性也非常小, 这些优点都是 Word 所无法比拟的。

在上个世纪 80 年代和 90 年代, LaTeX 的用户群体非常庞大, 然而, 在世纪之交, 随着微软推出的一系列 Windows 操作系统快速发展, 例如红极一时的 XP 系统,

相应的办公软件 Microsoft Office 也以其便捷性吸引了人们的视线，致使大量 LaTeX 用户转而使用 Microsoft Office。即便如此，时至今日，LaTeX 的用户群体依旧十分庞大，这主要得益于 LaTeX 强大的文档排版能力，虽然 LaTeX 复杂的语法结构、不容配置的编译环境让很多初学者望而却步，但 LaTeX 能让用户更专注于内容创作，而非锦上添花的“排版”，这一显著特点契合了人们对质量和效率的追求，使得 LaTeX 在文档排版、论文撰写等方面占有重要地位。在此基础上，具体来说，使得 LaTeX 历久弥新的关键可以归纳为以下五点：

1. LaTeX 是专门用于制作文档的计算机程序语言。在众多计算机程序语言中，LaTeX 可以制作排版连贯性极好的专业文档。
2. 独特的创作方式。尽管 LaTeX 沿用了 TeX 排版系统，但使用 LaTeX 制作文档时，内容创作和文档生成却是分开的，有需要的时候，我们可以预览创作的文档。因此，在创作的过程中，创作者不再像使用办公软件 Word 那样，既要关注创作内容，又要同步关注繁琐的排版和格式，使用 LaTeX 制作文档能在真正意义上让创作者专注于创作内容本身。更值得一提的是，当文档篇幅较大时，使用 LaTeX 无疑会让我们节省大量的时间和精力。
3. 简单的逻辑结构。使用 LaTeX 制作文档时，创作者可以通过一些非常简单的逻辑结构进行创作，如 chapter (章)、section (节)、table (表格)。因此，LaTeX 的使用门槛并不像真正的程序语言那么高。很多人或许在使用 LaTeX 的过程中都不会用到 for 等基本的循环语句。
4. 对数学公式以及特殊符号的支持程度。众所周知，LaTeX 在开发之初，是作为数学和计算机相关研究人员的创作工具，这类群体喜欢使用 LaTeX 的原因无外乎是 LaTeX 可以通过一些简单的代码生成复杂的数学表达式和特殊符号。
5. 编译以.tex 为拓展名的 LaTeX 文件后会得到一个 PDF 文档，PDF 文档不存在跨平台、兼容性等问题，可以在各种操作系统上打开。

当然，除了上述五点，实际上可能还有十分重要的一点，那就是 LaTeX 能够制作各类文档，从科技论文、技术报告、著作、学位论文、幻灯片甚至到科技绘图一应俱全，当然它也支持嵌入图片、绘制图形、设计表格、插入参考文献等。

从 LaTeX 的出现到当下，它已经形成了一套非常高效的文档制作环境：

- 文档类型 (document class)。文档类型是文档排版样式的基调，这些类型包括文章 (article)、报告 (report)、幻灯片 (beamer) 等，在.tex 文件中申明文档类型后，我们就可以开始文档创作了。

- 宏包 (package)。它是 *LaTeX* 中的重要辅助工具，也可以把它理解为一般意义上的工具包。在使用时，调用宏包的基本命令为 `\usepackage{}`，举例来说，包含颜色命令的宏包为 *color*，其调用语句为 `\usepackage{color}`。随着 *LaTeX* 的发展，越来越多的宏包被开发出来，这些宏包能满足特定的需求（如制表、插图、绘图），同时也能让 *LaTeX* 代码变得更加简洁，我们只需要用简单的 `\usepackage{}` 命令就能调用所我们需要用到的宏包。
- 模板 (template)。*LaTeX* 的发展催生了很多视觉和审美效果极好的模板，包括论文模板、幻灯片模板、报告模板甚至著作模板，这些模板在一定程度上能减少创作者在文档排版上的时间开销，也有很多学术刊物会给投稿作者提供相应的 *LaTeX* 模板。

通过对比 *LaTeX* 和 Word，我们还会看到：

- 第一，*LaTeX* 的.tex 源文件是无格式的，编译之后，根据特定的模板和指定的格式形成最终的 PDF 文档，因此，使用 *LaTeX* 制作各类文档能够很方便地切换模板和修改格式；
- 第二，*LaTeX* 对公式、图表以及文献引用的支持是 Word 所无法比拟的，尤为特殊的是，当文献数量达到上百篇时，在 Word 中修改参考文献可能是“牵一发而动全身”，费时耗力，而 *LaTeX* 根据已经整理好的.bib 文件可自动完成文献引用和生成。

1.2.3 *LATEX* 编辑器

实际上，配置 *LaTeX* 环境包括两部分，即编译器和编辑器，对应的英文表达分别是 editor 和 complier，两者不是一回事。*LaTeX* 编译器又称为 *LaTeX* 编译工具，可根据系统安装相应的编译工具：

- Linux 系统：可安装 TeX Live，该编辑器拥有 *LaTeX* 编辑器；
- Mac OS 系统：可安装 Mac TeX，该编译器拥有完整的 TeX/*LaTeX* 环境和 *LaTeX* 编辑器；
- Windows 系统：可安装 MiKTeX 或 TeX Live，两者都拥有完整的 TeX/*LaTeX* 环境和 *LaTeX* 编辑器。

目前，我们可以接触到很多 *LaTeX* 编辑器，这些编辑器的界面大致有两部分组成，即 *LaTeX* 源码编译区域和 PDF 文档预览区域。前面也提到了几款 *LaTeX* 编译器，但如果想要提高 *LaTeX* 的使用体验，以下几款 *LaTeX* 编辑器比较受人推崇：

- TeXworks：这是 TeX Live 自带的一款轻量级编辑器。
- TeXstudio：这款编辑器集代码编译与文档预览于一身。
- WinEdt：这是 CTeX 自带的一款编辑器。
- VS Code：这是微软推出的一款免费文本编辑器，功能包括文本编辑、日常开发等。
- Atom：这是一款开源的跨平台编辑器（GitHub 网址为 <https://github.com/atom/atom>），支持多种程序语言。

1.3 应运而生的在线系统

1.3.1 LATEX 在线系统的出现

上世纪 80 年代，*LaTeX* 作为一件新生事物，在发布之初便引起了人们极大的兴趣，虽然在制作文档方面拥有很多办公软件都无法比拟的强大优势，尤其在数学公式编写及高效排版上具有很大优势，但是由于其较高的使用门槛（使用计算机程序语言进行编译）和安装成本（本地安装需要花费大量的时间配置相应的环境），在很长一段时间里，*LaTeX* 主要用户都是科研工作者。然而，*LaTeX* 在线系统的出现已实实在在地改变了这一尴尬局面。

随着信息技术快速发展、互联网深度普及，人们的工作生活方式也在发生着很大改变，很多过去安装在本地的操作软件都被搬到了浏览器上，人们无须在个人计算机上安装各类办公软件就能进行办公，这带来了极大的便利。不过这类在线系统也存在一些先决条件，例如，出于计算资源方面的考虑，通常要求在线系统的类型不能是计算密集型，因为计算密集型的在线系统往往需要大量的计算资源作支撑。反观 *LaTeX*，尽管我们可以认为 *LaTeX* 是一种计算机程序语言，但实际上，其对计算资源的需求并不是很大。

在过去，受网速限制，使用线上系统几乎是一件难以想象的事。然而，在线系统的兴起并非空穴来风，一方面是目前的网速已经跟过去发生了质的变化，另一方面则是上网成本在急剧降低，互联网触手可及，已经成为人们日常生活和工作中不可或缺的一部分。以前，我们可能已经习惯了在本地计算机上安装和使用各类软件或者集成开发环境，不过以 *LaTeX* 为例，在本地计算机上安装的集成开发环境也有很多缺陷：

- 第一，我们需要为安装 *LaTeX* 编辑器腾出很大的存储空间；

- 第二，某些特定的宏包需要额外安装和配置，但安装过多宏包之后又会使 LaTeX 变得很臃肿，甚至是不友好；
- 第三，当我们在本地计算机使用 LaTeX 制作文档时，我们很难与合作者进行协同创作。

在这个背景下，一些成熟的 LaTeX 在线系统逐渐走进人们的视野，并受到很多用户的喜爱，其中，最为著名的 LaTeX 在线系统便是 *overleaf.com*。这些 LaTeX 在线系统不仅支持各种语言、各种拓展宏包等复杂的 LaTeX 环境，同时也支持实时编译和实时预览编译后的文档，就算是换一台电脑，也丝毫不会影响创作过程，创作完成之后，可以选择下载压缩文件包（如.zip），也可以只导出 PDF 文档，毫无疑问，这些人性化的设计都是为了让 LaTeX 更加便捷和高效。除此之外，现有的 LaTeX 在线系统还提供大量的 LaTeX 模板库，科技论文、毕业设计、幻灯片、海报、简历等参考模板一应俱全，就连 LaTeX 使用文档也数不胜数。

在线系统 overleaf

Overleaf 是一个初创的科技企业，它的主要业务是构建现代化协作创作工具，即 LaTeX 在线系统，旨在让科学研究变得更加便捷和高效。目前，Overleaf 已合并另一款著名的 LaTeX 在线系统 ShareLaTeX，在全球范围内拥有超过 600 万用户，这些用户大多是来自于高校和研究机构的研究人员、老师以及学生，只要打开网址 [overleaf.com](https://www.overleaf.com/about)，用户无需在本地计算机配置 LaTeX 环境就可以创建各种 LaTeX 项目。

关于 Overleaf 的介绍可参考 <https://www.overleaf.com/about>。

1.3.2 LATEX 在线系统的特点

以 Overleaf 为例，该 LaTeX 在线系统往往具备以下几点特征：

- 免费和开源。可以免费注册和使用，不用下载和安装 LaTeX 编辑器，这一点对于初学者来说无疑是非常友好的；
- 使用简单。不管是在计算机、手机还是其他终端上，我们只需要使用浏览器打开 [overleaf.com](https://www.overleaf.com) 就可以开始创作，另外，由于 Overleaf 界面非常简洁，所以用户使用起来也非常便利；
- 支持实时在线编辑。有各类 LaTeX 插件，编辑功能十分完善，且具有实时编译和预览功能；

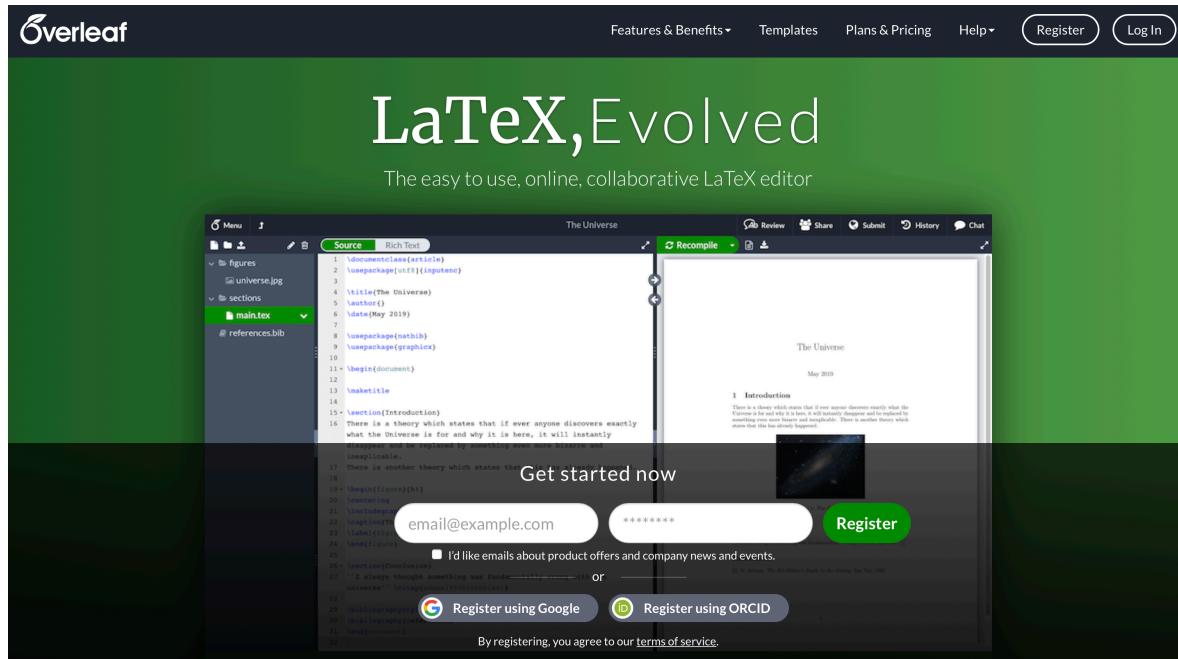


图 1.3: Overleaf 首页, 图片来源于 Overleaf 官网

- 支持在线协作。创作文档时, 我们可以将文档项目分享给合作者进行协作, Overleaf 支持实时编译, 不会出现版本控制混乱等问题;
- 支持双向定位。可以在 LaTeX 代码与 PDF 文档内容之间进行双向定位;
- 提供丰富的模板库。Overleaf 有着非常庞大的模板库, 不仅有正式的学术论文、学位论文和书籍的参考模板, 还有很多美观的报告、简历、幻灯片模板。就论文写作来说, 用户可以在 Overleaf 官网找到众多期刊的 LaTeX 模板, 根据使用说明, 用户很容易就能用于撰写自己的论文;
- 提供大量的帮助文档。LaTeX 提供了齐全的帮助文档, 从 LaTeX 快速入门、基础操作到编译数学公式, 应有尽有、一应俱全, 且这些文档内容具有很强的实操性。

LaTeX 在线系统的出现大大降低了 LaTeX 的使用门槛, 也为用户省去了繁琐的安装和配置过程。其实, LaTeX 在线系统的出现并非个例, 很多办公软件为迎合用户需求和时代发展趋势, 陆续转变了产品研发思路, 包括微软在线 Office 系统、腾讯在线文档等在内的很多在线系统都走进了人们的视野, 这些在线系统能够在线备份、满足人们对随时随地办公的需求, 在确保便捷和高效的同时, 在线和共享的理念正在潜移默化地影响着人们的办公模式。

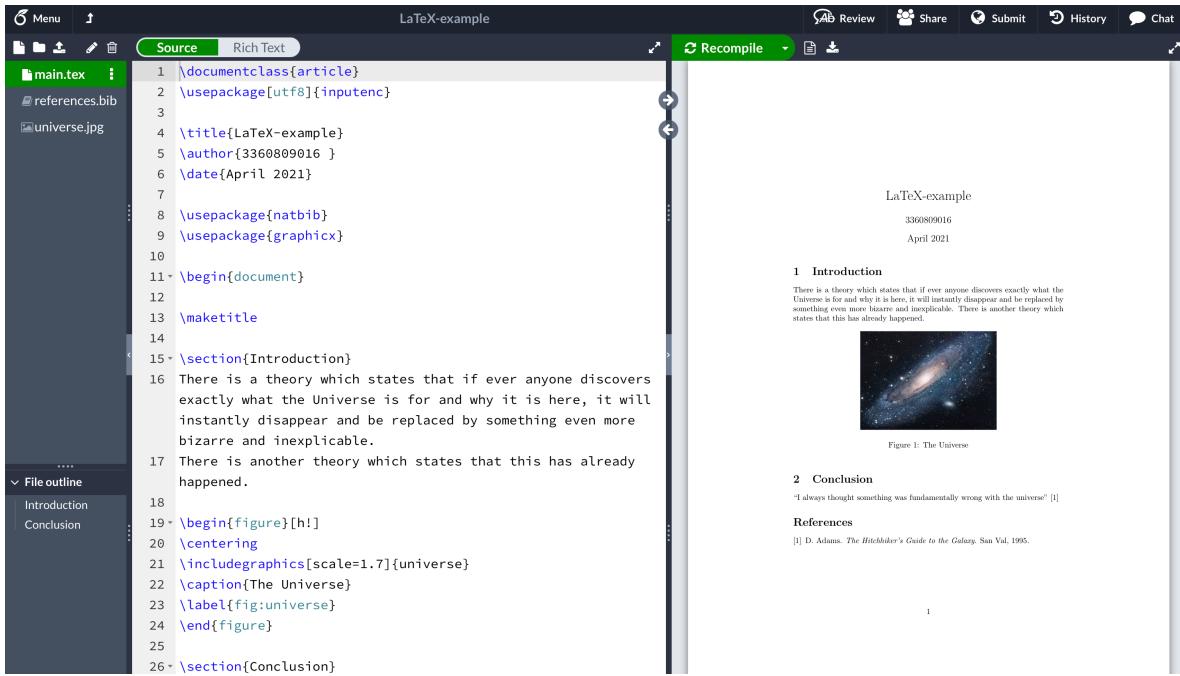


图 1.4: Overleaf 编辑器界面，主要由代码区域和文档预览区域组成

1.4 L^AT_EX 问答社区

在 L^AT_EX 刚被发布和推出的那个年代，相关资源如使用手册、教程等远没有今天这么丰富，同时，获取渠道也没有今天这么便捷。在互联网触手可及的今天，我们能通过一个浏览器访问到各种相关学习素材，遇到代码 bug，也能在一些专业的问答社区找到最佳的解决方案。毫无疑问，对于今天的我们来说，如何利用好互联网资源至关重要。

1.4.1 问答社区的介绍

对于从事计算机相关的技术人员来说，专业的技术问答社区往往是不可多得的资源，它能帮助很多技术人员提升个人编程能力、学习新技术，同时解决一些技术困扰，例如解决编程时遇到的 bug。对于各类计算机程序语言而言，*Stack Exchange* 作为一个著名的技术问答社区，覆盖了大量编程相关的技术问题和优质回答，就算是一些较为细致的问题，我们往往都能找到想要的答案。*Stack Exchange* 问答社区按计算机程序语言类型进行划分，我们所关心的 L^AT_EX 问答通常被分配在 TeX Stack Exchange 社区（网址为 <https://tex.stackexchange.com/>）。截至目前，TeX Stack Exchange 已经覆盖了关于 TeX、L^AT_EX 以及其他排版系统的用户在使用中遇到的诸多问题，并且这些问题多与 L^AT_EX 相关。

Stack Exchange

网址为 <https://stackexchange.com/>，它与 Stack Overflow 问答社区在全球范围内拥有广泛的用户群体

除了 Stack Exchange 这种涵盖了多种计算机程序语言的技术问答社区，LaTeX forum 社区 (<https://latex.org/forum/>) 是一个专门面向 LaTeX 的技术交流平台，它拥有非常活跃的用户群体和丰富的问答资源，并拥有超过 10 万篇分门别类的技术帖子，我们可以根据浏览量从该平台一览高频访问问题。

实际上，不管是 LaTeX 初学者还是高级用户，在遇到 LaTeX 使用问题时，去问答社区寻找解决方案都是一种非常有效的方式。TeX Stack Exchange 社区非常活跃，每天都会有大量关于 LaTeX 的问题和回答，且每个问题下面的回答都会根据用户的认可度进行排序，回答一般比较细致。

1.4.2 高频访问问题

顾名思义，高频访问问题是指访问量较高的问题。LaTeX forum 社区已经将众多问答帖子进行了分门别类整理，针对某一特定话题，展开内容即可看到各类问题的访问情况。

1.5 关于 *LATEX* 的开源项目

实际上，不管是 TeX 还是 LaTeX，它们作为计算机程序语言是完全开源的，我们可以轻松获取到 TeX 和 LaTeX 的源代码。近些年来，互联网催生了一些用户体验以及评价很好的开源社区，比如当下非常受欢迎的开源平台 GitHub，它们都在实实在在地影响着我们对于计算机技术开放的态度。开源社区有很多优质的“仓库”(repository)，这些仓库会提供诸如源代码、工具、案例甚至使用文档。与 LaTeX 在线系统相似的是，这些开源社区非常支持协作功能，在有一些优质的仓库中，我们或许能看到几十甚至上百个贡献者，“汇聚集体智慧”也是这些开源社区广受开发者青睐的一个重要原因。

1.5.1 开源社区 GitHub

GitHub 是一个面向开源项目及私有项目的云端托管平台，官方网址为 <http://github.com/>，旨在为开发者储存、管理代码以及控制代码的更新，只支持 Git 作为唯一的版本库格式进行托管。GitHub 已经走进人们的视野已经有十几年了，它于 2008 年 4 月 10 日

正式上线，除了 Git 代码仓库托管和基本的网页管理界面以外，还提供了在线文件编辑器、版本控制、协作报表等功能。截至 2021 年初，GitHub 在全球范围内拥有的注册用户已经超过 5000 万，托管项目的总量更是超过 1 亿，是很多开发者远程协作的重要工具。2018 年 6 月，GitHub 被微软以 75 亿美元的价格收购。

GitHub 本着开源、共享、协作的理念正影响着开发社区生态，吸引了包括微软和谷歌等国外技术巨头在这里进行开源，另外，我们也能在 GitHub 上看到一些国内互联网企业的身影。实际上，GitHub 开源社区的项目类型及展现形式也异常丰富，既有大量的应用型项目，也有很多研究探索型项目。

为了方便开发者追踪整个开源社区的开发动态，GitHub 根据计算机程序语言类型提供了趋势分析，这个功能称为 GitHub trending，它会定期更新趋势榜、帮助开发者追踪当下较为流行的 GitHub 仓库。

GitHub 开源社区提供了很多关于 LaTeX 的优质开源项目，这些项目使得 LaTeX 的用户群体越来越广泛。实际上，LaTeX 在线系统 Overleaf 也是开源的，它托管在 GitHub 上，网址为 <https://github.com/overleaf/overleaf>。

为便于读者认识和掌握 LaTeX，以下简单归纳一下关于 LaTeX 的各类开源项目。需要注意的是，优质的开源项目是非常多的，限于篇幅，这里只列举一部分，感兴趣的读者不妨在 GitHub 平台上探寻更多优质项目。

1.5.2 学位论文 *LATEX* 模板

在 GitHub 开源社区中，我们可以找到很多国内外高校官方和非官方的学位论文 LaTeX 模板，学位论文类型包括本科毕业设计、硕士学位论文和博士学位论文等。以下将列举一部分国内高校的学位论文 LaTeX 模板开源项目：

- 开源项目 <https://github.com/tuna/thuthesis> 提供了清华大学学位论文 LaTeX 模板，模板库涵盖本科综合论文训练、硕士论文、博士论文以及博士后出站报告，截至目前，已在 GitHub 上获得超过 3000 次标星。
- 开源项目 <https://github.com/mohuangrui/ucastheses> 提供了中国科学院大学学位论文 LaTeX 模板，包括本科、硕士和博士学位论文模板，截至目前，已在 GitHub 上获得超过 2000 次标星。
- 开源项目 <https://github.com/mohuangrui/ucastheses> 提供了中国科学院大学学位论文 LaTeX 模板，包括本科、硕士和博士学位论文模板，截至目前，已在 GitHub 上获得超过 2000 次标星。
- 开源项目 <https://github.com/ustctug/ustctheses> 提供了中国科学技术大学学位

论文 LaTeX 模板，包括本科毕业论文（设计）和研究生学位论文，截至目前，已在 GitHub 上获得近 1000 次标星。

- 开源项目 <https://github.com/TheNetAdmin/zjuthesis> 提供了浙江大学学位论文 LaTeX 模板，包含本科、硕士和博士学位论文模板，也提供了英文版的硕博学位论文模板，截至目前，已在 GitHub 上获得近 1000 次标星。
- 开源项目 <https://github.com/dustincys/hithesis> 提供了哈尔滨工业大学学位论文 LaTeX 模板，包括本科、硕士、博士开题、中期和毕业论文，也提供了博后出站报告和英文毕业论文格式，截至目前，已在 GitHub 上获得近 1000 次标星。
- 开源项目 <https://github.com/x-magus/ThesisUESTC> 提供了电子科技大学毕业论文 LaTeX 模板，截至目前，已在 GitHub 上获得超过 500 次标星。

上述开源项目提供的学位论文模板绝大多数支持在 LaTeX 在线系统 overleaf.com 上直接进行编辑和使用。

1.5.3 L^AT_EX 绘制图形

目前，LaTeX 有很多关于绘制图形方面的宏包，这些宏包会提供图形的基本元素、颜色等。由于 LaTeX 绘制出来的图形可以以 PDF 文档保存，所以一般不会出现分辨率不足等问题。这里对 LaTeX 绘制图形相关的一些开源项目稍作整理：

- 工具类：
 - BayesNet*: 用于绘制贝叶斯网络、图模型和因素图形的 TikZ 库。开源网址为 <https://github.com/jluttine/tikz-bayesnet>。一般而言，使用 BayesNet 绘制特定图形时，需要使用以下两行命令调用该库：\usepackage{tikz} 和 \usetikzlibrary{bayesnet}
 - matlab2tikz*: 将由 Matlab 代码绘制的图形转换成 PGFPlots 图形。开源网址为 <https://github.com/matlab2tikz/matlab2tikz>。
 - tikzplotlib*: 将由 Python 绘图工具 matplotlib 绘制的图形转换成 PGFPlots 图形。开源网址为 <https://github.com/nschloe/tikzplotlib>。
 - svg2tikz*: 将 SVG 图形转换成 TikZ 代码。开源网址为 <https://github.com/xyz2tex/svg2tikz>。
- 实例类：
 - TeXample*: 该项目提供了大量的 TikZ 绘图实例，开源网址为 <https://texample.net/tikz/>。

- 开源项目 <https://github.com/walmes/Tikz> 提供了大约 200 个关于统计学的 TikZ 绘图实例。
- 开源项目 <https://github.com/MartinThoma/LaTeX-examples/tree/master/tikz> 提供了大约 350 个 TikZ 绘图实例。
- 开源项目 <https://github.com/PetarV-/TikZ> 提供了大量的 Tikz 绘图实例，囊括了各类神经网络模型的图形。
- TeX Stack Exchange 社区中提供的可视化效果很好的 TikZ 科技绘图实例，网址为 <https://tex.stackexchange.com/questions/158668/nice-scientific-pictures-show-off>。
- 开源项目 <https://github.com/FriendlyUser/LatexDiagrams> 提供了大量的 TikZ 绘图实例，包括流程图、图模型。
- 开源项目 https://github.com/alemelis/tikz_drawings 提供了一些 TikZ 绘图实例。

1.5.4 *LATEX* 制作简历

LaTeX 可用于制作各类文档，这也包括简历。

- 开源项目 <https://github.com/sb2nov/resume> 提供了简历模板。
- 开源项目 <https://github.com/xdanaux/moderncv> 提供了 moderncv 简历样式。
- 开源项目 <https://github.com/jankapunkt/latexcv> 提供了一些美观大方且容易使用的简历模板，支持包括中文在内的多种语言编译。
- 开源项目 <https://github.com/hijiangtao/resume> 提供了中文简历的模板。

1.5.5 *LATEX* 制作幻灯片

- 开源项目 <https://github.com/matze/mtheme> 提供了现代风格主题的幻灯片模板。
- 开源项目 <https://github.com/wzpan/BeamerStyleSlides> 提供了很丰富的模板，包含了 PowerPoint 和 Keynote 两套格式。
- 开源项目 <https://github.com/XiangyunHuang/awesome-beamers> 提供了多种制作幻灯片的模版，同时也支持中文制作。

- 开源项目 <https://github.com/josephwright/beamer> 提供了多种风格的幻灯片模板。

1.5.6 LATEX 制作海报

- 开源项目 <https://github.com/jkjaer/aauLatexTemplates> 提供了 beamer 海报模板。
- 开源项目 <https://github.com/mloesch/baposter> 提供了简洁的海报模板。
- 开源项目 <https://github.com/HolgerGerhardt/TeXTemplates> 提供了适合做学术报告的海报模板，同时也包含幻灯片模板。

1.5.7 推荐指导本书的开源项目

开源网址为 <https://github.com/xinychen/latex-cookbook>。同时该作者还有图形绘制和幻灯片制作的开源项目：

- LaTeX 图形绘制：<https://github.com/xinychen/awesome-latex-drawing> 提供大量绘图实例，实例包括贝叶斯网络、框架图、流程图、机器学习模型示意图。
- LaTeX 幻灯片制作：<https://github.com/xinychen/awesome-beamer> 提供大量使用 Beamer 制作幻灯片的实例。

参考资料

开源项目 <https://github.com/xiaohanyu/awesome-tikz>。

1.6 LATEX 制作中文文档

LaTeX 最初只提供英文的编译环境，随着其在文档编辑领域的优势越来越深入人心，更多国家的学者希望用到这一工具，因此出现了许多支持各国语言的工具包或编写环境，作为全世界使用人数最多的语言，LaTeX 有 *CJKutf8* 宏包、*CTEX* 宏包等多种方式可以实现中文文档编辑，均能在开源网站 Overleaf 中调用并实现中文文档制作。

1.6.1 使用 CJKutf8 宏包

CJKutf8 宏包提供了两种中文简体字体制作中文文档：使用 `\begin{CJK}{UTF8}{gbsn}` 和 `\end{CJK}` 环境将以宋体（gbsn）制作文档，而使用 `\begin{CJK}{UTF8}{gkai}` 和 `\end{CJK}` 环境则以楷体（gkai）制作文档内容。在默认的 pdfLaTeX 编译环境中即可得到文档编译结果。

【例】使用 *CJKutf8* 红包制作中文文档：

```

1 \documentclass{article}
2 \usepackage{CJKutf8}
3
4 \begin{document}
5
6 \begin{CJK}{UTF8}{gbsn} %字体是gbsn
7 你好，LaTeX!。
8 \end{CJK}
9
10 \end{document}

```

Listing 1.1: CJKutf8 示例

1.6.2 使用 CTEX 宏包

CJKutf8 宏包只提供了两种字体，可选择的余地太小。如果想要使用更丰富的字体编辑 Latex 中文文档，可以调用 *CTEX* 宏包，并设置 UTF8 选项使其支持 utf-8 编码。在 Overleaf 中使用 *CTEX* 宏包时，需要先将编译环境从 *pdfLaTeX* 调整为 *XeLaTeX*。【例】使用 *CTEX* 宏包制作中文文档：

```

1 \documentclass[UTF8]{ctexart}
2
3 \begin{document}
4
5 {\kaishu 你好，LaTeX! (楷体) }
6
7 {\songti 你好，LaTeX! (宋体) }
8
9 {\heiti 你好，LaTeX! (黑体) }
10

```

```

11  {\fangsong 你好, \LaTeX! (仿宋) }。
12
13 \end{document}

```

Listing 1.2: CTEX 示例

目前在 Overleaf 上已经出现了许多中文文档 *LaTeX* 模板，除了一些学位论文模板，一些中文学术期刊如《计算机学报》也提供了科技论文的 *LaTeX* 模板。

期刊模板：

- 《中国科学:信息科学》<https://www.overleaf.com/project/5e99712a0916c900018d11af>
- 《中国科学:信息科学》<https://www.overleaf.com/project/5e99712a0916c900018d11af>

中文学位论文模板：

- 《浙江大学学位论文模板》<https://www.overleaf.com/project/610fa05007d0073d5405a04f>
- 《武汉大学博士学位论文模板》<https://www.overleaf.com/project/610fa09e07d007fa5605a1e9>
- 《武汉大学博士学位论文模板》<https://www.overleaf.com/project/610fa09e07d007fa5605a1e9>
- 《南京大学研究生毕业论文模板》<https://www.overleaf.com/project/610fa1d007d00704c305a3eb>

另外，开源项目 <https://github.com/MCG-NKU/NSFC-LaTex> 提供了国家自然科学基金申报书的 *LaTeX* 模板。

第二章 LATEX 基础介绍

LaTeX 作为专门用于制作文档的计算机程序语言，其语法规则是由命令 (command) 和环境 (environment) 构成，基于一些基本命令，如\usepackage{}，我们能调用制作文档所需的宏包。一般而言，在制作文档时，LaTeX 的代码结构分为前导 (preamble) 和主体 (body) 两部分，前导部分的代码主要用于申明文档类型、排版样式、所使用的宏包等；主体部分主要用于确定标题、章节、目录等文章结构及创作文档内容。

掌握一些常用命令之后，我们便可上手编辑简单的文档。从功能上来说，LaTeX 能满足人们对文档制作多样性的要求。我们可以用 LaTeX 制作各类文档，包括科技文档、技术报告、学位论文、书籍著作、幻灯片、个人简历、信件以及海报。不同的文档类型在文档大小、排版、章节样式方面略有不同，所以在使用 LaTeX 制作文档时，我们首先需要对文档类型进行申明，然后再进行文档内容的创作。

本章节要介绍的内容主要包括：Latex 语法规则、Latex 代码结构、文档类型的介绍、简单文档的制作以及全局格式的设置。

2.1 LATEX 语法规则

对于任何一种计算机程序语言，严格的语法规则是让程序语言实现功能的保障。不同于一般性的计算机程序语言，LaTeX 的语法规则非常简单，是由命令和环境构成。在 LaTeX 中，不管是命令还是环境，都离不开计算机符号反斜线\，例如调用宏包会用到 \usepackage{宏包名称}，这里的花括号 {} 也是非常常用的一个符号。在调用宏包后，命令的用法都是由宏包约定的。

2.1.1 命令

LaTeX 中有很多命令，它们用法大同小异，功能却千差万别。既有申明文档类型的命令，如\documentclass{article}，也有输入特殊符号的命令，例如\copyright。一般而言，LaTeX 中的命令通常由三部分组成\命令名 [可省略参数]{不可省略参数}，具

有以下特点：

- 通常都是以反斜线作为开始，通过紧跟的既定字符（命令名）实现相应的功能，例如\LaTeX 和\copyright 可生成特殊字符。
- 一些命令需要申明一些参数，通过大括号中的不可省略参数进行申明，例如\color{blue} 命令中需要申明具体的颜色名称。
- 一些命令包括一些选项和一些参数一般情况下是默认的，有需要时可以通过中括号中的可省略参数进行调整，例如在\documentclass[a4paper]{article} 中，中括号 [] 作为额外的选项，既可以自行设置，也可以选择默认设置。
- 有些命令可以用反斜线作为终止，例如\copyright。

2.1.2 环境

这里所说的“环境”是指编译环境，使用 LaTeX 时，当我们想要编译出期望的效果，例如列表、插入图形、制作表格，我们就需要用到一些环境。举例来说，我们可以通过

```

1 \begin{itemize}
2   \item item 1 % 条目1
3   \item item 2 % 条目2
4 \end{itemize}
```

制作一个简单的无序列表，这里的 *itemize* 表示无序列表环境，\begin{...} 和\end{...} 表示环境的初始化和结束。当然，这些环境并非“一成不变”，我们也可以设置一些参数，从而改变编译之后的文档效果，例如，可以通过

```

1 \begin{spacing}{1.3}
2   paragraph 1 % 第1段
3
4   paragraph 2 % 第2段
5 \end{spacing}
```

将两段话之间的行间距设置为 1.3 倍。

【例】 使用基本命令\documentclass{article} 和文档环境\begin{document} 和\end{document} 创建一个简单文档。

```

1 \documentclass{article}

2

3 \begin{document}

4

5 Hello, LaTeXers! This is our first LaTeX document.

6

7 \end{document}

```

2.1.3 宏包

宏包是 LaTeX 的重要组成部分，用来扩展和增强 LaTeX 的功能，是支撑 LaTeX 实现一系列复杂文档编辑和排版的关键所在。宏包与 LaTeX 的关系和浏览器插件与浏览器的关系类似，通过调用不同的宏包可以实现一些复杂排版功能，例如插入表格、插入公式和特殊符号、插入程序源代码、设置文档样式等。一个宏包通常会提供一组 LaTeX 命令，有些特殊命令只能在调用宏包后使用。在 LaTeX 中，调用宏包的形式大同小异，如果想要使用某一特定宏包，最简单的办法就是用\usepackage{宏包名} 命令对相应宏包进行调用。

【例】 使用\usepackage{color} 命令调用颜色宏包调整文本字体颜色。

```

1 \documentclass{article}

2 \usepackage{color} % 调用颜色宏包

3 \begin{document}

4 \textcolor[rgb]{1,0,0}{Hello, LaTeXers! This is our first LaTeX document.}

5 \end{document}

```

2.2 LATEX 代码结构

在使用 LaTeX 进行文档编辑时，我们通常在拓展名为.tex 的源文件中书写代码，然后通过编译，生成一个 PDF 格式的文档。LaTeX 源文件的代码结构主要包含两个部分，即前导代码（preamble）和主体代码（body），其结构示例如下：

```

1 \documentclass[] {}

2

```

```

3   ..... % 前导代码 (preamble)
4
5   \begin{document}
6
7   ..... % 主体代码 (body)
8
9   \end{document}

```

2.2.1 前导代码

前导代码是指从源文件第一行代码到\begin{document}之间的所有命令语句，一般为 LaTeX 代码的第一部分。在前导代码部分，既可设置文档类型全局参数，包括字体大小、纸张大小、文字分栏、单双面打印设置等，也可声明主体代码中需要用到的宏包，如插入图形、新增表格所要用到的宏包。当全局格式没有特殊申明时，前导代码中的文档类型申明语句可以简写成\documentclass{B}，其中，位置 B 的作用在于申明文档类型，如 article（常规文档）、book（书籍）、report（报告）、beamer（幻灯片）等。

下面以常规文档（article）为例，简要介绍各常用全局参数的设置方式。

字体大小

article 类型文档字体大小默认为 10pt，可在\documentclass[可省参数]{article}的中括号中根据需要设置成 11pt 和 12pt。

通常，我们按照\documentclass[fontsize = 12pt]{article}这样的形式设置文档参数，有时候，为了方便起见，可以把文档类型申明做一定的简写，如\documentclass[12pt]{article}。需要注意的是，字体大小设置中的基本单位 pt 是英文单词 point 的缩写，是一个物理长度单位，指的是 72 分之一英寸，即 1pt 等于 1/72 英寸。

纸张大小

article 类型文档纸张大小默认为 letterpaper，同样可在\documentclass [可省参数]{article}的中括号中根据需要设置成 a4paper、a5paper、b5paper、legalpaper 和 executivepaper。

文字分栏

article 类型文档的文字分栏默认为 *onecolumn* (不分栏), 也可以使用 *twocolumn* 参数设置为 *twocolumn* (两栏)。

单双面打印设置

article 类型文档打印时默认单面打印, 同样可以使用\documentclass[可省参数]{article} 中括号中的可选参数, 通过添加 *twoside* 参数进行双面打印的设置。

文档属性设置小结

基于上面介绍可供调整的参数, 我们可以进行任意无序组合, 例如\documentclass [a4paper, 11pt, twoside]{article} 对应着文档类型为 article、纸张大小为 A4、字体大小为 11pt 的双面文档。

2.2.2 主体代码

主体代码为\begin{document} 及\end{document} 之间所有的命令语句和文本, 一般由文档的创作内容构成, 按照一般书写顺序, 主要包含文档标题、目录、章节、图表及具体文字内容等, 通常配合一些基本命令的使用, 来形成我们期望的文档。下面给出了一个简单例子, 让读者对如何制作一个包含标题、章节及其文字内容的简单文档能有一个比较粗略的认识, 更具体的语法命令我们将在后续章节中依次介绍。

```
1 \documentclass{article}
2 \title{LaTeX cook-book}
3
4 \begin{document}
5 \maketitle
6 \section{Introduction}
7
8 Hello, LaTeXers! This is our first LaTeX document.
9
10 \end{document}
```

2.3 文档类型介绍

在 L^AT_EX 代码结构中，申明文档类型往往是制作文档的第一步，也是最基本的第一步。事实上，不同文档类型对应的文档样式略有不同，但制作不同类型的文档时，L^AT_EX 中的绝大多数命令和环境却是通用的，完成对文档内容的创作后，使用文档类型的申明语句可以让我们在不同类型的文档间切换自如。

2.3.1 基本介绍

对 L^AT_EX 熟悉的读者会知道，L^AT_EX 实际上支持非常多的文档类型，例如，撰写科技论文会用到的 *article*（常规文档）类型、制作演示文稿会用到的 *beamer*（幻灯片）类型。在众多文档类型中，常见的文档类型包括 *article*（常规文档）、*report*（报告）、*book*（书籍）、*beamer*（幻灯片）等，如果使用支持中文编译的 *ctex* 文档类型，还会有 *ctexart*（中文常规文档）、*ctexrep*（中文报告）、*ctexbook*（中文书籍）等，文档类型会直接决定整个文档的样式和风格。

使用 L^AT_EX 制作文档时，申明文档类型是作为前导代码，其一般格式为：

```
1 \documentclass[A]{B}
```

在这一申明语句中，位置 A 的作用主要是设置控制全文的文档参数，我们可以调整全文的字体大小、纸张大小、分栏设置等，因为各种文档类型都有一整套的默认参数，所以一般情况可以省略掉位置 A。在位置 B，我们需要键入特定的文档类型，例如，\documentclass[a4paper, 12pt]{article} 即表示申明一个纸张大小为 A4、字体大小为 12pt 的常规文档。

以下将逐一介绍比较常用的三种文档类型，包括 *article*（常规文档）、*report*（报告）、*book*（书籍）。其中，*report* 和 *book* 这两种文档类型的文档结构是一致的，可以使用的结构命令有\part{}, \chapter{}, \section{}, \subsection{}, \subsubsection{}, \paragraph{}, \ subparagraph{}, 举例来说，包含颜色命令的而 *article* 文档类型中除了没有\chapter{} 这一结构命令之外，其他都与 *report* 和 *book* 文档类型是一样的。

- *article* 是 L^AT_EX 制作文档时最为常用的一种文档类型，撰写科技论文往往需要用到 *article* 文档类型。
- *report* 主要是面向撰写各类技术报告的文档类型。
- *book* 是用于制作书籍等出版物的文档类型。

2.4 简单文档的制作

LaTeX 不但适合制作篇幅较大的文档，在制作篇幅较小的文档比如手稿、作业等时也十分方便。在 LaTeX 的各类文档中，最为常用的文档类型为 article (文章)，以下将介绍如何制作一个简单文档。

2.4.1 制作封面

添加标题、日期、作者信息一般是在\begin{document} 之前，格式如下：

```
1 % 输入空格表示空的  
2 \title{标题}  
3 \author{作者名字}  
4 \date{日期} % 如果不设置则会自动设置为编译时的时间，如果不展示日期则使用\data{}
```

如果要显示添加的相关信息，需要在\begin{document} 之后使用\maketitle 命令。

2.4.2 开始创建文档

在 LaTeX 中，以\begin{document} 命令为分界线，该命令之前的代码都统称为前导代码，这些代码能设置全局参数。位于\begin{document} 和 \end{document} 之间的代码被视为主体代码，我们所创作文档的具体内容也都是放在这两个命令之间。

设置章节

文档的章节是文档逻辑关系的重要体现，无论是中文论文还是英文论文都会有严谨的格式，章、节、段分明。在 LaTeX 中，不同的文档类设置章节的命令有些许差别，\chapter 命令只在 book、report 两个文档类中有定义，article 类型中设置章节可以通过\section{name} 及\subsection{name} 等简单的命令进行实现。

段落

段落是文章的基础，在 LaTeX 中，可以在文档中间键入空行文本作为段落，也可以使用 \paragraph{name} 和\subparagraph{name} 插入带标题的段落和亚段落。

生成目录

在 L^AT_EX 中, 我们可以通过一行简单的命令便可以生成文档的目录, 即 \tableofcontents。命令放在哪里, 就会在哪里自动创建一个目录。默认情况下, 该命令会根据用户定义的篇章节标题生成文档目录。目录中包含 \subsubsection 及其更高层次的结构标题, 而段落和子段信息则不会出现在文档目录中。注意如果有带 * 号的章节命令, 则该章节标题也不会出现在目录中。如果想让文档正文内容与目录不在同一页, 可在 \tableofcontents 命令后使用 \newpage 命令或者 \clearpage 命令。

类似对章节编号深度的设置, 我们通过调用计数器命令 \mintinline{tex}{\setcounter} 也可以指定目录层次深度。例如:

- \setcounter{tocdepth}{0} 目录层次仅包括 part
- \setcounter{tocdepth}{1} 目录层次深入到 section
- \setcounter{tocdepth}{2} 目录层次深入到 subsection
- \setcounter{tocdepth}{3} 目录层次深入到 subsubsection, 默认值

除此之外, 我们还可以在章节前面添加 \addtocontents{toc}{\setcounter{tocdepth}{}} 命令对每个章节设置不同深度的目录。另外还有一些其他的目录格式调整命令, 如果我们想让创建的目录在文档中独占一页, 只需要在目录生成命令前后添加 \newpage; 如果我们需要让目录页面不带有全文格式, 只需要在生成目录命令后面加上 \thispagestyle{empty} 命令; 如果我们想设置目录页之后设置页码为 1, 则需要在生成目录命令后面加上 \setcounter{page}{1} 命令。

如果我们想要创建图目录或表目录, 分别使用 \listoffigures、\listoftables 命令即可, 与创建章节目录的过程类似, 这两个命令会根据文档中图表的标题产生图表目录, 但不同之处在于, 图目录或表目录中所有标题均属于同一层次。

【例】 基于当前已掌握的知识的一个简单的 article:

```

1 \documentclass[a4paper,utf8,12pt]{ctexart}
2
3 %% 导言区
4 \usepackage{hyperref} % 超链接
5 \hypersetup{hidelinks} % 超链接隐藏默认的红框
6 \title{\LaTeX Cookbook} % 设置标题
7 \author{XXX} % 作者
8 \date{2021/12/19} % 设置日期, 不设置默认生成编译当天日期
9

```

```
10 \tableofcontents % 生成目录  
11  
12 %% 文档主体  
13 \begin{document}  
14 \section*{摘要} % 默认不生产目录；带*号的不设置编号  
15  
16 \section{XX背景介绍} % 第一节  
17  
18 \section{XX原理} % 第二节  
19 \subsection{XX原理论证} % 2.1 小节  
20 \subsection{基于XX的优化} % 2.2 小节  
21 \paragraph{一个段落...}  
22  
23 \paragraph{另一个段落...}  
24  
25 段落也可以使用空行生成，比如这是第三个段落...  
26  
27 这是第四个段落...  
28  
29 \end{document}
```

2.5 一些基本命令

当我们运用 LateX 进行文档编辑时，需要用到一些基本命令。

2.5.1 全局格式设置

在前面，我们介绍了一些全局设置的命令在申明文档类型时可以进行的一些全局参数设置，使用方法为在\documentclass[可省参数]{article} 的中括号中根据需要进行参数设置，然而有些全局设置需要用到一些其他方法进行调整。例如，纸张方向、页边距等需要调用宏包进行参数调整。同样地，我们以 article 类型文档进行举例说明。

纸张方向

article 类型文档的纸张方向默认为 *portrait* (纵向)，也可以设置成 *landscape* (横向)。在文档中可以调用 *lscape* 宏包中的\begin{landscape} \end{landscape} 环境将默认的纵向文档调整为横向。

How to change certain pages into landscape/portrait mode

<https://tex.stackexchange.com/q/337/227605>

页边距

article 类型文档的页边距可以通过调用 *geometry* 宏包进行调整

```
1 \usepackage{geometry} % 使用页面设置宏包
2 \geometry{left=3.0cm,right=3.0cm,top=2.5cm,bottom=2.5cm} % 设置页边距
```

章节标题的字体格式

article 类型文档的章节标题的字体格式可以通过调用 *sectsty* 宏包进行调整。

一些设置例举：

```
1 \usepackage{sectsty}
2 \sectionfont{\fontfamily{phv}\fontseries{b}\fontsize{11pt}{20pt}\selectfont} % 一级标题字体格式设置
3 \subsectionfont{\fontfamily{phv}\fontseries{b}\fontsize{11pt}{20pt}\selectfont} % 二级标题字体格式设置
4 \subsubsectionfont{\fontfamily{phv}\fontseries{b}\fontsize{11pt}{20pt}\selectfont} % 三级标题字体格式设置
```

图、表、公式格式全局设置

当我们需要批量设置图、表及公式的格式时，可以通过调用 *caption* 宏包进行全局设置。例：

```
1 \usepackage[labelfont=bf,labelsep=period,font={bf,sf,normalsize}]{caption}
```

Changing/Defining Fonts for an Entire Document

<https://tex.stackexchange.com/q/337/227605>

自定义命令全局设置

有时，我们需要也可以使用一些自定义命令来更改全局设置。例如在更改整个文档的字体格式时，我们也可以使用：

```
1 \renewcommand{\sfdefault}{\fontencoding{T1}\fontfamily{phv}\selectfont}
2 \renewcommand{\familydefault}{\sfdefault}
```

等命令。

在更改目录标题时，我们可以使用：

```
1 \renewcommand{\contentsname}{new name of Contents}
```


第三章 文本编辑

文本编辑是组成科技论文或科技报告的重要主要内容。而对于论文或报告中，简洁美观的文本对于阅读者的理解以及感受是非常重要的。因此，我们需要认真学习在 Latex 中如何编辑出清晰明了的文本内容。

文本编辑的内容一般包括文本章节的设定、文本段落的编辑、文字格式的编辑、列表的创建、页眉页脚和脚注的创建。其中文本段落的编辑又包括段落首行缩进、段行间距、段落对齐方式的调整等，而文字的编辑主要包括字体样式的调整、字体大小的调整、字体颜色的调整、字体本身的设计、字体中的下划线和删除线、以及一些特殊字符的书写等内容。

列表的创建是方便读者对文本进行阅读，而这一类文本中的内容一般处于并列关系。合适的列表可以方便读者知道文本的框架和内容关系。列表内容主要包括无序列表、排序列表、阐述性列表和自定义列表格式。对于文本编辑，页眉页脚和脚注是很重要的，页眉一般可以用于显示题目或者章节名称等提醒内容，而页脚可以显示文本页码等重要信息，脚注可以用于备注一些重要特别内容，如作者个人信息以及项目资助信息等。本章将详细介绍文本编辑的相关内容，学好文本编辑有助于我们完成一个阅读性更好的高质量文档。

3.1 创建标题部分、摘要及关键词

文档主体代码是指位于 document 环境的部分。在文档正文章节内容及目录之前，一般先创建标题部分（包括文档标题、作者和日期）、摘要、以及关键词信息，这也是文档主体代码中最开始的部分。下面分别介绍这部分的创建过程。

3.1.1 创建标题部分

- 使用\title{} 命令设置文档标题，对于较长的文档标题，可以使用\\对标题内容进行分行。

- 使用\author{} 命令设置作者，如果有多个作者，作者之间可以使用\and 进行分隔。
- 使用\date{} 命令设置日期信息，在实际使用时，有时需要省略日期信息，那么在 {} 中不写任何内容即可。如果想要使用默认值（当前日期），则应使用\date 命令。
- 使用\maketitle 命令完成标题部分的创建。

仅仅执行上述三行语句无法在文档编译时生成标题部分，还必须在之后加上\maketitle 命令，表示对标题部分内容进行排版才能真正实现标题部分的创建。

3.1.2 创建摘要及关键词

在 LaTeX 中，使用 *abstract* 环境撰写文档摘要部分，并在其后使用\textbf{} 命令设置文档关键词。

3.2 创建章节

在制作文档时，不管是学术论文还是书籍，都需要创建章节来优化文章的层次结构。LaTeX 提供了不同层次章节的创建命令，从高到低包括：使用\part{} 命令创建不同篇、使用\chapter{} 命令创建不同章、使用\section{} 命令创建一级节、使用\subsection{} 命令创建二级节、以及使用\subsubsection{} 命令创建三级节。各篇章的标题填写在 {} 中。

对于 article 类型文档，可以调用上述除了\chapter{} 命令之外的其他章节命令。而在 book、report 类型文档中，则可以调用上述所有章节命令。

在撰写学术论文时，不同出版商所要求的章节标题格式可能是不一样的。因此，我们需要根据不同需要调整章节标题的格式。

- 自动编号与取消自动编号：在 LaTeX 中使用上述命令创建章节时，默认会对各章节进行自动编号。如果用户想要对某章节取消自动编号，只需要在章节命令后加上星号即可，如\section*{} 命令。
- 设置章节自动编号深度：用户也可以通过在导言区使用\setcounter{secnumdepth}{ } 计数器命令设置章节自动编号深度，从而达到批量取消自动编号的效果。在 {} 中填写编号深度值，编号深度值从 0 开始设置，表示章节自动编号深入层次：
 - \setcounter{secnumdepth}{0} 表示自动编号章节层次仅包括\part 和\chapter；

- `\setcounter{secnumdepth}{1}` 表示自动编号层次深入到`\section` 级；
 - `\setcounter{secnumdepth}{2}` 表示自动编号层次深入到`\subsection` 级；
 - `\setcounter{secnumdepth}{3}` 表示自动编号层次深入到`\subsubsection` 级，
默认值。
- 改变字体样式：对于改变字体样式，我们需要使用 `titlesec` 宏包，使用该宏包中的命令`\titleformat*{}{}` 来改变字体样式。

当然，有时我们也会遇到一些出版商要求我们将标题居中显示，在 LaTeX 的 `article` 文档类型中，我们可以调用 `sectsty` 宏包，并用到`\sectionfont{\centering}` 使标题居中。

3.3 生成目录

章节目录一般在摘要之后创建，使用`\tableofcontents` 命令即可。命令放在哪里，就会在哪里自动创建一个章节目录。默认情况下，该命令会根据用户定义的篇章节标题生成文章目录，目录中将包含`\subsubsection` 及其更高层次的结构标题。但对于带星号的章节命令，其章节标题不会出现在目录中。

根据需要，用户可以对目录格式进行各项调整。

3.3.1 调整章节层次深度

在前一节中我们介绍了使用`\setcounter{secnumdepth}{}` 计数器命令调整章节自动编号深度，类似地，我们可以通过在导言区使用`\setcounter{tocdepth}{}` 命令指定目录中的章节层次深度。

- `\setcounter{tocdepth}{0}`, 目录层次仅包括 `part` 和 `chapter`;
- `\setcounter{tocdepth}{1}`, 设置目录层次深入到 `section` 级;
- `\setcounter{tocdepth}{2}`, 设置目录层次深入到 `subsection` 级;
- `\setcounter{tocdepth}{3}`, 设置目录层次深入到 `subsubsection` 级，
默认值;

上面的语句可以为所有章节指定了相同的目录层次深度。此外，我们也可以为每个章节设置不同的目录层次，具体是通过在每个章节创建命令前，使用`\addtocontents{toc}\setcounter{}` 命令为该章节指定目录层次深度。

3.3.2 给目录设置别名

对于章节标题特别长的情况，直接在目录中显示完整标题显然可视化效果不佳，因此需要为长章节标题设置一个比较短的“目录别名”。通过这种设置，在正文中可以显示完整标题，而在目录中将显示“短标题”。为此，只需要在章节创建命令中添加目录别名选项即可。例：\section[FS]{First Section}

3.3.3 给目录设置链接

如果想要为目录中的章节引用添加链接，使得点击链接后就能跳转到相应章节所在页面，那么只需要在导言区调用 *hyperref* 宏包。如果设置 `colorlinks=true` 选项，此时文档中章节引用及其它交叉引用均会被自动添加链接（添加了链接的引用将显示为红色）。

3.4 编辑段落

3.4.1 段落首行缩进

许多出版社要求文章段落必须首行缩进，若想调整段落首行缩进的距离，可以使用`\setlength{\parindent}{长度}` 命令，在`{长度}` 处填写需要设置的距离即可。例：
`\setlength{\parindent}{2em}`

当然，如果不让段落自动首行缩进，在段落前使用命令`\noindent` 即可。

需要注意的是，在段落设置在章节后面时，每一节后的第一段默认是不缩进的，为了使第一段向其他段一样缩进，可以在段落前使用`\hspace*{\parindent}` 命令，也可以在源文件的前导代码中直接调用宏包`\usepackage{indentfirst}`。

3.4.2 段落间距调整

在使用 LaTeX 排版时，有时为了使段落与段落之间的区别更加明显，我们可以在段落之间设置一定的间距，最简单的方式是使用`\smallskip`、`\medskip` 和`\bigskip` 等命令。

设置段落间距的几种方法

<https://tex.stackexchange.com/questions/41476/lengths-and-when-to-use-them>

<https://latex.org/forum/viewtopic.php?f=44&t=6934>

3.4.3 段落添加文本框

有时因为文档没有图全都是文字，版面显得极其单调。如果想让版面有所变化，可以通过给文字加边框来实现对段落文本新增边框。在 LaTeX 中，我们可以使用\fbox{}命令对文本新增边框。

How to put a box around multiple lines?

<https://latex.org/forum/viewtopic.php?f=44&t=4117>

3.4.4 段落对齐方式调整

LaTeX 默认的对齐方式是两端对齐，有时在进行文档排版的过程中，我们为了突出某一段落的内容，会选择将其居中显示，在 LaTeX 中，我们可以使用 *center* 环境对文本进行居中对齐。另外还有一些出版商要求文档是左对齐或者右对齐，这时我们同样可以使用 *flushleft* 环境和 *flushright* 环境将文档设置为左对齐或右对齐。

3.5 文字编辑

文字编辑是制作文档非常重要的一部分，主要关注如何调整字体样式、字体设置、增加下划线、突出文字、调整字体大小、调整对齐格式等内容。

3.5.1 调整字体样式

调整文字的样式有很多对应的命令，这些命令包括\textit、\textbf、\textsc、\texttt，在使用的过程中，需要用到花括号 {}。具体而言，\textit 对应着斜体字，\textbf 对应着粗体字，\textsc 对应着小型大写字母，\texttt 对应着打印机字体（即等宽字体）。

除了这几种字体样式，有时候，如果想对文本中的英文字母进行全部大写，可用\uppercase 和 \MakeUppercase 两个命令中的任意一个。

一般而言，当我们需要对段落、句子、关键词等做特殊标记时，往往会用到上述几种字体样式，其中，打字机字体主要用于代码的排版，有时候，如果需要添加一个网站，通常也会选用打字机字体对文本进行突出，例如\texttt{https://www.overleaf.com}。

3.5.2 调整字体大小

字体大小的设置一方面可以在申明文档类型的命令\documentclass[]{} 中指定具体的字体大小（如 11pt、12pt）来实现，另一方面也可以通过一些简单的命令来调整。

```

1 \documentclass[12pt]{article}
2 \begin{document}
3
4 Produce {\tiny tiny word}
5
6 Produce {\scriptsize script size word}
7
8 Produce {\footnotesize footnote size word}
9
10 Produce {\normalsize normal size word}
11
12 Produce {\large large word}
13
14 Produce {\Large Large word}
15
16 Produce {\LARGE LARGE word}
17
18 Produce {\huge huge word}
19
20 Produce {\Huge Huge word}
21
22 \end{document}

```

在这里，这些命令对应的字体依次是从小到大。当然，这些命令也有另外一种使用方法，以 `\large`、`\Large`、`\LARGE` 为例，我们可以使用`\begin{...}` `\end{...}` 语句来实现对字体的加大：

```

1 \documentclass[12pt]{article}
2 \begin{document}
3
4 Produce \begin{large}large word\end{large}
5
6 Produce \begin{Large}large word\end{Large}
7
8 Produce \begin{LARGE}large word\end{LARGE}
9
10 \end{document}

```

3.5.3 调整字体颜色

一般而言，文本默认的颜色是黑色，但有时候，我们可以根据需要改变字体的颜色，这通过 LaTeX 一些拓展的宏包就可以实现，例如 *xcolor*。

使用颜色宏包时，我们也可以根据需要自定义颜色，相应的命令为\definecolor{A}{B}{C}，其中位置 A 是颜色标签，位置 B 是制定颜色系统为 RGB（英文缩写 RGB 是红色、绿色和蓝色三种颜色的英文单词首字母），位置 C 是具体的 RGB 数值。

```

1 \documentclass[12pt]{article}

2

3 \usepackage{color}
4 \definecolor{kugreen}{RGB}{50, 93, 61}
5 \definecolor{kugreenlys}{RGB}{132, 158, 139}
6 \definecolor{kugreenlysls}{RGB}{173, 190, 177}
7 \definecolor{kugreenlyslslysls}{RGB}{214, 223, 216}

8

9 \begin{document}

10

11 This is a simple example for using \LaTeX.

12 {\color{kugreen}This is a simple example for using \LaTeX.}

13 {\color{kugreenlys}This is a simple example for using \LaTeX.}

14 {\color{kugreenlysls}This is a simple example for using \LaTeX.}

15 {\color{kugreenlyslslysls}This is a simple example for using \LaTeX.}

16

17 \end{document}
```

3.5.4 字体设置

不管是英文还是中文，我们都会遇到各种各样的字体，因此，使用 LaTeX 编译出想要的字体对于整个文档也非常的重要。对于英文文档的编译，一般会用宏包 *fontspec* 设置具体的字体，调用格式为\usepackage{fontspec}，需要说明的是，这个宏包只能设置英文的字体。例如：

```

1 \setmainfont{Times New Roman}
```

```

2 \setsansfont{DejaVu Sans}
3 \setmonofont{Latin Modern Mono}
4 \setsansfont{[foo.ttf]}

```

如果文档输入的是中文，首先需要申明文档类型为 ctex 中的 ctexart、ctexrep 之类的。

在 LaTeX 中，编译文档一般默认的英文字体为 Computer Modern，如果要将其调整为其他特定类型的字体，可以在前导代码中使用各种字体对应的工具包。

更多字体设置参考

https://www.overleaf.com/learn/latex/Font_typefaces

3.5.5 下划线与删除线

有时候，为了突出特定的文本，我们也会使用到各种下划线。最常用的下划线命令是\underline，然而，这个命令存在一个缺陷，即当文本过长，超过页面宽度时，下划线不会自动跳到下一行，因此，我们需要用到一个叫 *ulem* 的宏包，使用这个宏包中的命令\uline 可以增加单下划线，使用\uuline 可以增加双下划线，而使用\ulwave 则可以增加波浪线。

```

1 \documentclass[12pt]{article}
2 \usepackage{ulem}
3 \begin{document}
4
5 Generate \underline{underlined} text. \\ % 生成带下划线的文本（使用\underline命令）
6
7 Generate \uline{underlined} text. \\ % 生成单下划线的文本（使用\uline命令）
8
9 Generate \uuline{double underlined} text. \\ % 生成双下划线的文本
10
11 Generate \ulwave{wavy underlined} text. \\ % 生成波浪线的文本
12
13 \end{document}

```

删除线是文字中间划出的线段，常见于文档的审阅。在 LaTeX 中，我们可以使用宏包 *soul* 中的\st 命令生成删除线。

下划线在 LaTeX 环境中属于特殊字符，如果需要在文本中使用下划线，则需要加上反斜线进行转义，例：

```

1 \documentclass[12pt]{article}
2 \begin{document}
3
4 This\_is\_text\_with\_underscores.
5
6 \end{document}
```

3.5.6 特殊字符

在 LaTeX 中，有很多特殊字符的编译需要遵循一定的规则，例如：

- 反斜杠 (backslash) 符号是 LaTeX 中定义和使用各类命令的基本符号，如果要在文档中编译出反斜杠，可使用\textbackslash;
- 百分号通常用于注释代码，如果要在文档中编译出百分号，可使用\%;
- 美元符号通常用于书写公式，如果要在文档中编译出美元符号，可使用\\$。

带圆圈数字可用于各类编号，我们可以根据需要插入这种特殊符号。在 LaTeX 中，比较常用的一种生成带圆圈数字的方法是使用宏包 *pifont*，在前导代码中申明使用宏包，即\usepackage{pifont}，根据该宏包所提供的命令\ding{} 可以生成从 1 到 10 的带圆圈数字，且圆圈样式也各异。

【例】 使用 pifont 宏包中的命令生成从 1 到 10 的带圆圈数字：

```

1 \documentclass[12pt]{article}
2 \usepackage{pifont}
3 \begin{document}
4
5 How to write a number in a circle? \\
6 Type 1: \ding{172}-\ding{173}-\ding{174}-\ding{175}-\ding{176}-\ding{177}-\ding{178}-\ding{179}-\ding{180}-\ding{181} \\ % 样式1是从172开始
7 Type 2: \ding{182}-\ding{183}-\ding{184}-\ding{185}-\ding{186}-\ding{187}-\ding{188}-\ding{189}-\ding{190}-\ding{191} \\ % 样式2是从182开始
8 Type 3: \ding{192}-\ding{193}-\ding{194}-\ding{195}-\ding{196}-\ding{197}-\ding{198}-\ding{199}-\ding{200}-\ding{201} \\ % 样式3是从192开始
```

```

9 Type 4: \ding{202}-\ding{203}-\ding{204}-\ding{205}-\ding{206}-\ding{207}-\ding
10   {208}-\ding{209}-\ding{210}-\ding{211} \\ % 样式4是从202开始
11 \end{document}

```

3.6 创建列表

在内容表达上，列表是一种非常有效的方式，它将某一论述内容分成若干个条目进行罗列，能达到简明扼要、醒目直观的表达效果。在论文写作中，列表不失为一种让内容清晰明了的论述方式。

通常来说，列表有单层列表和多层列表，多层次列表无外乎是最外层列表中嵌套了一层甚至更多层列表。具体来说，列表主要有三种类型，即无序列表、排序列表和阐述性列表，其中，无序列表和排序列表是相对常用的列表类型，LaTeX 针对这三种列表提供了一些基本环境：

- itemize，无序列表环境；
- enumerate，有序列表环境；
- description，阐述列表环境；

在这三种列表中，我们创建的每一项列表内容都需要紧随\item 命令之后。当然，有时候，我们也可以根据需要选择合适的列表类型、调整列表符号甚至行间距等。

3.6.1 无序列表

LaTeX 中的无序列表环境一般用特定符号（如圆点、星号）作为列表中每个条目的起始标志，以示有别于常规文本。可以忽略主次或者先后顺序关系的条目排列都可以使用无序列表环境来编写，无序列表时很多文档最常用的列表类型，也被称为常规列表。

在无序列表环境中，每个条目都是以条目命令\item 开头的，一般默认的起始符号是 *textbullet*，即大圆点符号，当然，也可以根据需要调整起始符号。

【例】 在无需列表环境中使用星号作为条目的起始符号：

```

1 \documentclass[12pt]{article}
2 \begin{document}
3 \begin{itemize}

```

```
4  
5     \item Python \% 条目1, 起始符号为大圆点  
6     \item LaTeX \% 条目2, 起始符号为大圆点  
7     \item[*] GitHub \% 条目3, 起始符号为星号  
8  
9     \end{itemize}  
10    \end{document}
```

如果想要将所有条目的符号都进行调整，并统一为某一个特定符号，则可以使用\renewcommand 命令进行自定义。

【例】自定义条目的起始符号为黑色方块（black square）：

```
1 \documentclass[12pt]{article}  
2 \usepackage{amssymb}  
3 \begin{document}  
4  
5     \begin{itemize}  
6         \renewcommand{\labelitemi}{\scriptsize$\blacksquare$}  
7         \item Python \% 条目1  
8         \item LaTeX \% 条目2  
9         \item GitHub \% 条目3  
10        \end{itemize}  
11  
12    \end{document}
```

其中，命令\labelitemi 是由三部分组成，即 label（标签）、item（条目）、i（一级），有时候，如果需要创建多级列表，则可以类似这里使用命令\labelitemii（对应于二级列表）甚至\labelitemiii（对应于三级列表）。

3.6.2 排序列表

排序列表也被称为编号列表。在排序列表中，每个条目之前都有一个标号，它是由标志和序号两部分组成：序号自上而下，从 1 开始升序排列；标志可以是括号或小圆点等符号。相互之间有密切的关联，通常是按过程顺序或是重要程度排列的条目都可以采用排序列表环境编写。排序列表环境 *enumerate* 以序号作为列表的起始标志，每个条目命令 item 将在每个条目之前自动加上一个标号条目命令 item 生成的默认标号样式为阿拉伯数字加小圆点。

排序列表同样可以进行相互嵌套，最多可以达到四层，为了便于区分，不仅每层列表的条目段落都有不同程度的左缩进，而且每层列表中条目的标号也各不相同，其中序号的计数形式与条目所在的层次有关，标志所用的符号除第 2 层是圆括号外，其他各层都是小圆点。

3.6.3 阐述性列表

相比无序列表和排序列表，阐述性列表的使用频率较低，它常用于对一组专业术语进行解释说明。阐述性列表环境为 *description*。在 *description* 环境命令中，每个词条都是需要分别进行阐述的词语，每个阐述可以是一个或多个文本段落。这种形式很像词典，因此诸如名词解释说明之类的列表就可以采用解说列表环境来编写。在阐述性列表环境中，被解释的词条的格式是用 *descriptionlabel* 定义的。

【例】 创建一个简单的阐述性列表：

```

1 \documentclass[12pt]{article}
2 \begin{document}
3
4 \begin{description}
5
6   \item [CNN] Convolutional Neural Networks
7   \item [RNN] Recurrent Neural Network
8   \item [CRNN] Convolutional Recurrent Neural Network
9
10  \end{description}
11
12  \end{document}
```

3.6.4 自定义列表格式

使用系统默认的 LaTeX 列表环境排版的列表与上下文之间以及列表条目之间都附加有一段垂直空白，明显有别于列表环境之外的文本格式，通常列表中的条目内容都很简短，这样会造成很多空白，使得列表看起来很稀疏，与前后文本之间的协调性较差。因此，我们需要自定义列表格式。使用 *enumitem* 宏包可以调整 *enumerate* 或 *itemize* 的上下左右缩进间距。

- 垂直间距

- topsep 列表环境与上文之间的距离
 - parsep 条目里面段落之间的距离
 - itemsep 条目之间的距离
 - partopsep 条目与下面段落的距离
- 水平间距
 - leftmargin 列表环境左边的空白长度
 - rightmargin 列表环境右边的空白长度
 - labelsep 标号与列表环境左侧的距离
 - itemindent 条目的缩进距离
 - labelwidth 标号的宽度
 - listparindent 条目下面段落的缩进距离

【例】使用 enumitem 宏包可以调整无序列表间距：

```
1 \documentclass[12pt]{article}
2 \usepackage{enumitem}
3 \begin{document}
4
5
6 Default spacing:
7
8 \begin{itemize}
9
10 \item Python \% 条目1
11 \item LaTeX \% 条目2
12 \item GitHub \% 条目3
13
14 \end{itemize}
15
16 Custom Spacing:
17
18 \begin{itemize}[itemsep= 15 pt,topsep = 20 pt]
19
20 \item Python \% 条目1
21 \item LaTeX \% 条目2
22 \item GitHub \% 条目3
```

```

23
24     \end{itemize}
25
26     \end{document}

```

3.7 创建页眉、页脚及脚注

在大多数文档中，我们往往需要页面、页脚及脚注来展示文档的附加信息，例如时间、图形、页码、日期、公司微标、页眉示意图文档标题、文件名或作者姓名等信息。在 LaTeX 中，我们常用 *Fancyhdr* 工具包进行页眉、页脚的设置。

【例】 使用 *Fancyhdr* 工具包进行页眉、页脚的设置：

```

1   \documentclass{article}
2   \usepackage{fancyhdr}
3   \pagestyle{fancy}
4   \lhead{}
5   \chead{}

6
7   \rhead{\bfseries latex-cookbook} %页眉内容
8   \lfoot{From: Xinyu Chen} %页脚内容
9   \cfoot{To: Jieling Jin} %页脚内容
10  \rfoot{\thepage} %在页脚处给出页码
11  \renewcommand{\headrulewidth}{0.4pt}
12  \renewcommand{\footrulewidth}{0.4pt}

13
14  \begin{document}
15  This is latex-cookbook!
16  \end{document}

```

如果想某一页不需要页眉页脚，可以在该页正文内容开始时使用\thispagestyle{empty}命令，去除该页页眉页脚。

在 LaTeX 中，我们常用\footnote{} 命令来添加脚注。

在论文撰写过程中，我们有时会需要在表格中添加脚注，但是在 table 环境中，\footnote 命令不起作用，这时我们可以使用 *minipage* 环境来化解。

【例】 使用 *minipage* 环境来给表格添加脚注：

```
1 \documentclass{article}
2 \begin{document}
3 \begin{center}
4 \begin{minipage}{.5\textwidth}
5 \begin{tabular}{l|l}
6 \textsc{Chapter} & \textsc{Author} \\ \hline
7 \textit{Introduction} & Xinyu Chen\footnote{Xinyu Chen is a PhD from the
8 University of Montreal.} \\
9 \textit{Methods} & Jieling Jin \footnote{Jieling Jin is a PhD from the
10 Central South University.} \\
11 \textit{Case Study} & Xinyu Chen \\
12 \textit{Conclusion} & Jieling Jin
13 \end{tabular}
14 \end{minipage}
15 \end{center}
16 \end{document}
```


第四章 公式编辑

在一些文档特别是科技文档写作的过程中，难免涉及复杂数学公式的编辑工作。Microsoft office 等办公软件繁琐低效的数学公式编辑问题长期困扰着广大科研工作者。为什么这些办公软件数学公式编辑效率低下，主要原因是数学公式本身就是复杂多变的，而这些软件采用交互式的操作会让用户使用起来效率变低。而 LaTeX 具有简洁又强大的数学公式编辑功能，可以通过调用特定的工具包、使用一些简单的代码就能生成优雅美观的数学公式，这也是 LaTeX 深受广大科研工作者喜爱的重要原因之一。

本章将介绍在 Latex 中进行编写公式编辑的方法和技巧。为了方便读者查用和学习，我们将公式编辑内容主要分为以下部分：公式编辑的基本介绍、常用的数学符号、希腊字母、微积分中的常用公式、线性代数中的常用公式、概率论和数理统计、优化理论中的公式。这样分类的目的就是便于读者根据需要，可以很快查找出合适的公式编辑方法。公式编辑的基本介绍主要是关于数学公式换的设定和基本格式的调整；而常用的数学符号包括运算符号、标记符号、各类括号、空心符号和特殊函数；希腊字母主要用于一些数学的变量表示。

而微积分部分主要包括极限、导数和积分；线性代数部分包括矩阵、符合和范数；概率论和数理统计部分包括概率论基础和概率分布；最后是优化理论部分。以上这些是科研工作者在数学公式编辑时高频率遇到的公式编辑内容，因此我们将其分类进行介绍，这样可以方便读者提高编辑的效率和质量。

4.1 基本介绍

由于 LaTeX 编辑的数学公式颜值非常高，很多理工科研究领域的顶级期刊甚至明确要求投稿论文必须按照给定的 LaTeX 模板进行论文排版，这样做一方面能保证论文整体排版的美观程度，另一方面也能让生成出来的数学公式更加规范。一般而言，使用 LaTeX 编辑公式的一系列规则与数学公式的编写原则是一致的，例如，在 LaTeX 中，我们可以用 $\frac{\partial f}{\partial x}$ 生成偏导数 $\frac{\partial f}{\partial x}$ 。

4.1.1 数学公式环境

美元符号

在 LaTeX 中生成数学公式也有一些基本规则，插入公式的方式有很多种，最基本的一种方式是使用美元符号，这种方式不仅在 LaTeX 适用，在 Markdown 中也是适用的，具体插入数学公式的方法是：

- 如果我们想插入行内公式，可以直接在两个美元符号中间编辑需要的公式。
- 如果想用美元符号插入行间公式，我们需要输入四个美元符号，与此同时，在四个美元符号中间编辑需要的公式。需要注意的是，这里生成的数学公式会自动居中对齐。

需要注意的是，LaTeX 源文件中的美元符号一般都默认为申明数学公式环境，如果想要在文档中编译出美元符号，需要在美元符号前加上一个反斜线，这种做法同样适用于百分号，百分号一般被默认为注释功能。

equation 环境

尽管美元符号可以在行间插入公式，但却没办法对公式进行编号。自动生成带有公式编号的行间公式需要用到 *equation* 环境，使用 *equation* 环境，LaTeX 编译时会自动将公式进行居中对齐。

在 *equation* 环境中，如果不需要公式编号，只需要在数学公式环境中加上一个星号，例如，使用`\begin{equation*}` `\end{equation*}` 就可以移除公式编号。

更进一步，在 *equation* 环境中，如果想对公式进行索引，可以使用`\label` 和`\eqref` 两个命令。

【例】 在数学公式环境中对数学公式进行索引：

```

1 \documentclass[12pt]{article}
2 \begin{document}
3
4 Equation~\eqref{eq1} shows a simple formula.
5
6 \begin{equation}\label{eq1}
7 x+y=2
8 \end{equation}
9
10 \end{document}

```

align 环境

在 LaTeX 中，除了 equation 数学公式环境，还有其他几种数学公式环境可以使用。我们要介绍的第一种是 *align* 环境，它主要用于数组型的数学表达式，align 环境可以将公式进行自动对齐，它也能对每一条数学表达式分别进行公式编号。

【例】使用 align 环境编译一个方程组：

```
1 \documentclass[12pt]{article}
2 \usepackage{amsmath}
3 \begin{document}
4
5 %% 使用align环境
6 \begin{aligned}
7 x+y=2 \\
8 2x+y=3
9 \end{aligned}
10
11 \end{document}
```

在 align 环境中，如果不需要公式编号，同样只需要在数学公式环境中加上一个星号即可，利用这一特性，我们可以使用 align 环境，定义多列公式。

需要注意的是，如果想对多行公式对齐，并且共用同一个公式编号，可以在 equation 环境内使用 *aligned* 环境，这里的 aligned 与 align 功能类似。

【例】在 equation 内使用 aligned 编译多列公式：

```
1 \documentclass[12pt]{article}
2 \usepackage{amsmath}
3 \begin{document}
4
5 \begin{equation}
6 \begin{aligned}
7 2x+1&=7 & 3y-2&=-5 & -5z+8&=-2 \\
8 2x&=6 & 3y&=-3 & -5z&=-10 \\
9 x&=3 & y&=-1 & z&=2
10 \end{aligned}
11 \end{equation}
12
13 \end{document}
```

当然，我们也能只对 align 环境中的某一些公式进行编号，而其他公式不作编号。

Eqnarray: numbering last line only.

<https://latex.org/forum/viewtopic.php?f=46&t=4543>

【例】使用 align 编译一个方程组，并且只对第 2 个方程进行编号：

```

1 \documentclass[12pt]{article}
2 \usepackage{amsmath}
3 \begin{document}
4
5 \begin{equation}
6 \begin{aligned}
7 2x+1&=7 & 3y-2&=-5 & -5z+8&=-2 \\
8 2x&=6 & 3y&=-3 & -5z&=-10 \\
9 x&=3 & y&=-1 & z&=2
10 \end{aligned}
11 \end{equation}
12
13 \end{document}
```

gather 环境

我们要介绍的第二种数学公式环境是 *gather*，它既可以将公式进行居中对齐，也能对每一条数学表达式分别进行公式编号。同样的，如果想要移除公式编号，只需要在公式环境中加上星号即可。

【例】使用 gather 编译一个方程组：

```

1 \documentclass[12pt]{article}
2 \usepackage{amsmath}
3 \begin{document}
4
5 \begin{gather}
6 x+y=2 \\
7 2x+y=3
8 \end{gather}
9
10 \end{document}
```

4.1.2 基本格式调整

字符类型

在文本编辑中，我们已经介绍了几种常见的字符类型，实际上，对于数学公式而言，书写时也可以设置不同的字符类型。以 X, Y, Z, x, y, z 为例，具体而言：

- 命令`\boldsymbol{X,Y,Z,x,y,z}`，编译后的效果为 $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{x}, \mathbf{y}, \mathbf{z}$ ，使用之前需申明`\usepackage{amsmath}`；
- 命令`\mathrm{X,Y,Z,x,y,z}`，编译后的效果为 X, Y, Z, x, y, z ；
- 命令`\mathit{X,Y,Z,x,y,z}`，编译后的效果为 X, Y, Z, x, y, z ；
- 命令`\mathbf{X,Y,Z,x,y,z}`，编译后的效果为 $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{x}, \mathbf{y}, \mathbf{z}$ ；
- 命令`\mathsf{X,Y,Z,x,y,z}`，编译后的效果为 $\mathsf{X}, \mathsf{Y}, \mathsf{Z}, \mathsf{x}, \mathsf{y}, \mathsf{z}$ ；
- 命令`\mathtt{X,Y,Z,x,y,z}`，编译后的效果为 $\mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{x}, \mathtt{y}, \mathtt{z}$ ；
- 命令`\boldmath{X,Y,Z,x,y,z}`，编译后的效果为 X, Y, Z, x, y, z ；依赖于特定工具包，使用之前需申明`\usepackage{amssymb}`；
- 命令`\mathcal{X,Y,Z}`，编译后的效果为 $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ ；
- 命令`\mathbb{X,Y,Z}`，依赖于特定工具包，使用之前需申明`\usepackage{amssymb, amsfonts}`，编译后的效果为 $\mathbb{X}, \mathbb{Y}, \mathbb{Z}$ ，概率论与数理统计中常见的数学期望符号 \mathbb{E} 也是用该命令编译的，即`\mathbb{E}`；
- 命令`\mathfrak{X,Y,Z,x,y,z}`，依赖于特定工具包，使用之前需申明`\usepackage{amssymb, amsfonts, eufrak}`，编译后的效果为 $\mathfrak{X}, \mathfrak{Y}, \mathfrak{Z}, \mathfrak{x}, \mathfrak{y}, \mathfrak{z}$ 。

除此之外，如果想在公式中插入正常的文本，可以使用`\text{}`命令，例如 $T_{\text{\texttt{\text{start}}}}$ 编译后的效果为 T_{start} 。

调整公式大小

如果想对单个公式调整公式字符大小，在美元符号插入的公式中，可以使用 *displaystyle*、*textstyle*、*scriptstyle* 和 *scriptscriptstyle* 等申明命令对公式大小进行调整，公式显示效果依次从小到大，这些命令一般放在公式前即可。

【例】 使用上述四种命令分别书写函数 $f(x) = \sum_{i=1}^n \frac{1}{x_i}$ ：

```

1  \documentclass[12pt]{article}
2  \begin{document}
3
4  \$\textstyle{f(x)=\sum_{i=1}^n\frac{1}{x_i}}$,
5  \$\textstyle{f(x)=\sum_{i=1}^n\frac{1}{x_i}}$,
6  \$\scriptstyle{f(x)=\sum_{i=1}^n\frac{1}{x_i}}$,
7  \$\scriptstyle{f(x)=\sum_{i=1}^n\frac{1}{x_i}}$.
8
9  \end{document}

```

在各类公式环境（如 *equation*、*align*、*gather*）中，可以外使用一系列字符大小命令进行调整，例如用 *\begingroup* *\endgroup* 限定字符区域。

【例】 在 *\begingroup* *\endgroup* 中使用字符大小命令 *\small* 和 *\Large* 对公式大小进行调整：

```

1  \documentclass[12pt]{article}
2  \usepackage{amsmath}
3  \begin{document}
4
5  %% Small size
6  \begingroup
7  \small
8  \begin{align}
9  x+y=2 \\
10 2x+y=3
11 \end{align}
12 \endgroup
13
14 %% Large size
15 \begingroup
16 \Large
17 \begin{align}

```

```

18 x+y=2 \\
19 2x+y=3
20 \end{align}
21 \endgroup
22
23 \end{document}

```

其他格式调整

在 equation、align 等公式环境中，我们也可以通过数组 array 环境对数学公式进行对齐：

【例】使用 array 环境编译一个方程组，并添加半边花括号，如：

```

1 \documentclass[12pt]{article}
2 \usepackage{amsmath}
3 \begin{document}
4
5 \begin{equation}
6 \left.\begin{array}{l}
7 x+y=2 \\
8 2x+y=3
9 \end{array}\right.
10 \end{equation}
11
12 \begin{align}
13 \left.\begin{array}{l}
14 x+y=2 \\
15 2x+y=3
16 \end{array}\right.
17 \end{align}
18
19 \end{document}

```

编译后：

$$\begin{cases} x + y = 2 \\ 2x + y = 3 \end{cases} \quad (4.1)$$

其中，对齐的方式有 l（左侧对齐）、c（居中对齐）和 r（右侧对齐）。

【例】使用 array 环境编译公式，并让公式居中对齐：

```

1 \documentclass[12pt]{article}
2 \usepackage{amsmath, mathtools}
3
4 \begin{document}
5
6 \begin{equation}
7 \begin{array}{c@{\qquad}c}
8 A = B + C \\
9 \qquad\qquad\Rightarrow \\
10 & D = E - F, \\ 
11 G = H \\
12 \qquad\qquad\Rightarrow \\
13 & K = P + Q + M. \\
14 \end{array}
15 \end{equation}
16
17 \end{document}

```

编译后：

$$A = B + C \quad \Rightarrow \quad D = E - F, \quad (4.2)$$

$$G = H \quad \Rightarrow \quad K = P + Q + M.$$

当公式过长时，还有一些工具包提供的环境可以让公式进行自动跨行，以工具包 *breqn* 为例，在使用时，用 `\begin{dmath}` `\end{dmath}` 即可。

4.2 常用数学符号

常用数学符号包括运算符号、标记符号、各类括号、空心符号及一些特殊函数。

4.2.1 运算符号

在初等数学中，最基本的运算规则是加减乘除。在 LaTeX 中，加法符号和减法符号就是 `+` 和 `-`；而乘法符号有两种，第一种是 `\times`，对应于符号 `*`，第二种是 `\cdot`，对应于符号 `,`，除法符号的命令为 `\div`。

- `+` 加

- - 减
- \times 叉乘
- \cdot 点乘
- \div 除
- / 除

在加减的基础上命令\pm (由 plus 和 minus 首字母构成) 和\mp (由 minus 和 plus 首字母构成) 分别对应着符号 \pm 和 \mp 。与加减乘除同样常用的运算符号还有大于号、小于号等。

- x < y 即 $x < y$
- x > y 即 $x > y$
- x \leq y 即 $x \leq y$
- x \geq y 即 $x \geq y$
- x \ll y 即 $x \ll y$ 远小于
- x \gg y 即 $x \gg y$ 远大于

对于集合而言：

- \cap 即 \cap
- \cup 即 \cup
- \supset 即 \supset
- \subset 即 \subset
- \supseteq 即 \supseteq
- \subseteq 即 \subseteq
- \in 即 \in
- \notin 即 \notin

4.2.2 标记符号

在数学中，还有一些基本数学表达式也非常 important，例如分式、上标、下标。LaTeX 中用于书写分数和分式的基本命令为\frac{分子}{分母}，根据场景需要，也可以选用\frac{分子}{分母} 和\tfrac{分子}{分母}。

- \frac{x+3}{y-5} 即 $\frac{x+3}{y-5}$
- \dfrac{x+3}{y-5} 即 $\frac{x+3}{y-5}$
- \tfrac{x+3}{y-5} 即 $\frac{x+3}{y-5}$
- x^{x+5} 即 x^{x+5}
- x^{x^3+5} 即 x^{x^3+5}
- x_{x+5} 即 x_{x+5}
- x_{x_3+5} 即 x_{x_3+5}
- x_1,x_2,\ldots,x_n 即 x_1, x_2, \dots, x_n

与上标和下标对应的还有各种“帽子”符号，即字母上面加符号：

- \hat{x} 即 \hat{x}
- \bar{x} 即 \bar{x}
- \tilde{x} 即 \tilde{x}
- \vec{x} 即 \vec{x}
- \dot{x} 即 \dot{x}
- \bar{xy} 即 \bar{xy}
- \overline{xy} 即 \overline{xy}
- \tilde{xy} 即 \tilde{xy}
- \widetilde{xy} 即 \widetilde{xy}

Tilde over the letter

<https://latex.org/forum/viewtopic.php?f=48&t=11388>

根号同样是数学表达式中的常见符号，在 LaTeX 中，根号的命令为`\sqrt{}`，使用默认设置，生成的表达式为二次方根，如果想要设置为三次方根，则需要调整默认设置，即`\sqrt[3]{}`，以此类推，可以设置四次方根等。

4.2.3 各类括号

在数学表达式中，括号的用处和种类都非常多，例如最常见的小括号、中括号、大括号（即花括号）。

【例】书写数学表达式 $\left(\frac{1}{y} + 1\right)$ 、 $\left[\frac{1}{y} + 1\right]$ 和 $\left\{\frac{1}{y} + 1\right\}$ ：

```

1   $$x\left(\frac{1}{y}+1\right)$$
2   $$x\left[\frac{1}{y}+1\right]$$
3   $$x\left\{\frac{1}{y}+1\right\}$$

```

有时候，由于公式过长等原因，我们也可以在需要分行处插入`\\\`将括号内的公式切分成多行。

在这里，我们可以使用一系列命令代替最常用的`\left` 和`\right` 组合，如`\bigl` 和`\bigr` 组合、`\Bigl` 和`\Bigr` 组合、`\biggl` 和`\biggr` 组合、`\Biggl` 和`\Biggr` 组合来控制括号大小。

【例】利用各组合控制括号大小：

```

1 \begin{equation}
2
3 \left(x+y=z \right) \\
4 \bigl(x+y=z \bigr) \\
5 \Bigl(x+y=z \Bigr) \\
6 \biggl(x+y=z \biggr) \\
7 \Biggl(x+y=z \Biggr)
8
9 \end{equation}

```

编译后：

$$(x + y = z) (x + y = z) \quad (4.3)$$

在数学公式编辑中，除了以上常见的括号，也有一些广义的“括号”。

【例】书写数学表达式：

```

1 \begin{equation}
2   x\left|\frac{1}{y}+1\right| \\
3   x\left|\frac{1}{y}+1\right| \\
4   \left<\frac{1}{x},\frac{1}{y}\right> \\
5   \langle\frac{1}{x},\frac{1}{y}\rangle
6 \end{equation}
```

编译后：

$$x \left| \frac{1}{y} + 1 \right| x \left| \frac{1}{y} + 1 \right| \left\langle \frac{1}{x}, \frac{1}{y} \right\rangle \langle \frac{1}{x}, \frac{1}{y} \rangle \quad (4.4)$$

在这里，使用半边括号，也能书写出导数的表达式。

【例】书写导数：

```

1 $$$\left.\frac{dy}{dx}\right|_{x=1}$$$
```

编译后：

$$\left. \frac{dy}{dx} \right|_{x=1}$$

4.2.4 空心符号

在数学表达式中，我们有时候会用到一些约定俗成的空心符号表示集合，这包括：

- 空心 R 符号 \mathbb{R} 表示由所有实数构成的集合。
- 空心 Z 符号 \mathbb{Z} 表示由所有整数构成的集合。
- 空心 N 符号 \mathbb{N} 表示由所有非负整数构成的集合，如果要表示正整数，使用符号 \mathbb{N}_+ 即可。
- 空心 C 符号 \mathbb{C} 表示由所有复数构成的集合。

上面的例子使用了`\(\mathbb{字母}\)`需要注意的是，要想让 LaTeX 成功编译出这些空心符号，我们需要调用特定的工具包，即`\usepackage{amsfonts}`，一般而言，为了保证公式的编译不出现意外，还会用到其他工具包，即`\usepackage{amsfonts, amssymb, amsmath}`。当然，除了这些，还有许多其他符号，这里不再一一赘述。

【例】书写表达式 $X \in \mathbb{R}^{m \times n}$ ：

```

1  \usepackage{amsfonts}
2  \begin{document}
3  $$X \in \mathbb{R}^{m \times n}$$
4  \end{document}

```

【例】使用工具包 `bbold` 中的`\mathbb`命令书写空心的 1、2、3、4、5：

```

1  \usepackage{bbold}
2  \begin{document}
3  $$\mathbb{1}, \mathbb{2}, \mathbb{3}, \mathbb{4}, \mathbb{5}$$
4  \end{document}

```

编译后：

1, 2, 3, 4, 5

4.2.5 特殊函数

上标很多时候是用来表示变量的幂，例如 x^2 表示 x 的平方，由此可以用上标书写出指数函数如 $y = x^2$ 等。与指数函数对应的一类常用函数被称为对数函数，是指数函数的反函数，LaTeX 提供了一些跟对数函数相关的命令，包括`\log`、`\ln`，在命令`\log`中，我们可以自行设置底数，而命令`\ln`则表示底数为自然数的对数。

有时候，为了简化数学表达式，我们可能会采用求和或者连乘的写法，在 LaTeX 中，求和符号对应的命令为`\sum_{下标}^{上标}`，连乘符号对应的命令为`\prod_{下标}^{上标}`。

- `\sum_{i=1}^n x_i` 求和公式即 $\sum_{i=1}^n x_i$
- `\prod_{i=1}^n x_i` 连乘公式即 $\prod_{i=1}^n x_i$

另外，在初等数学的几何中，我们学过了正弦、余弦等，这些在 LaTeX 都有定义好的命令可供直接使用。

- $y=\sin(x)$ 正弦函数 $y = \sin(x)$
- $y=\arcsin(x)$ 反正弦函数 $y = \arcsin(x)$
- $y=\cos(x)$ 余弦函数 $y = \cos(x)$
- $y=\arccos(x)$ 反余弦函数 $y = \arccos(x)$
- $y=\tan(x)$ 余弦函数 $y = \tan(x)$
- $y=\arctan(x)$ 反余弦函数 $y = \arctan(x)$

4.3 希腊字母

我们在初等数学中便已经学习到了一些常用的希腊字母，例如最常见的 π （对应于`\pi`），圆周率 π 约等于 3.14，圆的面积为 πr^2 、周长为 $2\pi r$ 。在几何学中，我们习惯用各种希腊字母表示度数，如 α （对应于`\alpha`）、 β （对应于`\beta`）、 θ （对应于`\theta`）、 ϕ （对应于`\phi`）、 ψ （对应于`\psi`）、 φ （对应于`\varphi`），使用希腊字母既方便，也容易区分子 x,y,z 等其他变量。

实际上，这些希腊字母也可以用来作为变量，在概率论与数理统计中常常出现的变量就包括：

- 正态分布中的 μ （命令为`\mu`）、 σ （命令为`\sigma`）；
- 泊松分布中的 λ （命令为`\lambda`）；
- 通常表示自由度的希腊字母为 ν （命令为`\nu`）。

另外，在不等式中经常用到的希腊字母有：

- δ 命令为`\delta`；
- ϵ 命令为`\epsilon`；
- γ 命令为`\gamma`；
- η 命令为`\eta`；
- κ 命令为`\kappa`；
- ρ 命令为`\rho`；

- τ 命令为 \tau;
- ω 命令为 \omega;

当然，前面提到的这些希腊字母在用途上并没有严格的界定，很多时候，我们书写数学表达式时可以根据需要选用适当的希腊字母。

【例】 书写椭圆 $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ 的面积公式 $S = \pi ab$:

```
1   $$S=\pi\ ab$$ % 椭圆面积公式
```

【例】 书写不等式 $a^\alpha b^\beta \cdots k^\kappa l^\lambda \leq a\alpha + b\beta + \cdots + k\kappa + l\lambda$:

```
1   $$a^{\alpha}b^{\beta}\cdots k^{\kappa}l^{\lambda}\leq a\alpha+b\beta+\cdots+k\kappa+l\lambda$$
```

【例】 书写不等式 $\phi\left(\frac{x_1+x_2+\cdots+x_n}{n}\right) \leq \frac{\phi(x_1)+\phi(x_2)+\cdots+\phi(x_n)}{n}$:

```
1   $$\phi\left(\frac{x_1+x_2+\cdots+x_n}{n}\right) \leq \frac{\phi(x_1)+\phi(x_2)+\cdots+\phi(x_n)}{n}$$
```

与英文字母类似的是，有些希腊字母不但有小写字母，还有大写字母，具体如下：

- 命令 \Gamma 对应于希腊字母 Γ ，命令 \varGamma 对应于 Γ ；
- 命令 \Delta 对应于希腊字母 Δ ，命令 \varDelta 对应于 Δ ；
- 命令 \Theta 对应于希腊字母 Θ ，命令 \varTheta 对应于 Θ ；
- 命令 \Lambda 对应于希腊字母 Λ ，命令 \varLambda 对应于 Λ ；
- 命令 \Pi 对应于希腊字母 Π ，命令 \varPi 对应于 Π ；
- 命令 \Sigma 对应于希腊字母 Σ ，命令 \varSigma 对应于 Σ ；
- 命令 \Phi 对应于希腊字母 Φ ，命令 \varPhi 对应于 Φ ；
- 命令 \Omega 对应于希腊字母 Ω ，命令 \varOmega 对应于 Ω 。

从这些大写希腊字母中可以看到：大写希腊字母的命令是将小写希腊字母的命令首字母进行大写，但这些大写希腊字母与小写希腊字母的区别却不仅仅是尺寸不同；当大写希腊字母作为变量时，可以采用斜体字。

【例】 书写 $\Delta x + \Delta y$ 和 $(i, j, k) \in \Omega$ ：

```

1  $$\Delta x+\Delta y$$
2  $$(i,j,k)\in\Omega$$

```

4.4 微积分

事实上，数学公式的范畴极为广泛，我们所熟知的大学数学课程中，微积分、线性代数、概率论与数理统计中数学表达式的符号系统均大不相同。本节将主要介绍如何使用 LaTeX 对微积分中的数学表达式进行书写和编译。

4.4.1 极限

求极限是整个微积分中的基石，例如 $\lim_{x \rightarrow 2} x^2$ 对应的 LaTeX 代码为 `\lim_{x \rightarrow 2} x^2`。

【例】 书写以下求极限的表达式：

$$\lim_{x \rightarrow -\infty} \frac{3x^2 - 2}{3x - 2x^2} = \lim_{x \rightarrow -\infty} \frac{x^2 \left(3 - \frac{2}{x^2}\right)}{x^2 \left(\frac{3}{x} - 2\right)} = \lim_{x \rightarrow -\infty} \frac{3 - \frac{2}{x^2}}{\frac{3}{x} - 2} = -\frac{3}{2}$$

```

1  $$\lim_{x \rightarrow -\infty} \frac{3x^2 - 2}{3x - 2x^2} = \lim_{x \rightarrow -\infty} \frac{x^2 \left(3 - \frac{2}{x^2}\right)}{x^2 \left(\frac{3}{x} - 2\right)} = \lim_{x \rightarrow -\infty} \frac{3 - \frac{2}{x^2}}{\frac{3}{x} - 2} = -\frac{3}{2}

```

【例】 书写极限 $\lim_{\Delta t \rightarrow 0} \frac{s(t + \Delta t) + s(t)}{\Delta t}$ & $\lim_{\Delta t \rightarrow 0} \frac{s(t + \Delta t) + s(t)}{\Delta t}$ ：

```

1  $$\lim_{\Delta t \rightarrow 0} \frac{s(t + \Delta t) + s(t)}{\Delta t} \& \displaystyle \lim_{\Delta t \rightarrow 0} \frac{s(t + \Delta t) + s(t)}{\Delta t}$$

```

4.4.2 导数

在微积分中，给定函数 (x) 后，我们能够将其导数定义为：

$$f'(a) = \lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a}$$

用 LaTeX 书写这条公式为：

```
1 $$f^{\prime}(a)=\lim _{x \rightarrow a}\{x \backslash \text { to } a\} \backslash \text { frac }\{f(x)-f(a)\}\{x-a\}$$
```

有时候，为了让分数的形式在直观上不显得过大，可以用：

$$f'(a) = \lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a}$$

用 LaTeX 书写这条公式为：

```
1 $$f^{\prime}(a)=\lim \backslash \text { limits }_{\{x \backslash \text { to } a\}} \backslash \text { frac }\{f(x)-f(a)\}\{x-a\}$$
```

其中，`\lim` 和 `\limits` 两个命令需要配合使用。

需要注意的是，`f\prime(x)` 中的`\prime` 命令是标准写法。

【例】 书写导数的定义 $f'(x) = \lim_{\Delta x \rightarrow 0} \frac{f(x+\Delta x)-f(x)}{\Delta x}$ ：

```
1 $$f^{\prime}(x)=\lim _{\{\Delta x \rightarrow 0\}} \backslash \text { frac }\{f(x+\Delta x)-f(x)\}\{\Delta x\}$$
```

【例】 书写函数 $f(x) = 3x^2 + 2x^3 + 1$ 的导数 $f'(x) = 15x^4 + 6x^2$ ：

```
1 $$f^{\prime}(x)=15 x^{\{4\}}+6 x^{\{2\}}$$
```

微分在微积分中举足轻重，`\mathrm{d}` 为微分符号 d 的命令，一般而言，微分的标准写法为 $\frac{d^n}{dx^n} f(x)$ 。

【例】 书写微分 $\frac{d}{dx} f(x) = 15x^4 + 6x^2$ 和 $\frac{d^2}{dx^2} f(x) = 60x^3 + 12x$ ：

```
1 $$\frac{\mathrm{d}}{\mathrm{d} x} f(x)=15 x^{\{4\}}+6 x^{\{2\}}$$
```

```
2 $$\frac{\mathrm{d}^2}{\mathrm{d} x^2} f(x)=60 x^{\{3\}}+12 x$$
```

在微积分中，偏微分符号 ∂ 的命令为 ∂ ，对于任意函数 $f(x, y)$ ，偏微分的标准写法为 $\frac{\partial^n}{\partial x^n} f(x, y)$ 或 $\frac{\partial^n}{\partial y^n} f(x, y)$ 。

【例】 书写函数 $f(x, y) = 3x^5y^2 + 2x^3y + 1$ 的偏微分 $\frac{\partial}{\partial x}f(x, y) = 15x^4y^2 + 6x^2y$ 和 $\frac{\partial}{\partial y}f(x, y) = 6x^5y + 2x^3$ ：

```
1   $$\frac{\partial}{\partial x}f(x, y)=15x^4y^2+6x^2y$$
2   $$\frac{\partial}{\partial y}f(x, y)=6x^5y+2x^3$$
```

【例】 书写偏导数 $z = \mu \left. \frac{\partial y}{\partial x} \right|_{x=0}$ ：

```
1   $$z=\left. \mu \frac{\partial y}{\partial x} \right|_{x=0}$$
```

4.4.3 积分

积分的标准写法为 $\int_a^b f(x) dx$ ，代码为 $\int_{a}^{b} f(x) \mathrm{d}x$ ，其中， \int 表示积分，是英文单词 integral 的缩写形式，使用 \backslash ，的目的是引入一个空格。

【例】 书写积分 $\int \frac{dx}{\sqrt{a^2+x^2}} = \frac{1}{a} \arcsin\left(\frac{x}{a}\right) + C$ 和 $\int \tan^2 x dx = \tan x - x + C$ ：

```
1   $$\int \frac{\mathrm{d}x}{\sqrt{a^2+x^2}}=\frac{1}{a} \arcsin\left(\frac{x}{a}\right)+C$$
2   $$\int \tan^2 x \mathrm{d}x=\tan x-x+C$$
```

【例】 书写积分 $\int_a^b [\lambda_1 f_1(x) + \lambda_2 f_2(x)] dx = \lambda_1 \int_a^b f_1(x) dx + \lambda_2 \int_a^b f_2(x) dx$ 和 $\int_a^b f(x) dx = \int_a^c f(x) dx + \int_c^b f(x) dx$ ：

```
1   $$\int \left[ \lambda_1 f_1(x) + \lambda_2 f_2(x) \right] \mathrm{d}x=\lambda_1 \int f_1(x) \mathrm{d}x+\lambda_2 \int f_2(x) \mathrm{d}x$$
2   $$\int f(x) \mathrm{d}x=\int_a^c f(x) \mathrm{d}x+\int_c^b f(x) \mathrm{d}x$$
```

【例】书写积分：

$$\begin{aligned}
 V &= 2\pi \int_0^1 x \left\{ 1 - (x-1)^2 \right\} dx \\
 &= 2\pi \int_0^2 \left\{ -x^3 + 2x^2 \right\} dx \\
 &= 2\pi \left[-\frac{1}{4}x^4 + \frac{2}{3}x^3 \right]_0^2 \\
 &= 8\pi/3
 \end{aligned} \tag{4.5}$$

```

1 \begin{equation}
2   \begin{aligned}
3     V&=2\pi\int_0^1 x\left(1-(x-1)^2\right)dx,\mathrm{d}x \\
4     &=2\pi\int_0^2 \left\{ -x^3+2x^2 \right\} dx,\mathrm{d}x \\
5     &=2\pi\left[ -\frac{1}{4}x^4+\frac{2}{3}x^3 \right]_0^2 \\
6     &=8\pi/3
7   \end{aligned}
8 \end{equation}

```

上述介绍的都是一重积分，在微积分课程中，还有二重积分、三重积分等，对于一重积分，LaTeX 提供的基本命令为`\int`，二重积分为`\iint`，三重积分为`\iiint`、四重积分为`\iiiint`，当积分为五重或以上时，一般使用`\idotsint`，即 $\int \cdots \int$ 。

【例】书写积分 $\iint_D f(x, y) d\sigma$ 和 $\iiint_{\Omega} (x^2 + y^2 + z^2) dv$ ：

```

1 $$\iint\limits_D f(x,y),\mathrm{d}\sigma$$
2 $$\iiint\limits_{\Omega} (x^2+y^2+z^2)\mathrm{d}v$$

```

在积分中，有一种特殊的积分符号是在标准的积分符号上加上一个圈，可用来表示计算曲线曲面积分，即 $\oint_C f(x) dx + g(y) dy$ ，代码为：

```

1 $$\oint\limits_C f(x),\mathrm{d}x+g(y),\mathrm{d}y$$

```

4.5 线性代数

4.5.1 矩阵

在线性代数和众多代数课程中，矩阵是最基础的代数结构，可以直观上理解为数表。使用 LaTeX 时，我们可以用`\begin{array} \end{array}` 环境来书写矩阵。

【例】 书写矩阵
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$
 和
$$\begin{array}{c|cc} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \end{array}$$
:

```
1   $$\left.\left[\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \end{array}\right]\right.\right]$$
2   $$\left.\left[\begin{array}{c|cc} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \end{array}\right]\right.\right]$$
```

当然，除了`\begin{array} \end{array}`，我们还可以用`\begin{matrix} \end{matrix}`等一系列环境来书写矩阵。

- *smallmatrix* 环境，编译结果为：

$$\begin{smallmatrix} 1 & 2 & 3 \\ 4 & 5 & 7 \end{smallmatrix}$$

- *matrix* 环境，编译结果为：

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 7 \end{matrix}$$

- *pmatrix* 环境，编译结果为：

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 7 \end{pmatrix}$$

- *bmatrix* 环境，编译结果为：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 7 \end{bmatrix}$$

- *Bmatrix* 环境，编译结果为：

$$\left\{ \begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 7 \end{array} \right\}$$

- *vmatrix* 环境，编译结果为：

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 7 \end{vmatrix}$$

- *Vmatrix* 环境，编译结果为：

$$\left\| \begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 7 \end{array} \right\|$$

在线性代数中，一般用粗体字母或者符号表示矩阵，例如 **A,B,C,X,Y,Z**。

```

1   $\\mathbf{A}$,
2   $\\mathbf{B}$,
3   $\\mathbf{C}$,
4   $\\mathbf{X}$,
5   $\\mathbf{Y}$,
6   $\\mathbf{Z}$
```

【例】书写矩阵 $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$

```

1   $$\\mathbf{A}=\\begin{bmatrix}
2     a_{11} & a_{12} & \\cdots & a_{1n} \\
3     a_{21} & a_{22} & \\cdots & a_{2n} \\
4     \\vdots & \\vdots & \\ddots & \\vdots \\
5     a_{m1} & a_{m2} & \\cdots & a_{mn}
6   \\end{bmatrix}$$
```

【例】书写矩阵 $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix} = \begin{bmatrix} c & e & f \\ b & d & e \\ a & b & c \end{bmatrix}$

```

1   $$\\begin{bmatrix}
2     0 & 0 & 1 \\
3     0 & 1 & 0 \\
4     1 & 0 & 0
5   \\end{bmatrix}\\begin{bmatrix}
6     a & b & c \\
7     b & d & e \\
8     c & e & f
9   \\end{bmatrix}=\\begin{bmatrix}
10    c & e & f \\
11    b & d & e \\
12    a & b & c
13  \\end{bmatrix}$$
```

4.5.2 符号

作用于矩阵的符号可分为标记符号和运算符号，就标记符号而言，有：

- 矩阵的逆，写作 \mathbf{A}^{-1} ；
- 矩阵的伪逆，写作 \mathbf{A}^+ 或 \mathbf{A}^\dagger ；
- 矩阵的转置，写作 \mathbf{A}^T 或 \mathbf{A}^\top ；
- 酉矩阵的转置，写作 \mathbf{A}^H ；
- 矩阵的秩，写作 $\text{rank}(\mathbf{A})$ ；
- 矩阵的迹，写作 $\text{Tr}(\mathbf{A})$ ；
- 矩阵的行列式，写作 $\det(\mathbf{A})$ ；

上面的相关代码如下：

```

1   $$\mathbf{A}^{-1}$$ % 矩阵的逆
2   $$\mathbf{A}^+$$ % 矩阵的伪逆
3   $$\mathbf{A}^\dagger$$ % 矩阵的伪逆
4   $$\mathbf{A}^T$$ % 矩阵的转置
5   $$\mathbf{A}^\top$$ % 矩阵的转置
6   $$\mathbf{A}^H$$ % 酉矩阵的转置
7   $$\text{rank}(\mathbf{A})$$ % 矩阵的秩
8   $$\text{Tr}(\mathbf{A})$$ % 矩阵的迹
9   $$\det(\mathbf{A})$$ % 矩阵的行列式

```

【例】书写矩阵运算规则

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$

$$(\mathbf{A} + \mathbf{B})^\top = \mathbf{A}^\top + \mathbf{B}^\top$$

$$(\mathbf{A} + \mathbf{B})^H = \mathbf{A}^H + \mathbf{B}^H$$

$$\text{Tr}(\mathbf{A} + \mathbf{B}) = \text{Tr}(\mathbf{A}) + \text{Tr}(\mathbf{B})$$

$$\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$$

```

1   $$\left(\mathbf{A}\mathbf{B}\right)^{-1}=\mathbf{B}^{-1}\mathbf{A}^{-1}$$
2   $$\left(\mathbf{A}+\mathbf{B}\right)^{\text{top}}=\mathbf{A}^{\text{top}}+\mathbf{B}^{\text{top}}$$
3   $$\left(\mathbf{A}+\mathbf{B}\right)^H=\mathbf{A}^H+\mathbf{B}^H$$
4   $$\operatorname{Tr}\left(\mathbf{A}+\mathbf{B}\right)=\operatorname{Tr}\left(\mathbf{A}\right)+\operatorname{Tr}\left(\mathbf{B}\right)$$
5   $$\det\left(\mathbf{A}\mathbf{B}\right)=\det\left(\mathbf{A}\right)\det\left(\mathbf{B}\right)$$

```

【例】书写 $\frac{\partial}{\partial \mathbf{x}} \operatorname{Tr}(\mathbf{AxB}) = \mathbf{A}^\top \mathbf{B}^\top$

```

1   $$\frac{\partial}{\partial \mathbf{x}} \operatorname{Tr}(\mathbf{AxB})=\mathbf{A}^{\text{top}}\mathbf{B}^{\text{top}}$$

```

【例】书写多元正态分布的概率密度函数

$$p(\mathbf{x}) = \frac{1}{\sqrt{\det(2\pi)}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

```

1   $$p\left(\mathbf{x}\right)=\frac{1}{\sqrt{\det\left(2\pi\right)}}\exp\left[-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}\right)^\top\boldsymbol{\Sigma}^{-1}\left(\mathbf{x}-\boldsymbol{\mu}\right)\right]$$

```

【例】书写混合高斯分布的概率密度函数

$$p(\mathbf{x}) = \sum_{k=1}^K \rho_k \frac{1}{\sqrt{\det(2\pi_k)}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right]$$

```

1   $$p\left(\mathbf{x}\right)=\sum_{k=1}^K\rho_k\frac{1}{\sqrt{\det\left(2\pi_k\right)}}\exp\left[-\frac{1}{2}\left(\mathbf{x}-\boldsymbol{\mu}_k\right)^\top\boldsymbol{\Sigma}_k^{-1}\left(\mathbf{x}-\boldsymbol{\mu}_k\right)\right]$$

```

【例】书写混合高斯分布的概率密度函数

$$p(\mathbf{x}) = \sum_{k=1}^K \rho_k \frac{1}{\sqrt{\det(2\pi_k)}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right]$$

```

1   $$p\left(\mathbf{x}\right)=\sum_{k=1}^K\rho_k\frac{1}{\sqrt{\operatorname{det}\left(2\pi\mathbf{\Sigma}_k\right)}}\exp\left[-\frac{1}{2}\left(\mathbf{x}-\mathbf{\mu}_k\right)^T\mathbf{\Sigma}_k^{-1}\left(\mathbf{x}-\mathbf{\mu}_k\right)\right]$$

```

就运算符号而言，有：

- Hadamard 积，如 $\mathbf{A} \odot \mathbf{B}$ ；
- Hadamard 积，如 $\mathbf{A} \otimes \mathbf{B}$ ；
- 内积，如 $\langle \mathbf{A}, \mathbf{B} \rangle$ ；

上面的相关代码如下：

```

1   $\mathbf{A}\cdot\mathbf{B}$ % Hadamard积
2   $\mathbf{A}\otimes\mathbf{B}$ % Kronecker积
3   $\langle\mathbf{A},\mathbf{B}\rangle$ % 内积

```

【例】 书写 Kronecker 积运算

$$\mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \cdots & x_{1n}\mathbf{Y} \\ x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \cdots & x_{2n}\mathbf{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1}\mathbf{Y} & x_{m2}\mathbf{Y} & \cdots & x_{mn}\mathbf{Y} \end{bmatrix} \quad (4.6)$$

```

1 \begin{equation}
2   \mathbf{X}\otimes\mathbf{Y}=\left[\begin{array}{cccc}
3     x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \cdots & x_{1n}\mathbf{Y} \\
4     x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \cdots & x_{2n}\mathbf{Y} \\
5     \vdots & \vdots & \ddots & \vdots \\
6     x_{m1}\mathbf{Y} & x_{m2}\mathbf{Y} & \cdots & x_{mn}\mathbf{Y}
7   \end{array}\right]
8 \end{equation}

```

【例】 书写李亚普诺夫方程

$$\begin{aligned} \mathbf{AX} + \mathbf{XB} &= \mathbf{C} \\ \Rightarrow \quad \text{vec}(\mathbf{X}) &= (\mathbf{I} \otimes \mathbf{A} + \mathbf{B}^\top \otimes \mathbf{I})^{-1} \text{vec}(\mathbf{C}) \end{aligned} \quad (4.7)$$

```

1 \begin{equation}
2 \begin{aligned}
3 &\mathbf{A} \mathbf{X} + \mathbf{X} \mathbf{B} = \mathbf{C} \\
4 \Rightarrow &\operatorname{vec}(\mathbf{left}(\mathbf{X})\mathbf{right}) = \mathbf{left}(\mathbf{I}) \\
&\mathbf{times} \mathbf{A} + \mathbf{B}^{\top} \mathbf{times} \mathbf{I} \mathbf{right})^{-1} \operatorname{vec}(\mathbf{left}(\mathbf{C})\mathbf{right}) \\
5 \end{aligned}
6 \end{equation}

```

4.5.3 范数

一般而言，代数结构有标量、向量、矩阵、张量这几种，对于这几种代数结构来说，有时候，为了描述数据特征等，会引入范数这个概念。对于向量而言，常用的范数包括 ℓ_0 范数、 ℓ_1 范数、 ℓ_2 范数，具体来说，给定任意向量 \mathbf{x} ，有：

- ℓ_0 范数为 $\|\mathbf{x}\|_0 = \sqrt[0]{\sum_i x_i^0}$;
- ℓ_1 范数为 $\|\mathbf{x}\|_1 = \sum_i |x_i|$;
- ℓ_2 范数为 $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$;

上面的相关代码如下：

```

1 $ \left\| \mathbf{x} \right\|_0 = \sqrt{\sum_i x_i^0} \% 范数 0
2 $ \left\| \mathbf{x} \right\|_1 = \sum_i |x_i| \% 范数 1
3 $ \left\| \mathbf{x} \right\|_2 = \sqrt{\sum_i x_i^2} \% 范数 2

```

对于矩阵而言，常用的范数包括 F 范数、核范数。给定任意矩阵 \mathbf{X} ，F 范数的表达式为 $\|\mathbf{X}\|_F$ ；核范数的表达式为 $\|\mathbf{X}\|_*$ 。

```

1 $ \left\| \mathbf{X} \right\|_F \% F 范数
2 $ \left\| \mathbf{X} \right\|_* \% 核范数

```

【例】 书写

$$\frac{1}{2} \mathbf{x}^\top \mathbf{A}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{A}^\top \mathbf{b} = \frac{1}{2} \|\mathbf{A} \mathbf{x} - \mathbf{b}\|_2^2 - \frac{1}{2} \|\mathbf{b}\|_2^2$$

```

1   $$\frac{1}{2}\mathbf{x}^{\top}\mathbf{A}^{\top}\mathbf{A}\mathbf{x}-\mathbf{x}^{\top}\mathbf{A}^{\top}\mathbf{A}\mathbf{x}=\frac{1}{2}\left(\mathbf{x}^{\top}\mathbf{A}\mathbf{x}-\mathbf{x}^{\top}\mathbf{B}\mathbf{x}\right)^2-\frac{1}{2}\left(\mathbf{x}^{\top}\mathbf{B}\mathbf{x}\right)^2$$

```

【例】书写

$$\begin{aligned}
 & \left| \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ \hline a_{51} & a_{52} & a_{53} & a_{54} \\ a_{61} & a_{62} & a_{63} & a_{64} \end{array} \right] - \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \right|_F \\
 = & \left| \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ \hline a_{51} & a_{52} & a_{53} & a_{54} \\ a_{61} & a_{62} & a_{63} & a_{64} \end{array} \right] - \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{21} & c_{12} & c_{22} \end{bmatrix} \right|_F
 \end{aligned} \tag{4.8}$$

```

1 \begin{equation}
2 \begin{aligned}
3 & \left| \left[ \begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ \hline a_{51} & a_{52} & a_{53} & a_{54} \\ a_{61} & a_{62} & a_{63} & a_{64} \end{array} \right] - \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{21} & c_{12} & c_{22} \end{bmatrix} \right|_F
4 & \left| \left[ \begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ \hline a_{51} & a_{52} & a_{53} & a_{54} \\ a_{61} & a_{62} & a_{63} & a_{64} \end{array} \right] - \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \right|_F
5 \end{aligned}
6 \end{equation}

```



第五章 表格制作

表格是一种针对信息和数据的有效呈现方式，一般采用行和列组合的结构。在撰写科技论文时，往往需要使用表格对信息或数据进行归纳、整理以及比较分析。本章主要介绍如何在 LaTeX 中制作和编辑表格，内容包括：

- 第 1 节是对制作表格的环境和命令进行基本介绍；
- 第 2、3 节介绍如何合并单元格、插入斜线与表注；
- 第 4 节介绍如何调整表格样式，包括表格尺寸、单元格自动对齐与换行、小数点对齐、行高、列宽、线宽、以及字体大小；
- 第 5~8 节介绍如何创建特殊表格，分别为彩色表格、三线表格、跨页表格、旋转表格（或纵向表格）；
- 第 9 节介绍如何从 csv 文件中导入表格。

通过本章的学习，读者会对 LaTeX 制作和编辑表格有较为全面的认识，能够制作出科技论文中的各类表格。

5.1 基本介绍

表格是展现数据的一种常用方式。LaTeX 提供了多种表格环境可用于制作各类表格，例如 *tabular*、*tabular**、*tabularx*、*tabulary*、*table* 和 *longtable* 等。其中比较常用的方法是将 *tabular* 环境嵌入到 *table* 环境中，可以创建包含表格内容、表格标题、引用标签等属性的完整表格。

5.1.1 tabular 环境：创建表格内容

通过创建 tabular 环境可以定义表格内容、对齐方式、外观样式等，使用方式与前面章节中介绍的使用 array 环境制作数表（即矩阵）的方式类似。例如：

$$\begin{array}{|c|c|c|} \hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline \end{array}$$

上面矩阵的代码为：

```

1 $$\left[ \begin{array}{|c|c|c|} \hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline \end{array} \right]$$
2
3
4
5
6
7

```

不妨将上述代码中的 array 环境改写为 tabular 环境，并去掉\left[和\right]命令，得到如下所示的代码语句：

```

1 \begin{tabular}{|c|c|c|} \hline
2 a & b & c \\ \hline
3 \hline
4 d & e & f \\ \hline
5 \hline
6 g & h & i \\ \hline
7 \end{tabular}

```

编译上述代码，可以得到类似的结果：

$$\begin{array}{|c|c|c|} \hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline \end{array}$$

不同之处在于，array 环境制作的数表是属于数学公式，而使用 tabular 环境制作得到的表格则属于文本内容，但两者的用法及命令格式极其相似。在 tabular 环境下：

- 在\begin{tabular} 命令后的 {} 内设置表格的列类型参数，包括：

	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = x$	1	2	3	4
$y = x^2$	1	4	9	16
$y = x^3$	1	8	27	64

表 5.1: 编译后的表格

- 设置每列的单元格对齐方式。对齐方式选项包括 l 、 c 和 r , 即 left、center 和 right 的首字母, 分别对应左对齐、居中对齐和右对齐, 每个字母对应一列;
- 创建表格列分隔线。表格列分隔线以 | 符号表示, | 符号的个数表示列分隔线中线的个数, 如 | 表示使用单线分隔列, || 表示使用双线分隔列, 以此类推。分割线符号可以设置在列对齐方式选项的左侧或右侧, 分别表示创建列的左分隔线和右分隔线。
- 使用\\符号表示一行内容的结束;
- 使用 & 符号划分行内的单元格;
- 使用\hline 命令创建行分隔线。

下面给出一个示例让读者对 tabular 环境的使用方式能有更深刻的印象。

【例】使用 tabular 环境制作一个简单的表格, {l|cccr} 命令调整列对齐方式:

```

1 \begin{tabular}{l|cccr} % 调整对齐方式
2   \hline
3   & $x=1$ & $x=2$ & $x=3$ & $x=4$ \\
4   \hline
5   $y=x$ & 1 & 2 & 3 & 4 \\
6   $y=x^2$ & 1 & 4 & 9 & 16 \\
7   $y=x^3$ & 1 & 8 & 27 & 64 \\
8   \hline
9 \end{tabular}

```

编译上述代码, 得到表格如表5.1所示。

5.1.2 table 环境：自动编号与浮动表格

使用 table 环境嵌套 tabular 环境，能够为创建的表格进行自动递增编号。此外，可以使用\caption{} 命令设置表格标题、使用\label{} 命令为表格建立索引标签、使用 \centering 命令将表格置于文档中间，如下所示：

```

1  \begin{table}
2    \centering
3      \caption{Title of a table.}
4      \label{Label of the table}
5      \begin{tabular}
6        % 表格内容
7        \end{tabular}
8    \end{table}

```

下例创建表格嵌入到 table 环境中，创建了一个位置居中、并且具有标题、索引、自动编号的表格。

【例】 使用 table 和 tabular 环境制作一个简单的表格：

```

1  \begin{table}
2    \centering
3      \begin{tabular}{l|cccr}
4        \hline
5        & $x=1\$ & $x=2\$ & $x=3\$ & $x=4\$ \\
6        \hline
7        $y=x\$ & 1 & 2 & 3 & 4 \\
8        $y=x^{2}\$ & 1 & 4 & 9 & 16 \\
9        $y=x^{3}\$ & 1 & 8 & 27 & 64 \\
10       \hline
11       \end{tabular}
12   \end{table}

```

编译上述代码，得到表格如5.2。

事实上，在 table 环境中创建的表格属于浮动元素：

浮动元素（floating）是指不能跨页分割的元素，比如图片和表格。一般而言，浮动元素的显示位置未必是代码的位置，比如，当页面空间不足时，LaTeX 会根据内置的算法尝试将浮动元素放置到后面的页面中，避免出

	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = x$	1	2	3	4
$y = x^2$	1	4	9	16
$y = x^3$	1	8	27	64

表 5.2: 编译后的表格

现内容跨页分割或者页面大量留白的情况，从而创建更协调也更专业的文档。

通过在`\begin{table}[]` 的`[]`中设置位置控制参数，可以为浮动表格指定期望放置位置，各参数值及其含义如下：

- `h`: 英文单词 `here` 的首写字母，表示代码当前位置；
- `t`: 英文单词 `top` 的首写字母，表示页面顶部位置；
- `b`: 英文单词 `bottom` 的首写字母，表示页面底部位置；
- `p`: 英文单词 `page` 的首写字母，表示后面的页面；
- `! : !`: ! 参数一般与其它位置参数配合使用，表示当空间足够时，强制将表格放在指定位置。如`!h` 表示将表格强制放到当前页面，但当页面空间不足时，也可能被放置到后续页面中；
- `H`: 表示将表格强制放在代码当前位置，具有比`!h` 更严格的效果，使用时需要先在导言区使用`\usepackage{float}` 声明语句调用 `float` 宏包。

根据需要，浮动元素的位置控制参数一般可以设置为 `h`、`b`、`t`、`p`、`!` 和 `H` 的任意无需组合。该参数的缺省值为 `tbp`，此时 LaTeX 会尝试将表格放在页面的顶端或者底端，否则会将表格放在下一页。

【例】在 `table` 环境中将表格的位置控制参数设置为 `htbp`：

```

1 \begin{table}[htbp] % 设置位置参数
2   \centering
3   % \caption{The values of some basic functions.}
4   \begin{tabular}{l|cccr}
5     \hline
6     & $x=1$ & $x=2$ & $x=3$ & $x=4$ \\
7     \hline

```

```

8   $y=x\$ & 1 & 2 & 3 & 4 \\
9   $y=x^{2\$} & 1 & 4 & 9 & 16 \\
10  $y=x^{3\$} & 1 & 8 & 27 & 64 \\
11  \hline
12  \end{tabular}
13  % \label{table1}
14  \end{table}

```

	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = x$	1	2	3	4
$y = x^2$	1	4	9	16
$y = x^3$	1	8	27	64

表 5.3: 编译后的表格

编译上述代码，得到表格如表5.3所示。

5.2 合并单元格

如果需要合并单元格，首先应在导言区声明\usepackage{multirow} 导入 *multirow* 宏包，并使用\multicolumn 命令合并同行不同列的单元格、使用 texttt\multirow 命令合并同列不同行的单元格。

5.2.1 合并不同列的单元格

合并不同列的单元格时，应在 tabular 环境中使用\multicolumn{合并列数}{合并后的列类型参数}{单元格内容} 语句定义合并单元格。此时，合并后的单元格的列类型将由\multicolumn 给出，而非\begin{tabular} 中预设的列类型参数。

【例】合并不同列的单元格：

```

1  \usepackage{multirow}
2  \begin{document}
3
4  \begin{tabular}{|1|1|1|1|}
5    \hline
6    Column1 & Column2 & Column3 & Column4 \\
7    \hline

```

```
8     \multicolumn{2}{|c|}{A1 and A2} & A3 & A4 \\
9     \hline
10    B1 & B2 & B3 & B4 \\
11    \hline
12    C1 & C2 & C3 & C4 \\
13    \hline
14 \end{tabular}
15
16 \end{document}
```

Column1	Column2	Column3	Column4
A1 and A2		A3	A4
B1	B2	B3	B4
C1	C2	C3	C4

表 5.4: 编译后的表格

编译上述代码，得到表格如5.4所示。

5.2.2 合并不同行的单元格

合并不同行的单元格时使用的语句为\multirow{合并行数}{合并后的宽度}{单元格内容}。如果把 {合并后的宽度} 参数设置为 {*}，那么 LaTeX 会根据文本内容自动设置单元格宽度。在绘制行分隔线时，使用\hline 命令会创建一条横跨表格左右两端的横线，显然不适用于合并单元格后的行。此时应用\cline{起始列号-终止列号} 命令，通过指定行分隔线的起始列和终止列，从而定制跨越了部分列的行分隔线。

【例】合并不同列的单元格，并定制行分隔线的起始点：

```

1 \begin{tabular}{|l|l|l|l|l|}
2   \hline
3   Column1 & Column2 & Column3 & Column4 \\
4   \hline
5   \multicolumn{2}{|c|}{\text{*}\{A1 and B1}} & A2 & A3 & A4 \\
6   \cline{2-4} % 创建一条从第2列到第4列的行分隔线
7   & B2 & B3 & B4 \\
8   \hline
9   C1 & C2 & C3 & C4 \\
10  \hline

```

```
11 \end{tabular}
```

Column1	Column2	Column3	Column4
A1 and B1	A2	A3	A4
	B2	B3	B4
C1	C2	C3	C4

表 5.5: 编译后的表格

编译上述代码，得到表格如5.5所示。

从上例可以看出，合并多行的单元格时，除了第一个单元格处使用\multirow 命令定义单元格，其余被合并的单元格处均留空。

5.2.3 合并不同行不同列的单元格

通过嵌套使用\multicolumn 和\multirow 命令可以实现对不同行不同列单元格的合并操作，具体语句为\multicolumn{合并列数}{合并后的列类型参数}{\multirow{合并行数}{合并后的宽度}{单元格内容}}。

【例】合并不同行不同列的单元格：

```

1 \begin{tabular}{|l|l|l|l|l|}
2   \hline
3   Column1 & Column2 & Column3 & Column4 \\
4   \hline
5   \multicolumn{2}{|c|}{\multirow{2}{*}{A1, A2, B1 and B2}} & A3 & A4 \\ % 合并
6     不同行不同列的单元格
7   \cline{3-4} % 创建一条从第3列到第4列的行分隔线
8   \multicolumn{2}{|c|}{B3 & B4} \\
9   \hline
10  C1 & C2 & C3 & C4 \\
11  \hline
12 \end{tabular}

```

编译上述代码，得到表格如表5.6所示。

Column1	Column2	Column3	Column4
A1, A2, B1 and B2	A3	A4	
	B3	B4	
C1	C2	C3	C4

表 5.6: 编译后的表格

5.3 插入斜线与表注

5.3.1 插入斜线

在制作斜线表头或填充空白单元格时，经常需要用到斜线。在 LaTeX 中，我们可以通过调用 *diagbox* 宏包及其提供的\diagbox[参数]{单元格内容 1}…{单元格内容 n} 命令将一个单元格划分为 n 个部分（即插入 (n-1) 条斜线），并且可以在 [] 中设置不同参数，从而对斜线宽度、高度、方向等属性进行调整，主要包括：

- width: 设置斜线宽度；
- height: 设置斜线高度；
- font: 设置单元格字体大小和字体类型；
- linewidth: 设置线宽；
- linecolor: 设置线的颜色（需结合 xcolor 或其他宏包使用）；
- dir: 设置斜线方向，包括 NW（默认）、NE、SW 和 SE，分别表示西北方向、东北方向、西南方向、东南方向。当仅插入一个斜线时，dir=NW 与 dir=SE、dir=NE 与 dir=SW 效果相同，分别表示插入反斜线和斜线；

当插入两个斜线时，如\diagbox[设置 dir 参数]{A}{B}{C}，使用 NW、NE、SW 和 SE 的效果分别如表5.7所示：

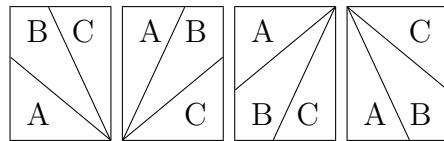


表 5.7: 将方向参数设置为 NW、NE、SW 和 SE 的效果示意图

【例】制作双斜线表头：

```

1 \begin{table}[htbp] % 设置位置参数
2   \centering
3   \caption{The values of some basic functions.}
4   \begin{tabular}{l|cccr}
5     \hline
6     \diagbox[width=5em]{y}{x} & $x=1$ & $x=2$ & $x=3$ & $x=4$ \\
7     \hline
8     $y=x$ & 1 & 2 & 3 & 4 \\
9     $y=x^2$ & 1 & 4 & 9 & 16 \\
10    $y=x^3$ & 1 & 8 & 27 & 64 \\
11    \hline
12  \end{tabular}
13  \label{table1}
14 \end{table}

```

编译上述代码，得到表格如表5.8所示。

	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = x$	1	2	3	4
$y = x^2$	1	4	9	16
$y = x^3$	1	8	27	64

表 5.8: 斜线表示例

5.3.2 插入表注

为了对表格内容进行解释说明，通常会在表格底部添加注释，即表注。在 LaTeX 中添加表注的方式有多种，其中比较常用的一种是使用 *threeparttable* 宏包及其相关命令，可以在表格底部生成与表格内容同宽的表注，并且当注释内容过长时可以实现自动换行，相比于其它方式更协调一致。

具体是在 tabular 环境外嵌套一层 *threeparttable* 环境，并在 tabular 环境之后将表注内容添加在 *tablenotes* 环境中，由此得到的表注将会显示在表格底部。如果需要将表格内容与表注建立关联关系，可以在表格内容的相应位置使用 \tnote{索引标记} 添加表注的索引标记，并且在 *tablenotes* 环境中使用 item[索引标记] 命令创建这项表注。

【例】添加表注：

```
1 \usepackage{booktabs}
2 \usepackage{threeparttable}
3 \begin{document}
4
5 \begin{table}
6   \centering
7   \begin{threeparttable}
8     \begin{tabular}{l|cccr}
9       \toprule
10      & $x=1$ & $x=2$ & $x=3$ & $x=4$ \\
11      \midrule
12      $y=x$ & 1\tnote{*} & 2 & 3 & 4 \\
13      $y=x^2$ & 1 & 4 & 9 & 16 \\
14      $y=x^3$ & 1 & 8 & 27 & 64 \\
15      \bottomrule
16    \end{tabular}
17    \begin{tablenotes}
18      \footnotesize
19      \item[1] This is a remark example.
20      \item[2] This is another remark example and with a very long content,
21          but the contents will be wrapped.
22      \item[*] This is 1.
23    \end{tablenotes}
24    \end{threeparttable}
25  \end{table}
26 \end{document}
```

编译上述代码，得到表格如表5.9所示。

5.4 调整表格样式

通过调用一些宏包及命令可以定制表格样式，从而创建更符合要求的表格。对表格样式的调整可以分为以下 7 个方面：表格尺寸、单元格自动对齐与换行、小数点对齐、行高、列宽、线宽、以及表格字体大小。

	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$y = x$	1 [*]	2	3	4
$y = x^2$	1	4	9	16
$y = x^3$	1	8	27	64

¹ This is a remark example.

² This is another remark example and with a very long content, but the contents will be wrapped.

* This is 1.

表 5.9: 表注示例

5.4.1 表格尺寸

如果想要修改表格尺寸，首先使用\usepackage{graphicx}语句调用 graphicx 宏包，并使用\resizebox{宽度}{高度}{内容}命令，该命令以 tabular 环境构建的表格作为内容。为了避免产生不协调的尺寸，在设置参数时只需要设置 {宽度} 和 {高度} 中的其中一个即可，另一个以! 作为参数，表示根据宽高比进行自动调整。

【例】调整表格尺寸：

```

1 \usepackage{graphicx}
2
3 \begin{table}[h]
4   \centering
5   \resizebox{0.8\textwidth}{!}{%
6     \begin{tabular}{|l|l|l|l|}
7       \hline
8       Column1 & Column2 & Column3 & Column4 \\
9       \hline
10      A1 & A2 & A3 & A4 \\
11      \hline
12      B1 & B2 & B3 & B4 \\
13      \hline
14      C1 & C2 & C3 & C4 \\
15      \hline
16    \end{tabular}}
17 \end{table}
```

编译后得到表5.10。

Column1	Column2	Column3	Column4
A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4

表 5.10: 调整表格尺寸 =0.8

5.4.2 单元格自动对齐与换行

使用列类型参数 l 、 c 或 r 可以对每列的单元格设置左对齐、横向居中对齐和右对齐，但由此创建的单元格不仅无法设置顶部对齐、纵向居中对齐、以及底部对齐方式，而且单元格内容不论长短都被拉长为一行，显得不够灵活。下面介绍几种方式用于实现单元格自动对齐与换行。

使用 array 宏包实现单元格自动对齐与换行

首先在导言区使用\usepackage{array} 语句声明调用 *array* 宏包，该宏包提供了以下 6 个列类型参数分别对应不同的对齐方式：

- p{列宽}：单元格内容将根据设置的列宽自动换行，并且对齐方式为顶部对齐；
- m{列宽}：单元格内容将根据设置的列宽自动换行，并且对齐方式为纵向居中对齐；
- b{列宽}：单元格内容将根据设置的列宽自动换行，并且对齐方式为底部对齐；
- >{\raggedright\arraybackslash}：将一列的单元格内容设置为左对齐；
- >{\centering\arraybackslash}：将一列的单元格内容设置为横向居中对齐；
- >{\raggedleft\arraybackslash}：将一列的单元格内容设置为右对齐。

默认情况下，如果单独使用 p 、 m 或 b 参数，默认为左对齐。我们可以对上述参数进行组合使用，从而获得不同的对齐效果。需要注意的是，此时应使用\tabularnewline 取代\\符号作为表格一行的结束。

【例】调用 array 宏包及其提供的列类型参数实现单元格自动对齐与分行：

```

1 \usepackage{array}
2
3 \begin{table}[h]
4 \centering
5 \begin{tabular}{|>{\raggedright\arraybackslash}m{2.3cm}|>{\centering\arraybackslash}m{2.3cm}|>{\centering\arraybackslash}m{2.3cm}|>{\raggedleft\arraybackslash}m{2.3cm}|}
6 \hline
7 Column1 & Column2 Column2 & Column3 Column3 Column3 & Column4 Column4
8 Column4 Column4 \tabularnewline
9 \hline
10 Value1 & Value2 Value2 & Value3 Value3 Value3 & Value4 Value4 Value
11 4 \tabularnewline
12 \hline
13 Value1 & Value2 Value2 & Value3 Value3 Value3 & Value4 Value4 Value
14 4 \tabularnewline
15 \hline
16 \end{tabular}
17 \end{table}

```

编译后得到表5.11。

Column1	Column2 Column2	Column3 Column3 Column3	Column4 Column4 Column4 Column4
Value1	Value2 Value2	Value3 Value3 Value3	Value4 Value4 Value4 Value4
Value1	Value2 Value2	Value3 Value3 Value3	Value4 Value4 Value4 Value4

表 5.11: 单元格自动对齐与分行

通过调用 array 宏包的方式虽然可以实现自动换行，但常常需要经过反复试验才能获得想要的宽度，更方便的方式是使用 *tabularx* 宏包或 *tabulary* 宏包及其相关命令自动计算列宽。对于涉及文本的表格，更推荐使用 *tabulary* 宏包。下面分别介绍通过这两个宏包及其命令如何实现自动换行。

使用 *tabularx* 宏包实现自动换行

首先在导言区声明调用 *tabularx* 宏包，然后使用 *tabularx* 环境取代 *tabular* 环境创建表格内容，*tabularx* 环境的使用方式与 *tabular* 类似，不同之处主要在于：`\begin{tabularx}{表格宽度}{列类型}` 中应设置表格宽度；在 *tabularx* 环境中，对于需要自动换行的列，其列类型应设置为大写的 X。`X` 参数可以与 `>{\raggedright\arraybackslash}`、`>{\centering\arraybackslash}` 或 `>{\raggedleft\arraybackslash}` 进行组合使用，从而修改单元格的对齐方式。

【例】 调用 *tabularx* 宏包并设置列类型参数 X 从而实现单元格内容自动换行：

```

1 \usepackage{tabularx} % 调用tabularx宏包
2
3 \begin{table}[h]
4   \centering
5   \caption{Title of a table.}
6   \label{first_label}
7   \begin{tabularx}{\linewidth}{|X|X|X|>{\centering\arraybackslash}X|} % 将需要自
8     动换行的列的列类型参数设为X
9     \hline
10    Column1 & Column2 & Column3 & Column4 \\
11    \hline
12    This is Value1. This is Value1. & This is Value2. This is Value2. & This
13      is Value3. This is Value3. & This is Value4. This is Value4. \\
14    \hline
15    This is Value1. This is Value1. This is Value1. & This is Value2. This is
16      Value2. This is Value2. & This is Value3. This is Value3. This is
        Value3. & This is Value4. This is Value4. This is Value4. \\
17    \hline
18  \end{tabularx}
19 \end{table}

```

编译上述代码，得到表5.12。

Column1	Column2	Column3	Column4
This is Value1.	This is Value2.	This is Value3.	This is Value4.
This is Value1.	This is Value2.	This is Value3.	This is Value4.
This is Value1.	This is Value2.	This is Value3.	This is Value4.
This is Value1.	This is Value2.	This is Value3.	This is Value4.

表 5.12: 使用宏包 tabularx 实现单元格内容自动换行

使用 tabulary 宏包实现自动换行

类似地，调用 *tabulary* 宏包并使用\begin{tabulary}{表格宽度}{列类型} 环境创建表格。对于需要自动换行的列，只需将列类型改为大写字母即可，即，大写 *L* 表示左对齐并自动换行、大写 *C* 表示居中对齐并自动换行、大写 *R* 表示右对齐并自动换行。

【例】调用 *tabulary* 宏包并设置大写列类型参数（L、C 和 R）从而实现单元格内容自动换行：

```

1 \usepackage{tabulary} % 调用tabulary宏包
2
3 \begin{table}[h]
4 \centering
5 \caption{Title of a table.}
6 \label{first label}
7 \begin{tabulary}{\linewidth}{|L|C|C|R|} % 将需要自动换行的列的列类型参数改为大
8   写
9   \hline
10  Column1 & Column2 & Column3 & Column4 \\
11  \hline
12  This is Value1. This is Value1. & This is Value2. This is Value2. & This
13    is Value3. This is Value3. & This is Value4. This is Value4. \\
14  \hline
15  This is Value1. This is Value1. This is Value1. & This is Value2. This is
16    Value2. This is Value2. & This is Value3. This is Value3. This is
      Value3. & This is Value4. This is Value4. This is Value4. \\
17  \hline
18  \end{tabulary}
19 \end{table}

```

编译上述代码，得到表5.13。

Column1	Column2	Column3	Column4
This is Value1.	This is Value2.	This is Value3.	This is Value4.
This is Value1.	This is Value2.	This is Value3.	This is Value4.
This is Value1.	This is Value2.	This is Value3.	This is Value4.
This is Value1.	This is Value2.	This is Value3.	This is Value4.
This is Value1.	This is Value2.	This is Value3.	This is Value4.

表 5.13: 使用 tabulary 宏包让单元格自动换行

使用 parbox 命令实现人工换行

我们也可以通过使用\parbox 命令对表格内容进行强制换行。

【例】用 parbox 命令来实现单元格中文本强制换行:

```

1 \begin{center}
2   \begin{tabular}{|c|c|c|c|c|}
3     \hline
4     a & b & c & d \\
5     \hline
6     a & b & c & \parbox[t]{5cm}{In probability theory and statistics, the
7       continuous uniform distribution\\ or rectangular distribution is a
8       family of symmetric probability distributions.} \\
9     \hline
10    \end{tabular}
11  \end{center}

```

编译上述代码，得到表5.14。

5.4.3 小数点对齐

为了更好地描述数据，在表格中常常将数据在小数点处进行对齐，在 LaTeX 中我们可以通过使用 *dcolumn* 包实现这一目的。这个包提供了一个名为 *D* 的列类型，可以方便实现基于小数点的数字对齐以及基于其它符号的对齐，使用方式为 *D{输入符号}{输出符号}{符号后的数字位数}*。对于基于小数点的数字对齐，输入符号一般为“.”；有时需要根据特定符号进行数字对齐，比如千分位逗号，这时输入符号即为

a	b	c	d
a	b	c	In probability theory and statistics, the continuous uniform distribution or rectangular distribution is a family of symmetric probability distributions.

表 5.14: 使用 parbox 强制换行

“.”。例如, `D{.}{\cdot}{2}` 表示将某列的数据根据 “.” 符号对齐, 输出时将该符号显示为点乘符号, 并且显示 2 个小数位数。

列类型 D 可以像其它列类型一样在表格环境的开始命令处直接进行设置, 但会导致语句过长, 所以一般使用 array 宏包的`\newcolumntype` 命令定义一个新的列类型, 并为这个列类型赋予一个比较短的名称以方便调用。定义新的列类型的语句为`\newcolumntype {新列类型名称}[新列类型的参数个数]{定义新列类型}`, 例如:
`\newcolumntype{d}[1]{D{.}{\cdot}{#1}}` 表示创建一个名为 d 的新列类型, 该列类型的内容为 `D{.}{\cdot}{符号后的数字位数}`, 其中数字位数是传给 d 的参数。

【例】调用 dcolumn 宏包和列类型 D 来实现表格数据的小数点对齐:

```

1  \usepackage{dcolumn}
2  \newcolumntype{d}[1]{D{.}{\cdot}{#1}}
3  \begin{document}
4      \begin{tabular}{|l|c|r|d{3}|}
5          \hline
6          Left & Center & Right & \mathbf{Decimal} \\
7          \hline
8          1.1 & 1.1 & 1.1 & 1.1 \\
9          \hline
10         33.3 & 33.3 & 33.3 & 33.3 \\
11         \hline
12         3.333 & 3.333 & 3.333 & 3.333 \\
13         \hline
14     \end{tabular}
15 \end{document}

```

编译上述代码, 得到表5.15。

Left	Center	Right	Decimal
1.1	1.1	1.1	1.1
33.3	33.3	33.3	33.3
3.333	3.333	3.333	3.333

表 5.15: dcolumn 宏包实现小数点对齐

5.4.4 行高

如果需要调整表格整体行高,可以在导言区使用\renewcommand{\arraystretch}{行高倍数} 命令,从而根据设置的行高倍数在默认值的基础上对行高进行扩大或缩小。

【例】将表格整体行高设为两倍行距:

```

1 \documentclass[12pt]{article}
2 \renewcommand{\arraystretch}{2}
3 \begin{document}
4 % 文章内容
5 \end{document}
```

另一种调整行高的方式是通过在每行的结束标志\\ 后加上行高增减量选项,即\\[行高增减量],从而在默认值的基础上对各行行高进行增减。

【例】为表格各行设置不同的行高:

```

1 \begin{tabular}{|c|c|c|c|c|}
2   \hline
3   Column1 & Column2 & Column3 & Column4\\[0cm]
4   \hline
5   A1 & A2 & A3 & A4\\[0.2cm]
6   \hline
7   B1 & B2 & B3 & B4\\[0.4cm]
8   \hline
9   C1 & C2 & C3 & C4\\[0.6cm]
10  \hline
11 \end{tabular}
```

编译上述代码,得到表5.16。

Column1	Column2	Column3	Column4
A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4

表 5.16: 为表格各行设置不同的行高

5.4.5 列宽

在上面，我们介绍了使用 array 宏包提供的列类型参数可以在设置单元格对齐方式的同时对列宽进行调整。此外，也可以在导言区使用\setlength{\textbackslash tabcolsep}{文本和列分隔线的间距} 命令修改表格列宽，默认情况下，单元格内容与列分隔线的间距为 6pt。

【例】将表格单元格文本和列分隔线的间距设为 12pt：

```

1 \setlength{\tabcolsep}{12pt}
2 \begin{document}
3 % 文章内容
4 \end{document}
```

5.4.6 线宽

通过在导言区使用\setlength{\arrayrulewidth}{线宽} 命令，可以修改表格线宽，默认为 0.4pt。然而当线宽设置过大时，可能导致表格线交叉处不连续的情况。对此，在导言区调用 xcolor 宏包、并设置 table 选项可以解决。

【例】设置表格线宽：

```

1 \usepackage[table]{xcolor} % 调用设置了table选项的xcolor宏包
2 \setlength{\arrayrulewidth}{2pt} % 修改表格线宽
3 \begin{document}
4 % 文章内容
5 \end{document}
```

5.4.7 表格字体大小

在文本编辑中我们知道，调整字体大小的方式既有全局方式也有局部方式，其中，全局方式是通过在文档类型中指定字体大小，而局部方式则是通过一系列设置字体大小的命令，例如`\large`、`\Large`、`\huge`、`\fontsize`等，在全局字体大小的基础上作进一步的调整。类似地，在使用 LaTeX 创建表格时，我们也可以对表格字体大小做全局或局部调整。

【例】使用 Large 调整表格局部字体大小：

```
1 \begin{table}[htp]
2   \Large % 调整表格局部字体大小
3   \centering
4   \begin{tabular}{l|cccr}
5     % 表格内容
6   \end{tabular}
7 \end{table}
```

【例】使用 fontsize 调整表格局部字体大小：

```
1 \begin{table}[htp]
2   \fontsize{18pt}{24pt}\selectfont % 将字体大小设为18pt、行距设为24pt
3   \centering
4   \begin{tabular}{l|cccr}
5     % 表格内容
6   \end{tabular}
7 \end{table}
```

5.4.8 文字环绕表格

如果想要实现文字环绕表格效果，可以使用 `wrapfig` 宏包，并使用其提供的 `wraptabular` 环境嵌套 `tabular` 环境创建表格，从而达到文字环绕表格的效果。

【例】使用 `wraptabular` 环境嵌套 `tabular` 环境创建表格，实现文字环绕表格；并使用 `\begin{wraptabular}{r}{8cm}` 将表格置于文字右侧，同时将表格和文字的距离设为 8cm：

```
1 \usepackage{wrapfig}
```

```
2 \begin{document}
3 % 段落
4 \begin{wraptable}{r}{8cm}
5     \centering
6     \begin{tabular}{lcccc}
7         % 表格内容
8         \end{tabular}
9     \end{wraptable}
10    % 段落
11    \end{document}
```

5.5 创建色彩表格

通过对表格的单元格、行或列填充颜色，可以创建不同的彩色表格。为此，首先应在导言区使用 `\usepackage[table]{xcolor}` 声明语句，通过调用 xcolor 宏包提供的相关命令可以实现颜色填充。

填充单元格时，使用\cellcolor{单元格填充颜色} 单元格内容命令定义单元格内容即可。

【例】 定义具有颜色填充效果的单元格：

```
1 \documentclass[12pt]{article}
2 \usepackage[table]{xcolor} % 调用设置了table选项的xcolor宏包
3 \begin{document}
4 \begin{table}
5   \centering
6   \begin{tabular}{|l|l|l|l|l|l}
7     \hline
8     Column1 & Column2 & Column3 & Column4 \\
9     \hline
10    \cellcolor{red!80}A1 & A2 & A3 & A4 \\ % 使用\cellcolor命令设置单元格填充颜色
11    \hline
12    \cellcolor{red!50}B1 & B2 & B3 & B4 \\
13    \hline
14    \cellcolor{red!20}C1 & C2 & C3 & C4 \\
15    \hline
16  \end{tabular}
17  \caption{表格示例1}
```

Column1	Column2	Column3	Column4
A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4

表 5.17: 表格示例 1

```
18     \label{table1}
19     \end{table}
20 \end{document}
```

编译上述代码，得到表5.17：

为了达到更好的可视化效果，有时候需要为表格的奇数行和偶数行交替设置不同的填充颜色，那么只需要在 tabular 环境前使用\rowcolors{开始填充的行编号}{第一个行填充颜色}{第二个行填充颜色} 命令即可：

编译上述代码，得到表5.18：

Column1	Column2	Column3	Column4
A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4

表 5.18: 表格示例 2

5.6 创建三线表格

booktabs 宏包提供了更美观的行分隔线创建命令，常用于创建三线表格。其中，\toprule 命令常用于创建表格顶线、\bottomrule 命令常用于创建表格底线、\midrule 命令常用于创建表格标题栏和表格内容的分隔线、以及 \cmidrule{起始列号-终止列号} 命令用于创建标题栏内部的分隔线并设置分隔线的跨越范围。

【例】调用 *booktabs* 宏包及其相关命令创建三线表：

```

1  \usepackage{booktabs}
2  \usepackage{multirow}
3  \begin{document}

4

5  \begin{tabular}{cccc}
6      \toprule
7      \multicolumn{2}{c}{\textbf{Type1}} & \\
8      \cmidrule{1-2}
9      Column1 & Column2 & Column3 & Column4\\
10     \midrule
11     A1 & A2 & A3 & A4\\
12     B1 & B2 & B3 & B4\\
13     C1 & C2 & C3 & C4\\
14     \bottomrule
15 \end{tabular}
16
17 \end{document}
```

编译上述代码，得到表5.19。

Type1			
Column1	Column2	Column3	Column4
A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4

表 5.19: 三线表示例

5.7 创建跨页表格

当表格太长时，使用 `tabular` 环境创建的表格会被裁剪掉页面放不下的部分。如果想让表格在行数太多时实现自动分页，可以通过调用 `longtable` 宏包并使用 `longtable` 环境创建表格。

在 `longtable` 环境中创建表格的方式与使用 `table` 和 `tabular` 嵌套环境类似，我们也能使用 `\caption`、`\label` 命令分别创建表格标题和索引标签。不同之处主要在于，`longtable` 环境中可以设置跨页表格在每一页的重复表头和表尾，按照使用顺序，包括以下四个命令：

- `\endfirsthead`: `\begin{longtable}` 和 `\endfirsthead` 之间的内容只会出现在表格第一页的表头部分；
- `\endhead`: `\endfirsthead` 和 `\endhead` 之间的内容将会出现在表格除第一页之外的表头部分；
- `\endfoot`: `\endhead` 和 `\endfoot` 之间的内容将会出现在除表格最后一页之外的表尾部分；
- `\endlastfoot`: `\endfoot` 和 `\endlastfoot` 之间的内容只会出现在表格最后一页的表尾部分。

以上四个命令需要放置在 `longtable` 环境的开始处。

【例】 调用 `longtable` 宏包及环境创建跨页表格：

```

1 \usepackage{longtable}
2 \usepackage{multirow}
3 \begin{document}
4
5 \begin{longtable}[c]{cccc}

```

```

6  % 创建表格第一页的表头部分
7  \caption{Title of a table} \\
8  \hline
9  Column1 & Column2 & Column3 & Column4 \\
10 \hline
11 \endfirsthead
12 % 创建表格除第一页之外的表头部分
13 \caption{Title of a table - Continued} \\
14 \hline
15 Column1 & Column2 & Column3 & Column4 \\
16 \hline
17 \endhead
18 % 创建表格除最后一页之外的表尾部分
19 \hline
20 \endfoot
21 % 创建表格最后一页的表尾部分
22 \multicolumn{4}{c}{\textbf{End of table.}} \\
23 \hline
24 \endlastfoot
25 % 表格内容
26 A1 & A2 & A3 & A4 \\
27 B1 & B2 & B3 & B4 \\
28 C1 & C2 & C3 & C4 \\
29 A1 & A2 & A3 & A4 \\
30 B1 & B2 & B3 & B4 \\
31 C1 & C2 & C3 & C4 \\
32 ... % 省略中间部分
33 A1 & A2 & A3 & A4 \\
34 B1 & B2 & B3 & B4 \\
35 C1 & C2 & C3 & C4 \\
36 \hline
37 \end{longtable}

```

5.8 旋转表格

当表格列数太多时，横向表格的展现效果较差，这时需要将表格旋转 90 度，以纵向表格的形式展现。在 LaTeX 中可以通过调用 *rotating* 宏包，并使用 *sidewaystable* 环境取代 *table* 环境、嵌套 *tabular* 环境创建纵向表格（表格逆时针旋转 90 度）。

【例】调用 rotating 宏包及 sidewaystable 环境创建纵向表格：

```

1  \usepackage{rotating}
2  \usepackage{booktabs}
3  \begin{document}

4

5  \begin{sidewaystable}[h]
6  \centering
7      \caption{Title of a table.}
8      \label{first_label}
9      \begin{tabular}{cccc}
10         \toprule
11         Column1 & Column2 & Column3 & Column4\\
12         \midrule
13         A1 & A2 & A3 & A4\\
14         B1 & B2 & B3 & B4\\
15         C1 & C2 & C3 & C4\\
16         \bottomrule
17     \end{tabular}
18 \end{sidewaystable}
```

5.9 导入表格

一个表格有时候可能涉及大量数据，这时在 LaTeX 中手动输入数据并创建表格显然不够灵活，对此，可以通过导入 csv 文件数据的方式创建表格。LaTeX 提供了 *csvsimple*、*pgfplotstable*、*csvtools* 等宏包可以帮助用户实现基于 csv 文件快速导入表格，其中，使用 *csvsimple* 宏包及其命令是一种比较常用的方式，我们下面将对其展开介绍。

5.9.1 快速创建表格

首先在导言区使用 \usepackage{csvsimple} 语句声明对 *csvsimple* 宏包的调用，然后在文档主体内容中使用 \csvautotabular{csv 文件名或文件路径} 命令即可导入 csv 文件从而快速创建表格。作为导入数据的 csv 文件，既可以预先放在指定目录下，也可以在 *filecontents* 环境中创建，示例语句如下：

```

1  \begin{filecontents}{dataimport.csv}
```

```

2   COLUMNa,COLUMNb,COLUMNc,COLUMNd
3   1.1,2.2,3.3,4.4
4   11.1,22.2,33.3,44.4
5   1.111,2.222,3.333,4.444
6   \end{filecontents}

```

通过上述语句，创建了一个名为“dataimport.csv”的 csv 文件。

【例】 使用 filecontents 环境创建一个名为“dataimport.csv”的 csv 文件，并调用 csvsimple 宏包及\csvautotabular 命令快速根据 csv 文件创建表格：

```

1   \begin{filecontents}{dataimport.csv}
2   COLUMNa,COLUMNb,COLUMNc,COLUMNd
3   1.1,2.2,3.3,4.4
4   11.1,22.2,33.3,44.4
5   1.111,2.222,3.333,4.444
6   \end{filecontents}

7
8   \documentclass{article}
9   \usepackage{csvsimple}
10  \begin{document}
11  \begin{table}
12  \centering
13  \caption{A table imported from csv file}
14  \label{labeloftable1}
15  \csvautotabular{dataimport.csv}
16  \end{table}
17  \end{document}

```

5.9.2 创建三线表

也可以结合 booktabs 宏包，使用\csvautobooktabular 命令自动读取 csv 文件并创建更专业的三线表格。

【例】 使用 filecontents 环境创建一个名为“dataimport.csv”的 csv 文件，并调用 csvsimple 和 booktabs 宏包，使用\csvautobooktabular 命令快速根据 csv 文件创建三线表格：

```

1   \begin{filecontents}{dataimport.csv}

```

```

2 COLUMNSa,COLUMNSb,COLUMNSc,COLUMNSd
3 1.1,2.2,3.3,4.4
4 11.1,22.2,33.3,44.4
5 1.111,2.222,3.333,4.444
6 \end{filecontents}

7
8 \documentclass{article}
9 \usepackage{csvsimple}
10 \usepackage{booktabs}
11 \begin{document}
12 \begin{table}
13 \centering
14 \caption{A table imported from csv file}
15 \label{labeloftable1}
16 \csvautobooktabular{dataimport.csv}
17 \end{table}
18 \end{document}

```

5.9.3 设置表格属性

如果想要调整导入的表格样式、表头、指定导入列等属性，可以使用\csvreader[属性设置] {csv 文件名或文件路径}{定义数据列名}{需要导入的数据列名} 命令读取 csv 文件创建表格，并通过设置属性选项、指定需要导入的数据列名从而调整表格属性。在属性设置选项中，主要包括以下属性：

- tabular：定义列类型。列类型个数应与需要导入的列数一致；
- table head：定义表头，包括标题行的顶线、列名、以及底线。由此可以对各列名进行重定义或省略；
- late after line：定义行分隔线。如，单行分隔线设置表示为 late after line=\hline。

【例】使用 csvreader 命令读取 csv 文件创建表格，并将表格列名从“COLUMNSa,COLUMNSb,COLUMNSc,COLUMNSd”重命名为“column1,column2,column3,column4”，并分别设置数据列名“ca,cb,cc,cd”用于指定导入哪些列：

```

1 \begin{filecontents*}{dataimport.csv}
2 COLUMNSa,COLUMNSb,COLUMNSc,COLUMNSd
3 1.1,2.2,3.3,4.4

```

```

4   11.1,22.2,33.3,44.4
5   1.111,2.222,3.333,4.444
6   \end{filecontents}

7
8   \documentclass{article}
9   \usepackage{csvsimple}
10  \begin{document}
11  \begin{table}
12  \centering
13  \caption{A table imported from csv file}
14  \label{labeloftable1}
15  \csvreader[tabular=|l|l|l|l|,
16  table head=\hline column1 & column2 & column3 & column4\\\hline,
17  late after line=\hline] % 表格属性设置
18  {dataimport.csv} % csv文件名
19  {COLUMNa =\ca, COLUMNb =\cb, COLUMNc =\cc, COLUMNd =\cd} % 定义数据列名
20  {\ca & \cb & \cc & \cd} % 需要导入的数据列名
21  \end{table}
22  \end{document}

```

如果需要对导入的表格增加行标签列，那么在设置需要导入的数据列名时增加\thecsvrow 命令即可。

【例】 使用 csvreader 命令读取 csv 文件创建表格，指定导入的数据列为前两列，并使用 \thecsvrow 命令增加行标签列：

```

1  \begin{filecontents}{dataimport.csv}
2  COLUMNa,COLUMNb,COLUMNc,COLUMNd
3  1.1,2.2,3.3,4.4
4  11.1,22.2,33.3,44.4
5  1.111,2.222,3.333,4.444
6  \end{filecontents}

7
8  \documentclass{article}
9  \usepackage{csvsimple}
10 \begin{document}
11 \begin{table}
12 \centering
13 \caption{A table imported from csv file}
14 \label{labeloftable1}
15 \csvreader[tabular=|l|l|l|,

```

```
16 |     table head=\hline & column1 & column2\\hline,
17 |     late after line=\\hline] % 表格属性设置
18 |     {dataimport.csv} % csv文件名
19 |     {COLUMNa =\ca, COLUMNb =\cb, COLUMNc =\cc, COLUMNd =\cd} % 定义数据列名
20 |     {\thecsvrow & \ca & \cb} % 需要导入的数据列名
21 | \end{table}
22 | \end{document}
```


第六章 插入图形

图形是论文中描述实验结果及相关情况的一种重要方式，也是论文中经常使用的一种特殊信息语言。一般来说，在科技论文中，通过正确地使用图形来辅助论文内容的理解与结果的表达，可以更直观地展现论文内容，有助于得出正确结论。尤其在科研的数据分析中，能形直观地反映变量与变量之间的关系，能清楚地表达某一变量的发展趋势等。很多时候图形能帮助读者领会文字难以表达的内容，同样重要的是它往往可以起到减少篇幅、方便读者阅读的作用。许多图形是在做实验时用相应的软件生成，在撰写论文时需要将其插入文档中，*LaTeX* 可以较为方便地插入各种格式的图形。

本章将介绍如何在 *LaTeX* 中插入图形：

- 首先在第 1 节中介绍如何以浮动体的形式插入图片，允许 *LaTeX* 根据内置的算法对图片位置进行调整；
- 第 2 节中介绍以非浮动体形式插入图片的环境命令；
- 第 3 节中介绍在文档中插入图目录、表目录的方式；
- 第 4 节中介绍对图表标题样式的调整方式；
- 第 5 节中介绍如何插入子图，并对子图的横纵向间距进行调整；
- 第 6 节中介绍图片的排列布局方式。

6.1 插入浮动图片

LaTeX 中可以支持插入.*pdf*、*.jpg*、*.jpeg*、*.png*、**.eps* 等常见格式的图片，而对于 *LaTeX* 不支持的图片文件格式，如 SVG 格式的矢量图，则需要先转换再插入。一般而言，读者可以通过截图、MS Visio 等绘图工具、或者 Matlab 等编程工具制作并导出目标图片。

在 LaTeX 中插入图片可以使用 *graphicx* 宏包, 该宏包提供的\includegraphics[参数]{文件名或文件路径} 命令可以用于插入图片, 以及设置参数以调整图片样式, 常用参数包括:

- width: 设置图片宽度;
- height: 设置图片高;
- scale: 设置图片的缩放倍数;
- angle: 设置图片的顺时针旋转角度 (负值表示逆时针旋转) 等。

一般而言, 对于参数 height 和 width, 只需要调整其中一个即可, 另一个参数将根据图片比例进行自动缩放。而如果同时调整了参数 height 和 width (不推荐), 可能会改变图片比例, 导致图片变形。

【例】在导言区使用\usepackage{graphicx} 声明语句, 在主体代码中使用\includegraphics 命令插入图片, 并调整图片样式参数:

```

1 \usepackage{graphicx}
2 \begin{document}
3
4 The following figure shows a beautiful butterfly.
5
6 \includegraphics[width = 0.5\textwidth]{butterfly.JPG} % 插入第一张图片
7
8 \vspace{12pt}
9
10 Rotate the figure by 90 degrees.
11
12 \includegraphics[width = 0.5\textwidth,angle = 90]{butterfly.JPG} % 插入第二张图
片

```

此外, *graphicx* 宏包提供了 *figure* 环境语句, 通过嵌套\includegraphics 命令可以以浮动体的形式插入图片, 从而能够实现自动递增编号、设置位置控制参数、利用\caption 命令创建标题名称等。

【例】 使用 *figure* 环境嵌套\includegraphics 命令插入浮动图片, 并使用\label 命令为图片创建索引标签, 然后在文本内容中使用 \ref 命令引用该图片:

```

1 \usepackage{graphicx}

```

```

2 \begin{document}
3
4 Figure \ref{fig:1} shows a beautiful butterfly.
5
6 \begin{figure}[htbp]
7 \centering
8 \includegraphics[width = 0.8\textwidth]{butterfly.JPG}
9 \caption{A beautiful butterfly.}
10 \label{fig:1}
11 \end{figure}

```

如果想要创建取消编号的标题，则应调用 *caption* 宏包、并使用 \caption* 命令创建标题名称。此时，LaTeX 内置的自动编号计数器将暂停，直到遇到下一个 \caption 命令才会继续递增计数，例如：

```

1 \caption{The first figure.} % 创建标题 “Figure 1: The first figure.”
2 \caption*{The second figure.} % 创建标题 “The second figure.”
3 \caption{The third figure.} % 创建标题 “Figure 2: The third figure.”

```

6.2 插入非浮动图片

通过 *figure* 环境插入图片虽然能够实现自动编号和创建图片标题，但创建结果为浮动图片，图片的显示位置与在代码中的位置未必一致。然而有时我们想要以非浮动体的形式插入图片，使得图片显示位置与代码中的位置一致，同时能够实现自动编号和创建标题，要实现这一效果，我们可以使用 *minipage* 环境或 *center* 环境替代 *figure* 环境插入图片，同时使用 *caption* 宏包提供的 \captionof{figure}{图片标题名称} 命令创建图片标题。

使用 *minipage* 环境插入图片的方式与 *figure* 环境类似，不同之处主要在于使用 *minipage* 环境插入的图片与上下文中的文本内容紧挨着，为了避免这种情况，*minipage* 环境前后可以使用 \vspace{纵向距离} 调整图片与文本的纵向空间距离。

【例】 使用 *minipage* 环境语句取代 *figure* 环境语句插入非浮动图片，使用 *captionof* 命令创建图片标题，并使用 *vspace* 命令调整图片与文本的纵向距离：

```

1 \usepackage{graphicx}
2 \usepackage{caption}

```

```
3 \begin{document}
4
5 Figure \ref{fig:1} shows a beautiful butterfly.
6
7 \vspace{12pt}
8
9 \begin{minipage}{\linewidth}
10 \centering
11 \includegraphics[width = 0.6\linewidth]{butterfly.JPG}
12 \captionof{figure}{A beautiful butterfly.}
13 \label{fig:1}
14 \end{minipage}
15
16 \vspace{12pt}
17
18 According to the picture, we know ...
19
20 \end{document}
```

使用 center 环境语句插入图片时可以自动将图片居中放置。

【例】使用 center 环境语句取代 figure 环境语句插入非浮动图片，并使用 captionof 命令创建图片标题：

```
1 \usepackage{graphicx}
2 \usepackage{caption}
3 \begin{document}
4
5 Figure \ref{fig:1} shows a beautiful butterfly.
6
7 \begin{center}
8 \includegraphics[width = 0.6\linewidth]{butterfly.JPG}
9 \captionof{figure}{A beautiful butterfly.}
10 \label{fig:1}
11 \end{center}
12
13 According to the picture, we know ...
14
15 \end{document}
```

6.3 插入图表目录

插入图目录或表目录的命令语句分别为`\listoffigures` 和`\listoftables`，可以罗列`\caption` 命令创建的图表标题名称，但对于使用`\caption*` 命令创建的无编号的标题名称而言，则不会出现在目录中。

在一份专业文档中，目录总是和正文内容在不同页显示，为此可以使用`\newpage` 命令进行分页；此外目录页中一般无页码编号，为此可以使用`\thispagestyle{empty}` 取消当页页码设置，并在正文页之前使用`\pagenumbering{页码样式}` 命令表示重新从 1 开始设置页码、同时设置页码样式。

默认的图目录名和表目录名分别是“List of Figures”、“List of Tables”，读者可以在导言区通过使用`\renewcommand{\listfigurename}{新图目录名}` 命令修改图目录名、使用`\renewcommand{\listtablename}{新表目录名}` 命令修改表目录名。

【例】使用`listoffigures` 命令和`listoftables` 命令创建图目录和表目录，使用`renewcommand` 命令修改图目录名和表目录名，并使用`newpage`、`thispagestyle` 和`pagenumbering` 命令对页码进行相应调整：

```
1 \usepackage{graphicx}
2
3 \renewcommand{\listfigurename}{Figures}
4 \renewcommand{\listtablename}{Tables}
5
6 \begin{document}
7
8 \thispagestyle{empty} % 取消页码编号
9
10 \listoffigures
11 \listoftables
12
13 \newpage % 插入新页
14 \pagenumbering{arabic} % 设置页码样式为小写的阿拉伯数字
15
16 Here are three created tables...
17 \begin{table}[h!]
18 % ...
19 \caption{The first table.}
20 \caption{The second table.}
21 \caption{The third table.}
22 \end{table}
23
```

```

24 Here are three inserted figures...
25 \begin{figure}[h!]
26 % ...
27 \caption{The first figure.}
28 \caption*{The second figure.}
29 \caption{The third figure.}
30 % ...
31 \end{figure}
32
33 \end{document}
```

6.4 定制图表标题样式

使用\caption 命令为浮动图片或者浮动表格创建的标题主要包含四部分： 标题头部、 自动编号、 编号分隔符、 以及 标题名称，如下图所示。其中，对标题名称的设置比较简单，下面分别就标题前三部分的调整方式展开介绍。



图 6.1: 图片标题的四个部分示意图

6.4.1 调整标题头部

默认情况下，图片和表格标题头部分别为“Figure”和“Table”，我们可以分别使用\renewcommand{\figurename}{新的图片标题头部} 和\renewcommand{\tablename}{新的表格标题头部} 命令对其进行修改。

6.4.2 调整编号

创建浮动表格或者浮动图片时，LaTeX 会根据内置的计数器对其进行自动递增计数，计数值即为图表标题编号和引用编号，默认为小写的阿拉伯数字。

如果需要取消图表的自动编号，可以使用 caption 宏包提供的\captionsetup[浮动体类型]{labelformat=empty} 命令，其中浮动体类型可以为 figure、subfigure、table、或 subtable，分别表示图片、子图、表格、子表。使用该命令后，指定浮动体类型下的所有浮动体将取消自动编号，但其标题和编号仍会显示在图表目录中。

【例】使用 caption* 命令取消部分表格的自动编号，同时使用 captionsetup 命令取消所有图片的自动编号：

```
1 \usepackage{graphicx}
2 \usepackage{caption}
3 \begin{document}

4
5 \captionsetup[figure]{labelformat=empty} % 取消所有图片的自动编号
6
7 Here are three created tables...
8 \begin{table}[h!]
9 %
10 \caption{The first table.}
11 \caption*{The second table.} % 取消该标题的自动编号
12 \caption{The third table.}
13 \end{table}
14
15 Here are three inserted figures...
16 \begin{figure}[h!]
17 %
18 \caption{The first figure.}
19 \caption{The second figure.}
20 \caption{The third figure.}
21 %
22 \end{figure}
23
24 \end{document}
```

如果想要修改图表编号样式，可以在导言区使用\renewcommand{浮动体的自动计数器}{计数器样式} 命令。其中，浮动体的自动计数器名称可以为\thefigure、\thesubfigure、\thetable、或\thesubtable；定义计数器样式可以使用\alph{浮动体类型}、\Alph{浮动体类型}、\Roman{浮动体类型}、\arabic{浮动体类型} 等命令。

【例】使用 renewcommand 命令调整图表标题头部和编号样式：

```
1 \usepackage{graphicx}
2
3 \renewcommand{\figurename}{Fig} % 调整图片头部
4 \renewcommand{\tablename}{Tab} % 调整新表格头部
5 \renewcommand{\thefigure}{\Alph{figure}} % 调整图片编号样式为大写字母
6 \renewcommand{\thetable}{\alph{table}} % 调整表格编号样式为小写字母
7
8 \begin{document}
9
10 Here are three created tables...
11 \begin{table}[h!]
12 % ...
13 \caption{The first table.}
14 \caption{The second table.}
15 \caption{The third table.}
16 \end{table}
17
18 Here are three inserted figures...
19 \begin{figure}[h!]
20 % ...
21 \caption{The first figure.}
22 \caption{The second figure.}
23 \caption{The third figure.}
24 % ...
25 \end{figure}
26
27 \end{document}
```

6.4.3 调整编号分隔符

图表中的编号分隔符默认为英文冒号“：“，如果需要对其进行调整，可以使用 caption 宏包提供的 \captionsetup[浮动体类型]{设置 labelsep 选项} 命令，通过设置不同的 labelsep 选项实现，各选项及其含义如下：

- colon：默认值，即英文冒号“：“；
- none：无编号分隔符；
- period：英文句号“.”；

- space: 一个空格;
- quad: 一个字符“M”大小的空格;
- newline: 换行。

6.5 插入子图

有时候需要将一组图片以子图的方式呈现，达到对比或者互补的效果。在 LaTeX 中，插入子图比较常用的方式是使用 *subcaption* 宏包及其相关命令。

6.5.1 基本介绍

子图一般在 *subfigure* 环境中创建，多个子图环境嵌套在 *figure* 环境中从而形成同一组子图。*subfigure* 环境与 *figure* 环境的使用方式基本类似，可以为每个子图分别创建标题和索引标签，方便说明和引用。

【例】在导言区使用 \usepackage{subcaption} 语句，在代码主体区域使用 *figure* 环境嵌套三个 *subfigure* 环境从而创建三个子图，并为各子图分别创建索引和标题：

```
1  \usepackage{graphicx}
2  \usepackage{subcaption}
3  \begin{document}

4

5  Figure \ref{fig:fig1} contains sub-figure \ref{subfig:subfig1}, sub-figure \ref{
6      subfig:subfig2} and sub-figure \ref{subfig:subfig3}.

7  \begin{figure}[h!]
8      \centering
9          % 插入第一张子图
10         \begin{subfigure}{.3\linewidth}
11             \centering
12             \includegraphics[width=.5\linewidth]{redflower.png}
13             \caption{A red flower.}
14             \label{subfig:subfig1}
15         \end{subfigure}
16          % 插入第二张子图
17         \begin{subfigure}{.3\linewidth}
18             \centering
19             \includegraphics[width=.5\linewidth]{yellowFlower.png}
```

```

20   \caption{A yellow flower.}
21   \label{subfig:subfig2}
22 \end{subfigure}
23 % 插入第三张子图
24 \begin{subfigure}{.3\linewidth}
25   \centering
26   \includegraphics[width=.5\linewidth]{blueFlower.png}
27   \caption{A blue flower.}
28   \label{subfig:subfig3}
29 \end{subfigure}
30 \caption{Three flowers with different colors.}
31 \label{fig:fig1}
32 \end{figure}

```

编译后的图片如图6.2所示。

Figure 1 contains sub-figure 1a, sub-figure 1b and sub-figure 1c.



(a) A red flower.



(b) A yellow flower.



(c) A blue flower.

Figure 1: Three flowers with different colors.

图 6.2: 编译后的插图

在上例中，每个子图都用到了两次宽度设置选项，分别具有不同含义：

- `\begin{subfigure}{.3\linewidth}` 表示将该子图环境的宽度设置为页面宽度的 0.3 倍；
- `\includegraphics[width=.5\linewidth]` 表示将该图片的宽度设置为当前子图环境宽度的 0.5 倍。

6.5.2 调整子图间距

通过调整子图的横向和纵向间距，可以创建更协调美观的图片。

具体而言，存在以下三类命令可用于调整图片的横向间距：

- `\hfill`: 对于位于相同行的子图，通过在相邻的 `subfigure` 环境间使用该命令，可以实现多个子图横向等距分布的效果；（通过调整子图环境的宽度，可以让子图位于相同行）

- `\hspace{横向距离}`: 定制任意长度的横向图片距离。当该值设为负值时，可以产生图片重叠的效果；
- `\quad`、`\qquad` 等：设置不同预设长度的横向图片距离。

【例】使用`\hfill`命令实现子图横向等距分布，以及使用`\hspace{}`命令实现子图重叠效果：

```
1  \usepackage{graphicx}
2  \usepackage{subcaption}
3  \usepackage{amssymb}
4  \usepackage{amsmath}
5  \begin{document}

6
7  % 使用\hfill命令调整子图横向间距

8
9  The horizontal space among Sub-figures in figure \ref{fig:fig1} is controlled by
10   \$\backslash\$textit{hfill}.

11 \begin{figure}[h!]
12 \centering
13 % 插入第一张子图
14 \begin{subfigure}{.3\linewidth}
15   \centering
16   \includegraphics[width=.5\linewidth]{redflower.png}
17   \caption{A red flower.}
18 \end{subfigure}
19 \hfill
20 % 插入第二张子图
21 \begin{subfigure}{.3\linewidth}
22   \centering
23   \includegraphics[width=.5\linewidth]{yellowFlower.png}
24   \caption{A yellow flower.}
25 \end{subfigure}
26 \hfill
27 % 插入第三张子图
28 \begin{subfigure}{.3\linewidth}
29   \centering
30   \includegraphics[width=.5\linewidth]{blueFlower.png}
31   \caption{A blue flower.}
32 \end{subfigure}
33 \caption{Three flowers with different colors.}
```

```
34 \label{fig:fig1}
35 \end{figure}
36
37 % 使用\space{}命令调整子图横向间距
38
39 The horizontal space among Sub-figures in figure \ref{fig:fig2} is controlled by
$\\backslash\$\\textit{space}.
40
41 \begin{figure}[h!]
42 \centering
43 % 插入第一张子图
44 \begin{subfigure}{.3\linewidth}
45 \centering
46 \includegraphics[width=.5\linewidth]{redflower.png}
47 \end{subfigure}
48 \hspace{-5cm}
49 % 插入第二张子图
50 \begin{subfigure}{.3\linewidth}
51 \centering
52 \includegraphics[width=.5\linewidth]{yellowFlower.png}
53 \end{subfigure}
54 \hspace{-5cm}
55 % 插入第三张子图
56 \begin{subfigure}{.3\linewidth}
57 \centering
58 \includegraphics[width=.5\linewidth]{blueFlower.png}
59 \end{subfigure}
60 \caption{Three flowers with different colors.}
61 \label{fig:fig2}
62 \end{figure}
63
64 \end{document}
```

编译后的图片如图6.3所示。

The horizontal space among Sub-figures in figure 1 is controlled by `\hfill`.



(a) A red flower.



(b) A yellow flower.



(c) A blue flower.

Figure 1: Three flowers with different colors.

The horizontal space among Sub-figures in figure 2 is controlled by `\space`.



Figure 2: Three flowers with different colors.

图 6.3: 编译后的插图

如果想让图片实现纵向等距分布，可以使用`\vfill`命令。

【例】 使用`\vfill`命令实现子图纵向等距分布：

```

1 \usepackage{graphicx}
2 \usepackage{subcaption}
3 \begin{document}
4
5 \begin{figure}[h!]
6 \centering
7 % 插入第一张子图
8 \begin{subfigure}{.3\linewidth}
9   \centering
10  \includegraphics[width=.5\linewidth]{redflower.png}
11  \caption{A red flower.}
12 \end{subfigure}
13 \vfill
14 % 插入第二张子图
15 \begin{subfigure}{.3\linewidth}
16   \centering
17  \includegraphics[width=.5\linewidth]{yellowFlower.png}
18  \caption{A yellow flower.}
19 \end{subfigure}
20 \vfill
21 % 插入第三张子图
22 \begin{subfigure}{.3\linewidth}
```

```
23  \centering
24  \includegraphics[width=.5\linewidth]{blueFlower.png}
25  \caption{A blue flower.}
26  \end{subfigure}
27  \caption{Three flowers with different colors.}
28  \label{fig:fig1}
29  \end{figure}
30
31  \end{document}
```

编译后的图片如图6.4所示。



(a) A red flower.



(b) A yellow flower.



(c) A blue flower.

Figure 1: Three flowers with different colors.

图 6.4: 编译后的插图

6.6 图片排列与布局

通过调整图片排列与布局方式可以创建更专业、更规范的文档。

6.6.1 多图并排

在论文写作中，有时需要将多个图片放在同一行进行排列以便于比较。在 figure 环境中，使用 minipage 环境即可实现图片并排显示，并连续编号。

【例】使用 figure 环境嵌套 minipage 环境实现多图并排显示：

```
1 \usepackage{graphicx}
2
3 The two figures are displayed side by side.
4
5 \begin{figure}[htbp]
6   \centering
7   \begin{minipage}[t]{0.48\textwidth}
8     \centering
9     \includegraphics[width=6cm]{graphics/butterfly.jpg}
10    \caption{Butterfly-1}
11    \end{minipage}
12
13   \begin{minipage}[t]{0.48\textwidth}
14     \centering
15     \includegraphics[width=6cm]{graphics/butterfly.jpg}
16     \caption{Butterfly-2}
17     \end{minipage}
18   \end{figure}
```

编译后的图片如图6.5所示。

The two figures are displayed side by side.



Figure 1: Butterfly-1

Figure 2: Butterfly-2

图 6.5: 编译后的插图

第七章 图形绘制

科技绘图是论文中不可或缺的重要部分，信息丰富的图形一方面可以达到很好的视觉效果。另一方面，图形辅以文字说明也能让读者更为直观地理解一些复杂的问题、过程以及结果。当论文中需要绘制一些图形时，制作这些图形所需要花费的时间就成为影响科研进度的一个重要因素了。在日常科研工作中，通过阅读大量文献，我们或许也会发现这样一种现象：在高质量期刊上发表学术论文，除了对论文本身的贡献和质量有严格的把控外，通常还需要一些高质量的配图用于直观阐述一些复杂原理或者实验结果，同时用于提高读者的阅读体验。

因此，我们应该对论文中的图形绘制引起足够的重视。如果需要，花费足够的时间和精力是值得的，因为这既能加深自己对研究的认识和理解，也能提高审稿人和潜在读者的阅读体验。一般而言，在科技论文中，制作图形的过程可以大致概括为：

- 确定绘图内容
- 设计图形雏形
- 完善图形细节
- 根据内容适当调整图形

7.1 基本介绍

TikZ 宏包是在 *LaTeX* 中创建图形元素的最复杂和最强大的工具。在本节中，我们将通过一些简单的示例来介绍如何在 *tikzpicture* 环境中创建基本的图形元素，如：线、点、曲线、圆、矩形等。

7.1.1 使用 *tikzpicture* 环境创建图形元素

首先，我们需要通过\usepackage{tikz} 命令调用 *TikZ* 宏包。在绘制图形之前，需要声明 *tikzpicture* 环境。在此我们先给出两个用 *TikZ* 绘图的例子，其后再进一步详细介绍具体的绘图命令。

【例】使用 tikzpicture 环境制作一个简单的图形：

```

1 \usepackage{tikz}
2 \begin{document}
3
4 \begin{tikzpicture}
5
6 \draw[red,fill=red] (0,0) .. controls (0,0.75) and (-1.5,1.00) .. (-1.5,2) arc
7   (180:0:0.75) -- cycle;
8 \draw[red,fill=red] (0,0) .. controls (0,0.75) and ( 1.5,1.00) .. ( 1.5,2) arc
9   (0:180:0.75) -- cycle;
10
11 \end{tikzpicture}
```

编译后的绘制图形如图7.1所示。



图 7.1: 绘制后的图形

【例】使用 tikz 宏包中的 tikzpicture 环境创建一个张量网络图：

```

1 \documentclass[border=0.3cm, 11pt]{standalone}
2 \usepackage{tikz}
3 \usepackage{amsmath, amssymb, amsfonts}
4 \usepackage{color}
5
6 \begin{document}
7 \begin{tikzpicture}
8
9 \node[circle, line width = 0.4mm, draw = black, fill = red!45, inner sep = 0pt,
10   minimum size = 0.4cm] (w) at (0, 0) {};
11 \node at (0, 0.5) {\small{\$\\boldsymbol{W}\\$}};
12
13 \node[circle, line width = 0.4mm, draw = black, fill = red!45, inner sep = 0pt,
14   minimum size = 0.4cm] (g) at (1.5, 0) {};
15 \node at (1.5, 0.5) {\small{\$\\boldsymbol{G}\\$}};
```

```

14
15 \node [circle, line width = 0.4mm, draw = black, fill = red!45, inner sep = 0pt,
16   minimum size = 0.4cm] (v) at (3, 0) {};
17 \node at (3, 0.5) {\small{\boldsymbol{V}}};
18
19 \path [draw, line width = 0.4mm, -] (w) edge (g);
20 \node at (0.75, 0.25) {\small{R}};
21 \path [draw, line width = 0.4mm, -] (g) edge (v);
22 \node at (2.25, 0.25) {\small{K}};
23
24 \draw [line width = 0.4mm] (w) -- (0, -0.8);
25 \node at (-0.25, -0.4) {\small{N}};
26 \draw [line width = 0.4mm] (g) -- (1.5, -0.8);
27 \node at (1.5-0.25, -0.4) {\small{N}};
28 \draw [line width = 0.4mm] (v) -- (3, -0.8);
29 \node at (3-0.25, -0.4) {\small{d}};
30
31 \end{tikzpicture}
\end{document}

```

编译后的绘制图形如图7.2所示。

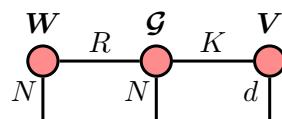


图 7.2: 绘制后的图形

7.1.2 绘制直线

我们在这两个示例中可以感受到 TikZ 功能的强大之处。但是，这些复杂的图形都是由最基本的点、线和面所构成。在本小节中，我们将从绘制一条直线开始，入门这个强大的 LaTeX 绘图工具。首先，画一条直线需要给出起始点坐标和终止点坐标，我们可以简单地通过如下代码：

```

1 \begin{tikzpicture}
2   \draw (x1,y1) -- (x2,y2); % 这里(x1,y1)和(x2,y2)在编译时均需替换成具体坐标数
3   值。
\end{tikzpicture}

```

来实现绘制一条从 (x_1, y_1) 到 (x_2, y_2) 的直线的功能。值得注意的是，在默认情况下，坐标系均以 cm 为单位。

【例】 尝试绘制一条直线：

```

1 \begin{tikzpicture}
2   \draw (-2,0) -- (2,0);
3 \end{tikzpicture}
```

进一步地，我们可以通过设定一系列的坐标点，来实现多条线段的连续绘制。

【例】 多条线段连续绘制：

```

1 \begin{tikzpicture}
2   \draw (-2,0) -- (2,0) -- (2,2) -- (-2,2) -- (-2,0);
3 \end{tikzpicture}
```

也可以通过增加多行命令，实现多段线条的分开绘制。

【例】 多段线条分开绘制：

```

1 \begin{tikzpicture}
2   \draw (-2,0) -- (2,0) -- (2,2) -- (-2,2) -- (-2,0);
3   \draw (0,4) -- (0,-2);
4   \draw (3,-2) -- (3,4) -- (7,4) -- (7,-2) -- (3,-2);
5   \draw (4,3) -- (6,3); \draw (4,1) -- (6,1); \draw (4,-1) -- (6,-1);
6   \draw (5,3) -- (5,-1); \draw (5.75,0.25) -- (6.25,-0.25);
7 \end{tikzpicture}
```

编译后的绘制图形如图7.3所示。

值得注意的是，在 tikzpicture 环境中，像 Matlab 语言一样，我们需要采用 ; 符号来标记一个指令的结束。这样的指令结束标记让我们不但可以在多行完成一条指令，同时也可以在一行内实现多条指令。

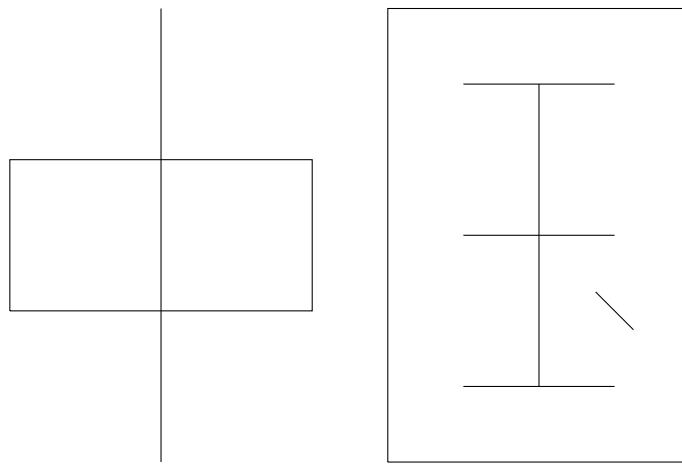


图 7.3: 绘制后的图形

7.1.3 图形缩放

在上小节中，我们绘制图形需要给出精确的坐标点。但是在绘制好之后，如果需要调整图形大小，我们可以采用 scale 的方式对图形进行缩放。

【例】整体缩放：

```
1 \begin{tikzpicture}[scale=0.5]
2     % 绘制内容
3 \end{tikzpicture}
```

【例】横向缩放：

```
1 \begin{tikzpicture}[xscale=1.5]
2     % 绘制内容
3 \end{tikzpicture}
```

【例】分别调整横向、纵向缩放：

```
1 \begin{tikzpicture}[xscale=1.5, yscale = 2]
2     % 绘制内容
3 \end{tikzpicture}
```

7.1.4 绘制箭头

在绘制直线的基础上，我们往往需要通过绘制箭头来指向性地表达意图。箭头的绘制只需要在直线绘制的基础上，增加 [option] 进行声明即可。

【例】 绘制箭头：

```

1 \begin{tikzpicture}
2
3   \draw [->] (0,0) -- (2,0);
4   \draw [<-] (0, -0.5) -- (2,-0.5);
5   \draw [|->] (0,-1) -- (2,-1);
6
7 \end{tikzpicture}
```

编译后的绘制图形如图7.4所示。

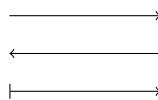


图 7.4: 绘制后的图形

【例】 利用绘制箭头的例子以及多条线段连续绘制的例子，用一行命令绘制一个直角坐标系：

```

1 \begin{tikzpicture}
2
3   \draw [<->] (0,2) -- (0,0) -- (3,0);
4
5 \end{tikzpicture}
```

编译后的绘制图形如图7.5所示。



图 7.5: 绘制后的图形

7.1.5 调整线条粗细

采用\draw 命令时，增加的 [option] 声明也可以用来调整线条的粗细。

【例】绘制不同粗细的线条：

```

1 \begin{tikzpicture}
2
3   \draw [ultra thick] (0,1) -- (2,1);
4   \draw [thick] (0,0.5) -- (2,0.5);
5   \draw [thin] (0,0) -- (2,0);
6
7 \end{tikzpicture}
```

编译后的绘制图形如图7.6所示。

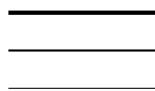


图 7.6: 绘制后的图形

其中，线条的粗细可以通过不同的指令来控制，从细到粗分别可调用：*ultra thin*, *very thin*, *thin*, *semithick*, *thick*, *very thick*, *ultra thick*。

【例】绘制上述七种不同粗细的线条：

```

1 \begin{tikzpicture}
2
3   \draw [ultra thin] (0,0) -- (2,0);
4   \draw [very thin] (0,0.5) -- (2,0.5);
5   \draw [thin] (0,1) -- (2,1);
6   \draw [semithick] (0,1.5) -- (2,1.5);
7   \draw [thick] (0,2) -- (2,2);
8   \draw [very thick] (0,2.5) -- (2,2.5);
9   \draw [ultra thick] (0,3) -- (2,3);
10
11 \end{tikzpicture}
```

编译后的绘制图形如图7.7所示。

除此之外，我们也可以自行定义线条的粗细，如 [line width=5]、[line width=0.2cm]。值得注意的是，当我们直接声明数值而不声明单位时，其默认单位均为 pt。

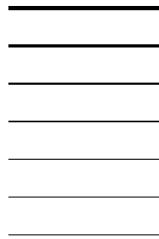


图 7.7: 绘制后的图形

【例】 使用 `line width` 参数绘制不同粗细线条：

```

1 \begin{tikzpicture}
2   \draw [line width=3] (0,0) -- (2,0);
3   \draw [line width=0.2cm] (0,0.5) -- (2,0.5);
4 \end{tikzpicture}
```

编译后的绘制图形如图7.8所示。



图 7.8: 绘制后的图形

7.1.6 虚线

我们也可以在 `[option]` 声明中增加对于线条形状的定义。如虚线 `dashed` 和点线 `dotted`。

【例】 绘制虚线：

```

1 \begin{tikzpicture}
2
3   \draw [dashed, ultra thick] (0,1) -- (2,1); %我们可以通过组合多种option来声明
4   %线条的多种特征。
5   \draw [dashed] (0, 0.5) -- (2,0.5);
6   \draw [dotted] (0,0) -- (2,0);
7
8 \end{tikzpicture}
```

编译后的绘制图形如图7.9所示。



图 7.9: 绘制后的图形

7.1.7 颜色

我们也可以在 [option] 声明中增加对于线条颜色的定义。如红色 *red*、绿色 *green*、蓝色 *blue* 等等。

【例】绘制不同颜色的直线：

```

1 \begin{tikzpicture}

2
3   \draw [red] (0,1) -- (2,1);
4   \draw [green] (0, 0.5) -- (2,0.5);
5   \draw [blue] (0,0) -- (2,0);
6
7 \end{tikzpicture}

```

编译后的绘制图形如图7.10所示。



图 7.10: 绘制后的图形

7.2 节点介绍

节点是 TikZ 中的一个常用功能。在绘制节点时，通常需要声明其位置和形状，部分节点可以在其中添加文字，同时也为节点赋予一个名称，用于后续参考。在本节中，我们将详细介绍节点的相关功能及其应用。

7.2.1 节点基本介绍

【例】绘制节点，方法一：

```

1 \begin{tikzpicture}
2   \path (0,2) node [shape=circle,draw=blue!50,fill=blue!20,thick] {}

```

```

3   (0,1) node [shape=circle,draw=blue!50,fill=blue!20,thick] {$C$}
4   (0,0) node [shape=circle,draw=blue!50,fill=blue!20,thick] {}
5   (1,1) node [shape=rectangle,draw=black!50,fill=black!20] {}
6   (-1,1) node [shape=rectangle,draw=black!50,fill=black!20] {};
7 \end{tikzpicture}

```

等价于

【例】 绘制节点，方法二：

```

1 \begin{tikzpicture}
2   \node [shape=circle,draw=blue!50,fill=blue!20,thick] at (0,2) {};
3   \node [shape=circle,draw=blue!50,fill=blue!20,thick] at (0,1) {$C$};
4   \node [shape=circle,draw=blue!50,fill=blue!20,thick] at (0,0) {};
5   \node [shape=rectangle,draw=black!50,fill=black!20] at (1,1) {};
6   \node [shape=rectangle,draw=black!50,fill=black!20] at (-1,1) {};
7 \end{tikzpicture}

```

编译后的绘制图形如图7.11所示。

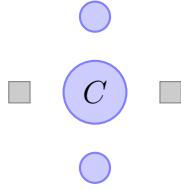


图 7.11：绘制后的图形

这里需要注意的是，在 [shape=circle,draw=blue!50,fill=blue!20] 中，shape 指令声明节点形状，draw 指令声明是否显现该形状的边框，并用 draw= 来声明边框颜色，fill 命令指明该节点是否要填充，并用 fill= 来声明填充颜色。若要在节点中显示文字，需在后面 {} 中填写对应的文字即可。

7.2.2 节点样式

当某一种形状及颜色的节点需要在不同位置多次出现时，上述代码显得不够优美。我们可以通过一段代码提前声明该节点的样式，并反复调用这段代码即可。

【例】 绘制节点，方法三：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]

```

```

2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]
3 \begin{tikzpicture}
4
5   \path (0,2) node [aaa] {}
6     (0,1) node [aaa] {$C$}
7     (0,0) node [aaa] {}
8     (1,1) node [bbb] {}
9     (-1,1) node [bbb] {};
10
11 \end{tikzpicture}

```

编译上述代码，得到图形与图7.11所示相同。

7.2.3 节点命名

为了将节点相互连接起来，我们需要指明连接哪两个节点。因此，每个节点需要我们声明一个名称。声明名称有两种方式，一种是采用 `name=` 的方式，另一种是在`\node` 后用括号声明。

【例】 绘制节点，并声明节点名称，方法一：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]
2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]
3 \begin{tikzpicture}
4
5   \path (0,2) node [aaa,name=a1] {}
6     (0,1) node [aaa,name=a2] {$C$}
7     (0,0) node [aaa,name=a3] {}
8     (1,1) node [bbb,name=b1] {}
9     (-1,1) node [bbb,name=b2] {};
10
11 \end{tikzpicture}

```

【例】 绘制节点，并声明节点名称，方法二：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]
2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]
3 \begin{tikzpicture}
4

```

```

5   \node (a1) [aaa] at (0,2) {};
6   \node (a2) [aaa] at (0,1) {$C$};
7   \node (a3) [aaa] at (0,0) {};
8   \node (b1) [bbb] at (1,1) {};
9   \node (b2) [bbb] at (-1,1) {};

10
11 \end{tikzpicture}

```

7.2.4 基于相对位置绘制节点

给每个节点命名后，我们便可以通过 above of (上)、below of (下)、left of (左)、right of (右) 等命令来声明新节点与某个节点的相对位置来绘制图形。

【例】利用相对位置绘制节点：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]
2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]
3 \begin{tikzpicture}
4
5   \node (a1) [aaa] {};
6   \node (a2) [aaa] [below of=a1] {$C$};
7   \node (a3) [aaa] [below of=a2] {};
8   \node (b1) [bbb] [right of=a2] {};
9   \node (b2) [bbb] [left of=a2] {};

10
11 \end{tikzpicture}

```

编译上述代码，得到图形与图7.11所示相同。

7.2.5 连接节点

有了节点名称了，我们就可以对节点进行连接。我们拿连接 A 与 B 节点为例，在连接时，我们通常需要声明 A 节点的哪个位置与 B 节点的哪个位置连接。位置声明通常采用 east (右)、west (左)、north (上)、south (下)、center (中心) 等命令。

【例】利用相对位置连接节点：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]
2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]

```

```

3   \begin{tikzpicture}

4

5     \node (a1) [aaa] {$a_1$};
6     \node (a2) [aaa] [below of=a1] {$C$};
7     \node (a3) [aaa] [below of=a2] {$a_3$};
8     \node (b1) [bbb] [right of=a2] {$b_1$};
9     \node (b2) [bbb] [left of=a2] {$b_2$};
10    \draw [->] (a2.west) -- (b2.east);
11    \draw [->] (a2.east) -- (b1.west);
12    \draw [->] (a2.north) -- (a1.south);
13    \draw [->] (a2.south) -- (a3.north);

14
15 \end{tikzpicture}

```

该代码等价于

【例】利用相对位置连接节点，不声明节点的连接位置：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]
2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]
3 \begin{tikzpicture}

4
5     \node (a1) [aaa] {$a_1$};
6     \node (a2) [aaa] [below of=a1] {$C$};
7     \node (a3) [aaa] [below of=a2] {$a_3$};
8     \node (b1) [bbb] [right of=a2] {$b_1$};
9     \node (b2) [bbb] [left of=a2] {$b_2$};
10    \draw [->] (a2) -- (b2);
11    \draw [->] (a2) -- (b1);
12    \draw [->] (a2) -- (a1);
13    \draw [->] (a2) -- (a3);

14
15 \end{tikzpicture}

```

编译后的绘制图形如图7.12所示。

【例】利用 edge 命令连接节点，方法一：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]
2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]
3 \begin{tikzpicture}

```

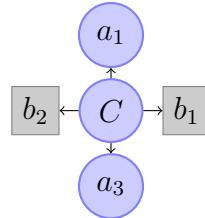


图 7.12: 绘制后的图形

```

4
5      \node (a1) [aaa] {$a_1$};
6      \node (a2) [aaa] [below of=a1] {$C$} edge [->] (a1);
7      \node (a3) [aaa] [below of=a2] {$a_3$} edge [<-] (a2);
8      \node (b1) [bbb] [right of=a2] {$b_1$} edge [<-] (a2);
9      \node (b2) [bbb] [left of=a2] {$b_2$} edge [<-] (a2);
10
11  \end{tikzpicture}

```

等价于

【例】用 edge 命令连接节点，方法二：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]
2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]
3 \begin{tikzpicture}
4
5     \path (0,2) node [aaa,name=a1] {$a_1$}
6     (0,1) node [aaa,name=a2] {$C$} edge [->] (a1)
7     (0,0) node [aaa,name=a3] {$a_3$} edge [<-] (a2)
8     (1,1) node [bbb,name=b1] {$b_1$} edge [<-] (a2)
9     (-1,1) node [bbb,name=b2] {$b_2$} edge [<-] (a2);
10
11 \end{tikzpicture}

```

编译上述代码，得到图形与图7.12所示相同。

【例】声明每个节点的连接位置，将周围节点边缘连接至中间节点的中心：

```

1 \tikzstyle{aaa}=[circle,draw=blue!50,fill=blue!20,thick]
2 \tikzstyle{bbb}=[rectangle,draw=black!50,fill=black!20]
3 \begin{tikzpicture}

```

```

4
5   \node (a1) [aaa] {$a_1$};
6   \node (a2) [aaa] [below of=a1] {};
7   \node (a3) [aaa] [below of=a2] {$a_3$};
8   \node (b1) [bbb] [right of=a2] {$b_1$};
9   \node (b2) [bbb] [left of=a2] {$b_2$};
10  \draw [->] (a2.center) -- (b2.east);
11  \draw [->] (a2.center) -- (b1.west);
12  \draw [->] (a2.center) -- (a1.south);
13  \draw [->] (a2.center) -- (a3.north);
14
15 \end{tikzpicture}

```

编译上述代码，得到图形如图7.13所示。

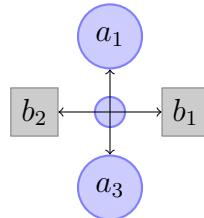


图 7.13: 绘制后的图形

7.3 高级功能

在本节中，我们将介绍除直线以外的复杂功能，如：非规则曲线、复杂函数绘图、区域填充、填写标签等等。

7.3.1 矩形、圆形、曲线

我们可以通过`\draw (x,y) rectangle (w,h);`的方式绘制一个矩形，其左下角坐标位于点 (x,y) 处，长度为 w ，高度为 h 。类似地，我们也可以通过`\draw (x,y) circle [radius=r];`的方式绘制一个圆形，其圆心落在点 (x,y) 处，半径为 r 。除此之外，我们可以通过`\draw (x,y) arc [radius=r, start angle=a1, end angle=a2];`的方式绘制一条弧线，它从点 (x,y) 处开始绘制，该弧线曲率半径为 r ，其起始角度为所对应曲率圆的1处，终止角度为所对应曲率圆的2处。

【例】按上述介绍，绘制矩形、圆形以曲线：

```

1 \begin{tikzpicture}
2
3   \draw [red] (0,0) rectangle (1.5,1);
4   \draw [blue, ultra thick] (3,0.5) circle [radius=0.5];
5   \draw [green] (6,0) arc [radius=1.5, start angle=45, end angle= 100];
6
7 \end{tikzpicture}

```

编译上述代码，得到图形如图7.14所示。

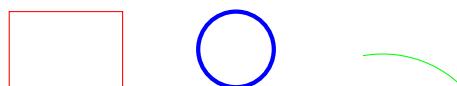


图 7.14: 绘制后的图形

7.3.2 平滑过渡曲线

在绘图时，一种不突兀地连接两条直线的方式是采用平滑过渡圆角/曲线。在本节中，我们将介绍两种平滑过渡曲线的绘制方法：绘制带圆角的曲线和绘制过渡曲线。

【例】绘制带圆角的坐标系：

```

1 \begin{tikzpicture}
2
3   \draw [<->, rounded corners, thick, purple] (0,2) -- (0,0) -- (3,0);
4
5 \end{tikzpicture}

```

编译上述代码，得到图形如图7.15所示。



图 7.15: 绘制后的图形

【例】绘制过渡曲线：

```

1 \begin{tikzpicture}
2
3   \draw[<-, thick] (0,2) -- (0,0.5);
4   \draw[thick,red] (0,0.5) to [out=270,in=180] (0.5,0);
5   \draw[->, thick] (0.5,0) -- (3,0);
6
7 \end{tikzpicture}

```

编译上述代码，得到图形如图7.16所示。

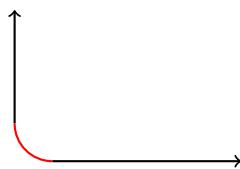


图 7.16: 绘制后的图形

【例】利用多段过渡曲线绘制 S 曲线：

```

1 \begin{tikzpicture}
2
3   \draw [<->,thick, blue] (0,0) to [out=90,in=180] (1,1) to [out=0,in=180]
4     (3,0) to [out=0,in=-90] (4,1) ;
5
6 \end{tikzpicture}

```

编译上述代码，得到图形如图7.17所示。



图 7.17: 绘制后的图形

7.3.3 根据函数绘制曲线

TikZ 宏包的强大之处在于，它还提供了可供绘制函数的数学引擎。在此我们先给出一个示例，再详细讲解如何利用该宏包绘制函数所对应的曲线。

【例】利用函数绘制正弦曲线：

```

1 \begin{tikzpicture}[xscale=0.01,yscale=1]
2
3   \draw [->] (0,1) -- (0,0) -- (370,0);
4   \draw[green, thick, domain=0:360] plot (\x, {sin(\x)}); % 这里需要注意带上{}
5
6 \end{tikzpicture}

```

编译上述代码，得到图形如图7.18所示。

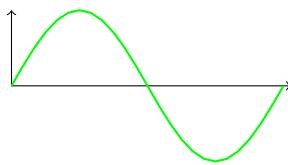


图 7.18: 绘制后的图形

在上述例子中，*domain* 指令声明了横坐标 *x* 的范围。在本示例中，我们利用 *sin* 函数绘制了一段正弦曲线。

除了本示例中的正弦曲线 *sin* 函数，我们还可以调用大量其他函数，在此列举一部分作为示例：

- 阶乘函数: `factorial(\x);`
- 平方根函数: `sqrt(\x);`
- 幂函数: `pow(\x,y);`
- 指数函数: `exp(\x);`
- 对数函数: `ln(\x)`、`log10(\x)`、`log2(\x)`；
- 绝对值函数: `abs(\x);`
- 取余函数: `mod(\x,y)`（即求 被 除后的余数）；
- 圆整函数: `round(\x)`、`floor(\x)`、`ceil(\x)`； 三角函数: `sin(\x)`、`cos(\x)`、`tan(\x)`。;

值得注意的是，在三角函数中，通常默认自变量 以度（°）为单位。若要采用弧度制，则需要将函数分别改写为 `sin(\x r)`、`cos(\x r)`、`tan(\x r)`。除了这部分常用函数之外，我们通常还会使用两个常数 *e* (= 2.718281828) 和 *pi* (= 3.141592654)。

通过组合以上基本函数，我们可以实现更复杂的函数功能。

【例】利用函数绘制正弦曲线：

```

1 \begin{tikzpicture}[yscale=1.5]
2
3   \draw [thick, ->] (0,0) -- (6.5,0);
4   \draw [thick, ->] (0,-1.1) -- (0,1.1);
5   \draw [green,domain=0:2*pi] plot (\x, {(\sin(\x r)* ln(\x+1))/2});
6   \draw [red,domain=0:pi] plot (\x, {\sin(\x r)});
7   \draw [blue, domain=pi:2*pi] plot (\x, {cos(\x r)*exp(\x/exp(2*pi))});
8
9 \end{tikzpicture}

```

编译上述代码，得到图形如图7.19所示。

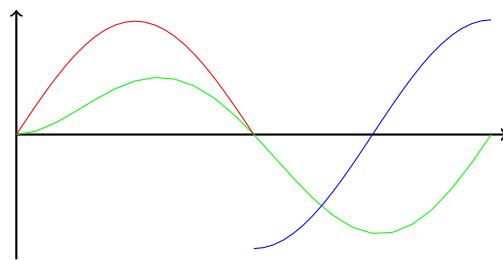


图 7.19: 绘制后的图形

7.3.4 简单图形的区域填充

我们可以在简单图形的基础上进行区域填充。

【例】简单形状的区域填充：

```

1 \begin{tikzpicture}[yscale=1.5]
2
3   \draw [fill=red,ultra thick] (0,0) rectangle (1,1);
4   \draw [fill=red,ultra thick,red] (2,0) rectangle (3,1); % 这里的第二个red声明
5   \draw [blue, fill=blue] (4,0) -- (5,1) -- (4.75,0.15) -- (4,0);
6   \draw [fill] (7,0.5) circle [radius=0.1];
7   \draw [fill=orange] (9,0) rectangle (11,1);
8   \draw [fill=white] (9.25,0.25) rectangle (10,1.5);
9

```

```
10 \end{tikzpicture}
```

编译上述代码，得到图形如图7.20所示。



图 7.20: 绘制后的图形

如上例中的注释所提，我们可以通过声明图形边框线的颜色来对边框线进行个性化更改。若并不希望出现边框线，我们可以采用 path 命令替换\draw 命令。

【例】简单形状的区域填充：

```
1 \begin{tikzpicture} [yscale=1.5]
2
3     \path [fill=red,thick] (0,0) rectangle (1.5,1);
4     \draw [fill=red,thick] (2,0) rectangle (3.5,1);
5
6 \end{tikzpicture}
```

编译上述代码，得到图形如图7.21所示。

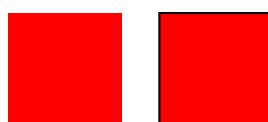


图 7.21: 绘制后的图形

7.3.5 在图形中填写标签

在绘图时，在合适的位置加入适当的文字进行说明，对内容的表达具有很重要的作用。在本节中，我们将通过\node 来实现这一功能。

【例】在直角坐标系中插入文字：

```
1 \begin{tikzpicture} [yscale=1.5]
2
```

```

3   \draw [thick, <->] (0,2) -- (0,0) -- (2,0);
4   \node at (1,1) {good};
5
6 \end{tikzpicture}

```

编译上述代码，得到图形如图7.22所示。

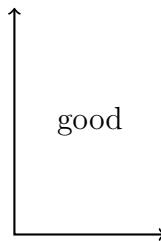


图 7.22: 绘制后的图形

在上例中，我们给出了一个简单的示范，我们将文字”good” 的中心位置固定在坐标 (1,1) 点处。当然，我们也可以通过命令，控制文字与所声明坐标的相对位置，如：在坐标上方、下方、左侧、右侧。

【例】 在 (1,1) 点下方插入文字：

```

1 \begin{tikzpicture}
2
3   \draw [thick, <->] (0,2) -- (0,0) -- (2,0);
4   \draw [fill] (1,1) circle [radius=0.025];
5   \node [below] at (1,1) {good};
6
7 \end{tikzpicture}

```

编译上述代码，得到图形如图7.23所示。

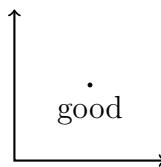


图 7.23: 绘制后的图形

【例】 在 (1,1) 点上方、下方、左侧、右侧插入文字：

```

1 \begin{tikzpicture}

2
3   \draw [thick, <->] (0,2) -- (0,0) -- (2,0);
4   \draw [fill] (1,1) circle [radius=0.025];
5   \node [below] at (1,1) {below};
6   \node [above] at (1,1) {above};
7   \node [left] at (1,1) {left};
8   \node [right] at (1,1) {right};

9
10 \end{tikzpicture}

```

编译上述代码，得到图形如图7.24所示。

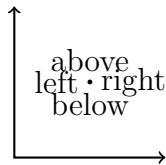


图 7.24: 绘制后的图形

【例】 在 (1,1) 点处插入数学符号 θ :

```

1 \begin{tikzpicture}

2
3   \draw [thick, <->] (0,2) -- (0,0) -- (2,0);
4   \node [below right] at (2,0) {$x$};
5   \node [left] at (0,2) {$y$};
6   \draw[fill] (1,1) circle [radius=.5pt];
7   \node[above right] at (1,1) {$\theta$};

8
9 \end{tikzpicture}

```

编译上述代码，得到图形如图7.25所示。

7.4 复杂模型实战解析

在本节中，我们将给出一些科研论文中的复杂模型供读者们进一步解析学习。

【例】 BCPF:

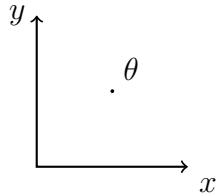


图 7.25: 绘制后的图形

```

1 \documentclass[border=0.1cm]{standalone}
2 \usepackage[utf8]{inputenc}
3
4 \usepackage{tikz}
5 \usepackage{amsfonts}
6 \usepackage{amsmath,amssymb}
7 \usepackage{systeme,mathtools}
8 \usetikzlibrary{positioning,arrows.meta,quotes}
9 \usetikzlibrary{shapes,snakes}
10 \usetikzlibrary{bayesnet}
11 \tikzset{>=latex}
12 \tikzstyle{plate caption} = [caption, node distance=0, inner sep=0pt,
13 below left=5pt and 0pt of #1.south]
14
15 \begin{document}
16 \begin{tikzpicture}
17
18     \node [obs] (x) at (0,0) {\large $\mathbf{x}_{\boldsymbol{i}}$};
19     \node [circle,draw=black,fill=white,inner sep=0pt,minimum size=0.6cm] (u1) at
20         (-1.2,1.6) { $\mathbf{u}_{\boldsymbol{i}_1}^{(1)}$};
21     \node [circle,draw=black,fill=white,inner sep=0pt,minimum size=0.6cm] (u3) at
22         (1.2,1.6) { $\mathbf{u}_{\boldsymbol{i}_d}^{(d)}$};
23     \node [circle,draw=black,fill=white,inner sep=0pt,minimum size=0.65cm] (
24         lambda) at (0,3.0) {\large $\boldsymbol{\lambda}$};
25
26     \node [mark size=1pt,color=black] at (0,1.6) {\pgfuseplotmark{*}};
27     \node [mark size=1pt,color=black] at (-0.2,1.6) {\pgfuseplotmark{*}};
28     \node [mark size=1pt,color=black] at (0.2,1.6) {\pgfuseplotmark{*}};
29
30     \node [text width=0.5cm] (c0) at (0,4) {$\alpha, \beta$};
31     \node [text width=0.5cm] (a0) at (2.5,2.6) {$\alpha, \beta$};
32     \node [circle,draw=black,fill=white,inner sep=0pt,minimum size=0.65cm] (tau_
33         epsilon) at (2.5,1.6) {\large $\boldsymbol{\tau}_{\epsilon}$};

```

```

30
31     \path [draw,->] (u1) edge (x);
32     \path [draw,->] (u3) edge (x);
33     \path [draw,->] (lambda) edge (u1);
34     \path [draw,->] (lambda) edge (u3);

35
36     \path [draw,->] (c0) edge (lambda);
37     \path [draw,->] (tau_epsilon) edge (x);
38     \path [draw,->] (a0) edge (tau_epsilon);
39     \plate [color=red] {part1} {(x)(u1)} {};
40     \plate [color=blue] {part3} {(x)(u3)(part1.north east)} {};

41
42     \node [text width=2cm] at (-0.6,-0.5) {\large $n_{-1}$};
43     \node [text width=2cm] at (2,-0.5) {\large $n_d$};

44
45 \end{tikzpicture}
46 \end{document}

```

编译上述代码，得到图形如图7.26所示。

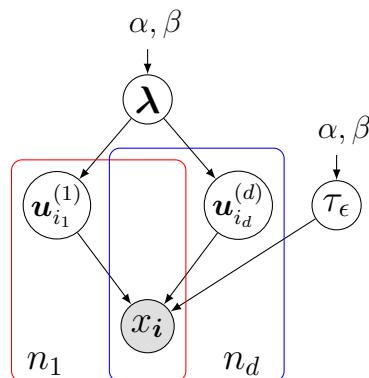


图 7.26: 绘制后的图形

【例】矩阵分解示意图：

```

1 \documentclass[border=0.1cm]{standalone}
2 \usepackage[utf8]{inputenc}
3
4 \usepackage{tikz}
5 \usepackage{amsfonts}
6 \usepackage{amsmath,amssymb}
7 \usepackage{systeme,mathtools}

```

```

8  \usetikzlibrary{positioning,arrows.meta,quotes}
9  \usetikzlibrary{shapes,snakes}
10 \usetikzlibrary{bayesnet}
11 \tikzset{>=latex}

12
13 \begin{document}
14 \begin{tikzpicture}

15
16   \draw [very thick] (0,0) rectangle (3.6/2,2.4/2);
17   \filldraw [fill=green!20!white,draw=green!40!black] (0,0) rectangle
18     (3.6/2,2.4/2);
19   \filldraw [fill=white] (0.4/2,0.4/2) rectangle (0.8/2,0.8/2);
20   \filldraw [fill=white] (2.4/2,0.4/2) rectangle (2.8/2,0.8/2);
21   \filldraw [fill=white] (0.8/2,1.2/2) rectangle (1.2/2,1.6/2);
22   \filldraw [fill=white] (2.0/2,1.6/2) rectangle (2.4/2,2.0/2);
23   \filldraw [fill=white] (0.4/2,2.0/2) rectangle (0.8/2,2.4/2);
24   \filldraw [fill=white] (2.4/2,2.0/2) rectangle (2.8/2,2.4/2);
25   \filldraw [fill=white] (2.8/2,1.2/2) rectangle (3.2/2,2.0/2);
26   \draw [step=0.4/2, very thin, color=gray] (0,0) grid (3.6/2,2.4/2);
27   \draw (1.8/2,-0.3) node {{\color{red}\scriptsize{$Y \in \mathbb{R}^{m \times n}$}}};
28
29   \draw (4.4/2,1.2/2) node {{\color{black}\large{$\approx$}}};
30   \draw [very thick] (5.2/2,0) rectangle (6.0/2,2.4/2);
31   \filldraw [fill=green!20!white,draw=green!40!black] (5.2/2,0) rectangle
32     (6.0/2,2.4/2);
33   \draw [step=0.4/2, very thin, color=gray] (5.2/2,0) grid (6.0/2,2.4/2);
34   \draw (5.6/2,-0.3) node {{\color{black}\scriptsize{$W \in \mathbb{R}^{r \times m}$}}};
35
36   \draw (6.8/2,1.2/2) node {{\color{black}\large{$\times$}}};
37   \draw [very thick] (7.6/2,0.8/2) rectangle (11.2/2,1.6/2);
38   \filldraw [fill=green!20!white,draw=green!40!black] (7.6/2,0.8/2) rectangle
39     (11.2/2,1.6/2);

```

编译上述代码，得到图形如图7.27所示。

$$\begin{array}{c} \text{A 5x5 grid of green squares with white centers, labeled } Y \in \mathbb{R}^{m \times f} \\ \approx \end{array} \begin{array}{c} \text{A vertical column of 5 green squares, labeled } W \in \mathbb{R}^{m \times r} \\ \times \end{array} \begin{array}{c} \text{A horizontal row of 5 green squares, labeled } X^T \in \mathbb{R}^{r \times f} \end{array}$$

图 7.27: 绘制后的图形

更多模型例子可以参考开源项目：

<https://github.com/xinychen/awesome-latex-drawing>

第八章 建立索引及文献引用

科研论文或科技报告中的图片、表格、公式和参考文献往往会被编号，方便读者进行查看。在实际写作过程中，有时图片、表格、公式会不在文本引用位置附近，而参考文献更是一般都总结放在文末结尾处。因此，读者在阅读过程中，为了查看该处引用内容的详细信息，需要翻看全文查找被引用的内容。这个过程一定是非常繁琐低效、并且会影响读者的阅读流畅性。为此，建立索引和文献引用就非常有必要了。建立索引一般指对文档中的图片、表格、公式等进行索引的设置，这个建立是自动编号完成；而文献引用同样是对文中存在引用参考文献的地方自动编号建立索引。因此，建立索引和文献引用的过程较为简单，并且建立索引和引用后的效果非常好，可以实现读者根据索引和文献引用直接跳转至想要查看的内容，能够有效提高读者的阅读效率和流畅性。

本章节主要包括以下部分：公式和图表的索引、创建超链接并调整链接格式、利用 BibTeX 完成参考文献引用以及引用格式的设定。

8.1 公式和图表的索引

在第 4、5、6、7 章中，分别介绍了如何制作或插入公式及图表，在文档中，我们往往需要对公式及图表进行引用以辅助文档的叙述、描述结果以及佐证一些结论。而有时为了排版的美观，所插入的公式或图表不一定直接放在引用位置旁边，因此对公式及图表进行索引就显得尤为重要了。

8.1.1 公式的索引

LaTeX 中，公式的索引分为主要分为两个部分，一部分是给公式添加标签，可以使用`\label{标签名}` 命令。根据第 4 章，可以使用 `equation` 环境插入带标签的公式；另一部分是在文档中进行引用，可以使用`\ref{标签名}` 命令。

【例】用 `label` 及 `ref` 在文中对公式进行索引：

```

1 (\ref{eq1}) is a binary equation
2
3 (\ref{eq2}) is a binary quadratic equation.
4
5 \begin{equation}
6 x+y=2\label{eq1}
7 \end{equation}
8
9 \begin{equation}
10 x^2+y^2=2\label{eq2}
11 \end{equation}

```

8.1.2 图形的索引

根据第 6 章，插入图片需要使用 graphicx 宏包，图形的索引与公式的索引类似。同样分为两部分，一部分是使用 \label{标签名} 命令给添加图形标签，另一部分是使用 \ref{标签名} 在文档中进行引用。

【例】 使用 label 及 ref 在文中对图形进行索引：

```

1 \usepackage{graphicx}
2 \begin{document}
3
4 Figure \ref{butterfly} is a photo of butterfly.
5
6 \begin{figure}
7 \centering
8 \includegraphics[width = 0.8\textwidth]{graphics/butterfly.JPG}
9 \caption{There is a beautiful butterfly.}
10 \label{butterfly}
11 \end{figure}
12
13 \end{document}

```

8.1.3 表格的索引

表格的索引与公式及图形的索引类似，同样分为两部分，一部分是使用 \label{标签名} 命令给添加表格标签。根据第 5 章，可以使用 tabular 和 table 两种环境制作

带标签的表格；另一部分是使用`\ref{标签名}`在文档中进行引用。

【例】 使用 label 及 ref 在文中对图形进行索引：

```

1  Table-\ref{table1} shows the values of some basic functions.

2

3  \begin{table}
4      \centering
5      \caption{The values of some basic functions.}
6      \begin{tabular}{l|cccc}
7          \hline
8          & $x=1$ & $x=2$ & $x=3$ & $x=4$ \\
9          \hline
10         $y=x$ & 1 & 2 & 3 & 4 \\
11         $y=x^2$ & 1 & 4 & 9 & 16 \\
12         $y=x^3$ & 1 & 8 & 27 & 64 \\
13         \hline
14     \end{tabular}
15     \label{table1}\% 索引标签
16 \end{table}

```

8.2 创建超链接

超链接指按内容链接，可以从一个文本内容指向文本其他内容或其他文件、网址等。超链接可以分为文本内链接、网页链接以及本地文件链接。LaTeX 提供了 `hyperref` 宏包，可用于生成超链接。在使用时，只需在前导代码中申明宏包即可，即`\usepackage{hyperref}`。

8.2.1 超链接类型

文本内链接

在篇幅较大的文档中，查阅内容会比较繁琐，因此，往往会在目录中使用超链接来进行文本内容的快速高效浏览。可以使用 `hyperref` 宏包创建文本内超链接。

【例】 使用 label 及 ref 在文中对图形进行索引：

```

1  \documentclass{book}
2  \usepackage{blindtext}

```

```

3  \usepackage{hyperref} %超链接包
4
5  \begin{document}
6
7  \frontmatter
8  \tableofcontents
9  \clearpage
10
11 \addcontentsline{toc}{chapter}{Foreword}
12 {\huge \bf Foreword}
13
14 This is foreword.
15 \clearpage
16
17 \mainmatter
18
19 \chapter{First Chapter}
20
21 This is chapter 1.
22
23
24 \clearpage
25
26 \section{First section} \label{second}
27
28 This is section 1.1.
29 \end{document}

```

在导入 *hyperref* 时必须非常小心，一般而言，它必须是最后一个要导入的包。

网址链接

众所周知，在文档中插入网址之类的文本同样需要用到超链接，同样的，使用 *hyperref* 宏包可以创建网页超链接。有时我们需要将超链接命名并隐藏网址，这时我们可以使用 href 命令进行插入；有时，我们插入的网址链接太长，但 LaTeX 不会自动换行，往往会造成格式混乱的问题，这时，我们可以使用 url 工具包，并在该工具包中声明一个参数即可解决这个问题，相关命令为\usepackage [hyphens]{url}。

【例】 在 LaTeX 中使用 *hyperref* 及 url 工具包插入网页链接并设置自动换行：

```
1 \usepackage[hyphens]{url}
2 \usepackage{hyperref}
3
4 \begin{document}
5
6 This is the website of open-source latex-cookbook repository: \href{https://
7     github.com/xinychen/latex-cookbook}{\LaTeX-cookbook} or go to the next url: \
8     url{https://github.com/xinychen/latex-cookbook}.
9
10 \end{document}
```

本地文件链接

有时，需要将文本与本地文件进行链接，`href` 命令也可用于打开本地文件。

【例】在 LaTeX 中使用 `href` 命令打开本地文件：

```
1 \usepackage[hyphens]{url}
2 \usepackage{hyperref}
3
4 \begin{document}
5
6 This is the text of open-source latex-cookbook repository: \href{run:./\LaTeX-
7     cookbook.dox}{\LaTeX-cookbook}.
8
9 \end{document}
```

8.2.2 超链接格式

当然，有时候为了突出超链接，也可以在工具包 `hyperref` 中设置特定的颜色，设置的命令为`\hypersetup`，一般放在前导代码中，例如 `colorlinks = true, linkcolor=blue, urlcolor = blue, filecolor=magenta`。默认设置以单色样式的空间字体打印链接，`\urlstyle{same}` 命令将改变这个样式，并以与文本其余部分相同的样式显示链接。

【例】在 LaTeX 中使用 `hyperref` 工具包插入超链接并设置超链接颜色为蓝色：

```
1 \usepackage{blindtext}
2 \usepackage{hyperref} %超链接包
```

```
3  \hypersetup{colorlinks = true, %链接将被着色， 默认颜色是红色
4      linkcolor=blue, % 内部链接显示为蓝色
5      urlcolor = cyan, % 网址链接为青色
6      filecolor=magenta} % 本地文件链接为洋红色
7  \urlstyle{same}
8
9  \begin{document}
10
11 \frontmatter
12 \tableofcontents
13 \clearpage
14
15 \addcontentsline{toc}{chapter}{Foreword}
16 {\huge \bf Foreword}
17
18 This is foreword.
19 \clearpage
20
21 \mainmatter
22
23 \chapter{First Chapter}
24
25 This is chapter 1.
26 \clearpage
27
28 \section{First section} \label{second}
29
30 This is section 1.1.
31
32 This is the website of open-source latex-cookbook repository: \href{https://
github.com/xinychen/latex-cookbook}{\LaTeX-cookbook} or go to the next url: \
url{https://github.com/xinychen/latex-cookbook}.
33
34 This is the text of open-source latex-cookbook repository: \href{run:./LaTeX-
cookbook.dox}{\LaTeX-cookbook}
35
36 \end{document}
```

8.3 BibTeX 用法

LaTeX 主要有两种管理参考文献的方法，第一种方法是在`.tex` 文档中嵌入参考文献，参考文献格式需符合特定的文献引用格式；另一种方法则是使用 *BibTeX* 进行文献管理，文件的拓展名为`.bib`。其中，使用外部文件 BibTeX 管理文献更加便捷高效。

8.3.1 创建参考文献

在 LaTeX 中，插入参考文献的一种直接方式是使用 *thebibliography* 环境，以列表的形式将参考文献进行整理起来，配以标签，以供正文引用，文档中引用的命令为`\cite{}`。

【例】使用 *thebibliography* 环境在文档中插入参考文献并进行编译：

```
1 Some examples for showing how to use \texttt{\begin{thebibliography}} environment:  
2 \begin{itemize}  
3     \item Book reference sample: The \LaTeX\ companion book \cite{latexcompanion}  
3     }.  
4     \item Paper reference sample: On the electrodynamics of moving bodies \cite{einstein}.  
5     \item Open-source reference sample: Knuth: Computers and Typesetting \cite{knuthwebsite}.  
6 \end{itemize}  
7  
8 \begin{thebibliography}{9}  
9 \bibitem{latexcompanion}  
10 Michel Goossens, Frank Mittelbach, and Alexander Samarin.  
11 \textit{The \LaTeX\ Companion}.  
12 Addison-Wesley, Reading, Massachusetts, 1993.  
13  
14 \bibitem{einstein}  
15 Albert Einstein.  
16 \textit{Zur Elektrodynamik bewegter K\"orper}. (German)  
17 [\textit{On the electrodynamics of moving bodies}].  
18 Annalen der Physik, 322(10):891–921, 1905.  
19  
20 \bibitem{knuthwebsite}  
21 Knuth: Computers and Typesetting,  
22 \\ \texttt{http://www-cs-faculty.stanford.edu/\~{}uno/abcde.html}  
23 \end{thebibliography}
```

【例】使用 `thebibliography` 环境在文档中插入参考文献并进行编译：

```

1 \LaTeX{} \cite{lamport94} is a set of macros built atop \TeX{} \cite{texbook}.
2
3 \begin{thebibliography}{9}
4   \bibitem{texbook}
5     Donald E. Knuth (1986) \emph{The \TeX{} Book}, Addison-Wesley Professional.
6
7   \bibitem{lamport94}
8     Leslie Lamport (1994) \emph{\LaTeX: a document preparation system}, Addison
9     Wesley, Massachusetts, 2nd ed.
10  \end{thebibliography}
```

参考：

https://www.overleaf.com/learn/latex/Bibliography_management_with_bibtex

8.3.2 使用 BibTeX 文件

BibTeX 是 LaTeX 最为常用的一个文献管理工具，它通常以一个独立的文件出现，其拓展名为 `.bib`。它是伴随着 LaTeX 文档排版系统出现的，1985 年兰伯特博士与其合作者开发了这一工具。作为一种特殊的且独立于 LaTeX 文件 `.tex` 之外的数据库，它能大大简化 LaTeX 文档中的文献引用。实际上，BibTeX 文件中的文献都是用列表的形式罗列的，且不分先后顺序。通过使用引用命令如 `\cite{}` 即可在文档中自动生成特定格式的参考文献，其中，文档中的参考文献格式一般是提前设定好的。

【例】使用 BibTeX 命令一个文献管理文件为 `sample.bib`，将文献按照指定格式进行整理，插入参考文献并进行编译：

```

1 % 创建Bibtex文件，并将其命名为sample.bib
2
3 @article{einstein,
4   author = "Albert Einstein",
5   title = "{Zur Elektrodynamik bewegter K{"o}rper}. ({German})",
6   journal = "Annalen der Physik",
7   volume = "322",
8   number = "10",
9   pages = "891--921",
```

```

11     year = "1905",
12     DOI = "http://dx.doi.org/10.1002/andp.19053221004"
13 }
14
15 @book{latexcompanion,
16     author = "Michel Goossens and Frank Mittelbach and Alexander Samarin",
17     title = "The \LaTeX\ Companion",
18     year = "1993",
19     publisher = "Addison-Wesley",
20     address = "Reading, Massachusetts"
21 }
22
23 @misc{knuthwebsite,
24     author = "Donald Knuth",
25     title = "Knuth: Computers and Typesetting",
26     url = "http://www-cs-faculty.stanford.edu/~uno/abcde.html"
27 }
```

在这三条文献中, einstein、latexcompanion、knuthwebsite 是文献的标签, 在文档中, 只需要在适当位置用引用命令如\cite{} 便可以引用这些文献, 例如,\cite{einstein}。

```

1 Some examples for showing how to use \texttt{\thebibliography} environment:
2 \begin{itemize}
3     \item Book reference sample: The \LaTeX\ companion book \cite{latexcompanion}.
4     \item Paper reference sample: On the electrodynamics of moving bodies \cite{einstein}.
5     \item Open-source reference sample: Knuth: Computers and Typesetting \cite{knuthwebsite}.
6 \end{itemize}
7
8 \bibliographystyle{unsrt}
9 \bibliography{sample}
```

在 sample.bib 文件中, 根据文献类型可定义文献列表, 对于每篇文献, 需要整理 author (作者信息)、title (文献标题) 等基本信息。在 LaTeX 文档中, 我们需要使用\bibliographystyle 命令申明参考文献的格式, 如本例中的 unsrt, 同时, 使用\bibliography 命令申明参考文献的源文件, 即 sample.bib。

当然, BibTeX 文献管理也有很多优点:

- 无需重复输入每条参考文献。文献放在 BibTeX 之后，引用文献的标签即可在文档中显示参考文献。
- 文档中的参考文献格式是根据文档样式自动设置的，且所有文献的引用风格是一致的。
- 引用同一作者同年的文献过多时，引用格式会自动调整。
- BibTeX 文件中的文献只有在文档中明确引用才会显示在文档的参考文献中。

在 BibTeX 文件中，不同类型的文献是需要进行分类的：

- article：对应着期刊或杂志上发表的论文，必须添加的信息有 author（作者）、title（标题）、journal（期刊）、year（年份）、volume（卷），可供选择添加的信息包括 number（期）、pages（页码）、month（月份）、doi（数字对象识别码）等。
- book：对应着书籍，必须添加的信息有 author/editor（作者或主编）、title（书名）、publisher（出版社）、year（年份），可供选择添加的信息包括 volume/number（卷/期）、series（系列）、address（出版地址）、edition（版号）、month（月份）、url（网址）等。
- inbook：书籍中的一部分或者某一章节，必须添加的信息有 author/editor、title（标题）、chapter/pages（章节/页码）、publisher（出版社）、year（年份），其他可供选择添加的信息与 book 一致。
- inproceedings：对应着会议论文，必须添加的信息有 author（作者）、title（论文标题）、booktitle（论文集标题）、year（年份），可供选择添加的信息包括 editor（版号）、volume/number（卷或期）、series（系列）、pages（页码）、address（地址）、month（月份）、organization（组织方）、publisher（出版社）等。
- conference：对应着会议论文，与 inproceedings 用法一致。
- mastersthesis 和 proceedings：分别对应着硕士学位论文和博士学位论文，必须添加的信息有 author（作者）、title（标题）、school（学校或研究机构）、year（年份）。

8.4 文献引用格式

Bibtex 的最大特点是采用了标准化的数据库，对于论文、著作以及其他类型的文献，我们可以自定义文献的引用格式。Bibtex 的样式会改变所引用文献的引用格式。

8.4.1 几种标准样式

一般而言，LaTeX 中有一系列标准样式 (standard styles) 可供选择和使用。具体而言，这些标准样式对应的文件包括：

- plain.bst
- acm.bst：对应于 Association for Computing Machinery 期刊。
- ieeetr.bst：对应于 IEEE Transactions 期刊。
- alpha.bst
- abrv.bst
- siam.bst：对应于 SIAM。

当然，实际上还有很多 *.bst* 文件，这里给出的几个文件只是最为常用的。不得不提的是 *natbib* 工具包，这一工具包对一系列引用命令进行了标准化，而这种标准化不受不同文献样式的影响。

Choosing a BibTeX Style:

<https://www.reed.edu/cis/help/LaTeX/bibtexstyles.html>

第九章 幻灯片制作

2003 年，柏林工业大学的博士生 Till Tantau 发布了一款用于制作演示文稿的工具 Beamer。Beamer 是 Till Tantau 利用业余时间开发的，他的初衷是方便自己使用 LaTeX 制作幻灯片，不过出乎意料的是，后来 Beamer 的受欢迎程度完全超出了他的想象。在 Till Tantau 开发 Beamer 期间，他收到了很多人的建议和反馈，这些都直接推动了开发工作。2010 年，Till Tantau 将 Beamer 交由他人维护和管理。

Beamer 作为 LaTeX 的一种特殊文档类型，它的出现无疑丰富了 LaTeX 制作演示文稿的功能。虽然 Beamer 并非 LaTeX 中第一款用于制作演示文稿的工具，但直到今日，Beamer 却是最受大家欢迎的一款。Beamer 的使用方式简单灵活，只需设定 LaTeX 文档类型为 beamer，便可开始创作。同时，Beamer 提供了大量的幻灯片模板，这些模板包含了丰富多彩的视觉效果，创作者可以直接使用。毫不夸张地说，Beamer 的出现极大地提高了人们使用 LaTeX 制作幻灯片的热情。值得一提的是，在 2005 年，Till Tantau 又发布了一款非常便捷的 LaTeX 绘图工具 TikZ。TikZ 不仅可以辅助 Beamer 幻灯片的制作，也可以应用于科技文档中的各类绘图任务。

Beamer 是随着 LaTeX 的发展而衍生出来的一种特殊文档类型，也常常被看作是一个功能强大的宏包，可以支撑科研工作者制作幻灯片的需求。使用 Beamer 制作幻灯片与 Office 办公软件（如 PowerPoint）完全不同，从本质上来说，使用 Beamer 制作幻灯片其实和 LaTeX 中的其他文档类型没有太大区别：代码结构都是由前导代码和主体代码组成，完全沿用了 LaTeX 的文档环境与基本命令。因此，使用 Beamer 制作幻灯片也有一个缺点，那就是必须掌握 LaTeX 制作文档的基础。

在呈现方式上，Beamer 制作的幻灯片最终会在 LaTeX 中被编译成 PDF 文档，在不同的系统上打开幻灯片时不存在不兼容等问题。在功能上，使用 Beamer 制作幻灯片时，我们可以对常规文本、公式、列表、图表甚至动画效果、视觉效果和主题样式等进行调整，最终达到想要的视觉效果。

事实上，LaTeX 中用于制作演示文稿的工具并非只有 Beamer，但相比其他工具，Beamer 具有如下优点：

- 拥有海量的模板和丰富的主题样式，且使用方便；

- 能满足制作幻灯片的功能性需求，从创建标题、文本和段落到插入图表、参考文献等操作，且与常规文档中的使用规则几乎一致；
- 使用方式简单，在主体代码中使用 frame 环境就能创建一页幻灯片。

本章主要包括以下内容：Beamer 的基本使用方式、在演示稿中添加动画效果、添加文本框等框元素、设置主题样式、插入程序源代码、添加参考文献、插入表格、插入与调整图片。

9.1 基本介绍

Beamer 是一款灵活的幻灯片制作工具，我们可以在 LaTeX 中将它作为一种文档类型进行使用。本节主要介绍 Beamer 的基本使用方式，包括创建幻灯片、创建章节、生成目录等操作。

9.1.1 Beamer 简介

在上述章节中，我们主要介绍了 LaTeX 中比较常用的文档类型 article，可用于创建期刊论文、技术报告等。本章中我们将介绍另一种文档类型：beamer。Beamer 的开发者 Till Tantau 说，“BEAMER is a LATEX class for creating presentations”，显然，Beamer 是一种用于制作演示文稿或者幻灯片的文档类型。

从使用角度来说，beamer 文档类型和 book、article 等文档类型一样，都是在以.tex 为拓展名的文件上编写程序和文档内容，然后再通过编译生成 PDF 文档。当然，Beamer 也兼具常用演示文稿如 PowerPoint 的主要功能，可以自行设置动态效果、甚至使用主题样式修改幻灯片的外观。

与其他文档类型相似的是，beamer 文档类型中拥有很多视觉效果极好的模板，这些模版已经设置好了特定的主题样式，有时候甚至只需要加入创作内容即可得到心仪的幻灯片。使用 Beamer 制作幻灯片时，我们可以体验 LaTeX 排版论文的几乎所有优点，公式排版、图表排版、参考文献设置等也非常便捷，有时候甚至可以将常规文档中的内容直接复制到 Beamer 文档类型中，稍加调整便能得到样式合适的幻灯片。另外，我们也可以根据需要，在前导代码中使用全局设置调整幻灯片的主题样式、颜色主题、字体主题等。

使用 beamer 制作幻灯片仍然遵循着 LaTeX 的一般使用方法，代码结构分为前导代码和主体代码，前导代码除了申明文档类型为 beamer 外，即\documentclass{beamer}，调用宏包等与常规文档的制作基本是一致的。

【例】使用 beamer 文档类型创建一个简单的幻灯片：

```
1 \documentclass{beamer}\n2\n3 \title{A Simple Beamer Example}\n4 \author{Author's Name}\n5 \institute{Author's Institute}\n6 \date{\today}\n7\n8 \begin{document}\n9\n10 \frame{\titlepage}\n11\n12 \end{document}
```

编译后得到的幻灯片如图9.1所示。

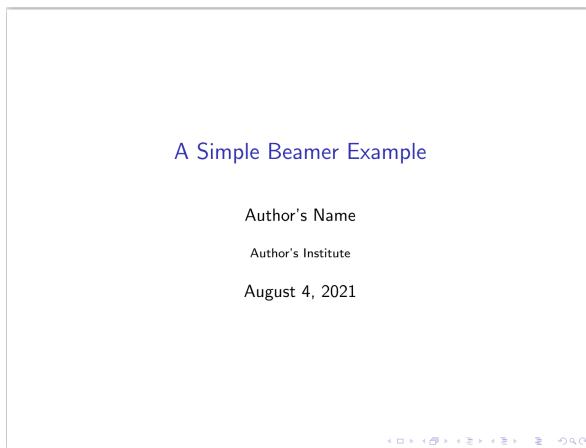


图 9.1: 编译后的幻灯片效果

在例子中, \title{}、\author{} 和 \date{} 这几个命令分别对应着标题、作者以及日期, 一般放在标题页, 如果想在幻灯片首页显示这些信息, 可以在使用 \frame{\titlepage} 命令新建标题页。

总结来说, 标题及作者信息对应的特定命令包括:

- 标题: 对应的命令为 \title[A]{B}, 其中, 位置 A 一般填写的是简化标题, 而位置 B 则填写的是完整标题, 这里的完整标题有时候可能会很长。
- 副标题: 对应的命令为 \subtitle[A]{B}, 其中, 位置 A 一般填写的是简化副标题, 而位置 B 则填写的是完整副标题, 这里的完整副标题有时候也可能会很长。

- 作者：对应的命令为\author{A}{B}，用法类似。
- 日期：对应的命令为\date{A}{B}，用法类似。
- 单位：对应的命令为\institution{A}{B}，用法类似。

我们知道，在常规文档 article 中，申明文档类型时可以指定正文字体大小，在文档类型的申明语句 \documentclass{beamer} 中，我们也可以通过特定选项调整幻灯片内容的字体大小，一般默认为 11pt，我们也可以根据需要使用 8pt、9pt、10pt、12pt、14pt、17pt、20pt 字体大小，例如使用\documentclass[12pt]{beamer} 可以将字体大小设置为 12pt。

制作幻灯片时，有时候为了达到特定的投影效果，会设置幻灯片的长宽比例，比较常用的两种长宽比例分别为 4:3 和 16:9。一般来说，Beamer 制作出来的幻灯片默认大小为长 128 毫米、宽 96 毫米，对应着默认的长宽比例 4:3，有时候，我们也可以根据需要将幻灯片的长宽比例调整为 16:9、14:9、5:4 甚至 3:2。

【例】 使用 beamer 文档类型创建一个简单的幻灯片，将幻灯片的长宽比例调整为 16:9：

```

1  \documentclass[aspectratio = 169]{beamer}
2
3  \title{A Simple Beamer Example}
4  \author{Author's Name}
5  \institute{Author's Institute}
6  \date{\today}
7
8  \begin{document}
9
10 \frame{\titlepage}
11
12 \end{document}
```

编译后得到的幻灯片如图9.2所示。

在例子中，选项 aspectratio 对应着长宽比例，数字 169 对应着长宽比例 16:9，类似地，149、54、32 分别对应着长宽比例 14:9、5:4、3:2。

9.1.2 创建幻灯片

frame 这个词在计算机编程中非常常见，这一英文单词的字面意思可以翻译为“帧”，假如我们将幻灯片视作“连环画”，是由一页一页单独的幻灯片组成，那么每

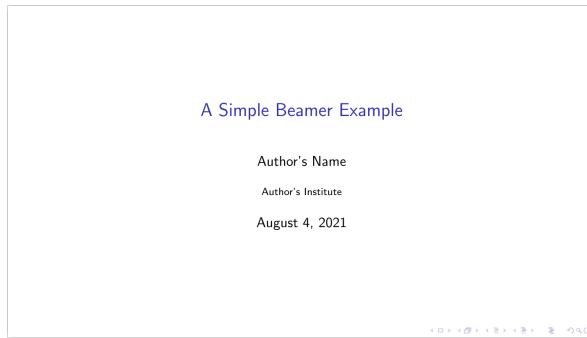


图 9.2: 编译后的幻灯片效果

页幻灯片则对应着连环画中的帧。使用 Beamer 制作幻灯片时，幻灯片就是用 frame 环境创建出来的，然而，有时候为了让幻灯片产生动画视觉效果，Beamer 中的帧（即 frame）与每页幻灯片并非严格意义上的一一对应。

在 beamer 文档类型中，制作幻灯片的环境一般为 frame。在 document 环境构成的主体代码中，一个 frame 环境一般对应着一页幻灯片。

每张幻灯片一般都有一个标题，有时也会有一个副标题。若要创建标题和副标题，用户可以通过使用 `\begin{frame}{}{}` 的命令格式，其中第一、二个 {} 中分别为幻灯片的标题和副标题；此外，用户也可以通过在 frame 环境中，使用`\frametitle{}` 和`\framesubtitle{}` 命令分别创建标题和副标题。由此创建的标题和副标题一般位于幻灯片的顶部，标题相对于副标题字体稍大一点。

实际上，Beamer 与其他文档类型并没有特别大的差异，常规文档中的基本列表环境都可以在 Beamer 中使用，包括：有序列表环境 enumerate、无序列表环境 itemize 以及解释性列表环境 description。

【例】使用 beamer 文档类型创建一个简单的幻灯片：

```
1 \documentclass{beamer}
2 \usefonttheme{professionalfonts}
3
4 \begin{document}
5
6 \begin{frame}
7 \frametitle{Parent function}
8 \framesubtitle{A short list}
9
10 Please check out the following parent function list.
11 \begin{enumerate}
12 \item $y=x$
```

```

13 \item $y=|x|$
14 \item $y=x^{\{2\}}$
15 \item $y=x^{\{3\}}$
16 \item $y=x^{\{b\}}$
17 \end{enumerate}
18
19 \end{frame}
20
21 \end{document}

```

编译后得到的幻灯片如图9.3所示。

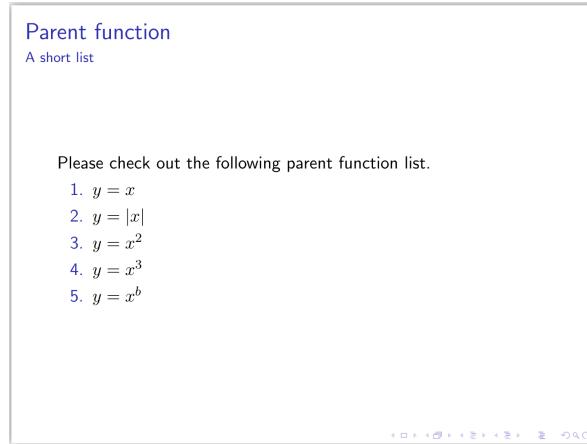


图 9.3: 编译后的幻灯片效果

有时为了简化代码，也可以直接用`\frame{}`命令取代 frame 环境囊括幻灯片内容。

【例】 使用 beamer 文档类型中的 frame 简化环境命令创建一个简单的幻灯片：

```

1 \documentclass{beamer}
2 \usefonttheme{professionalfonts}
3
4 \begin{document}
5
6 \frame{
7   \frametitle{Parent function}
8   \framesubtitle{A short list}
9
10  Please check out the following parent function list.
11  \begin{enumerate}

```

```
12 \item $y=x$  
13 \item $y=|x|$\n14 \item $y=x^2$  
15 \item $y=x^3$  
16 \item $y=x^b$  
17 \end{enumerate}  
18  
19 \end{document}
```

使用 Beamer 制作幻灯片时，幻灯片内容会在标题下方自动居中对齐，如果想调整对其方式，可以在 frame 环境中设置参数，具体而言，有以下几种：

c 居中对齐，字母 c 对应着英文单词 center 的首字母，一般而言，[c] 作为默认参数，无需专门设置；

t 让幻灯片内容进行顶部对齐，其中，字母 t 对应着英文单词 top 的首字母；

b 让幻灯片内容进行底部对齐，其中，字母 b 对应着英文单词 bottom 的首字母。

【例】使用 beamer 文档类型中的 frame 环境创建一个简单的幻灯片，并让幻灯片内容进行顶部对齐：

```
1 \documentclass{beamer}  
2 \usefonttheme{professionalfonts}  
3  
4 \begin{document}  
5  
6 \begin{frame}[t]  
7 \frametitle{Parent function}  
8 \framesubtitle{A short list}  
9  
10 Please check out the following parent function list.  
11 \begin{enumerate}  
12 \item $y=x$  
13 \item $y=|x|$\n14 \item $y=x^2$  
15 \item $y=x^3$  
16 \item $y=x^b$  
17 \end{enumerate}  
18  
19 \end{frame}
```

```

20
21 \end{document}

```

编译后得到的幻灯片如图9.4所示。

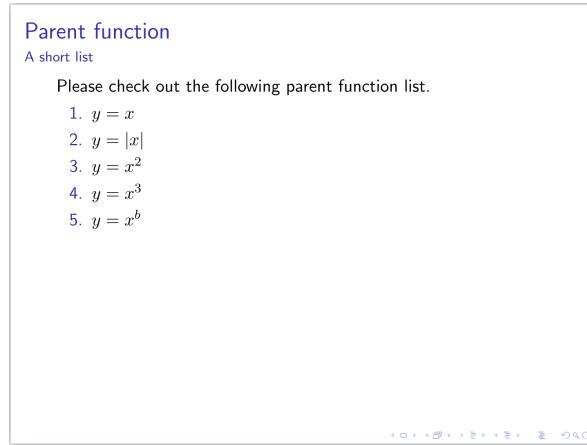


图 9.4: 编译后的幻灯片效果

上面例子介绍了如何创建单页幻灯片，类似地，可以使用多个 frame 环境制作多页幻灯片。

【例】 使用 beamer 文档类型中的 frame 环境创建一个多多页的幻灯片：

```

1 \documentclass{beamer}
2
3 \title{The title}
4 \subtitle{The subtitle}
5 \author{Author's name}
6
7 \begin{document}
8
9 \begin{frame}
10   \titlepage % 创建标题页
11 \end{frame}
12
13 \begin{frame}
14   \frametitle{Frame title}
15   The body of the frame.
16 \end{frame}
17
18 \end{document}

```

编译后得到的幻灯片如图9.5所示。

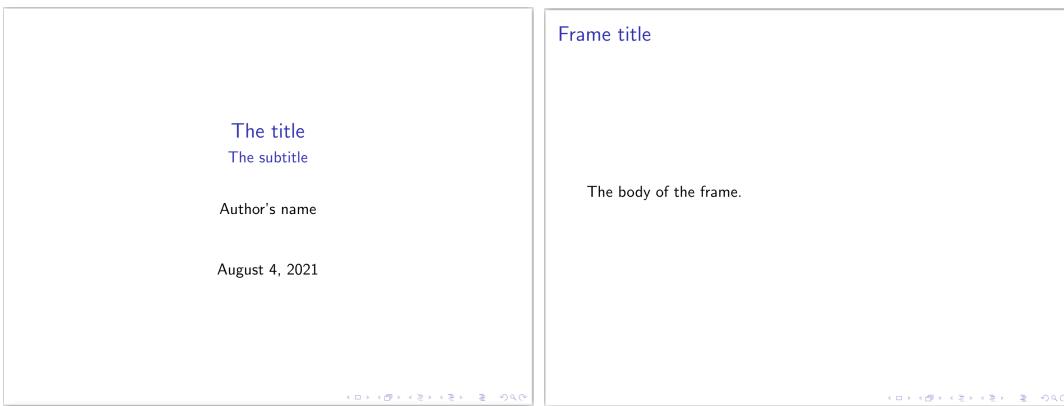


图 9.5: 编译后的幻灯片效果

9.1.3 创建章节与生成目录

类似 article 文档类, beamer 中可以利用\part{}、\section{}、\subsection{}、以及\subsubsection{} 等命令构建演示稿中的章节层次, 但此时\chapter{} 命令无效。其中, 章节标题写在 {} 中, 但编译后不会出现在创建章节的位置, 仅在目录和导航条中显示。类似地, 可以通过加 * 号使得章节标题不出现在目录中, 但仍然会在导航条中显示。

在 beamer 中, 可以使用\tableofcontents 命令自动生成演示稿目录, 通过在 frame 幻灯片页中添加该命令即可。由此生成的目录实际上是超链接, 点击之后会自动跳转到相应章节。

【例】在 beamer 文档类型中使用 tableofcontents 命令为幻灯片生成目录, 并使用 section 和 subsection 创建章节:

```
1 \documentclass{beamer}
2
3 \begin{document}
4
5 \begin{frame}{Table of contents}
6 \tableofcontents
7 \end{frame}
8
9 \section{Section A}
```

```

10  \begin{frame}{frame1}
11  \subsection{a1}
12  This is subsection a1. This is subsection a1.
13  \subsection{a2}
14  This is subsection a2. This is subsection a2.
15  \subsection{a3}
16  This is subsection a3. This is subsection a3.
17  \end{frame}

18
19  \section{Section B}
20  \begin{frame}{frame2}
21  \subsection{b1}
22  This is subsection b1. This is subsection b1. % 在下方插入空行，使得内容分行显示。
23
24  \subsection{b2}
25  This is subsection b2. % 在下方插入空行，使得内容分行显示。
26
27  This is subsection b2.
28  \end{frame}

29
30  \section*[Section C]
31  \begin{frame}{frame3}
32  \subsection*[c1]
33  This is subsection c1. This is subsection c1. % 在下方插入空行，使得内容分行显示。
34
35  \subsection*[c2]
36  This is subsection c2. This is subsection c2.
37  \end{frame}

38
39  \end{document}

```

编译后得到的幻灯片如图9.6所示。

从上例中可以看出，如果想让相邻章节或者同章节的内容分行显示，只需要在相应位置插入空行即可。

默认情况下，目录页中包含所有不含 * 号的章节标题，甚至是三级节标题。但有时目录只需要显示到一级节标题即可，而二级节标题及其次级标题则不需要显示，为此，只需要在\tableofcontents 命令后设置选项 [hideallsubsections] 即可。

【例】在 beamer 文档类型中使用\tableofcontents[hideallsubsections] 命令为幻灯片生成一级节目录：



图 9.6: 编译后的幻灯片效果

```
1 \documentclass{beamer}
2
3 \begin{document}
4
5 \begin{frame}{table of contents}
6 \tableofcontents[hideallsubsections]
7 \end{frame}
8
9 \section{Section A}
10 \begin{frame}{frame1}
11 \subsection{a1}
12 This is subsection a1. This is subsection a1.
13
14 \subsection{a2}
15 This is subsection a2. This is subsection a2.
16
17 \subsection{a3}
18 This is subsection a3. This is subsection a3.
19
20 \end{frame}
21
22 \section{Section B}
23 \begin{frame}{frame2}
24 \subsection{b1}
25 This is subsection b1. This is subsection b1.
26
27 \subsection{b2}
28 This is subsection b2. This is subsection b2.
29 \end{frame}
30
31 \end{document}
```

编译后得到的幻灯片如图9.7所示。

一般而言，使用\tableofcontents 命令生成的目录只会显示在相应的幻灯片页。有时候为了更好地梳理演示稿脉络，需要在各章节前均插入目录页，为此，一种更简便的方式是使用\AtBeginSection{}、\AtBeginSubsection{}、或\AtBeginSubsubsection{} 命令分别在一级节、二级节、三级节前均插入目录页。此外，使用\tableofcontents[currentsection] 命令或\tableofcontents[currentsubsection] 命令可以在各章节前的目录页中突

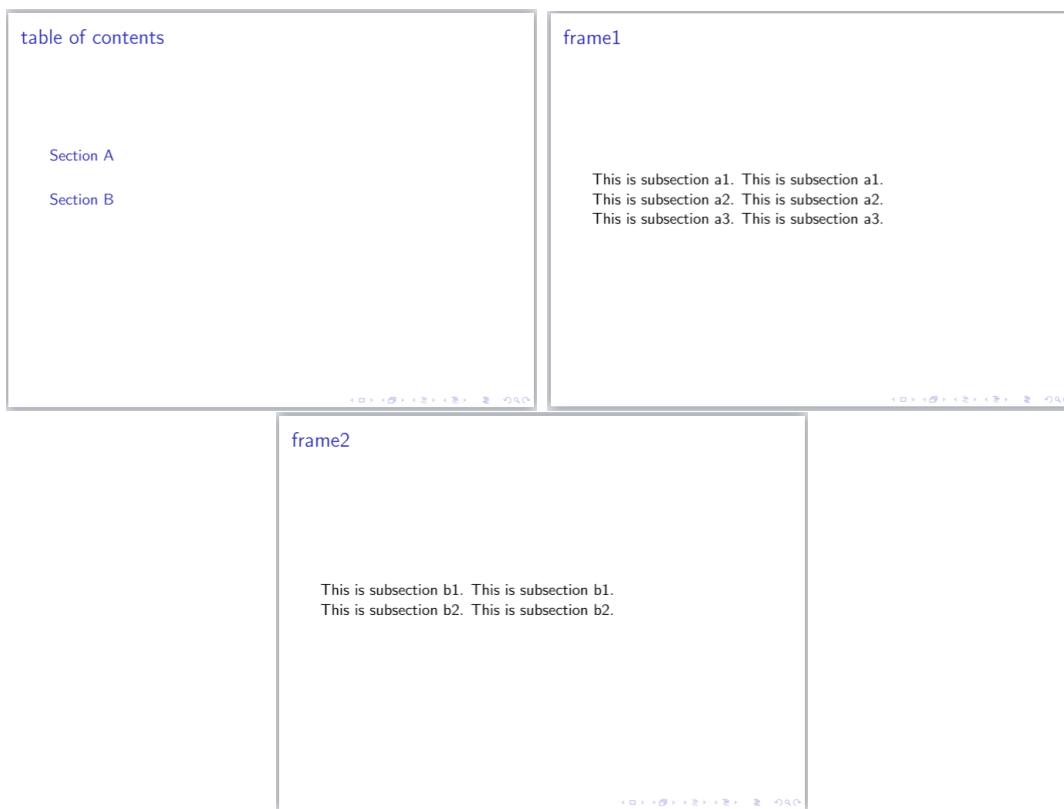


图 9.7: 编译后的幻灯片效果

出显示当前一级节标题或二级节标题。

【例】在 beamer 文档类型中使用\AtBeginSection{} 以及\tableofcontents[currentsection] 命令在幻灯片的各一级节前均插入目录页，并突出显示当前一级节标题：

```
1 \documentclass{beamer}
2
3 \begin{document}
4
5 \AtBeginSection
6 {
7 \begin{frame}{table of contents}
8 \tableofcontents[currentsection]
9 \end{frame}
10 }
11
12 \section{Section A}
13 \begin{frame}{frame1}
14 \subsection{a1}
15 This is subsection a1. This is subsection a1.
16
17 \subsection{a2}
18 This is subsection a2. This is subsection a2.
19
20 \subsection{a3}
21 This is subsection a3. This is subsection a3.
22 \end{frame}
23
24 \section{Section B}
25 \begin{frame}{frame2}
26 \subsection{b1}
27 This is subsection b1. This is subsection b1.
28
29 \subsection{b2}
30 This is subsection b2. This is subsection b2.
31 \end{frame}
32
33 \section{Section C}
34 \begin{frame}{frame3}
35 \subsection{c1}
36 This is subsection c1. This is subsection c1.
37
```

```
38 \subsection{c2}
39 This is subsection c2. This is subsection c2.
40 \end{frame}
41
42 \end{document}
```

编译后得到的幻灯片如图9.8所示。

The figure displays three slides from a Beamer presentation. The first slide, titled 'table of contents', shows a hierarchical table of contents with sections A, B, and C, each containing subsections. The second slide, titled 'frame1', contains the text 'This is subsection a1. This is subsection a1.
This is subsection a2. This is subsection a2.
This is subsection a3. This is subsection a3.'. The third slide, titled 'frame2', contains the text 'This is subsection b1. This is subsection b1.
This is subsection b2. This is subsection b2.'. Both frame slides include navigation icons at the bottom.

图 9.8: 编译后的幻灯片效果

生成目录时,我们也能自定义目录显示的动画格式,通过使用\tableofcontents[pausesections]

命令，同时在前导代码中申明`\setbeamercolor{dynamic}`语句即可。

【例】在 beamer 文档类型中使用`\tableofcontents` 命令生成幻灯片的目录，同时使用`\tableofcontents[pausesections]` 对目录设置动画格式：

```

1  \documentclass{beamer}
2  \setbeamercolor{dynamic}
3
4  \begin{document}
5
6  \begin{frame}
7  \frametitle{Table of Contents}
8
9  \tableofcontents[pausesections]
10
11 \end{frame}
12
13 \section{Intro to Beamer}
14 \subsection{About Beamer}
15 \subsection[Basic Structure]{Basic Structure}
16 \subsection{How to Compile}
17 \section{Overlaying Concepts}
18 \subsection{Specifications}
19 \subsection[Examples]{Examples: Lists, Graphics, Tables}
20 \section{Sparkle}{Adding that Sparkle}
21 \subsection{Sections}
22 \subsection{Themes}
23 \section*{References}
24
25 \begin{frame}
26
27 \end{frame}
28
29 \end{document}

```

编译后得到的幻灯片如图9.9所示。

9.1.4 幻灯片内容分栏

对幻灯片内容进行分栏有两种常用方式，第一种是使用 `multicol` 宏包中的`\begin{multicols}{A}`
`\end{multicols}` 环境，其中位置 A 可用于设定分栏列数；第二种是使用`\begin{columns}`

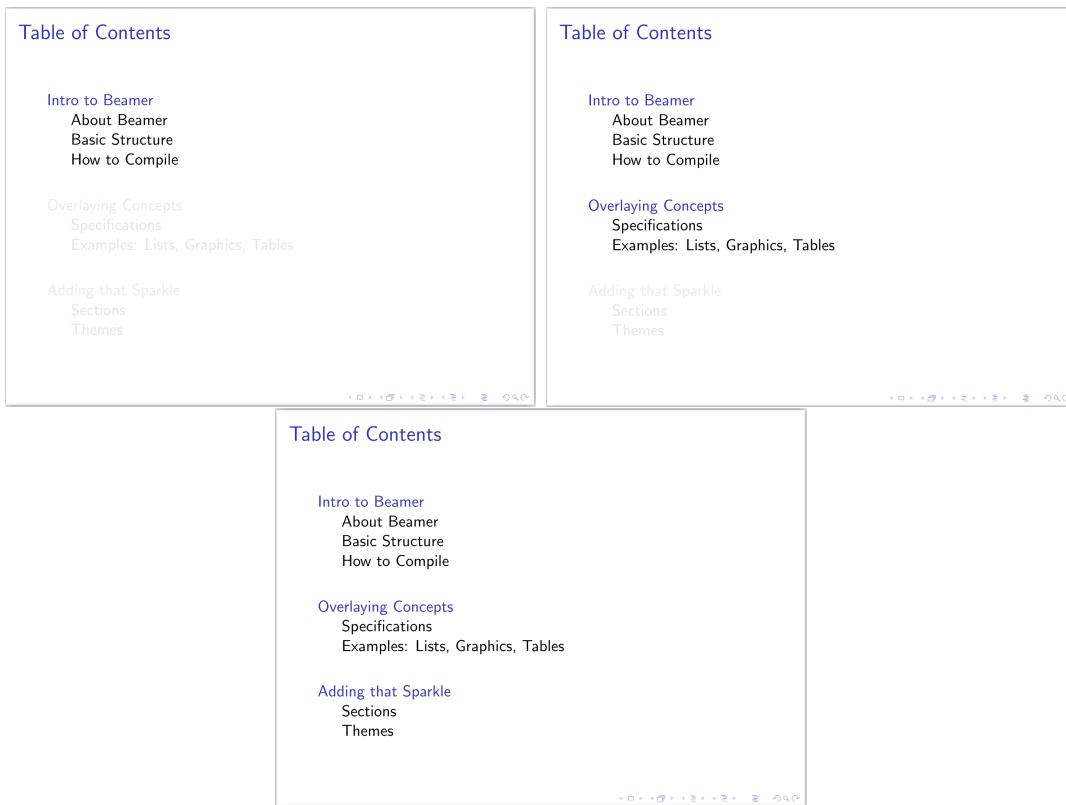


图 9.9: 编译后的幻灯片效果

\end{columns} 环境。

【例】在 beamer 文档类型中使用 multicol 宏包对列表内容进行分栏处理：

```

1  \documentclass{beamer}
2  \usefonttheme{professionalfonts}
3  \usepackage{multicol}
4
5  \begin{document}
6
7  \begin{frame}
8  \frametitle{Parent function}
9  \framesubtitle{A short list}
10
11 Please check out the following parent function list.
12 \begin{enumerate}
13 \begin{multicols}{3}
14 \item $y=x$%
15 \item $y=|x|$
16 \item $y=x^{2}$
17 \item $y=x^{3}$
18 \item $y=x^b$%
19 \end{multicols}
20 \end{enumerate}
21
22 \end{frame}
23
24 \end{document}
```

编译后得到的幻灯片如图9.10所示。

【例】在 beamer 文档类型中使用 columns 环境对幻灯片内容进行分栏处理：

```

1  \documentclass{beamer}
2  \usefonttheme{professionalfonts}
3
4  \begin{document}
5
6  \begin{frame}
7  \frametitle{Parent function}
8  \framesubtitle{A short list}
9
```

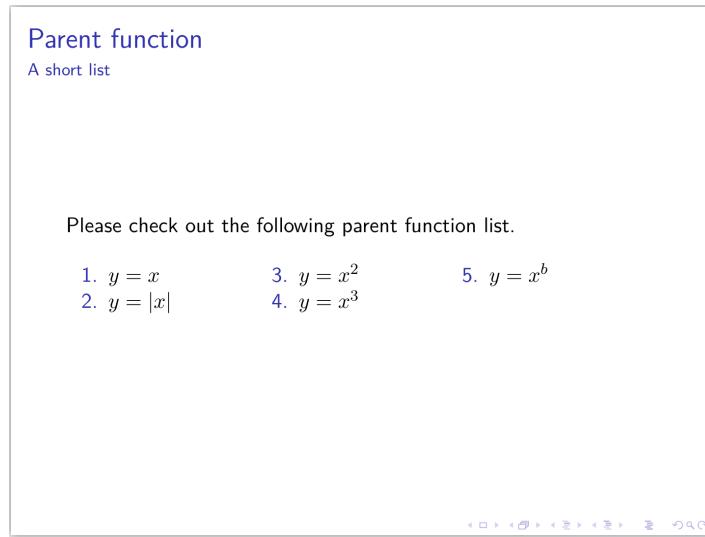


图 9.10: 编译后的幻灯片效果

```
10 \begin{columns}
11   \begin{column}{0.5\textwidth}
12
13     Please check out the following parent function list.
14     \begin{enumerate}
15       \item $y=x$%
16       \item $y=|x|$%
17       \item $y=x^{2}$%
18       \item $y=x^{3}$%
19       \item $y=x^{b}$%
20     \end{enumerate}
21
22   \end{column}
23
24   \begin{column}{0.5\textwidth}
25
26     Please check out the following parent function list.
27     \begin{enumerate}
28       \item $y=x$%
29       \item $y=|x|$%
30       \item $y=x^{2}$%
31       \item $y=x^{3}$%
32       \item $y=x^{b}$%
33     \end{enumerate}
34
35   \end{column}
```

```

36 \end{columns}
37
38 \end{frame}
39
40 \end{document}

```

编译后得到的幻灯片如图9.11所示。

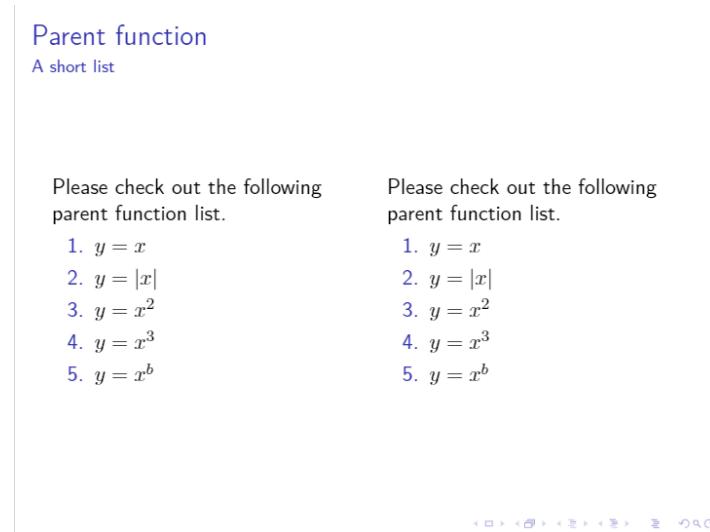


图 9.11: 编译后的幻灯片效果

参考资料

- Prathik Naidu, Adam Pahlavan. Fun with Beamer: An Epic Quest To Create the Perfect Presentation, June 28, 2017.

9.2 添加动画效果

在制作幻灯片时有时需要添加动画效果。由于 LaTeX 制作幻灯片会被编译成 PDF 文档，因此，在 Beamer 中，实现动画效果的方式是将具有动画内容的幻灯片按照次序拆分成若干页内容，在播放时通过翻页达到“动态”视觉效果。为了便于说明，以下将一个 frame 环境创建的内容称为一页幻灯片或幻灯片页、将动画效果拆分后得到的每一页内容称为该幻灯片的某一帧。

下面介绍在 Beamer 中常见的几种动画效果命令。

9.2.1 pause 命令

\pause 是 Beamer 中最常用的一种动画效果命令，它的使用方式极其简单，通过在文本或段落中添加\pause 命令，便可将一页幻灯片拆分成若干帧。一般来说，\pause 命令后的内容将会在下一帧中显示，从而使幻灯片在内容显示上呈现出动画效果。比如，一般情况下，使用列表环境创建的每项内容（使用\item 创建）都会在同一帧幻灯片中显示，为了达到各项内容逐个显示的动画效果，可以在两个相邻的\item 语句之间插入\pause 命令。

【例】在 beamer 文档类型中使用 pause 命令实现一个简单的动画效果：

```
1 \documentclass{beamer}
2 \usefonttheme{professionalfonts}
3
4 \begin{document}
5
6 \begin{frame}
7 \frametitle{Parent function}
8 \framesubtitle{A short list}
9
10 Please check out the following parent function list.
11 \begin{enumerate}
12 \item $y=x$ \pause
13 \item $y=|x|$ \pause
14 \item $y=x^2$ \pause
15 \item $y=x^3$ \pause
16 \item $y=x^b$ \end{enumerate}
17 \end{frame}
18
19 \end{document}
```

编译后得到的幻灯片如图9.12所示。



图 9.12: 编译后的幻灯片效果

9.2.2 item<> 命令

对于列表环境中的各项内容，也可以通过设置选项指定在该幻灯片的哪些步骤中显示该项内容，从而更加灵活地定制动画效果。具体是使用\item<>，其中 <> 用于指定显示步骤，对于没有指定 <> 显示范围的内容项默认会在所有幻灯片页面中显示。具体而言，<> 中的内容存在以下四种格式：

- <A,B,C>：表示内容项将在第 A、B、C 步中显示。如，\item<2, 3, 4> \$y=x\\$ 表示该项内容将出现在该页幻灯片放映的第 2、3、4 步；
- <A-B>：表示内容项将在第 A 至 B 步中显示。如，\item<1-4> \$y=x\\$ 表示该项内容将出现在该页幻灯片放映的第 1 4 步；
- <A->：表示内容项将在第 A 步及以后显示。如，\item<2-> \$y=x\\$ 表示该项内容将出现在该页幻灯片放映的第 2 步及之后的步骤中；
- <-A>：表示内容项将在第 A 步及之前显示。如，\item<-3> \$y=x\\$ 表示该项内容将出现在该页幻灯片放映的第 3 步及之前的步骤中。

由此创建的一张幻灯片将包含多帧，其帧数由所有\item<> 命令中设置的最大步骤决定。

如果想要在某一帧中突出某项内容，主要包括以下两种方式：

- 使用\alert 命令为该项内容指定需要高亮显示的步骤。具体用法如：\item<2-| alert@3-4> The second item.，此时内容项 “The second item.” 将出现在第 2 步之后的步骤中，并通过命令\alert 及前缀 @ 使其在第 3 4 步中高亮显示。
- 使用\color< 范围 >{显示颜色} 命令改变内容项的颜色。如\item<1-> \color<1>{red} The first item. 语句，内容 The first item. 将出现在第一步及之后的步骤中，通过\color<1>{red} 令该项内容在第一步显示颜色为红色，而在其它步骤中仍为默认颜色。

【例】 在 beamer 文档类型中使用 item 定制内容显示范围并使用 alert 对内容项进行高亮显示，从而实现一个简单的动画效果：

```
1 \documentclass{beamer}
2 \usefonttheme{professionalfonts}
3
4 \begin{document}
```

```

6  \begin{frame}
7    \frametitle{Parent function}
8    \framesubtitle{A short list}
9
10   Please check out the following parent function list.
11   \begin{enumerate}
12     \item<1-| alert@1> $y=x$  

13     \item<2-| alert@2> $y=|x|$
14     \item<3-| alert@3> $y=x^{2}$
15     \item<4-| alert@4> $y=x^{3}$
16   \end{enumerate}
17
18   \end{frame}
19
20   \end{document}

```

编译后得到的幻灯片如图9.13所示。

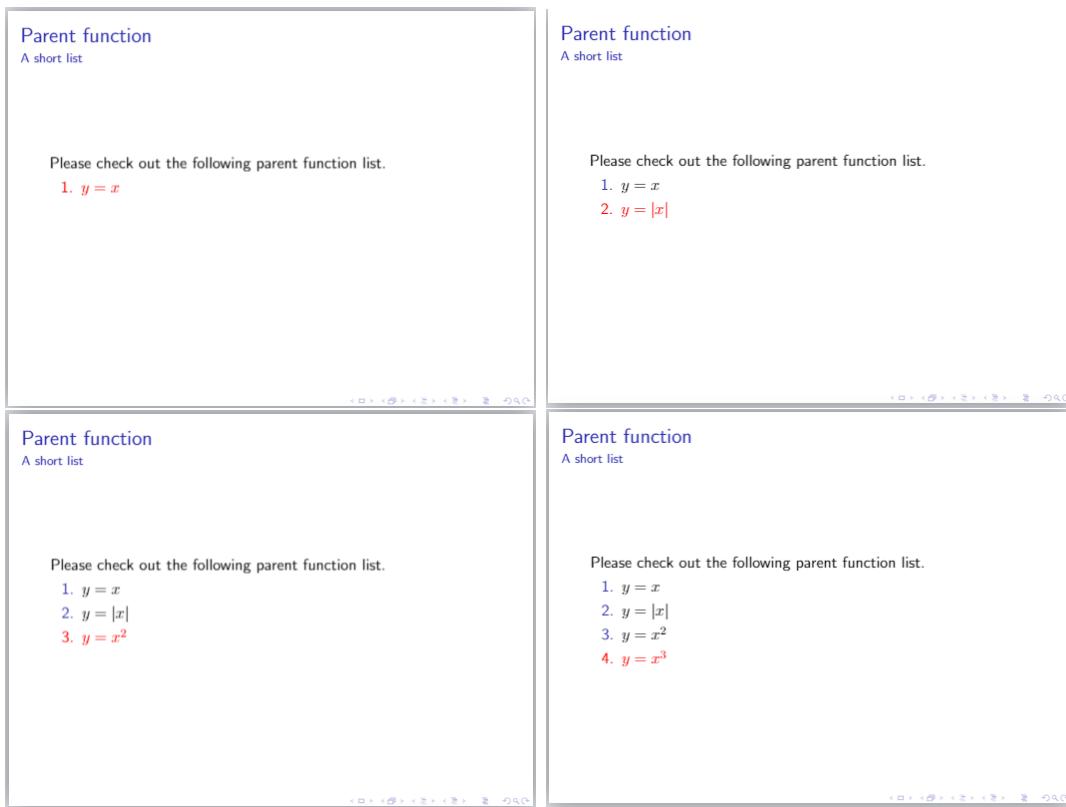


图 9.13: 编译后的幻灯片效果

【例】 在 beamer 文档类型中使用 item 定制内容显示范围并使用 color 对内容项

进行高亮显示：

```

1 \documentclass{beamer}
2 \usefonttheme{professionalfonts}
3
4 \begin{document}
5
6 \begin{frame}
7 \frametitle{Parent function}
8 \framesubtitle{A short list}
9
10 Please check out the following parent function list.
11 \begin{enumerate}
12 \item<1-> \color{red} $y=x$\\
13 \item<2-> \color{red} $y=|x|$
14 \item<3-> \color{red} $y=x^{2}$
15 \item<4-> \color{red} $y=x^{3}$
16 \end{enumerate}
17
18 \end{frame}
19
20 \end{document}
```

9.2.3 其它命令

LaTeX 还提供了其它命令可以实现类似的动画效果，同样可以在可选参数 $<>$ 中指定内容项或内容块的显示范围，主要包括\onslide、\uncover、\only、\visible、\invisible 等命令：

- \onslide $<>$ {}: 该命令可以指定内容在当前幻灯片页放映的第几步显示。使用该命令时不显示的内容将被“遮挡”，仍将占用其指定的位置（\uncover $<>$ { } 或\visible $<>$ { } 也能实现类似效果）；【例】在 beamer 文档类型中使用 onslide 命令实现一个简单的动画效果：

```

1 \documentclass{beamer}
2 \usefonttheme{professionalfonts}
3
4 \begin{document}
```

```

5   \begin{frame}
6     \frametitle{Parent function}
7     \framesubtitle{A short list}
8
9
10    \onslide<1->{Please check out the following parent function list.}
11
12    \onslide<2,4>{1. $y=x$}
13
14    \onslide<1-4>{2. $y=|x|$}
15
16    \onslide<2->{3. $y=x^{2}$}
17
18    \onslide<3->{4. $y=x^{3}$}
19
20    \onslide<4>{5. $y=x^{\{b\}}$}
21
22    \end{frame}
23
24    \end{document}

```

- `\only<>{}`: 该命令可以指定内容在当前幻灯片页放映的第几步插入。使用该命令时，不显示的内容对应的位置将腾空出来，可以用于显示其它内容；【例】在 beamer 文档类型中使用 only 命令实现一个简单的动画效果：

```

1   \documentclass{beamer}
2   \usefonttheme{professionalfonts}
3
4   \begin{document}
5
6   \begin{frame}
7     \frametitle{Parent function}
8     \framesubtitle{A short list}
9
10    \only<1->{Please check out the following parent function list.}
11
12    \only<2,4>{1. $y=x$}
13
14    \only<1-4>{2. $y=|x|$}

```

```

15      \only<2->{3. $y=x^{2$}
16
17      \only<3->{4. $y=x^{3$}
18
19      \only<4>{5. $y=x^{b$}
20
21      \end{frame}
22
23
24      \end{document}

```

编译后得到的幻灯片如图9.14所示。

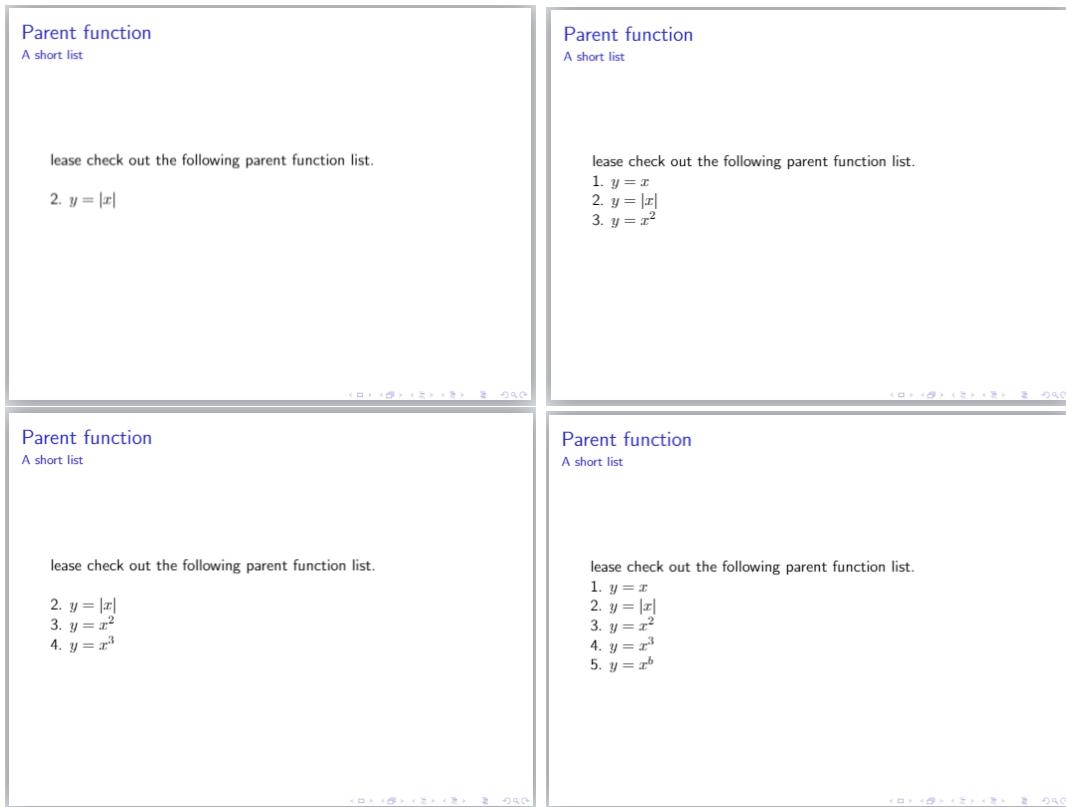


图 9.14: 编译后的幻灯片效果

- `\invisible{}`: 该命令的作用效果与`\visible{}`相反, 用于指定内容在当前幻灯片放映的第几步不可见。但与`\visible{}`相同的是, 使用`\invisible{}`命令时, 不可见的内容仍占据着对应的位置, 不可用于显示其它内容。**【例】** 在 beamer 文档类型中使用 invisible 命令实现一个简单的动画效果:

```

1   \documentclass{beamer}
2   \usefonttheme{professionalfonts}
3
4   \begin{document}
5
6   \begin{frame}
7   \frametitle{Parent function}
8   \framesubtitle{A short list}
9
10  \visible<1-4>{Please check out the following parent function list.}
11
12  \invisible<1,3>{1. $y=x$}
13
14  \invisible<>{2. $y=|x|$}
15
16  \invisible<1>{3. $y=x^{[2]}$}
17
18  \invisible<1-2>{4. $y=x^{[3]}$}
19
20  \invisible<1-3>{5. $y=x^{[b]}$}
21
22  \end{frame}
23
24  \end{document}

```

编译后得到的幻灯片如图9.15所示。

9.2.4 自动计数

上述介绍的动画效果定制命令均通过在 $<>$ 中指定具体的数字指定内容显示范围。此时，如果想要在开始处或中间插入新的内容项，则其余所有内容项的 $<>$ 显示范围都必须作出相应调整，显然不够灵活。LaTeX 提供了一种更巧妙的方式可以解决这类问题：使用“+”号代替具体数字，从 1 开始进行自动递增计数。就例 9-13 而言，可以用“+”符号代替各 $<>$ 中的具体数字，可以得到完全相同的编译效果。修改后的语句如下：【例】在 beamer 文档类型中使用 + 符号灵活定制幻灯片效果：

```

1   \documentclass{beamer}
2   \usefonttheme{professionalfonts}

```

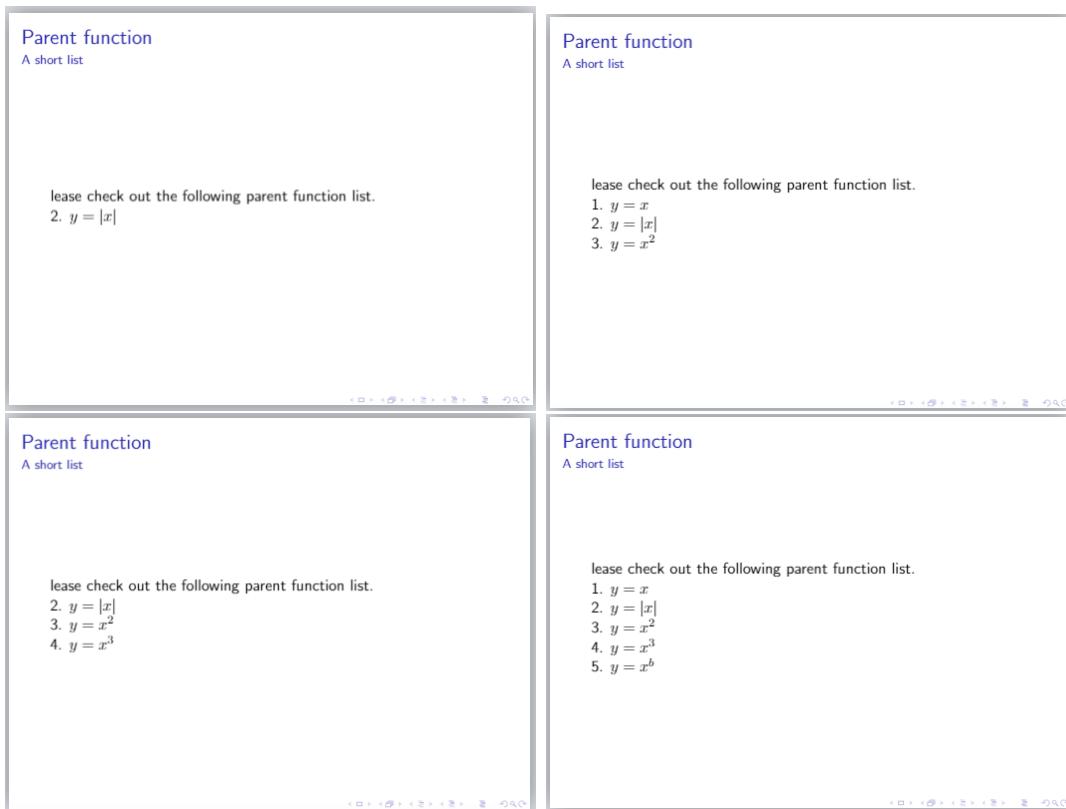


图 9.15: 编译后的幻灯片效果

```

3   \begin{document}
4
5   \begin{frame}
6     \frametitle{Parent function}
7     \framesubtitle{A short list}
8
9
10    Please check out the following parent function list.
11    \begin{enumerate}
12      \item<+-| alert@+> $y=x$ % 此时“+”号对应数字1
13      \item<+-| alert@+> $y=|x|$ % 此时“+”号对应数字2
14      \item<+-| alert@+> $y=x^{2}$ % 此时“+”号对应数字3
15      \item<+-| alert@+> $y=x^{3}$ % 此时“+”号对应数字4
16    \end{enumerate}
17
18    \end{frame}
19
20  \end{document}

```

上述语句每一条\item 格式相同，因此也可以简写为如下语句：

```

1   \documentclass{beamer}
2   \usefonttheme{professionalfonts}
3
4   \begin{document}
5
6   \begin{frame}
7     \frametitle{Parent function}
8     \framesubtitle{A short list}
9
10    Please check out the following parent function list.
11    \begin{enumerate}[<+-| alert@+>]
12      \item $y=x$%
13      \item $y=|x|$
14      \item $y=x^2$%
15      \item $y=x^3$%
16    \end{enumerate}
17
18    \end{frame}
19

```

20 \end{document}

有时在 `<>` 中使用的数字不总是从 1 开始递增，那么就需要使用“+(偏移量)”的命令格式。比如，如果当前“+”号对应的计数器值为 3，那么 `<+(2)->` 意味着在当前计数器值的基础上加 2，`<+(-2)->` 则意味着在当前计数器值的基础上减 2。

【例】 在 beamer 文档类型中使用 +(偏移量) 符号灵活定制任意显示步骤的幻灯片效果：

```
1 \documentclass{beamer}
2 \usefonttheme{professionalfonts}
3
4 \begin{document}
5
6 \begin{frame}
7 \frametitle{Parent function}
8 \framesubtitle{A short list}
9
10 Please check out the following parent function list.
11 \begin{enumerate}
12 \item<+(1)-> $y=x$ % 相当于`\item<2-> $y=x$`  

13 \item<+(2)-> $y=|x|$ % 相当于`\item<-4> $y=|x|$`  

14 \item<+(-1)-+(1)> $y=x^{2}$ % 相当于`\item<2-4> $y=x^{2}$`  

15 \item<+(-1)-> $y=x^{3}$ % 相当于`\item<3> $y=x^{3}$`  

16 \end{enumerate}
17
18 \end{frame}
19
20 \end{document}
```

编译后得到的幻灯片如图9.16所示。

9.3 块与盒子——添加框元素

在幻灯片中框选文本或图片等元素是常见的操作，可以对幻灯片内容进行划分或者突出重点内容。在 Beamer 中，可以通过添加区块环境（block environments）或创建盒子（box）结构的方式将文本等元素放在各式各样的框中。

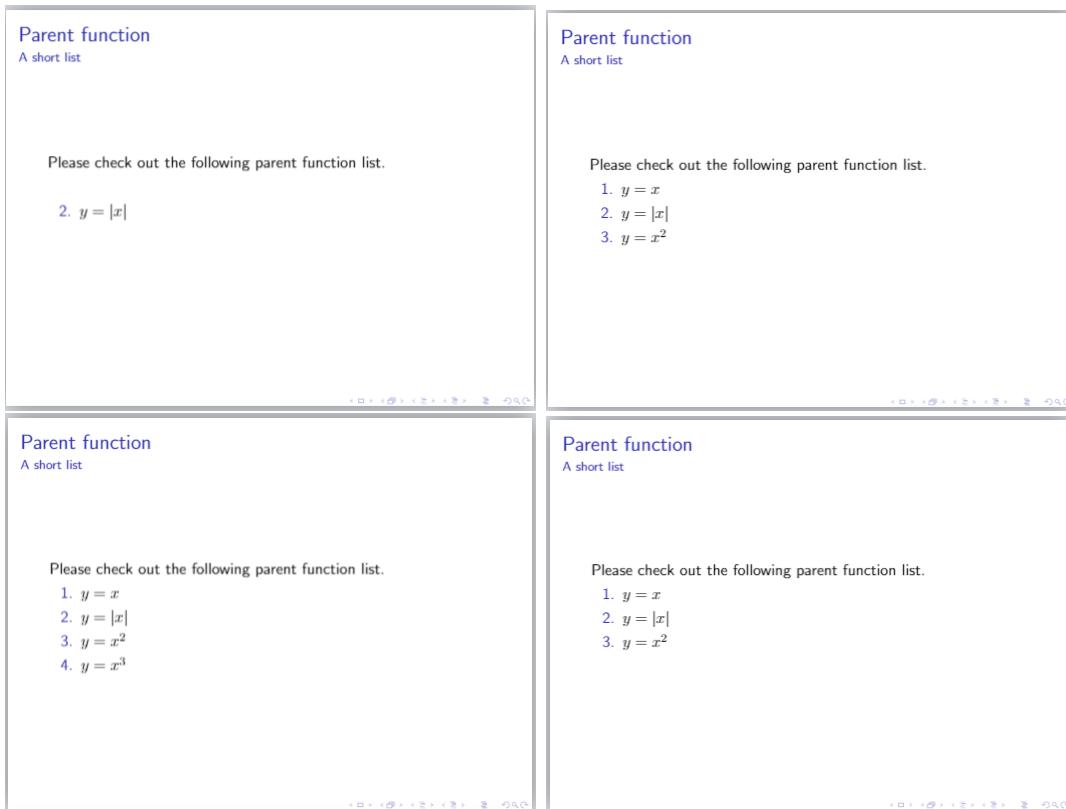


图 9.16: 编译后的幻灯片效果

9.3.1 区块环境

Beamer 提供了区块环境（block）可用于编辑文本内容，通过 block 环境创建的文本内容将放置在一个框中，使其与普通文本区分开。根据内容样式和使用目的的不同，包括三种区块环境：

- block：一般性区块环境。使用语法为`\begin{block}< 指定显示步骤 >{设置标题} \end{block}`；
- alertblock：警示性区块环境，主要用于创建警示信息。使用语法为`\begin{alertblock}< 指定显示步骤 >{设置标题} \end{alertblock}`；
- exampleblock：示例性区块环境，主要用于创建示例文本。使用语法为`\begin{exampleblock}< 指定显示步骤 >{设置标题} \end{exampleblock}`。

在三种区块环境的开始命令中（如：`\begin{block}< 指定显示步骤 >{设置标题}`），“<>”可用于指定当前区块内容显示的步骤，实现动画效果；第二个“{}”可用于设置该区块内容的标题，标题将显示在区块内容的上面。此外，区块内容的样式由使用的 Beamer 主题样式决定。

【例】在 beamer 文档类型中使用 block 环境插入一个一般文本框、使用 alertblock 环境插入一个警示性文本框、以及使用 exampleblock 环境插入一个示例性文本框：

```
1  \documentclass{beamer}
2  \usefonttheme{professionalfonts}
3  \usetheme{Copenhagen}
4
5  \begin{document}
6
7  \begin{frame}
8  \begin{block}<1>{Block1}
9  This is a generic block.
10 \end{block}
11
12 \begin{alertblock}<1>{Block2}
13 This is an alert block.
14 \end{alertblock}
15
16 \begin{exampleblock}<1>{Block3}
17 This is an example block.
18 \end{exampleblock}
```

```

19 \end{frame}
20
21 \end{document}

```

编译后得到的幻灯片如图9.17所示。

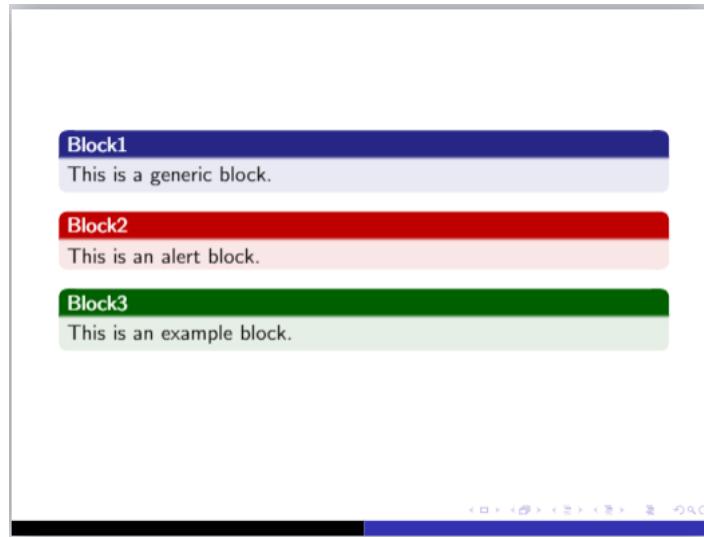


图 9.17: 编译后的幻灯片效果

9.3.2 定理类环境

对于定理、引理、推论、示例等定理类文本，除了可以考虑使用区块环境创建之外，Beamer 也预定义了相应的命令环境可供使用，包括：

- definition：定义环境。使用语法为\begin{definition}< 指定显示步骤 >\{设置名称\} \end{definition}；
- fact：事实环境。使用语法为\begin{fact}< 指定显示步骤 >\{设置名称\} \end{fact}；
- theorem：定理环境。使用语法为\begin{theorem}< 指定显示步骤 >\{设置名称\} \end{theorem}；
- lemma：引理环境。使用语法为\begin{lemma}< 指定显示步骤 >\{设置名称\} \end{lemma}；
- proof：证明环境。使用语法为\begin{proof}< 指定显示步骤 >\{设置名称\} \end{proof}；

- corollary：推论环境。使用语法为\begin{corollary}< 指定显示步骤 >\{设置名称\} \end{corollary}；
- example：示例环境等。使用语法为\begin{example}< 指定显示步骤 >\{设置名称\} \end{example}。

定理类环境的使用与区块环境类似：使用定理类环境可以创建文本框；开始命令中的“<>”可用于指定当前内容显示的步骤，实现动画效果；第二个“{}”可用于设置该定理类内容的名称。不同于区块环境的是：定理类内容的标题默认为对应的定理类型，如在 definition 环境下，标题即为“Definition”，显示在定理类内容的上方；而定理类内容的名称允许用户自行定制，通常位于定理类内容的左侧，以较大的斜体字标示。

【例】在 beamer 文档类型中使用 definition 环境插入一个定义文本框、使用 theorem 环境插入一个定理文本框、以及使用 example 环境插入一个示例文本框：

```
1  \documentclass{beamer}
2  \usefonttheme{professionalfonts}
3  \usetheme{Copenhagen}
4
5  \begin{document}
6
7  \begin{frame}{Definition, theorem and example}
8  \begin{definition}<1>{Definition Demo}
9  This is a definition.
10 \end{definition}
11
12 \begin{theorem}<1>{Theorem Demo}
13 This is a theorem.
14 \end{theorem}
15
16 \begin{example}<1>{Example Demo}
17 This is an example.
18 \end{example}
19 \end{frame}
20
21 \end{document}
```

编译后得到的幻灯片如图9.18所示。

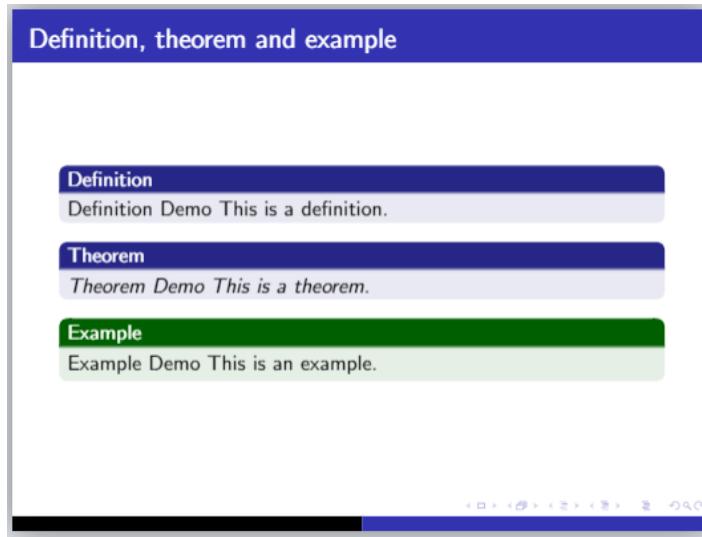


图 9.18: 编译后的幻灯片效果

9.3.3 Beamer 中的盒子

Beamer 也支持通过绘制外框的方式为幻灯片的元素（如，文本、图片等）加上外框，或者说创建盒子（box）。常用的语法包括调用`\fbox{}`命令绘制简单矩形框、或调用 fancybox 宏包提供的命令（`\shadowbox`, `\doublebox`, `\ovalbox` 和`\Ovalbox`）创建不同类型的外框。

使用 `fbox` 命令

使用`\fbox{}`命令可以创建简单的矩形盒子，调用以下命令可以对盒子参数进行修改：

- `\setlength{\fboxsep}{}`: 设置盒子内的元素与其边框之间的距离，默认值为 3pt；
- `\setlength{\fboxrule}{}`: 设置盒子边框线的粗细，默认值为 0.4pt。

此外，盒子之间的行间距可以使用`\vskip`命令进行修改。

【例】在 beamer 文档类型中使用 `fbox` 命令三个文本盒子、使用 `setlength` 命令设置不同参数、并使用 `vskip` 命令设置行间距：

```

1 \documentclass{beamer}
2
3 \begin{document}
4

```

```
5 \begin{frame}
6 \setlength{\fboxsep}{3pt}
7 \setlength{\fboxrule}{0.4pt}
8 \fbox{This is our 1st text box.}
9 \vskip 5mm
10 \setlength{\fboxsep}{6pt}
11 \setlength{\fboxrule}{0.8pt}
12 \fbox{This is our 2nd text box.}
13 \vskip 5mm
14 \setlength{\fboxsep}{9pt}
15 \setlength{\fboxrule}{1.2pt}
16 \fbox{This is our 3rd text box.}
17 \end{frame}
18
19 \end{document}
```

编译后得到的幻灯片如图9.19所示。

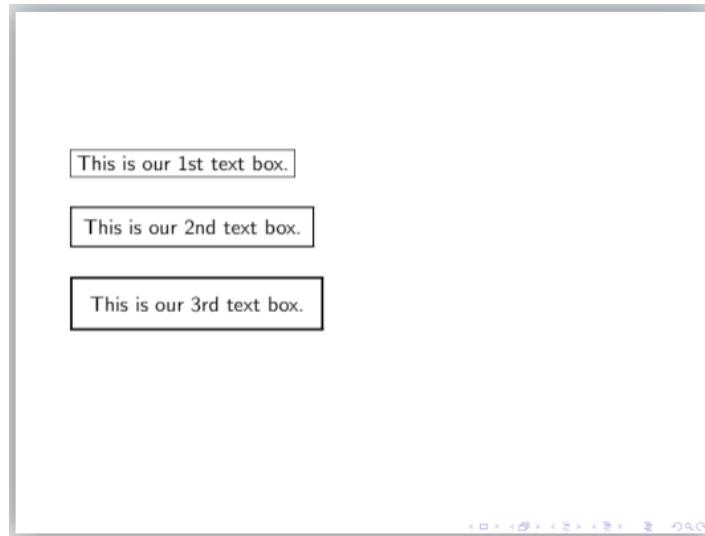


图 9.19: 编译后的幻灯片效果

对于较短的文本内容，使用\fbox{} 命令可以实现较好的效果。但由于在\fbox{} 命令中换行符\\不起作用，因此如果要对段落文本或长文本创建盒子，需要先将文本内容放置到段落环境中，然后再调用\fbox{} 命令。其中，\begin{minipage}[外部对齐方式][高度][内部对齐方式]{宽度}{内容} \end{minipage} 环境和\parbox[外部对齐][高度][内部对齐]{宽度}{内容} 命令是比较常用的处理段落的语法。

【例】在 beamer 文档类型中使用 fbox 命令和 minipage 环境创建段落文本盒子：

```

1 \documentclass{beamer}
2
3 \begin{document}
4
5 \begin{frame}
6
7 \fbox{
8 \begin{minipage}[c][1.8cm][t]{5cm}
9 {This is our paragraph text box. This is our paragraph text box. This is our
10    paragraph text box. This is our paragraph text box.}
11 \end{minipage}}
12
13 \end{frame}
14
15 \end{document}

```

编译后得到的幻灯片如图9.20所示。

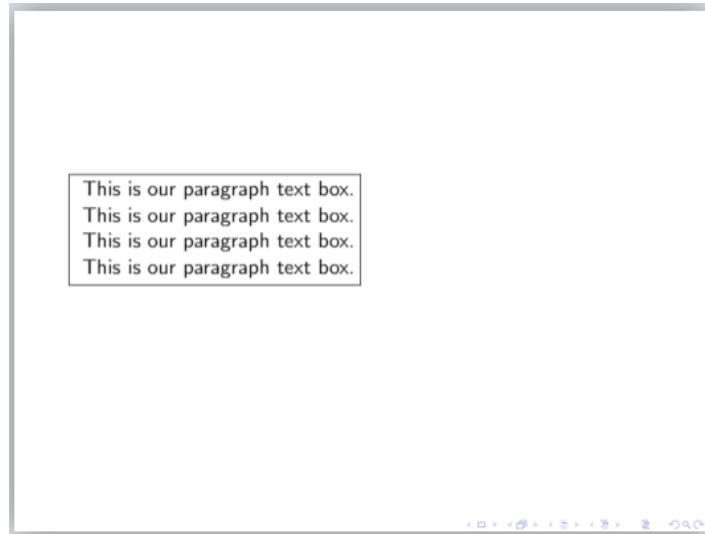


图 9.20: 编译后的幻灯片效果

类似地，也可以使用\parbox 命令处理段落文本，与 minipage 环境类似，该命令也能设置外部对齐方式、高度、内部对齐方式、以及宽度参数。用\parbox 命令改写上例代码，编译得到相同的结果，具体语句如下例所示：

【例】 在 beamer 文档类型中使用 fbox 命令和 parbox 命令创建段落文本盒子：

```
1 \documentclass{beamer}
2
3 \begin{document}
4
5 \begin{frame}
6
7 \fbox{
8 \parbox[c][1.8cm][t]{5cm}
9 {This is our paragraph text box. This is our paragraph text box. This is our
10 paragraph text box. This is our paragraph text box.}}
11 \end{frame}
12
13 \end{document}
```

当然，除了为文本内容创建盒子之外，`\fbox` 命令也能为图片等非文本内容创建盒子。

【例】在 beamer 文档类型中使用 `figure` 环境插入三张图片，并使用 `fbox` 命令将三种图片装入一个盒子中：

```
1 \documentclass{beamer}
2
3 \begin{document}
4
5 \begin{frame}
6
7 \begin{figure}
8 \centering
9 \fbox{
10 \includegraphics[width=0.2\linewidth]{redflower.png}
11 \includegraphics[width=0.2\linewidth]{yellowflower.png}
12 \includegraphics[width=0.2\linewidth]{blueflower.png}
13 }
14 \caption{Here is a figure box.}
15 \end{figure}
16
17 \end{frame}
18
19 \end{document}
```

编译后得到的幻灯片如图9.21所示。

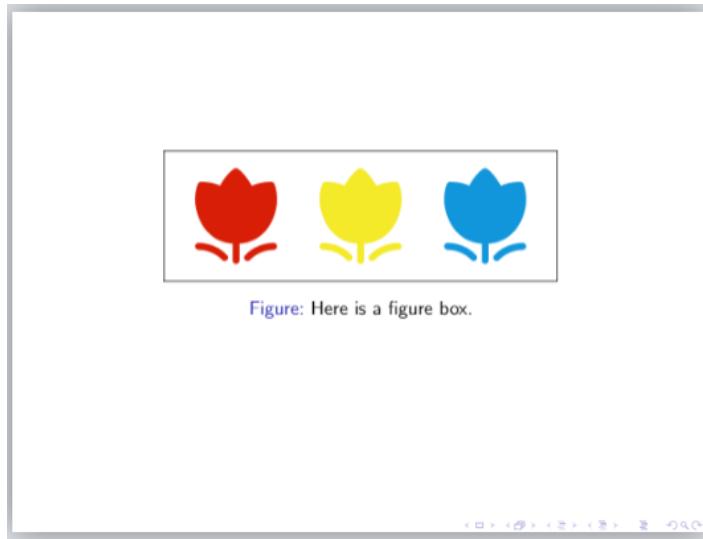


图 9.21: 编译后的幻灯片效果

调用 fancybox 宏包

在 fancybox 宏包中，提供了以下四个命令用来创建不同样式的盒子：

- \shadowbox{}： 创建阴影盒子；
- \doublebox{}： 创建两重线盒子；
- \ovalbox{}： 创建细边线椭圆盒子；
- \Ovalbox{}： 创建粗边线椭圆盒子。

【例】 在 beamer 文档类型中使用 shadowbox, doublebox, ovalbox 和 Ovalbox 命令创建不同样式的盒子：

```
1 \documentclass{beamer}
2 \usepackage{fancybox}
3 \begin{document}
4
5 \begin{frame}
6
7   \setlength{\fboxsep}{5pt}
8   \setlength{\fboxrule}{2pt}
9 
```

```
10 \shadowbox{This is a shadowbox.}  
11  
12 \vskip 5mm  
13  
14 \doublebox{This is a doublebox.}  
15  
16 \vskip 5mm  
17  
18 \ovalbox{This is an ovalbox.}  
19  
20 \vskip 5mm  
21  
22 \Ovalbox{This is an Ovalbox.}  
23  
24 \end{frame}  
25  
26 \end{document}
```

编译后得到的幻灯片如图9.22所示。

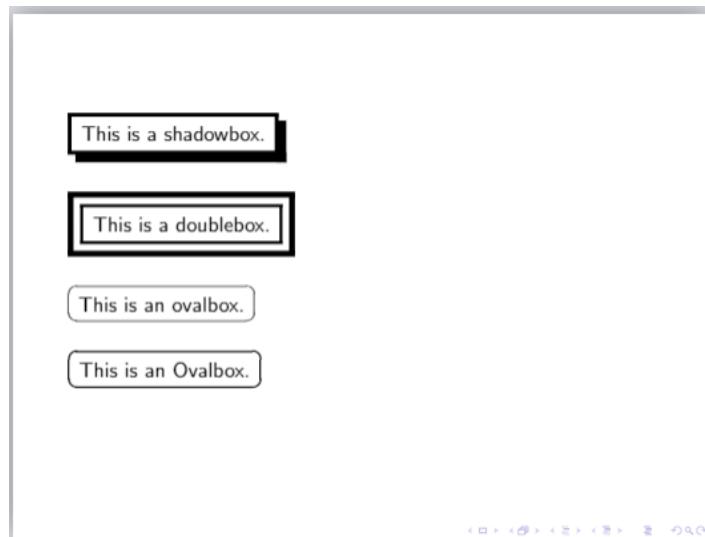


图 9.22: 编译后的幻灯片效果

类似地，也可以使用上述命令为图片等非文本元素创建不同样式的盒子。

【例】在 beamer 文档类型中使用 figure 环境插入四张图片，使用 shadowbox, doublebox, ovalbox 和 Ovalbox 命令分别为每张图片创建盒子，并使用 parbox 命令把图片和标题均包含在盒子中：

```
1 \documentclass{beamer}
2 \usepackage{fancybox}
3 \begin{document}
4
5 \setlength{\fboxsep}{5pt}
6 \setlength{\fboxrule}{2pt}
7
8 \begin{frame}
9 \begin{figure}
10 \centering
11 \shadowbox{
12 \parbox[c][6cm][t]{5cm}{
13 \includegraphics[width=1\linewidth]{redflower.png}
14 \caption{A red flower.}}}
15 \end{figure}
16 \end{frame}
17
18 \begin{frame}
19 \begin{figure}
20 \centering
21 \doublebox{
22 \parbox[c][6cm][t]{5cm}{
23 \includegraphics[width=1\linewidth]{yellowflower.png}
24 \caption{A yellow flower.}}}
25 \end{figure}
26 \end{frame}
27
28 \begin{frame}
29 \begin{figure}
30 \centering
31 \ovalbox{
32 \parbox[c][6cm][t]{5cm}{
33 \includegraphics[width=1\linewidth]{blueflower.png}
34 \caption{A blue flower.}}}
35 \end{figure}
36 \end{frame}
37
38 \begin{frame}
39 \begin{figure}
40 \centering
41 \ovalbox{
```

```
42 \parbox[c][6cm][t]{5cm}{  
43 \includegraphics[width=1\linewidth]{magentaflower.png}  
44 \caption{A magenta flower.}}}  
45 \end{figure}  
46 \end{frame}  
47  
48 \end{document}
```

编译后得到的幻灯片如图9.23所示。

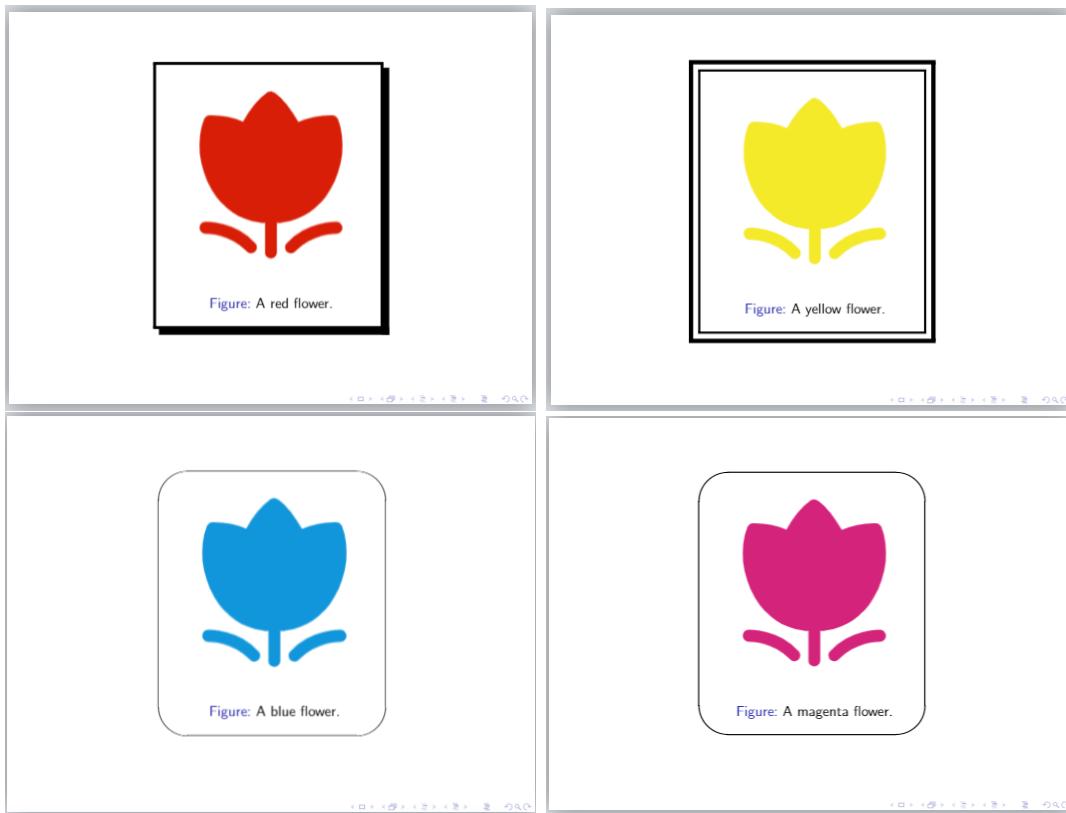


图 9.23: 编译后的幻灯片效果

9.4 设置主题样式

使用 Beamer 制作幻灯片的一道特色就是有现成的主题样式可供选择和直接使用，其中，主题样式对于幻灯片的演示效果而言十分重要，简言之，主题样式就是幻灯片的“外观”，改变幻灯片最简单的方式就是变换不同的主题样式。Beamer 中提供的每种主题样式都具有良好的可用性和可读性，这也使得 Beamer 制作出来的幻灯片

看起来十分专业，同时，反复使用的难度也不大。

在英文中，主题对应的英文单词为 theme。狭义来看，幻灯片主题是指幻灯片的主题样式；但从广义来看，其实幻灯片主题包括了包括主题样式、颜色主题、字体主题、内部主题、外部主题。

9.4.1 基本介绍

使用 Beamer 制作幻灯片时，我们可以选择很多已经封装好的幻灯片主题样式，不同样式可以达到不同的视觉效果。其实，使用这些主题样式的方法非常简单。通常来说，在前导代码中插入 \usepackage{theme} 命令即可，例如使用 Copenhagen（哥本哈根主题样式）只需要在前导代码中申明 \usepackage{Copenhagen}，这种方式调用主题样式是非常省事。



图 9.24: Beamer 文档类型中的主题样式

在 Beamer 文档类型中，有几十种主题样式可供选择和使用，比较常用的主题样式包括以下这些：

- Berlin：柏林主题样式，默认样式为蓝色调；
- Copenhagen：哥本哈根主题样式，默认样式为蓝色调；
- CambridgeUS：美国剑桥主题样式，默认样式为红色调；
- Berkeley：伯克利主题样式，默认样式为蓝色调；
- Singapore：新加坡主题样式；
- Warsaw：默认样式为蓝色调。

【例】 在 beamer 文档类型中使用 CambridgeUS 主题样式制作一个简单的幻灯片：

```
1 \documentclass{beamer}
2 \usepackage{CambridgeUS}
3
4 \begin{document}
5
6 \begin{frame}{Example}
7
8 This is a simple example for the CambridgeUS theme.
9
10 \end{frame}
11
12 \end{document}
```

编译上述代码，得到幻灯片如图9.25所示。

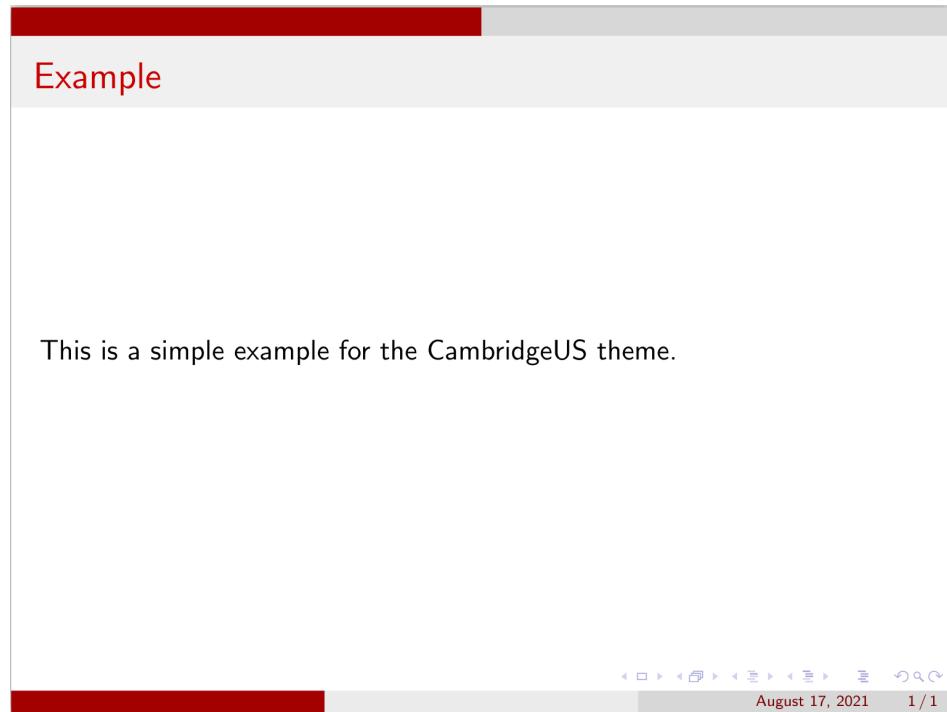


图 9.25: 编译后的幻灯片效果

当然，在这些主题样式基础上，我们也能够使用一些特定的主题样式如颜色主题、字体主题、内部主题、外部主题对幻灯片的显示效果进行调整。



图 9.26: Beamer 文档类型中的其他几种主题设置

9.4.2 颜色主题

使用 Beamer 制作幻灯片时，我们能够自行设置幻灯片主题样式的色调，使用`\usecolortheme{}`命令即可，这些色调包括 beetle、beaver、orchid、whale、dolphin 等。这里的色调又被称为颜色主题，它定义了幻灯片各部分的颜色搭配，设置特定的颜色主题后，我们能够得到不同的组合样式，具体可参考 <https://hartwork.org/beamer-theme-matrix/> 网站提供的组合样式矩阵。

【例】在 beamer 文档类型中使用 CambridgeUS 主题样式和 dolphin 色调制作一个简单的幻灯片：

```
1 \documentclass{beamer}
2 \usetheme{CambridgeUS}
3 \usecolortheme{dolphin}
4
5 \begin{document}
6
7 \begin{frame}{Example}
8
9 This is a simple example for the CambridgeUS theme with dolphin (color theme).
10
11 \end{frame}
12
13 \end{document}
```

编译上述代码，得到幻灯片如图9.27所示。

9.4.3 字体主题

实际上，对于幻灯片的文本字体，我们可以调用字体样式对其进行调整。在 Beamer 中，字体样式被称为字体主题，它定义了幻灯片的字体搭配。具体使用方法是：在前导代码中要用到的命令为`\usefonttheme{A}`，位置 A 填写的一般是字体类型，例如 serif。

【例】使用 beamer 文档类型创建一个简单的幻灯片，并在前导代码中申明使用 serif 对应的字体样式：

```
1 \documentclass{beamer}
2 \usefonttheme{serif}
3
```

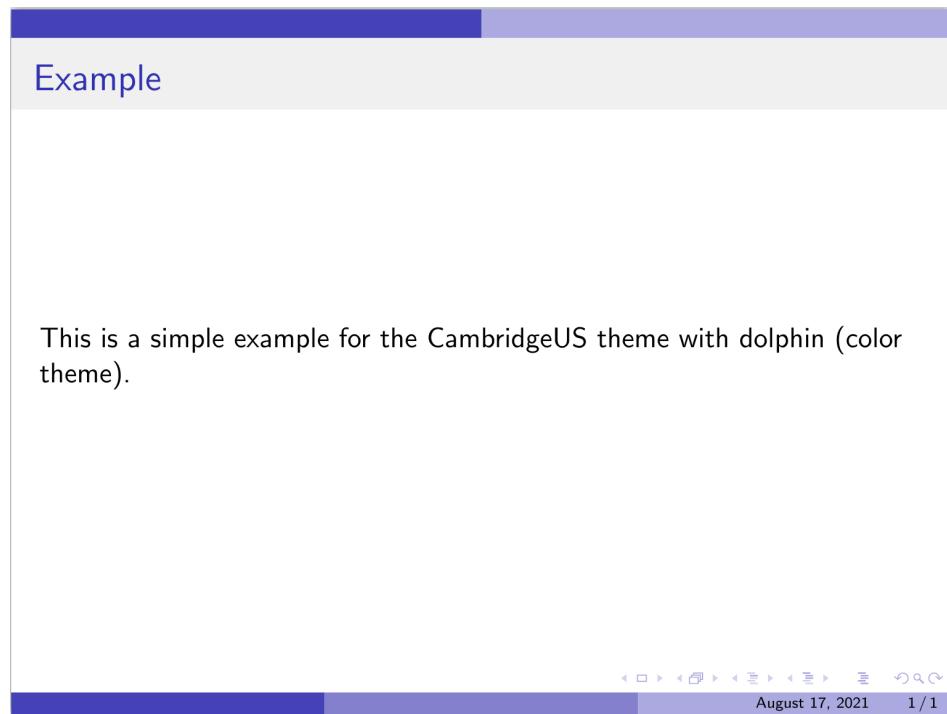


图 9.27: 编译后的幻灯片效果

```

4 \begin{document}
5
6 \begin{frame}
7
8 This is a simple example for using \alert{serif} font theme.
9
10 \end{frame}
11
12 \end{document}

```

编译上述代码，得到幻灯片如图9.28所示。

我们知道：在常规文档中，可以使用各种字体对应的宏包达到调用字体的作用，使用规则为\usepackage{A}，位置 A 填写的一般是字体类型，包括 serif、avant、bookman、chancery、charter、euler、helvet、mathtime、mathptm、mathptmx、newcent、palatino、pifont、utopia 等。

【例】 使用 beamer 文档类型创建一个简单的幻灯片，并在前导代码中申明使用字体 palatino 对应的宏包：

```

1 \documentclass{beamer}

```

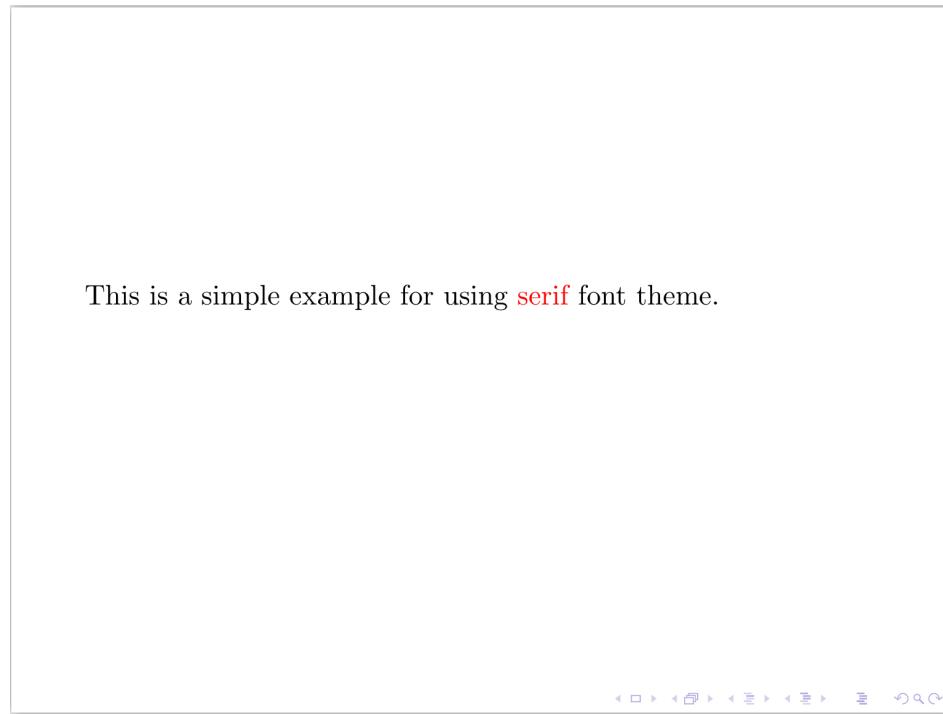


图 9.28: 编译后的幻灯片效果

```
2 \usepackage{palatino}
3
4 \begin{document}
5
6 \begin{frame}
7
8 This is a simple example for using \alert{palatino} font.
9
10 \end{frame}
11
12 \end{document}
```

编译上述代码，得到幻灯片如图9.29所示。

9.4.4 内部主题

内部主题定义了幻灯片展示区域的样式，如列表、定理等，内部主题不包括页眉、页脚、导航栏等部分。每一种主题样式都有默认的内部主题，更换内部主题需使用`\useinnertheme{A}` 命令，位置 A 可供选择的内部主题包括 `circles`、`rectangles`、`rounded` 和 `inmargin`。

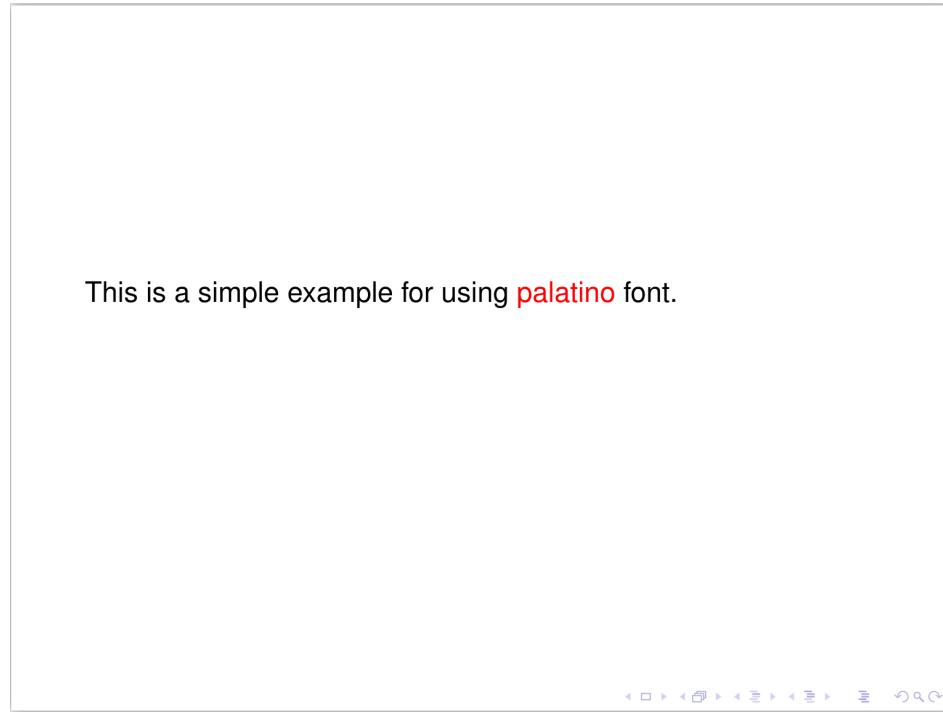


图 9.29: 编译后的幻灯片效果

【例】在 beamer 文档类型中分别使用 circles 内部主题制作幻灯片：

```
1 \documentclass{beamer}
2 \usepackage{CambridgeUS}
3 \usefonttheme{professionalfonts}
4 \useinnertheme{circles}
5
6 \begin{document}
7
8 \begin{frame}
9 \frametitle{Parent function}
10 \framesubtitle{A short list}
11
12 Please check out the following parent function list.
13 \begin{enumerate}
14 \item $y=x$ 
15 \item $y=|x|$
16 \item $y=x^{2}$
17 \item $y=x^{3}$
18 \item $y=x^{\{b\}}$ 
19 \end{enumerate}
```

```
20  
21     \end{frame}  
22  
23 \end{document}
```

编译上述代码，得到幻灯片如图9.30所示。

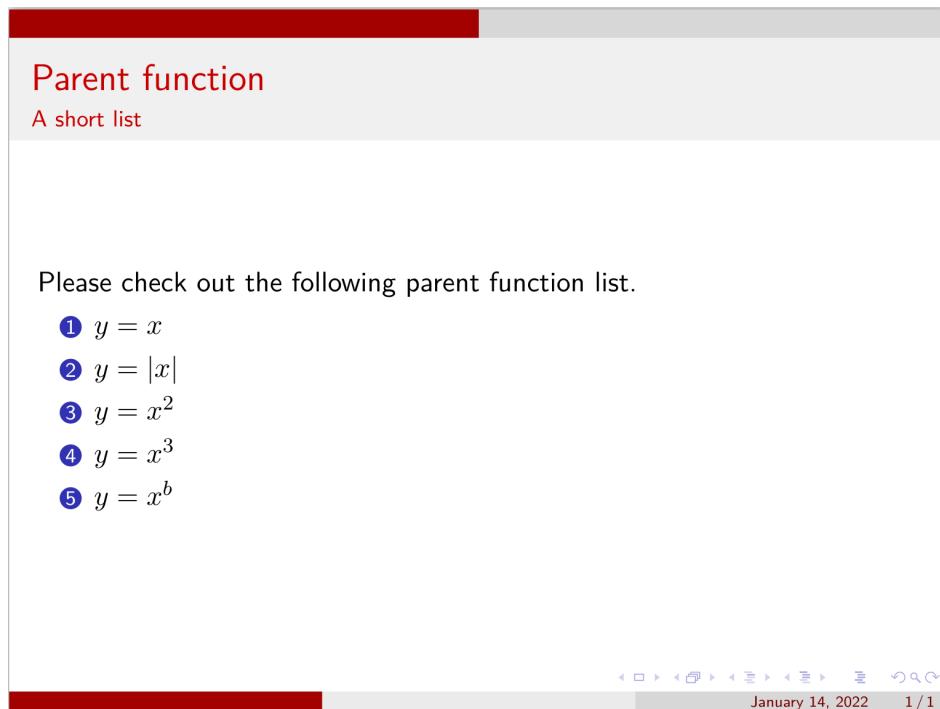


图 9.30: 编译后的幻灯片效果

【例】在 beamer 文档类型中分别使用 inmargin 内部主题制作幻灯片：

```
1 \documentclass{beamer}  
2 \usepackage{CambridgeUS}  
3 \usefonttheme{professionalfonts}  
4 \useinnertheme{inmargin}  
5  
6 \begin{document}  
7  
8 \begin{frame}  
9 \frametitle{Parent function}  
10 \framesubtitle{A short list}  
11  
12 Please check out the following parent function list.
```

```

13 \begin{enumerate}
14   \item $y=x$ 
15   \item $y=|x|$
16   \item $y=x^2$ 
17   \item $y=x^3$ 
18   \item $y=x^b$ 
19 \end{enumerate}

20
21 \end{frame}

22
23 \end{document}

```

编译上述代码，得到幻灯片如图9.31所示。

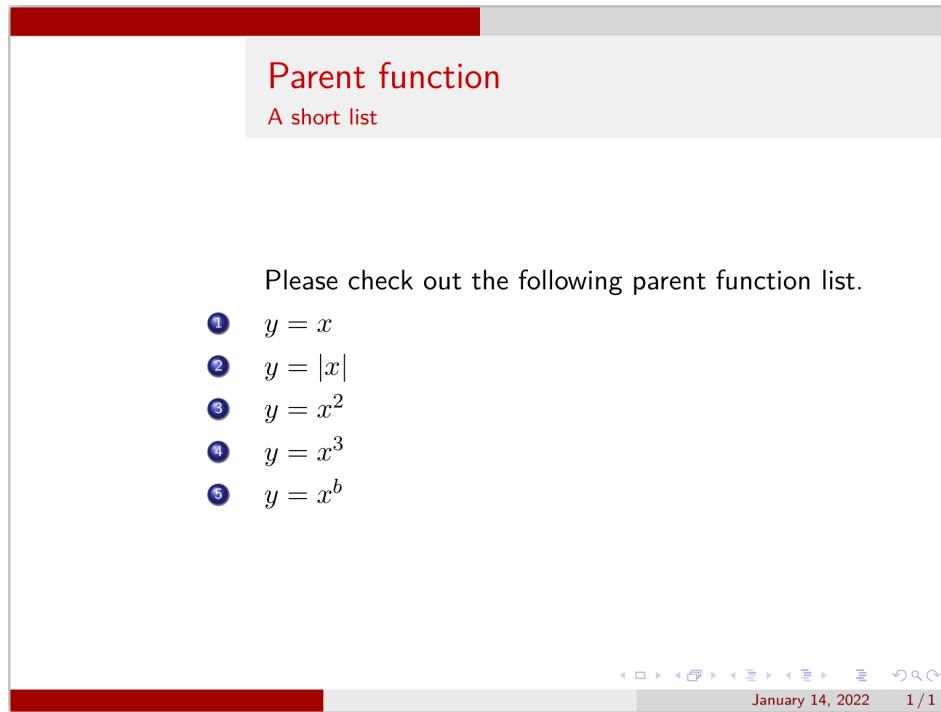


图 9.31: 编译后的幻灯片效果

9.4.5 外部主题

外部主题定义了幻灯片的边框、页眉、页脚、导航栏等部分的样式。更换外部主题需使用`\useoutertheme{A}`，位置 A 可供选择的外部主题包括 infolines、smoothbars、sidebar、split 和 tree。

9.4.6 表格字体大小

在 Beamer 中制作表格，当我们想对表头或者表格内容文字大小进行调整时，可以使用在前导代码中申明使用 *caption* 宏包，即`\usepackage{caption}`，然后设置具体的字体大小即可，如`\captionsetup{font = scriptsize, labelfont = scriptsize}`可以将表头和表格内容字体大小调整为 *scriptsize*。

【例】使用 *table* 环境创建一个简单表格，并使用 *caption* 宏包将表头字体大小设置为 *Large*、将表格内容字体大小设置为 *large*：

```
1 \documentclass{beamer}
2 \usepackage{booktabs}
3 \usepackage{caption}
4 \captionsetup{font = large, labelfont = Large}
5
6 \begin{document}
7
8 \begin{frame}
9
10 \begin{table}
11 \caption{A simple table.}
12 \begin{tabular}{l|ccc}
13 \toprule
14 & \textbf{header3} & \textbf{header4} & \textbf{header5} \\
15 \midrule
16 \textbf{header1} & cell1 & cell2 & cell3 \\
17 \midrule
18 \textbf{header2} & cell4 & cell5 & cell6 \\
19 \bottomrule
20 \end{tabular}
21 \end{table}
22
23 \end{frame}
24
25 \end{document}
```

编译上述代码，得到幻灯片如图9.32所示。

其中，单就设置表头字体大小而言，除了使用 *caption* 宏包之外，还可以通过对幻灯片设置全局参数达到调整字体大小的效果，例如`\setbeamertemplate{caption}{size = \Large}`。

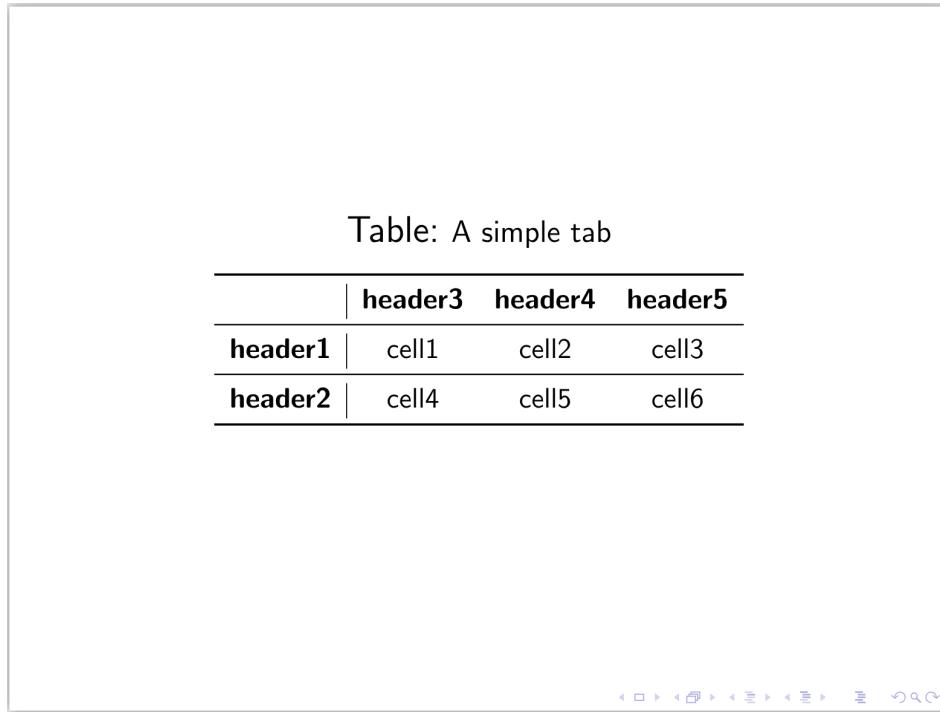


图 9.32: 编译后的幻灯片效果

9.4.7 样式调整

在 Beamer 文档类型中，除了可以使用各种主题样式，另外也可以根据幻灯片组成部分，分别对侧边栏、导航栏以及 Logo 等进行调整。其中，侧边栏是由所选幻灯片主题样式自动生成的，主要用于显示幻灯片目录。有时为了显示幻灯片的层次，使用侧边栏进行目录索引。

【例】 使用 Berkeley 主题样式，并将侧边栏显示在右侧：

```

1 \documentclass{beamer}
2 \PassOptionsToPackage{right}{beamerouterthemesidebar}
3 \usetheme{Berkeley}
4 \usefonttheme{professionalfonts}
5
6 \begin{document}
7
8 \begin{frame}
9 \frametitle{Parent function}
10 \framesubtitle{A short list}
11
12 Please check out the following parent function list.

```

```
13 \begin{enumerate}
14     \item $y=x$ 
15     \item $y=|x|$
16     \item $y=x^2$ 
17     \item $y=x^3$ 
18     \item $y=x^b$ 
19 \end{enumerate}
20
21 \end{frame}
22
23 \end{document}
```

编译上述代码，得到幻灯片如图9.33所示。

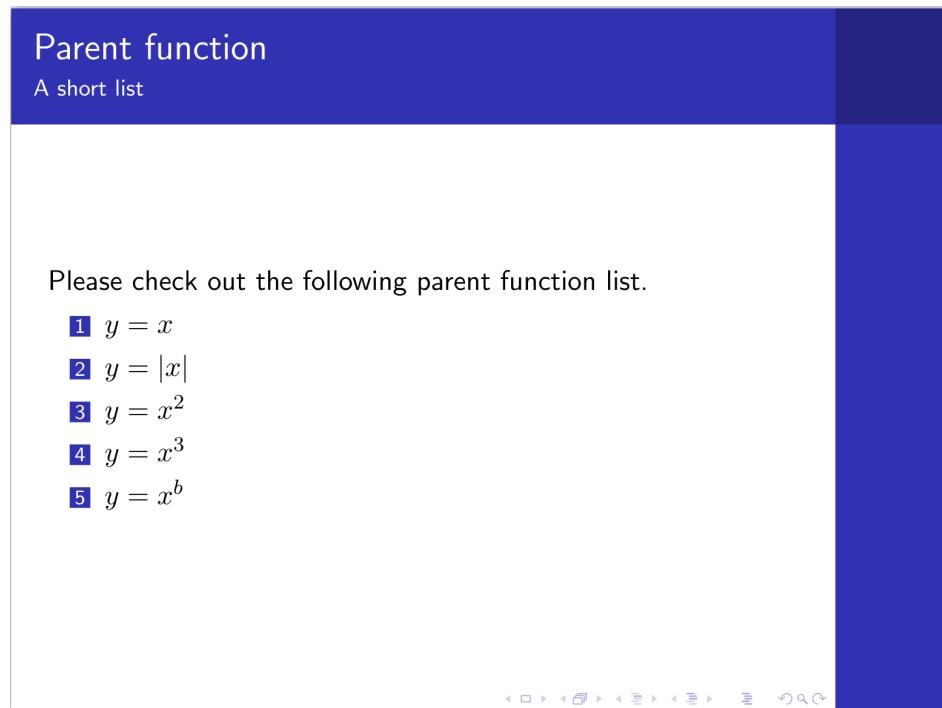


图 9.33: 编译后的幻灯片效果

很多时候我们会发现，在各类学术汇报中，幻灯片的首页通常会有主讲人所在的研究机构 Logo。在 Beamer 文档类型中，有`\logo` 和`\titlegraphic` 两个命令可供使用，使用`\logo` 命令添加的 Logo 会在每一页幻灯片中都显示，而使用`\titlegraphic` 命令添加的 Logo 只出现在标题页。

【例】 使用`\logo` 命令在幻灯片中添加 Logo：

```

1  \documentclass{beamer}
2  \usefonttheme{professionalfonts}
3
4  \title{A Simple Beamer Example}
5  \author{Author's Name}
6  \institute{Author's Institute}
7
8  \logo{\includegraphics[width=2cm]{logopolito}}
9
10 \begin{document}
11
12 \begin{frame}
13 \titlepage
14 \end{frame}
15
16 \begin{frame}{Parent function}{A short list}
17 Please check out the following parent function list.
18 \begin{enumerate}
19 \item $y=x$ 
20 \item $y=|x|$
21 \item $y=x^2$ 
22 \item $y=x^3$ 
23 \item $y=x^b$ 
24 \end{enumerate}
25 \end{frame}
26
27 \end{document}

```

编译上述代码，得到幻灯片如图9.34所示。

【例】 使用 titlegraphic 命令在幻灯片的标题页添加 Logo：

```

1  \documentclass{beamer}
2  \usefonttheme{professionalfonts}
3
4  \title{A Simple Beamer Example}
5  \author{Author's Name}
6  \institute{Author's Institute}
7
8  \titlegraphic{\includegraphics[width=2cm]{logopolito}\hspace*{4.75cm}~\includegraphics[width=2cm]{logopolito}}
9

```

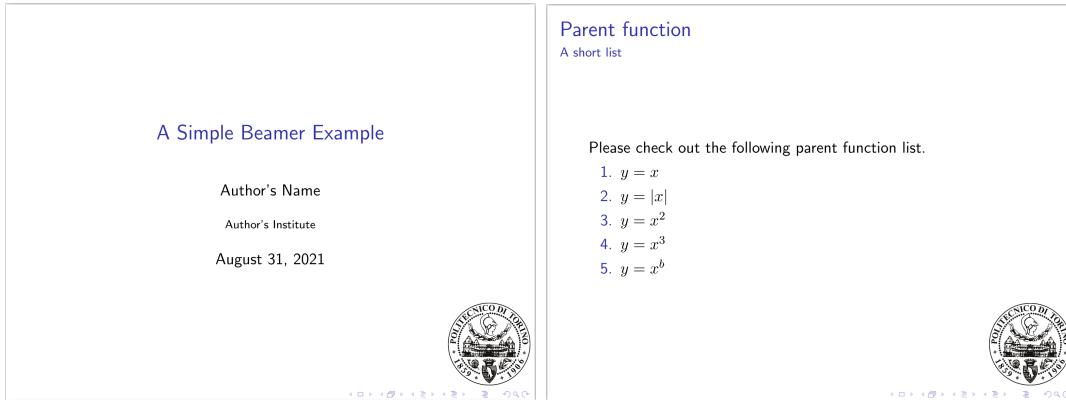


图 9.34: 编译后的幻灯片效果

```
10      }
11
12 \begin{document}
13
14 \begin{frame}
15 \titlepage
16 \end{frame}
17
18 \begin{frame}{Parent function}{A short list}
19 Please check out the following parent function list.
20 \begin{enumerate}
21 \item $y=x$%
22 \item $y=|x|$%
23 \item $y=x^{2}$%
24 \item $y=x^{3}$%
25 \item $y=x^b$%
26 \end{enumerate}
27 \end{frame}
28
29 \end{document}
```

编译上述代码，得到幻灯片如图9.35所示。

A Simple Beamer Example

Author's Name

Author's Institute

August 31, 2021



Parent function

A short list

Please check out the following parent function list.

1. $y = x$
2. $y = |x|$
3. $y = x^2$
4. $y = x^3$
5. $y = x^b$



图 9.35: 编译后的幻灯片效果