

▼ 基本概念

- 概述
- 命令与环境
- LaTeX源代码结构
- 文档类
- 宏包
- 导言区
- 文件组织方式

▼ 用Latex排版文字

- 排版中文
- ▼ Latex中的字符
 - 空格和分段
 - 特殊字符
 - 连字
- ▼ 标点符号
 - 引号
 - 连字号和破折号
 - 省略号
 - 波浪号
 - 其他符号

▼ 断行和断页

- 单词间距
- 手动断行和断页
- 断词

▼ 文档元素

- ### ▼ 章节和目录
- 章节标题
 - 目录
 - 文档结构的划分
- ### ▪ 标题页
- ### ▪ 交叉引用
- ### ▪ 脚注和边注
- ### ▼ 特殊环境
- 列表
 - 对齐环境
 - 引用环境
 - 摘要环境
 - 代码环境

▼ 表格

- 列格式
- 列宽
- 横线
- 合并单元格

- 嵌套表格
- 行距控制
- 图片
- ▼ 盒子
 - 水平盒子
 - 带框的水平盒子
 - 垂直盒子
 - 标尺盒子
- ▼ 浮动体
 - 浮动体的标题
 - 并排和子图表
- ▼ 排版数学公式
 - AMS宏集
 - ▼ 公式排版基础
 - 行内和行间公式
 - 数学模式
 - ▼ 数学符号
 - 指数、上下标和导数
 - 分式和根式
 - 关系符
 - 算符
 - 巨算符
 - 数学重音和上下括号
 - 箭头
 - 括号和定界符
 - ▼ 多行公式
 - 长公式折行
 - 公用编号的多行公式
 - 数组和矩阵
 - 公式中的间距
 - ▼ 数学符号的字体控制
 - 数学字母字体
 - 加粗的数学符号
 - 数学符号的尺寸
 - ▼ 定理环境
 - `latex`原始的定理环境
 - `amsthm`宏包
 - 证明环境和证毕符号
 - ▼ 符号表
 - `latex`普通符号
 - AMS符号
- ▼ 排版样式设定
 - ▼ 字体和字号
 - `latex`字体字号

- 字体宏包
 - 使用*unicode-math*宏包配置Unicode数学字体

- 文字装饰和强调

- ▼ 段落格式和间距

- 长度和长度变量
- 行距
- 段落格式
- 水平间距
- 垂直间距

- ▼ 页面和分栏

- 利用*geometry*宏包设置页面参数
- 页面内容的垂直对齐
- 分栏

- ▼ 页眉页脚

- 基本的页眉页脚样式
- 手动更改页眉页脚的内容
- *fancyhdr*宏包

- ▼ 特色工具和功能

- ▼ 参考文献和**bibtex**工具

- 基本的参考文献和引用
- *bibtex*数据库
- *bibtex*样
- 使用**bibtex**排版参考文献
- *natbib*宏包

- ▼ *biblatex*宏包

- 文档结构和*biblatex*相关命令
- 编译方式
- *biblatex*样式和其他选项

- ▼ 索引和 *makeindex* 工具

- 使用 *makeindex* 工具的方法
- 索引项的写法

- ▼ 使用颜色

- 颜色的表达方式
- 带颜色的文本和盒子

- ▼ 使用超链接

- *hyperref*宏包
- 超链接
- PDF书签
- PDF文档属性

- ▼ 绘图功能

- 绘图语言简介

- ▼ *TikZ*绘图语言

- *TikZ*和路径
- *TikZ*绘图命令和参数

- [TikZ文字结点](#)
- [在TikZ中使用循环](#)
- ▼ [自定义 latex 命令和功能](#)
 - ▼ [自定义命令和环境](#)
 - [定义新命令](#)
 - [定义环境](#)
 - [xparse宏包](#)
 - ▼ [编写自己的宏包和文档类](#)
 - [编写简单的宏包](#)
 - [在宏包中调用其它宏包](#)
 - [编写自己的文档类](#)
 - ▼ [计数器](#)
 - [定义和修改计数器](#)
 - [计数器的输出格式](#)
 - ▼ [latex中的计数器](#)
 - [secnumdepth](#)
 - [tocdepth](#)
 - [latex可定制的一些命令和参数](#)

基本概念

概述

经典hello world

```
% LaTeX的基本概念
% 最简短的Latex源代码示例
% Ltex: language=en
\documentclass{article}
\begin{document}
``Hello, world!'' from \LaTeX.
\end{document}
```

中文版

```
% LaTeX在排版中文时的一个最简示例
% Ltex: language=zh
\documentclass{ctexart}
\begin{document}
“你好，世界！”来自 \LaTeX{} 的问候。
\end{document}
```

命令与环境

LaTeX中命令以反斜线\开头，有

1. 跟着字母 如 \LaTeX。以任意非字母符号为界限
2. 跟着单个非字母符号 如 \\$

字母形式的LaTeX命令忽略其后的所有连续空格 可以使用 {} 或 \(\) 使命令后出现空格：

```
Shall we call ourselves \TeX users or \TeX{} users?
```

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

命令分为 [可选参数] 和 {必选参数}，带 * 的命令效果有一定差异，可以粗略视为一种特殊的可选参数。
以单个字符作为命令的参数时 可以不加括号，例如 \frac{1}{2} 与 \frac{1}{2}

环境用于部分效果在局部生效 用法为一对命令 \begin{...} 和 \end{...} :

```
\begin{<environment name>}[<optional arguments>]{<mandatory arguments>}  
...  
\end{<environment name>}
```

部分环境允许嵌套使用

有些命令(如 \bfseries)会对其后所有内容产生作用 在分组中使用的命令被限制在分组内，使用一对花括号 { 和 } 作为分组 环境隐含了一个分组 修改字体和字号的命令用法属于此类

个别命令在分组内仍会产生全局作用 例如 \setcounter 等命令

LaTeX源代码结构

```
\documentclass{...} % ...指定文档使用的**文档类**  
% 导言区  
% 可以使用 \usepackage 命令调用**宏包**  
% 或进行文档的全局设置  
\begin{document} % 文档正文环境  
% 正文内容  
\end{document} % 文档结束  
% 此后内容会被忽略
```

文档类

```
\documentclass[<options>]{<class-name>}
```

其中 <class-name> 为文档类的名称 如 latex 提供的

- article 文章格式的文档类
广泛用于科技论文 报告 说明文档等 页码在页底
- report 长篇报告的文档类
具有章节结构 用于综述 长篇论文 简单的书籍等 页码在页底
- book 书籍文档类
包含章节结构和前言 正文 后记等结构 页码在页顶外侧
- proc 基于article文档类的一个简单的学术文档模板
页码在页底外侧
- slides 幻灯格式的文档类
使用无衬线字体 无页码
- minimal 一个极其精简的文档类
只设定了纸张大小和基本字号 用作代码测试的最小工作示例 无页码
- 基于以上还派生出一些排版如支持中文的ctexart ctexrep ctexbook 或其他文档moderncv beamer等

可选参数 <options> 为文档类指定选项 以全局的规定一些排版的参数
如字号 纸张大小 单双面等等 例如调用 article 文档类排版文章 指定纸质为 A4 大小 基本字号为11pt 双面排版：

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

常用选项：

- 10pt 11pt 12pt 基本字号 默认为 10pt
- a4paper letterpaper ... 指定纸张大小 默认为美式信纸letterpaper(8.5in 21.6cm, 11in 28.0cm)
还有a5paper b5paper executivepaper legalpaper
- twoside oneside 指定单面/双面排版 双面排版时 奇偶页的页眉页脚页边距不同
article和report默认oneside book默认twoside
- onecolumn twocolumn 指定单栏/双栏排版 默认onecolumn
- openright openany 指定新的一章\chapter是否总是在奇数页(右侧right)或紧跟上一页(any)开始
report默认openany book默认openright article无效
- titlepage notitlepage 指定标题命令\maketitle是否生成单独的标题页
article默认notitlepage report和book默认titlepage
- fleqn 令行间公式左对齐 默认为居中对齐
- leqno 令行间公式编号在左侧 默认为右侧
- draft final 指定草稿/终稿模式
草稿模式下 断行不良(溢出)的地方会在行尾添加一个黑色方块 默认为final

宏包

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

宏包用于扩展LaTeX的功能 可以一次性调用多个宏包 <package-name> 中用逗号隔开,例如

```
\usepackage{tabulary, makecell, multirow}
```

下面列出了常用的宏包。

B.3.1 文字、公式和符号

<code>amsmath</code>	\mathcal{AM} S 数学公式扩展。
<code>mathtools</code>	数学公式扩展宏包，提供了公式编号定制和更多的符号、矩阵等。
<code>amsfonts</code>	\mathcal{AM} S 扩展符号的基础字体支持。
<code>amssymb</code>	在 <code>amsfonts</code> 基础上将 \mathcal{AM} S 扩展符号定义成命令。
<code>bm</code>	提供将数学符号加粗的命令 <code>\bm</code> 。
<code>unicode-math</code>	使用 Unicode 数学字体。
<code>nicematrix</code>	排版复杂矩阵。
<code>siunitx</code>	以国际单位规范排版物理量的单位。
<code>mhchem</code>	排版化学式和方程式。
<code>tipa</code>	排版国际音标。

B.3.2 排版元素

<code>ulem</code>	提供排版可断行下划线的命令 <code>\uline</code> 以及其它装饰文字的命令。
<code>endnote</code>	排版尾注。
<code>marginnote</code>	改善的边注排版功能。
<code>multicol</code>	提供将内容自由分栏的 <code>multcols</code> 环境。
<code>multitoc</code>	生成多栏排版的目录。
<code>minitoc</code>	为章节生成独立的小目录。
<code>glossaries</code>	生成词汇表。
<code>verbatim</code>	对原始的 <code>verbatim</code> 环境的改善。提供了命令 <code>\verb+input</code> 调用源文件。
<code>fancyvrb</code>	提供了代码排版环境 <code>Verbatim</code> 以及对版式的自定义。
<code>listings</code>	提供了排版关键字高亮的代码环境 <code>lstlisting</code> 以及对版式的自定义。类似宏包有 <code>minted</code> 等。
<code>algorithmic</code>	一个简单的实现算法排版的宏包。如果要生成浮动体的话，需要搭配 <code>algorithm</code> 宏包使用。
<code>algorithm2e</code>	较为复杂的、可定制的算法排版宏包。类似宏包有 <code>algorithmicx</code> 等。
<code>amsthm</code>	定制定理环境。类似宏包包括 <code>theorem</code> 、 <code>ntheorem</code> 、 <code>thmtools</code> 等。
<code>mdframed</code>	排版可自动断页的带边框文字段落，提供边框样式的定制功能。
<code>tcolorbox</code>	以 <code>TikZ</code> 为基础提供排版样式丰富的彩色盒子的功能。

B.3.3 图表和浮动体

array	对表格列格式的扩展。
booktabs	排版三线表。
tabularx	提供 <code>tabularx</code> 环境排版定宽表格，支持自动计算宽度的 X 列格式。
arydshln	支持排版虚线表格线。
colortbl	支持修改表格的行、列、单元格的颜色。
multirow	支持合并多行单元格。
makecell	支持在单元格里排版多行内容（嵌套一个单列的小表格）。
diagbox	排版斜线表头。
longtable	提供排版跨页长表格的 <code>longtable</code> 环境。
ltxtable	为跨页长表格提供 <code>tabularx</code> 的 X 列格式。
tabulararray	排版复杂表格（基于 L ^A T _E X3 实现）。
graphicx	支持插图。
bmpsize	<code>latex + dvipdfmx</code> 命令下支持 BMP/JPG/PNG 等格式的位图。
epstopdf	<code>pdflatex</code> 命令下支持 EPS 格式的矢量图。
wrapfig	支持简单的文字在图片周围的绕排。
caption	控制浮动体标题的格式。类似宏包有 <code>keyfloat</code> 等。
subcaption	提供子图表和子标题的排版。类似宏包有 <code>subfigure</code> 和 <code>subfig</code> 等。
bicaption	生成双语浮动体标题。
float	为浮动体提供不浮动的 H 模式；提供自定义浮动体结构的功能。

B.3.4 修改版式

geometry	修改页面尺寸、页边距、页眉页脚等参数。
fancyhdr	修改页眉页脚格式，令页眉页脚可以左对齐、居中、右对齐。
titlesec	修改章节标题 <code>\chapter</code> 、 <code>\section</code> 等的格式。
titletoc	修改目录中各条目的格式。类似宏包有 <code>tocloft</code> 等。
tocbibind	支持将目录、参考文献、索引本身写入目录项。
footmisc	修改脚注 <code>\footnote</code> 的格式。
indentfirst	令章节标题后的第一段首行缩进。
enumerate	提供简单的自定义标签格式的 <code>enumerate</code> 环境。
enumitem	修改列表环境 <code>enumerate</code> 和 <code>itemize</code> 等的格式。
lettrine	生成段落首字母大写的效果。

导言区

导言区用于载入宏包和进行全局设置，在 `\begin{document}` 前面，`\end{document}` 后面

文件组织方式

当编写长篇文档例如书籍或毕业论文时，单个源文件修改校对较困难，可以将源文件分割成若干个文件 例如每章内容单独写在一个文件里 通过插入文件的方式组织文档：

使用 `\input{<filename>}` 或 `\include{<filename>}`， `\include` 会生成新的一页， `\input` 则纯粹是将文件内的内容插入。

其中 `<filename>` 为文件名 不带.tex拓展名 不在同一目录需要加上相对或绝对路径 用正斜线 / 连接

导言区内容较多时 可以将其单独放置在一个`package.tex`文件中 再用 `\input` 命令插入

latex还提供了 `\includeonly` 命令 用于导言区 指定只载入某些文件 正文中不在其列表范围的`\include`命令不会起效：

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

宏包`syntonly`可以用于代码测试 在导言区使用 只编译文档内容而不输出结果 用于检查文档中LaTeX代码是否正确：

```
\usepackage{syntonly}
\syntonly % 若想生成文档 注释掉即可
```

用Latex排版文字

如今CJK宏包已不再推荐使用，因其使用非常不方便。

排版中文

`xeCJK`及`luatexja`是如今推荐的排版中文的宏包，而ctex文档类则对CJK等宏包进行了分装，直接使用ctex进行排版：

```
\documentclass{ctexart}
\begin{document}
在\LaTeX{}中排版中文。
汉字和English单词混排，通常不需要在中英文之间添加额外的空格。
当然，为了代码的可读性，加上汉字和 English 之间的空格也无妨。
汉字换行时不会引入多余的空格。
\end{document}
```

Latex中的字符

空格和分段

空格：在LaTeX中，空格会被忽略，多个空格被视为一个空格。一个换行也被视为一个空格

换行：在LaTeX中，换行会被忽略，多个换行被视为一个换行。或在行末使用 `\par` 或 `\\"{ }` 命令分段

特殊字符

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

% 分别为 # \$ % & { } _ ^ ~ \

其中 \^ 和 \~ 两个命令需要一个参数（详见[波浪号](#)小节），加一对花括号的写法相当于提供了空的参数。\\ 则是手动换行的命令，故需要用 \textbackslash 命令来输出 \ 字符。

连字

使用 {} 避免ff/fi/fl/ffi/ffi排版时连字

```
It's difficult to find \ldots\\
It's dif{}f{}icult to f{}ind \ldots
```

标点符号

中文标点符号直接使用中文输入法输入即可

西文标点符号则需要注意：

引号

单引号分别使用 ‘ 和 ’ 输入，双引号分别使用 ‘‘ 和 ‘‘ 输入

```
``Please press the `x' key.''
```

“Please press the ‘x’ key.”

连字号和破折号

三种长度：连字号 (hyphen)、短破折号 (en-dash) 和长破折号 (em-dash)。

连字号 - 用来组成复合词；短破折号 -- 用来连接数字表示范围；长破折号 --- 用来连接单词，语义上类似中文的破折号。

```
daughter-in-law, X-rated \\
pages 13--67 \\
yes---or no?
```

daughter-in-law, X-rated
pages 13–67
yes—or no?

省略号

有 \dots 命令表示省略号，相对于直接输入三个点更为合理。.\dots 与 \ldots 命令等效。

```
one, two, three, \ldots{} one hundred.
```

one, two, three, ... one hundred.

波浪号

用 `\~` 可以输入波浪号，单位置靠顶，但西文中很少用波浪号作为标点符号，主要用于拉丁文重音，中文环境中一般直接使用全角波浪号 (`~`) .

```
H\^otel, na\"i ve, '\el`eve, \\ % ^o 也可以写成 ^{o} 因为是带一个参数的命令  
sm\o rrebr\o d, !`Se\ norita!, \\ % !` 为倒着的感叹号  
Sch\"onrunner Schlo\ss{}  
Stra\ss{} e
```

Hôtel, naïve, élève,
smørrebrød, !‘Se norita!,
Schönrunner Schloß
Straße

其他符号

```
\P{} \S{} \dag{} \ddag{} \copyright{} \pounds{}
```

```
\textasteriskcentered  
\textperiodcentered  
\textbullet
```

```
\textregistered{} \texttrademark
```

¶ § † ‡
© £
* • •
® ™

以上分别为，`latex`预定了其他一些文本模式的符号，部分符号可参考[latex普通符号](#)的表格.

ó	\'{o}	ö	\"{o}	ô	\^{o}	ò	\'{o}	õ	\~{o}	ó	\={o}	§	\d{s}
ó	\.{o}	ó	\u{o}	ő	\H{o}	őo	\t{oo}	ó	\c{o}	ó	\d{o}	§	\r{s}
ó	\b{o}	À	\AA	å	\aa	ß	\ss	í	\i	í	\j	§	\H{s}
ø	\o	ſ	\t{s}	š	\v{s}	Ø	\O	¶	\P	§	\S		
æ	\ae	Æ	\AE	†	\dag	‡	\ddag	©	\copyright	£	\pounds		

断行和断页

单词间距

在西文排版实践中，断行的位置优先选取在两个单词之间，若有些单词不想在中间的空格处断开，可以用字符 ~ 输入一个不会断行的空格，通常用在英文人名、图标名称等上下文环境：

```
Fig.~2a \\  
Donald~E. Knuth
```

手动断行和断页

若需要手动断行，可以使用以下命令：

```
\\\[<length>]    \\\*[<length>]  
\newline
```

\\\ 可以带可选参数 <length>，用于在断行处向下增加垂直间距（见[垂直间距](#)小节），而 \newline 不带可选参数；\\\ 也会在表格、公式等地方用于换行，而 \newline 只用于文本段落中，带星号的 \\ 表示禁止在断行处分页；另外， \\ 断行命令不会令内容另起一段，而是在段落中直接开始新的一行。

断页的命令有两个：

```
\newpage  
\clearpage
```

通常情况下两个命令都起到另起一页的作用，区别在于：在双栏排版模式中 \newpage 起到另起一栏的作用，而 \clearpage 则能够另起一页；在涉及浮动体的排版上行为不同（见[浮动体](#)节以及[分栏](#)小节）。

latex默认的断行和断页位置需要进行微调时，可以通过以下命令告诉latex哪些地方适合断行或断页，哪些地方不适合：

```
\linebreak[<n>]    \nolinebreak[<n>]  
\pagebreak[<n>]    \nopagebreak[<n>]
```

其中 <n> 为可选参数，表示适合/不适合断行或断页的程度，取值范围为0-4，默认为4， linebreak[4] 意味着此处需要强行断行， nopagebreak[4] 意味着禁止在此处断页。

以上命令适合给出优先考虑断行/禁止断行断页的位置，但不适合直接拿来断行或断页，使用 \newline 或 \newpage 等命令是更好的选择。因为这些命令会在断行/断页位置填充适当的间距，但 \linebreak 和 \pagebreak 不能。

断词

latex遇到很长的英文单词，会在单词中间断开。如果一些单词没能自动断词，我们可以在单词内使用 \- 命令指定断词的位置：

I think this is: super-cal-i-frag-i-lis-tic-ex-pi-al-i-do-cious.

文档元素

结构化的文档有：章节、目录、列表、图表、交叉引用、脚注等等

章节和目录

章节标题

标准文档类article, report, book划分章节的命令：

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

其中 \chapter 只在report和book中有定义。这些命令用于生成章节标题，并能够自动编号。此外latex还提供了 \part 命令，用来将整个文档分割成大个分块（第·部分），不影响 \chapter 或 \section 等的编号。

上述命令除了生成带编号的标题外，还能向目录中添加条目，并影响页眉页脚的内容（详见[页眉页脚节](#)）：

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

较低层次如 \paragraph 和 \ subparagraph 即使不用带星号的变体，生成的标题默认也不带编号，事实上，除 \part 外：

- article文档类带编号的层级为 \section、\subsection、\subsubsection 三级；
- report和book文档类带编号的层级为 \chapter、\section、\subsection 三级。

具体的解释和调整方法见[latex中的计数器](#)小节。

latex及标准文档类并未提供为 \section 等章节命令定制格式的功能，这一功能由[titlessec](#)宏包提供。

目录

在latex中生成目录只需要在合适的地方使用命令：

```
\tableofcontents
```

这个命令会生成单独的一章（report/book）或一节（article），标题默认为“Contents”，可通过[latex可定制的一些命令和参数](#)节中的方法定制标题。 \tableofcontents 生成的章节默认不写入目录（\section* 或 \chapter*），可使用 [tocbibind](#)等宏包修改设置。

使用了 \chapter* 或 \section* 命令的章节标题不会生成目录项，可以在标题命令后使用：

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

其中 `<level>` 为章节层次chapter或section等，`<title>` 为出现于目录项的章节标题。

`titletoc`、`tocloft`等宏包提供了具体定制目录项格式的功能。

文档结构的划分

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

`book`文档类还提供了前言、正文、后记结构的划分命令：

```
\frontmatter % 前言部分，页码使用小写罗马数字；其后的 \chapter 不编号  
\mainmatter % 正文部分，页码使用阿拉伯数字，从1开始计数；其后的章节编号正常  
\backmatter % 后记部分，页码格式不变，继续正常计数；其后的 \chapter 不编号
```

还可与 `\appendix` 命令结合，生成有前言、正文、附录、后记四部分的文档，如下：

```
\documentclass{book}  
  
% 导言区，加载宏包和各项设置，包括参考文献、索引等  
\usepackage{makeidx}          % 调用 makeidx 宏包，用来处理索引  
\makeindex                   % 开启索引的收集  
\bibliographystyle{plain}    % 指定参考文献样式为 plain  
  
\begin{document}  
  
\frontmatter      % 前言部分  
\maketitle        % 标题页  
\include{preface} % 前言章节 preface.tex  
\tableofcontents  
  
\mainmatter       % 正文部分  
\include{chapter1} % 第一章 chapter1.tex  
\include{chapter2} % 第二章 chapter2.tex  
...  
\appendix         % 附录  
\include{appendixA} % 附录 A appendixA.tex  
...  
  
\backmatter        % 后记部分  
\include{epilogue} % 后记 epilogue.tex  
\bibliography{books} % 利用 BibTeX 工具从数据库文件 books.bib 生成参考文献  
\printindex        % 利用 makeindex 工具生成索引  
  
\end{document}
```

标题页

latex支持简单的标题页

Test title

Mary* Ted† Louis

May 5, 2023

*E-mail:*****@***.com

†Corresponding author

首先要先要给定标题和作者等信息：

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

前两个命令是必须的（不用 `\title` 会报错；不用 `\author` 会警告），`\date` 可选。 latex还提供了 `\today` 命令自动生成当前日期，`\date` 默认使用 `\today`。在 `\title`、`\author` 等命令中可以使用 `\thanks` 命令生成标题页的脚注，用 `\and` 隔开多个人名。

信息给定后通过 `\maketitle` 命令生成标题页，如下：

```
\title{Test title}
\author{ Mary\thanks{E-mail: *****@***.com}
\and Ted\thanks{Corresponding author}
\and Louis}
\date{\today}
```

article文档类的标题默认不单独成页，而report和book默认单独成页。可在 `\documentclass` 命令调用文档类时指定 `titlepage` 或 `notitlepage` 选项以修改。

latex标准类还提供了简单的 `titlepage` 环境，生成不带页眉页脚的一页。用户可以在该环境下自由发挥，生成自定义的标题页以代替 `\maketitle` 命令，或重定义：

```
\renewcommand{\maketitle}{\begin{titlepage}
... % 用户自定义的标题页排版内容
\end{titlepage}}
```

事实上，标准文档类的 `titlepage` 选项下的 `\maketitle` 本身就是一个 `titlepage` 环境。

交叉引用

能够被交叉引用的地方，如章节、公式、图表、定理等位置使用 `\label{<label-name>}` 命令：

之后可以在别处使用 `\ref{<label-name>}` 或 `\pageref{<label-name>}` 命令分别生成交叉引用的编号和页码：

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

A reference to this subsection looks like: “see section 3.3 on page 22.”

命令可使用的位置有：

- **章节标题** 在章节标题命令 `\section` 等之后紧接着使用
- **行间公式** 单行公式在公式内任意位置使用；多行公式在每一行公式的任意位置使用
- **有序列表** 在 `enumerate` 环境的每个 `\item` 命令之后、下一个 `\item` 命令之前的任意位置使用
- **图表标题** 在图表标题命令 `\caption` 之后紧接着使用
- **定理环境** 在定理环境内部任意位置使用

在使用不记编号的命令形式（`\section*`、`\caption*`（需加载相关宏包如`caption`）、带可选参数的 `\item` 命令等）不能使用 `\label` 命令，否则生成的引用编号不正确。

脚注和边注

使用 `\footnote{<footnote>}` 命令可以在页面底部生成一个脚注，例如输入：

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

在正文中的效果为：

“天地玄黄，宇宙洪荒。日月盈昃，辰宿列张。”¹

¹出自《千字文》。

有些情况下（比如在表格环境、各种盒子中）使用 `\footnote` 并不能生成脚注，可以先使用 `\footnotemark` 为脚注计数，再在合适的位置用 `\footnotetext` 生成脚注：

```
\begin{tabular}{l}
\hline
“天地玄黄，宇宙洪荒。日月盈昃，辰宿列张。”\footnotemark \\
\hline
\end{tabular}
\footnotetext{表格里的名句出自《千字文》。}
```

在表格中的效果为：

“天地玄黄，宇宙洪荒。日月盈昃，辰宿列张。”¹

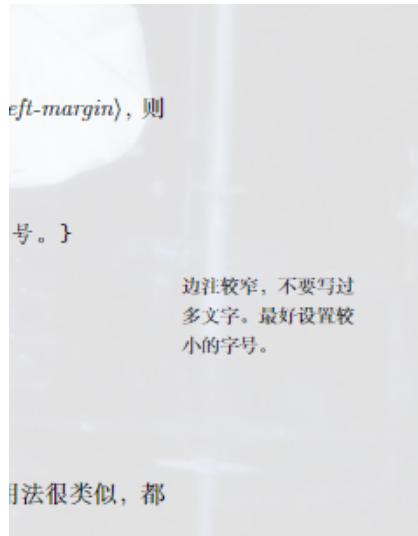
¹表格里的名句出自《千字文》。

使用 `\marginpar` 命令可以在边栏位置生成边注:

```
\marginpar[<left-margin>]{<right-margin>}
```

如果只给定了 `<right-margin>`，则边注在奇偶页文字相同；若给定了 `<left-margin>`，则奇偶页文字不同。例如：

```
\marginpar{\footnotesize 边注较窄，不要写过多文字，最好设置较小的字号。}
```



特殊环境

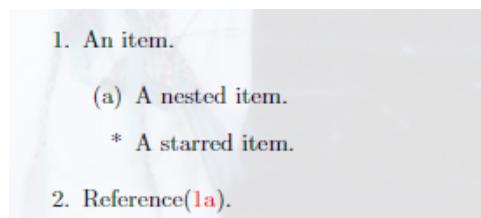
列表

基本的有序和无序列表：`enumerate` 和 `itemize`，都用 `\item` 表明每个列表项。`enumerate` 环境会自动对列表项编号。

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

```
\begin{enumerate}
    \item An item.
\begin{enumerate}
    \item A nested item.\label{itref}
    \item[*] A starred item.
\end{enumerate}
\item Reference(\ref{itref}).
\end{enumerate}
```

结果如下：



```
\begin{itemize}
    \item An item.
    \begin{itemize}
        \item A nested item.
        \item[+] A `plus' item.
    \end{itemize}
    \item Go back to upper lever.
\end{itemize}
```

结果如下：

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

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

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

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

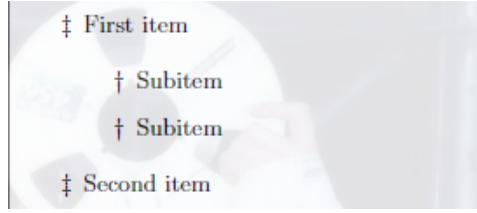
结果如下：

- Enumerate** Numbered list.
Itemize Non-numbered list.

各级无序列表的符号由命令 `\labelitemi` (i为1) 到 `\labelitemiv` (iv为4) 定义，可以简单地重新定义它们：

```
\renewcommand{\labelitemi}{\ddag}
\renewcommand{\labelitemii}{\dag}
\begin{itemize}
    \item First item
    \begin{itemize}
        \item Subitem
        \item Subitem
    \end{itemize}
    \item Second item
\end{itemize}
```

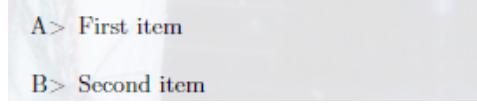
结果如下:



各级有序列表的符号由命令 `\labelenumi` (i为1) 到 `\labelenumiv` (iv为4) 定义, 重新定义这些命令需要用到[计数器](#)节的计数器相关命令:

```
\renewcommand{\labelenumi}{\Alpha{enumi}}
\begin{enumerate}
\item First item
\item Second item
\end{enumerate}
```

结果如下:

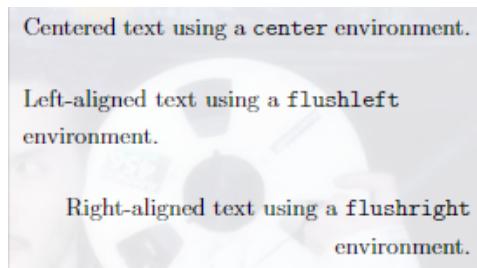


默认的**列表间距**比较宽, latex本身未提供方便的定制功能, 可用**enumitem**宏包定制各种列表间距. **enumitem**宏包还提供了对列表标签、引用等的定制.

对齐环境

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

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



除此之外, 还可以直接用 `\centering`、`\raggedright`、`\raggedleft` 命令改变文本的对齐方式:

```
\centering  
Centered text paragraph.
```

```
\raggedright  
Left-aligned text paragraph.
```

```
\raggedleft  
Right-aligned text paragraph.
```

Centered text paragraph.

Left-aligned text paragraph.

Right-aligned text paragraph.

*注意： center 等环境会在上下文产生一个额外间距，而 \centering 等命令不产生，只改变对齐方式。比如在浮动体环境 table 或 figure 内实现居中对齐，用 \centering 命令即可，没必要使用 center

引用环境

latex提供了两种引用的环境： quote 用于引用较短的文字，首行不缩进； quotation 用于引用若干段文字，首行缩进。引用环境一般文字有额外的左右缩进。

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

Francis Bacon says:

Knowledge is power.

《木兰诗》：

```
\begin{quotation}  
万里赴戎机，关山度若飞。  
朔气传金柝，寒光照铁衣。  
将军百战死，壮士一去兮。
```

归来见天子，天子坐明堂。

策勋十二转，赏赐百千强。……

```
\end{quotation}
```

《木兰诗》：

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

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

Rabindranath Tagore's short poem:

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

Rabindranath Tagore's short poem:

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

摘要环境

摘要环境 `abstract` 默认只在标准文档类中的 `article` 和 `report` 文档类可用，一般用于紧跟着 `\maketitle` 命令之后介绍文档的摘要。如果文档类指定了 `titlepage` 选项，则单独成页；反之，单栏排版时相当于一个居中的小标题加一个 `quotation` 环境，双栏排版时相当于 `\section*` 定义的一节。

代码环境

`verbatim` 用于将代码原样输出，等宽字体排版，回车和空格也分别起到了换行和空位的作用；带星号的版本则更进一步将空格显示成“_”

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

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

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

```
for (int i=0; i<4; i++)
    printf("Number %d\n", i);
```

要排版**简短的代码或关键字**, 可使用 `\verb` 命令, 如

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

`<delim>` 标明代码的分解位置 (delimit), 其后必须一致, **除字母、空格或星号外**, 可任意选择使得不与代码本身冲突, 习惯上使用 `|` 符号.

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

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

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

表格

latex里排版表格最基本的 `tabular` 环境用法:

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

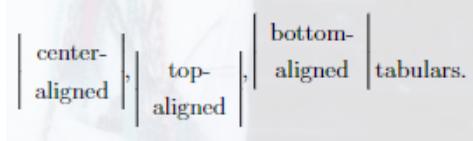
其中 `<column-spec>` 是列格式标记, 接下来会细讲; `&` 用来分隔单元格; `\backslash` 用来换行; `\hline` 用来在行与行之间绘制横线.

直接使用 `tabular` 环境的, 会与周围的文字混排. 此时可用可选参数 `<align>` 控制垂直对齐: `t` 和 `b` 分别表示按表功二顶部、底部对齐, 其他参数或省略不写 (默认) 表示居中对齐

```

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

```



通常情况下 `tabular` 环境很少直接与文字混排，而是放在 `table` 浮动体环境中，并用 `\caption` 命令加标题。

列格式

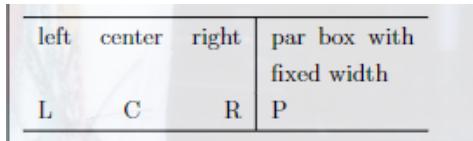
`tabular` 环境使用 `<column-spec>` 参数指定表格的列数以及每列的格式，基本的列格式见下表：

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

```

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

```



表格中每行的单元格数目不能多于列格式中的 `l / c / r / p` 的总数（可以少），否则出错。

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

*即： `l / c / r / p` 在单元格内（必选其中一个），`|` 和 `@{<string>}` 在单元格间（每个至多一个）。

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

1:1	one
11:3	eleven

另外，`latex`还提供了重复格式参数的写法：`*{<n>}{<column-spec>}`，例如以下两种写法：

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

有时需要整列修饰格式，比如整列改为粗体，可以用`array`宏包提供的辅助格式`>`和`<`，用于给列格式前后加上修饰命令：

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

italic*	normal
column*	column

辅助格式甚至支持插入`\centering`等命令改变`p`列格式的对齐方式，但需要额外加`\arraybackslash`命令，否则`\centering`等对齐命令会破坏表格环境里`\\\`换行命令的定义，`\arraybackslash`用来恢复之。若不加该命令，也可以用`\tabularnewline`代替`\\\`实现表格换行

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

italic*	normal
column*	column

`array`宏包还提供了类似`p`格式的`m`格式和`b`格式，三者分别在垂直方向上靠顶端对齐、居中以及底部对齐。

```
% \usepackage{array}
\newcommand{\txt}{\begin{array}{ccccccccc}a&b&c&d&e&f&g&h&i\end{array}}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}\hline pos & \txt & \txt & \txt & \\ \hline \end{tabular}
\end{array}
```

		a b c	
		a b c	d e f
pos	a b c	d e f	g h i
	d e f	g h i	
	g h i		

列宽

latex的 l / c / r 格式的列宽由文字内容的自然宽度决定，而 p 格式给定了列宽却不好控制对齐（可使用array的辅助格式），又考虑到列与列之间还有间距，所以直接生成给定总宽度的表格并不容易。

latex本身提供了 tabular* 环境用来排版定宽表格，但不太方便，例如用到 @ 格式插入额外命令，令单元格之间的间距为 \fill，但即便这样仍有瑕疵：

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

A	B	C	D
a	b	c	d

tabularx宏包提供了方便的解决方案。通过引入 x 列格式，类似 p 格式，不过会根据表格宽度自动计算列宽，多个 x 列格式平均分配列宽。结合array里的辅助格式修饰对齐方式：

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

A	B	C	D
a	b	c	d

横线

除了用 \hline 绘制表格线外，还可以用 \cline{<i>-<j>} 绘制跨越部分单元格的横线：

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

4	9	2
3	5	7
8	1	6

科技论文排版中广泛应用的表格形式是三线表，由`booktabs`宏包支持，提供了`\toprule`、`\midrule`和`\bottomrule`命令用以排版三线表的三条线，以及和`\cline`对应的`\cmidrule`。此外，最好不用其他横线以及竖线：

```
% \usepackage{booktabs}
\begin{tabular}{cccc}
\toprule
& \multicolumn{3}{c}{Numbers} \\
\cmidrule{2-4}
& 1 & 2 & 3 \\
\midrule
Alphabet & A & B & C \\
Roman & I & II & III \\
\bottomrule
\end{tabular}
```

Numbers			
	1	2	3
Alphabet	A	B	C
Roman	I	II	III

合并单元格

LaTeX是一行一行排版表格的，横向合并单元格较为容易，由`\multicolumn{<n>}{{<column-spec>}}{<item>}`命令实现。

其中`<n>`是要合并的列数，`<column-spec>`为合并单元格后的列格式，只允许出现一个`l / c / r / p`格式。如果合并前的单元格前后带表格线`|`，合并后的列格式也要带`|`以便表格的竖线一致。

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

1	2	Center
3		Right
4		C

以上还说明了，形如 `\multicolumn{1}{<column-spec>}{<item>}` 的命令可以用来修改某一单元格的列格式.

纵向合并单元格需要用到`multirow`宏包，通过 `\multirow{<n>}{<width>}{<item>}` 命令实现. 其中 `<width>` 是合并单元格后的列宽，可以填 * 以使用自然宽度.

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

Item	Value	
	First	Second
A	1	2

嵌套表格

拆分单元格可以通过单元格嵌套的方式得到，注意用 `\multicolumn` 命令配合 `@{}` 格式把单元格的额外边距去掉：

```
\begin{tabular}{|c|c|c|}
\hline
a & b & c \\ \hline
a & \multicolumn{1}{@{}c@{}}{%
\begin{tabular}{|c|c|}
\hline
e & f \\ \hline
e & g \\ \hline
\end{tabular}%
} & c \\ \hline
a & b & c \\ \hline
\end{tabular}
```

a	b	c
a	e f	c
a	b	c

若不需要为“拆分的单元格”画线，并且只在垂直方向“拆分”，可以使用`makecell`宏包提供的 `\makecell` 命令：

```
% \usepackage{makecell}
\begin{tabular}{|c|c|}
\hline
a & \makecell{d1 \\ d2} \\
b & c \\
\end{tabular}
```

a	d1 d2
b	c

行距控制

修改参数 `\arraystretch` 可以得到行距更加宽松的表格（具体参考[定义新命令](#)节）：

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

Really loose
tabular rows.

另一种增加间距的方式是换行命令 `\\"` 添加可选参数，产生额外的间距，适合用于在行间不加横线的表格：

```
\begin{tabular}{c}
\hline
Head lines \\
tabular lines \\
tabular lines \\
\end{tabular}
```

Head lines
tabular lines
tabular lines

但这种换行方式导致了——从第二行开始，表格的首个单元格不能直接使用中括号`[]`，否则 `\\"` 往往会将下一行的中括号当作自己的可选参数。若要使用中括号，应当放在花括号`{}`里面，或者可以选择将换行命令写成 `\\"[0pt]`。

图片

LaTeX需要由`graphicx`宏包辅助支持插图功能。

在调用了`graphicx`宏包后，可以使用 `\includegraphics[<options>]{<filename>}` 命令加载图片，其中 `<filename>` 为图片文件名，

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

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

在 `\includegraphics` 目录的可选参数 `<options>` 中可以使用 `<key>=<value>` 的形式，常用的参数如下：

参数	含义
<code>width=<width></code>	宽度为 <code><width></code>
<code>height=<height></code>	高度为 <code><height></code>
<code>scale=<scale></code>	缩放 <code><scale></code> 倍
<code>angle=<angle></code>	逆时针旋转 <code><angle></code> 度

`graphicx`宏包也支持 `draft / final` 选项，当`graphicx`宏包或文档类指定 `draft` 选项时，图片将不会实际插入，而是用一个包含文件名与原图片等大的方框取代之。

盒子

盒子是`latex`排版的基础单元。

水平盒子

```
\mbox{...}  
\makebox[<width>][<align>]{...}
```

`\mbox` 生成一个基本的水平盒子，内容只有一行，不允许分段（除非嵌套其他盒子，例如垂直盒子）。外表上该盒子的内容与正常文本无二，不过断行时文字不会从盒子里断开。

`\makebox` 更进一步加上可选参数控制盒子的宽度和对齐方式，可选居中 `c(default)`、左对齐 `l`、右对齐 `r` 和分散对齐 `s`（强行拉开单词的间距，往往会报 `Underfull \hbox` 的警告）。

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

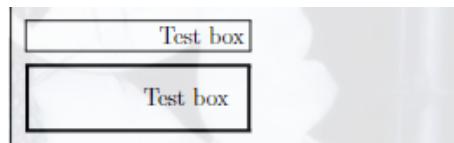
```
| Test some words.  
|   Test some words. |  
| Test some words. |  
|   Test some words.|  
| Test some words.|
```

带框的水平盒子

\fbox 和 \framebox 可以让水平盒子添加边框。使用的语法与 \mbox 和 \makebox 一致。

可以通过 \setlength 命令（见[长度和长度变量小节](#)）调节边框的宽度 \fboxrule 和内边距 \fboxsep：

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



垂直盒子

若要排版一个文字可以换行的盒子，\tex 提供了两种方式：

```
\parbox[<align>][<height>][<inner-align>]{<width>}{...}  
\begin{minipage}[<align>][<height>][<inner-align>]{<width>}  
...  
\end{minipage}
```

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

三字经：\parbox[t]{3em}{人之初 性本善 性相近 习相远}

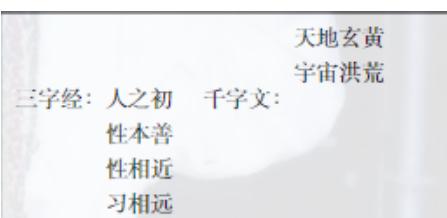
\quad

千字文：

```
\begin{minipage}[b][8ex][t]{4em}  
天地玄黄 宇宙洪荒  
\end{minipage}
```

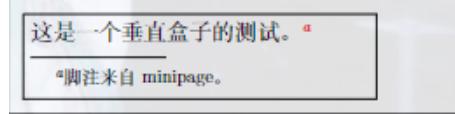
天地玄黄 宇宙洪荒

```
\begin{minipage}
```



若在 `minipage` 内使用 `\footnote` 命令，生成的脚注会出现在盒子底部，编号是独立的，并且用小写字母编号。而在 `\parbox` 里无法正常使用 `\footnote` 命令，只能在盒子里使用 `\footnotemark`，在盒子外使用 `\footnotetext`。

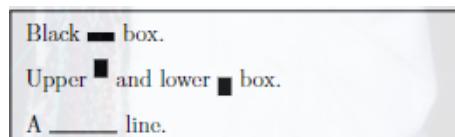
```
\fbox{  
  \begin{minipage}{15em}  
    这是一个垂直盒子的测试。  
    \footnote{脚注来自 minipage。}  
  \end{minipage}  
}
```



标尺盒子

`\rule[<raise>]{<width>}{<height>}` 命令用来画一个实心的矩形盒子，也可以适当调整用来画线（标尺）：

```
Black \rule{12pt}{4pt} box.  
  
Upper \rule[4pt]{6pt}{8pt} and lower \rule[-4pt]{6pt}{8pt} box.  
  
A \rule[-.4pt]{3em}{.4pt} line.
```



浮动体

latex的浮动体机制，令大块内容脱离上下文，放置在合适的位置。

latex预定义了两类浮动体环境 `figure` 和 `table`。习惯上 `figure` 里放图片，`table` 里放表格，但并没有严格限制，可以在任何一个浮动体里放置文字、公式、表格、图片等等任意内容。以 `table` 环境用法举例：

```
\begin{table}[<placement>]  
  ...  
\end{table}
```

`<placement>` 参数提供了一些符号用来表示浮动体允许排版的位置，`h` 表示当前位置；`t` 表示顶部；`b` 表示底部；`p` 表示单独成页；`!` 表示在决定位置时忽视限制。浮动体默认设置为 `tbp`，参数符号的顺序并不影响排版位置的选取，`latex` 总是以 `h-t-b-p` 的优先级顺序决定浮动体的位置。限制包括浮动体个数（除单独成页外，默认每页不超过3个浮动体，其中顶部不超过2个，底部不超过1个）以及浮动体空间占页面的百分比（默认顶部不超过70%，底部不超过30%）。

双栏排版环境下，`latex` 提供了 `table*` 和 `figure*` 环境用来排版跨栏的浮动体。其用法与 `table` 和 `figure` 一样，不同之处为双栏的 `<placement>` 参数只能用 `tp` 两个位置。

浮动体的位置选取受到先后顺序的限制. 若某个浮动体由于参数限制、空间限制等原因在当前页面无法放置，就要推迟到之后处理，并使得之后的同类浮动体一并推迟. `\clearpage` 命令会在另起一页之前，先将所有推迟处理的浮动体排版成页，此时 `htbp` 等位置限制会被完全忽略.

`float` 宏包为浮动体提供了 `H` 位置参数，不与 `htbp` 及 `!` 混用. 使用 `H` 位置参数时，会取消浮动机制，将浮动体视为一般的盒子插入当前位置. 这在一些特殊情况下很有用（如使用 `multicol` 宏包排版分栏内容的时候），但尺寸过大的浮动体可能使得分页比较困难.

浮动体的标题

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

`\caption` 的用法非常类似于 `\section` 等命令，可以用带星号的命令 `\caption*` 生成不带编号的标题（需加载相关宏包，如 `caption`），也可以使用带可选参数的形式 `\caption[...]{...}`，使得在目录里使用短标题. `\caption` 命令之后还可以紧跟着 `\label` 命令标记交叉引用.

`\caption` 生成的标题形如“Figure 1: ...”（`figure` 环境）或“Table 1: ...”（`table` 环境），可通过修改 `\figurename` 和 `\tablename` 的内容修改标题的前缀（详见 [latex 可定制的一些命令和参数](#) 节）. 标题样式的定制功能由 `caption` 宏包提供.

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

```
\listoftables  
\listoffigures
```

类似 `\tableofcontents` 生成单独的章节.

并排和子图表

一个浮动体里放置多张图可以通过直接并排放置，也可以通过分段或换行命令 `\\\` 排版多行多列的图片.

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



由于标题是横跨一行的，用 `\caption` 命令为每个图片单独生成标题需要借助 `\parbox` 或者 `\minipage` 环境

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

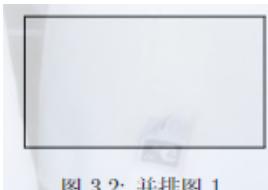


图 3.2: 并排图 1。

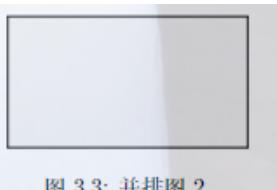
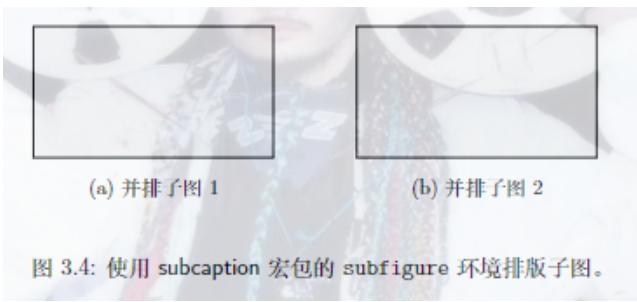


图 3.3: 并排图 2。

以上图标标题是单独编号的，若要进一步给每个图片定义小标题时，需要用到`subcaption`宏包。

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



*subcaption*宏包依赖上文提到的*caption*, 因此也支持子图表标题样式的定制. 并排子图表的功能也可通过*subfig*宏包的 `\subfloat` 命令实现.

排版数学公式

AMS宏集

AMS宏集合是美国数学学会 (American Mathematical Society) 提供的对latex原生的数学公式排版的拓展, 其核心是 *amsmath*宏包, 对多行公式的排版提供了有力支持. 此外, *amsfonts*宏包以及基于它的*amssymb*宏包提供了丰富的数学符号; *amsthm*宏包扩展了latex定理证明格式.

本章许多命令和环境依赖于*amsmath*宏包, 以**粗体**示意. 导言区需写有

```
\usepackage{amsmath}
```

公式排版基础

行内和行间公式

行内公式与文字混排, 而**行间公式**则单独列为一列排版。

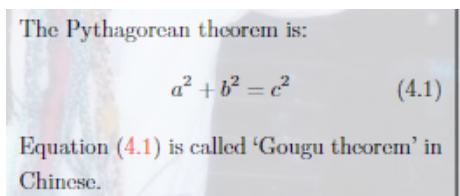
行内公式由一对 \$ 符号包裹:

```
The Pythagorean theorem is $a^2 + b^2 = c^2$.
```

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

```
The Pythagorean theorem is:
```

```
\begin{equation}
a^2 + b^2 = c^2 \label{pythagorean}
\end{equation}
Equation \eqref{pythagorean} is called 'Gougu theorem' in Chinese.
```

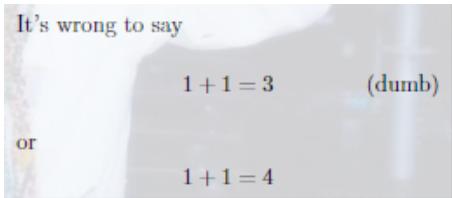


It's wrong to say

```
\begin{equation}
 1 + 1 = 3 \tag{dumb}
\end{equation}
```

or

```
\begin{equation}
 1 + 1 = 4 \notag
\end{equation}
```



若需要直接使用不带编号的行间公式，则将公式用命令 `\[` 和 `\]` 包裹（tex原生使用的是一对 `$$` 符号包裹，但无法通过指定 `fleqn` 选项控制左对齐，与上下文的间距也不好调整，故不推荐），与之等效的是 `displaymath` 环境，或 `equation*` 环境。

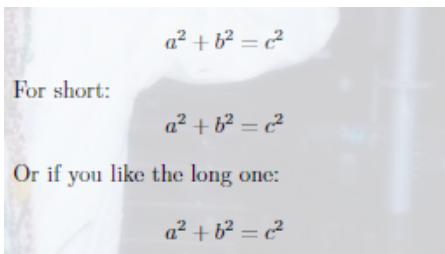
```
\begin{equation*}
 a^2 + b^2 = c^2
\end{equation*}
```

For short:

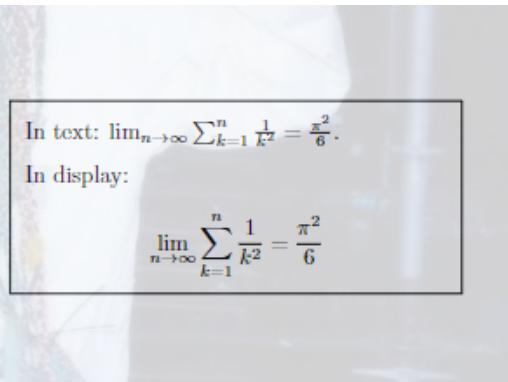
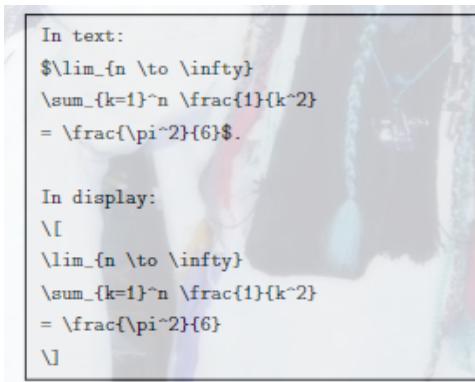
```
\[ a^2 + b^2 = c^2 \]
```

Or if you like the long one:

```
\begin{displaymath}
 a^2 + b^2 = c^2
\end{displaymath}
```



行内公式为了与文字相适应，公式元素会被压缩



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

数学模式

数学模式有以下特点：

1. 数学模式中输入的空格被忽略。数学符号的间距默认由符号性质决定，需要人为引入间距时，使用 `\quad` 和 `\quad` 命令。详见[公式中的间距](#)节。
2. **不允许有空行（分段）**。行间公式中也无法用 `\backslash` 命令手动换行。排版多行公式需要用到[多行公式](#)节介绍的各种环境。
3. 所有的字母都被当成变量处理，字母间距与文本模式不一致，也无法生成单词之间的空格。若要输入正体文本，简单情况可以使用[数学字母字体](#)小节中提供的 `\text{}` 命令，或者用`amsmath`提供的 `\text{}` 命令（仅适合在公式中穿插少量文字）。

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

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

数学符号

latex默认提供了常用的数学符号，`amssymb`宏包提供了一些次常用的符号。大多数常用的数学符号都能做本章末尾列出的各个表格里查到。

指数、上下标和导数

latex中用 `^` 和 `_` 表明上下标。上下标内容一般需要**用花括号包裹**，否则只对后一个符号起作用。

导数符号'见下：

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

分式和根式

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

```
In display style:
\[
  3/8 \quad \frac{3}{8} \quad \tfrac{3}{8}
\]
In text style:
$1\frac{1}{2}\text{~hours} \quad \dfrac{1}{2}\text{~hours}
```

In display style:

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

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

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

特殊的分式形式, 如二次项结构, 由`amsmath`宏包的 `\binom` 命令生成:

Pascal's rule is

```
\[
\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}
\]
```

Pascal's rule is

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

关系符

latex常见的关系符号见表格. latex提供自定义二元关系符的命令 `\stackrel{*}{\approx}`, 用于将一个符号叠加在原有的二元关系符之上:

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

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

算符

latex常见的算符见表格. ∇ (`\nabla`)、 ∂ (`\partial`)也是常用的算符.

latex将数学函数的名称作为常用的算符排版, 字体为直立字体, 如下表:

不带上下限的算符				
$\sin(\sin)$	$\arcsin(\arcsin)$	$\sinh(\sinh)$	$\exp(\exp)$	$\dim(\dim)$
$\cos(\cos)$	$\arccos(\arccos)$	$\cosh(\cosh)$	$\log(\log)$	$\ker(\ker)$
$\tan(\tan)$	$\arctan(\arctan)$	$\tanh(\tanh)$	$\lg(\lg)$	$\hom(\hom)$
$\cot(\cot)$	$\arg(\arg)$	$\coth(\coth)$	$\ln(\ln)$	$\deg(\deg)$
$\sec(\sec)$	$\csc(\csc)$			

带上下限的算符				
$\lim(\ \lim)$	$\limsup(\ \limsup)$	$\liminf(\ \liminf)$	$\sup(\sup)$	$\inf(\inf)$
$\min(\min)$	$\max(\max)$	$\det(\det)$	$\Pr(\Pr)$	$\gcd(\gcd)$

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

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

`amsmath`允许用户在导言区使用 `\DeclareMathOperator` 定义自己的算符, 其中带星号的命令定义带上下限的算符:

```
\DeclareMathOperator{\argh}{\argh}\argh  
\DeclareMathOperator*\{\nut\}{Nut}  
  
\[\argh 3 = \nut_{x=1} 4x\]
```

巨算符

积分号 $\int(\ \int)$ 、求和号 $\sum(\ \sum)$ 等符号称为**巨算符**. 巨算符在行内和行间公式的大小形状有区别.

巨算符的上下标位置可由 `\limits` 和 `\nolimits` 命令控制, 前者令上下标位于上下方; 后者令上下标位于右上和右下方.

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

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

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

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

数学重音和上下括号

数学符号可以像文字一样加重音，如求导 \dot{r} (`\dot{r}`)、 \ddot{r} (`\ddot{r}`)、向量 \vec{r} (`\vec{r}`)、单位向量 \hat{e} (`\hat{e}`)等，详见末尾表。

latex还能为多个字符加重音，包括直接画线 $0.\overline{3} = \underline{\overline{0}}.$ `\overline{3}` = `\underline{\overline{3}}`、宽重音符号 \widehat{XY} (`\widehat{XY}`)、向量箭头 \overrightarrow{AB} (`\overrightarrow{AB}`)等，详见末尾表。

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

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

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

meaning of life

箭头

常用的包括 `\rightarrow` (→, 或 `\to`)、`\leftarrow` (←, 或 `\gets`)等，更多见末尾表。

amsmath 的 `\xleftarrow[<under>]{<over>}` 和 `\xrightarrow[<under>]{<over>}` 命令提供了长度可以伸展的箭头，并且可以增加上下标：

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

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

括号和定界符

latex提供了多种括号和定界符表示公式块的边界，详见末尾表。

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

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

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

若不满意于`\textrm{Latex}`自动调节的定界符大小, 可以使用 `\big`、`\bigg` 等命令生成固定大小的定界符. 更常用的形式是 `\bigl`、`\biggl`、`\bigr`、`\biggr` 等 (`\bigl` 和 `\bigr` 等不必成对出现).

```
$\Bigl((x+1)(x-1)\Bigr)^2$ \\
$\bigl( \Bigl( \biggl( \Biggl( \quad
\biggr\} \Bigr\} \biggr\} \biggr\} \Biggr\} \quad
\bigl| \Bigl| \biggl| \Biggl| \quad
\bigl\Downarrow \Bigl\Downarrow \biggl\Downarrow \Biggl\Downarrow $
```

使用 `\big` 和 `\bigg` 等命令的另一个好处是：使用 `\left` 和 `\right` 分界符包裹公式块是不允许断行的（下文提到的 `array` 或者 `aligned` 等环境视为一个公式块），所以也不允许在多行公式里跨行使用，而 `\big` 和 `\bigg` 等命令不受限制。

多行公式

长公式折行

通常应该避免写超过一行而需要折行的长公式。若一定要折行，习惯上按照等号之前、加减号、乘除号的优先次序选择折行，避免在其它位置折行。

*amsmath*宏包的 `multiline` 环境提供了书写折行多公式的环境. 允许用 \\\ 折行, 将公式编号放在最后一行. 首行左对齐, 末行右对齐, 其余行居中.

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

$$\begin{aligned} a + b + c + d + e + f + g + h + i \\ = j + k + l + m + n \\ = o + p + q + r + s \\ = t + u + v + x + z \quad (4.2) \end{aligned}$$

与表格不同的是，公式的最后一行不写 \\.

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

多行公式

latex提供了 `eqnarray` 环境，按照等号左边——等号——等号右边呈三列对齐，但等号周围的空隙过大，加上公式编号等一些bug，目前不再推荐使用。

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

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

$$a = b + c \tag{1}$$

$$= d + e \tag{2}$$

`align` 环境会给每行公式都编号，我们可以用 `\notag` 去除某行的编号（为了对齐等号及公式，将分隔符放在等号右侧，此时需要在等号后面添加一对括号 `{}` 以产生正常的间距）：

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

$$a = b + c \tag{3}$$

$$= d + e + f + g + h + i + j + k + l \\ + m + n + o \tag{4}$$

$$= p + q + r + s \tag{5}$$

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

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

$$a = 1 \quad b = 2 \quad c = 3 \tag{6}$$

$$d = -1 \quad e = -2 \quad f = -5 \tag{7}$$

若不需要等号对齐，只需要罗列多行公式，`gather` 环境可以被使用：

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

$$a = b + c \tag{8}$$

$$d = e + f + g \tag{9}$$

$$h + i = j + k$$

$$l + m = n \tag{10}$$

`align` 和 `gather` 都有对应不带编号的版本 `align*` 和 `gather*`.

公用编号的多行公式

公用一个编号的多行公式，编号位于公式的居中位置。`amsmath`宏包提供了诸如 `aligned`、`gathered` 等环境，与 `equation` 环境套用，语法与前对应。

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

$$\begin{aligned} a &= b + c \\ d &= e + f + g \\ h + i &= j + k \\ l + m &= n \end{aligned} \tag{11}$$

`split` 环境与 `aligned` 环境用法类似，也与 `equation` 环境套用，区别是 `split` 只能将每行的一个公式分两栏，`align` 允许每行多个公式多栏（即允许不允许多于一个分隔符）。

数组和矩阵

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

```
\[
\mathbf{X} = \left(
\begin{array}{cccc}
x_{11} & x_{12} & \cdots & x_{1n} \\
x_{21} & x_{22} & \cdots & x_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
x_{n1} & x_{n2} & \cdots & x_{nn}
\end{array} \right)
\]
```

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix}$$

值得注意的是，`aligned` 等环境也可以用定界符包裹：

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

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

不过上例可以用`amsmath`提供的 `cases` 环境实现：

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

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

使用 `array` 环境排版各种矩阵也是可以的。`amsmath`宏包还直接提供了多种排版矩阵的环境，包括不带定界符的 `matrix`，以及带各种定界符的矩阵

`\pmatrix(())`、`\bmatrix([])`、`\Bmatrix({ })`、`\vmatrix(||)`、`\Vmatrix(|||)`. 使用这些环境时无需给定列格式
 (事实上这些矩阵内部也是用 `array` 环境生成的, 列格式默认为 `*[<n>]{c}`, `<n>` 默认为 `10`; `cases` 内部时一个列
 格式为 `@{}`1@{\quad}1@{`}` 的 `array` 环境) :

```
\[
\begin{matrix}
1 & 2 \\
3 & 4
\end{matrix} \quad
\begin{bmatrix}
x_{11} & x_{12} & \cdots & x_{1n} \\
x_{21} & x_{22} & \cdots & x_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
x_{n1} & x_{n2} & \cdots & x_{nn}
\end{bmatrix}
\]
]
```

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}$$

在矩阵中的元素里排版分式时, 一来要用到 `\frac` 等命令, 二来行与行之间有可能紧贴着:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

可以用[排版表格调整行间距的方法](#)来调整间距:

```
\[
\mathbf{H} = \begin{bmatrix}
\frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\
\frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2}
\end{bmatrix} \quad
\]
]
```

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

公式中的间距

绝大部分时候, 数学公式中元素的间距是根据符号类型自动生成的, 需要手动调整的情况极少. 公式中的间距包括两个
 生成间距的命令 `\quad` 和 `\quadquad` 及 `\, , \, , \,` 以及负间距 `\! ,` 文本中的 `\,` 也能使用在数学公式中.

公式	效果	公式	效果
无额外间距	aa	\,	$a a$
\quad	$a \quad a$	\colon	$a a$
\quad\quad	$a \quad a$	\;:	$a a$
\,	$a a$	\!	aa

一个常见的用途是用于修正积分的被积函数 $f(x)$ 和微元 dx 之间的距离：

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

$$\int_a^b f(x) \mathrm{d}x \quad \int_a^b f(x) \, dx$$

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

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

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

数学符号的字体控制

数学字母字体

LaTeX 允许一部分数学符号切换字体，主要是拉丁字母、数字、大写希腊字母以及重音符号等。

示例	命令	依赖宏包
ABCDEabcde1234	<code>\mathnormal{}</code>	

示例	命令	依赖宏包
ABCDEabcde1234	<code>\mathrm{}</code>	
<i>ABCDEabcde1234</i>	<code>\mathit{}</code>	
ABCDEabcde1234	<code>\mathbf{}</code>	
ABCDEabcde1234	<code>\mathsf{}</code>	
ABCDEabcde1234	<code>\mathtt{}</code>	
$\mathcal{ABCDEabcde}1234$	<code>\mathcal{}</code>	仅提供大写字母
$\mathscr{ABCDEabcde}1234$	<code>\mathscr{}</code>	<i>mathrsfs</i> 仅提供大写字母
$\mathfrak{ABCDEabcde}1234$	<code>\mathfrak{}</code>	<i>amssymb</i> 或 <i>eufrac</i>
$\mathbb{ABCDEabcde}1234$	<code>\mathbb{}</code>	<i>amssymb</i> 仅提供大写字母

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

$$\begin{array}{c} \mathcal{R} \quad \mathfrak{R} \quad \mathbb{R} \\ \mathcal{L} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu} \\ \mathfrak{su}(2) \text{ and } \mathfrak{so}(3) \text{ Lie algebra} \end{array}$$

一般来说，不同数学字体往往带有不同的语义，如矩阵向量使用粗体或粗斜体等等。出于内容与格式分离以及方便书写的考虑，可以为它们定义新的命令。具体方法详见[定义新命令](#)小节。

若要为所有的数学符号切换字体，则需要直接调用数学字体宏包（见[字体宏包](#)小节）。

加粗的数学符号

上 `\mathbf` 命令只能获得直立、加粗的字母。若想得到粗斜体，可以使用`amsmath`宏包提供的 `\boldsymbol` 命令：

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

或直接使用`bm`宏包提供的 `\bm` 命令：

```
% \usepackage{bm}
$\mu, M \quad \bm{\mu}, \bm{M}$
```

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

在 latex 默认的数学字体中，一些符号本身并没有粗体版本，使用 `\boldsymbol` 也得不到粗体。此时 `\bm` 命令会生成“伪粗体”。

数学符号的尺寸

数学符号按照符号排版的位置规定尺寸，从大到小包括行间公式尺寸 $\sum a$ (`\displaystyle`)、行内公式尺寸 $\sum a$ (`\textstyle`)、上下标尺寸 $\sum a$ (`\scriptstyle`)、次级上下标尺寸 $\sum a$ (`\scriptscriptstyle`)（例如在行间公式中使用分式时分子和分母均是行内公式尺寸，可以使用以上命令切换分子或分母的尺寸）

定理环境

latex原始的定理环境

latex 提供了一个基本的命令 `\newtheorem` 提供定理环境的定义：

```
\newtheorem{<theorem environment>}{<title>}[<section-level>]  
\newtheorem{<theorem environment>}[<counter>]{<title>}
```

`<theorem>` 为自定义定理环境的名称，原始的 latex 未提供现成的定理环境，`<title>` 是定理环境的标题。

定理的符号由两个可选参数之一决定，不可同时使用：

- `<section level>` 为章节级别，如 `chapter`、`section` 等，定理序号成为章节的下一级序号；
- `<counter>` 为用 `\newcounter` 自定义的计数器名称（详见计数器节），定理序号由这个计数器管理。

若两个可选参数均不用，默认使用与定理环境同名的计数器。

```
\newtheorem{mythm}{My Theorem}[section]  
\begin{mythm}\label{thm:light}  
The light speed in vacuum  
is $299,792,458\,\mathrm{m/s}$.  
\end{mythm}  
\begin{mythm}[Energy-momentum relation]  
The relationship of energy,  
momentum and mass is  
\[  
E^2 = m_0^2 c^4 + p^2 c^2  
\]  
where $c$ is the light speed  
described in theorem \ref{thm:light}.  
\end{mythm}
```

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

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

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

where c is the light speed described in theorem 4.8.1.

amsthm宏包

latex默认的定理环境格式为粗体标签、斜体内容、定理名用小括号包裹。若要修改格式，需要使用其他宏包如`amsthm`、`ntheorem`等等。

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

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

```
\theoremstyle{definition} \newtheorem{law}{Law}
\theoremstyle{plain} \newtheorem{jury}[law]{Jury} % 与law共用一个计数器law
\theoremstyle{remark} \newtheorem*{mar}{Margaret}
```

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

Law 1. Don't hide in the witness box.

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

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

Margaret. No, No, No

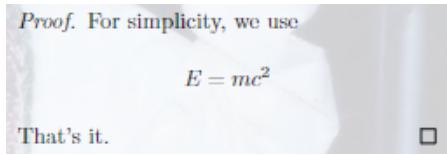
Margaret. Denis!

*amsthm*支持使用 `\newtheoremstyle` 命令自定义定理环境格式，事实上*ntheorem*宏包使用更为方便.

证明环境和证毕符号

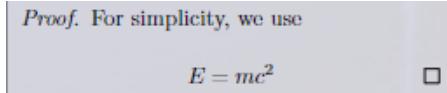
*amsthm*还提供了 `proof` 环境用于排版定理的证明过程. `proof` 环境末尾自动加上一个 \square 证毕符号:

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



若行末是不带一个不带编号的公式， \square 符号会另起一行，这时可使用 `\qedhere` 命令将 \square 符号放在公式末尾:

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



在使用带编号的公式时，最好不要在公式末尾使用 `\qedhere` 命令，因为该命令会使 \square 符号放在公式编号的下方，十分难看的位置.

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

证毕符号 \square 本身被定义在命令 `\qedsymbol` 中，若要修改证毕符号，需要用 `\renewcommand` 命令修改（只对*amsthm*宏包适用，其他宏包如*ntheorem*等需要使用各自文档中提供的修改方法）：

```
\renewcommand{\qedsymbol}{\rule{1ex}{1.5ex}}
\begin{proof}
For simplicity, we use
\[
E=mc^2 \qedhere
\]
\end{proof}
```

Proof. For simplicity, we use

$$E = mc^2$$

■

符号表

几个注意事项：

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

latex普通符号

表 4.4: 文本/数学模式通用符号

这些符号可用于文本和数学模式。

{	\{	}	\}	\$	\\$	%	\%
\dag	\dag	\S	\S	\circledC	\copyright	\dots	\dots
\ddag	\ddag	\P	\P	\pounds	\pounds		

表 4.5: 希腊字母

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

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

表 4.6: 二元关系符

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

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

表 4.7: 二元运算符

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

表 4.8: 巨算符

Σ	\sum	$\backslash sum$	\bigcup	\bigcup	$\backslash bigcup$	\bigvee	\bigvee	$\backslash bigvee$
\prod	\prod	$\backslash prod$	\bigcap	\bigcap	$\backslash bigcap$	\bigwedge	\bigwedge	$\backslash bigwedge$
\coprod	\coprod	$\backslash coprod$	\bigsqcup	\bigsqcup	$\backslash bigsqcup$	\biguplus	\biguplus	$\backslash biguplus$
\int	\int	$\backslash int$	\oint	\oint	$\backslash oint$	\odot	\odot	$\backslash bigodot$
\oplus	\oplus	$\backslash bigoplus$	\otimes	\otimes	$\backslash bigotimes$			
\iint	\iint	$\backslash iint$	\iiint	\iiint	$\backslash iiint$	\iiiiint	\iiiiint	$\backslash iiiint$
\cdots	\cdots	$\backslash idotsint$						

表 4.9: 数学重音符号

最后一个 \wideparen 依赖 yhmath 宏包。

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

表 4.10: 箭头

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

表 4.11: 作为重音的箭头符号

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

表 4.12: 定界符

`amsmath` 还定义了 `\lvert`、`\rvert` 和 `\lVert`、`\rVert`, 分别作为 `\vert` 和 `\lvert` 对应的开符号 (左侧) 和闭符号 (右侧) 的命令。

()	\uparrow	<code>\uparrowarrow</code>	\downarrow	<code>\downarrowarrow</code>
[]	\uparrow	<code>\Uparrow</code>	\downarrow	<code>\Downarrow</code>
{	}	\updownarrow	<code>\updownarrowarrow</code>	\Updownarrow	<code>\Updownarrow</code>
	\mid or <code>\lvert</code>	\parallel	<code>\lvert</code>	\lceil	<code>\lceil</code>
\langle	\rangle	\rangle	<code>\rangleangle</code>	\lfloor	<code>\lfloor</code>
/	/	\backslash	<code>\backslash</code>		

表 4.13: 用于行间公式的大定界符

$\left\{ \right\}$	<code>\lgroup</code>	<code>\rgroup</code>	$\left\{ \right\}$	<code>\lmoustache</code>
$\left \right $	<code>\arrowvert</code>	<code>\Arrowvert</code>	$\left \right $	<code>\bracevert</code>
$\left\{ \right\}$	<code>\rmoustache</code>			

表 4.14: 其他符号

...	<code>\dots</code>	...	<code>\cdots</code>	:	<code>\vdots</code>	⋮	<code>\ddots</code>
\hbar	<code>\hbar</code>	\imath	<code>\imath</code>	\jmath	<code>\jmath</code>	ℓ	<code>\ell</code>
\Re	<code>\Re</code>	\Im	<code>\Im</code>	\aleph	<code>\aleph</code>	\wp	<code>\wp</code>
\forall	<code>\forall</code>	\exists	<code>\exists</code>	\mho	<code>\mho</code>	∂	<code>\partial</code>
'		\prime	<code>\prime</code>	\emptyset	<code>\emptyset</code>	∞	<code>\infty</code>
∇	<code>\nabla</code>	\triangle	<code>\triangle</code>	\Box	<code>\Box</code>	\Diamond	<code>\Diamond</code>
\bot	<code>\bot</code>	\top	<code>\top</code>	\angle	<code>\angle</code>	\surd	<code>\surd</code>
◊	<code>\diamondsuit</code>	♥	<code>\heartsuit</code>	♣	<code>\clubsuit</code>	♠	<code>\spadesuit</code>
¬	<code>\neg</code> or <code>\lnot</code>	flat	<code>\flat</code>	natural	<code>\natural</code>	#	<code>\sharp</code>

AMS符号

本小节所有符号依赖`amssymb`宏包

表 4.15: *AMS* 希腊字母和希伯来字母

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

表 4.16: *AMS* 二元关系符

\triangleleft	\lessdot	\triangleright	\gtrdot	\doteqdot	\doteqdot
\trianglelefteqslant	\leqslant	\trianglerighteqslant	\geqslant	\risingdotseq	\risingdotseq
\trianglelefteqslantless	\eqslantless	\trianglerighteqslantgtr	\eqslantgtr	\fallingdotseq	\fallingdotseq
\trianglelefteq	\leqq	\trianglerighteq	\geqq	\eqcirc	\eqcirc
\llcorner	\lll or \llless	\ggm	\ggg	\circeq	\circeq
\lesssim	\lesssim	\gtrsim	\gtrsim	\triangleq	\triangleq
\lessapprox	\lessapprox	\gtrapprox	\gtrapprox	\bumpeq	\bumpeq
\lessdot	\lessdot	\gtreqless	\gtreqless	\Bumpeq	\Bumpeq
\lesseqgtr	\lesseqgtr	\gtreqqless	\gtreqqless	\thicksim	\thicksim
\lesseqqgtr	\lesseqqgtr	\gtreqqless	\gtreqqless	\thickapprox	\thickapprox
\preccurlyeq	\preccurlyeq	\succcurlyeq	\succcurlyeq	\approxeq	\approxeq
\curlyeqprec	\curlyeqprec	\curlyeqsucc	\curlyeqsucc	\backsimeq	\backsimeq
\precsim	\precsim	\succsim	\succsim	\backsimeq	\backsimeq
\precapprox	\precapprox	\succapprox	\succapprox	\vDash	\vDash
\subseteqq	\subsetneqq	\supseteqq	\supseteqq	\Vdash	\Vdash
\shortparallel	\shortparallel	\Supset	\Supset	\Vvdash	\Vvdash
\blacktriangleleft	\blacktriangleleft	\sqsupset	\sqsupset	\backepsilon	\backepsilon
\vartriangleright	\vartriangleright	\because	\because	\varpropto	\varpropto
\blacktriangleright	\blacktriangleright	\Subset	\Subset	\between	\between
\trianglerighteq	\trianglerighteq	\smallfrown	\smallfrown	\pitchfork	\pitchfork
\vartriangleleft	\vartriangleleft	\shortmid	\shortmid	\smallsmile	\smallsmile
\trianglelefteq	\trianglelefteq	\therefore	\therefore	\sqsubset	\sqsubset

表 4.17: *AMS* 二元运算符

\dotplus	\dotplus	\centerdot	\centerdot
\ltimes	\ltimes	\rtimes	\rtimes
\divideontimes	\divideontimes	\divideontimes	\divideontimes
\doublecup	\doublecup	\doublecap	\doublecap
\smallsetminus	\smallsetminus	\smallsetminus	\smallsetminus
\veebar	\veebar	\barwedge	\barwedge
\barwedge	\barwedge	\doublebarwedge	\doublebarwedge
\boxplus	\boxplus	\boxminus	\boxminus
\boxtimes	\boxtimes	\boxdot	\boxdot
\circledddash	\circledddash	\circledcirc	\circledcirc
\intercal	\intercal	\circledast	\circledast
\rightthreetimes	\rightthreetimes	\leftthreetimes	\leftthreetimes
\curlyvee	\curlyvee	\curlywedge	\curlywedge
\leftthreetimes	\leftthreetimes	\rightthreetimes	\rightthreetimes

表 4.18: \mathcal{AM} S 箭头

$\leftarrow\!\!\!-\!$	<code>\dashleftarrow</code>	$\rightarrow\!\!\!-\!$	<code>\dashrightarrow</code>
$\Leftarrow\!\!\!-\!$	<code>\leftleftarrows</code>	$\Rightarrow\!\!\!-\!$	<code>\rightrightarrows</code>
$\Leftrightarrow\!\!\!-\!$	<code>\leftrightarrows</code>	$\Leftrightarrow\!\!\!-\!$	<code>\rightleftarrows</code>
$\Leftarrow\!\!\!-\!$	<code>\Lleftarrow</code>	$\Rightarrow\!\!\!-\!$	<code>\Rrightarrow</code>
$\Leftarrow\!\!\!-\!$	<code>\twoheadleftarrow</code>	$\Rightarrow\!\!\!-\!$	<code>\twoheadrightarrow</code>
$\Leftarrow\!\!\!-\!$	<code>\leftarrowtail</code>	$\Rightarrow\!\!\!-\!$	<code>\rightarrowtail</code>
$\Leftarrow\!\!\!-\!$	<code>\leftrightharpoons</code>	$\Rightarrow\!\!\!-\!$	<code>\rightleftharpoons</code>
\nmid	<code>\Lsh</code>	\nmid	<code>\Rsh</code>
\looparrowleft	<code>\looparrowleft</code>	\looparrowright	<code>\looparrowright</code>
\curvearrowleft	<code>\curvearrowleft</code>	\curvearrowright	<code>\curvearrowright</code>
\circlearrowleft	<code>\circlearrowleft</code>	\circlearrowright	<code>\circlearrowright</code>
\multimap	<code>\multimap</code>	\upuparrows	<code>\upuparrows</code>
\downdownarrows	<code>\downdownarrows</code>	\upharpoonleft	<code>\upharpoonleft</code>
\upharpoonright	<code>\upharpoonright</code>	\downharpoonright	<code>\downharpoonright</code>
\rightsquigarrow	<code>\rightsquigarrow</code>	\leftrightsquigarrow	<code>\leftrightsquigarrow</code>

表 4.19: \mathcal{AM} S 反义二元关系符和箭头

$\not\sim$	<code>\nless</code>	$\not\simeq$	<code>\ngtr</code>	$\not\subseteq$	<code>\varsubsetneqq</code>
$\not\leq$	<code>\lneq</code>	$\not\geq$	<code>\gneq</code>	$\not\supseteq$	<code>\varsupsetneqq</code>
$\not\leqslant$	<code>\nleq</code>	$\not\geqslant$	<code>\ngeq</code>	$\not\subsetneq$	<code>\nsubsetneqq</code>
$\not\leqslant$	<code>\nleqslant</code>	$\not\geqslant$	<code>\ngeqslant</code>	$\not\supseteq$	<code>\nsupsetneqq</code>
$\not\leq\!\!\!-\!$	<code>\lneqq</code>	$\not\geq\!\!\!-\!$	<code>\gneqq</code>	$\not\mid$	<code>\nmid</code>
$\not\leq\!\!\!-\!$	<code>\lvertneqq</code>	$\not\geq\!\!\!-\!$	<code>\gvertneqq</code>	$\not\parallel$	<code>\nparallel</code>
$\not\leq\!\!\!-\!$	<code>\lvertneqq</code>	$\not\geq\!\!\!-\!$	<code>\gvertneqq</code>	$\not\shortmid$	<code>\nshortmid</code>
$\not\sim\!\!\!-\!$	<code>\lnsim</code>	$\not\sim\!\!\!-\!$	<code>\gnsim</code>	$\not\shortparallel$	<code>\nshortparallel</code>
$\not\approx\!\!\!-\!$	<code>\lnapprox</code>	$\not\approx\!\!\!-\!$	<code>\gnapprox</code>	$\not\sim$	<code>\nsim</code>
$\not\prec$	<code>\nprec</code>	$\not\succ$	<code>\nsucc</code>	$\not\cong$	<code>\ncong</code>
$\not\preceq$	<code>\npreceq</code>	$\not\sucess$	<code>\nsuccceq</code>	$\not\dash$	<code>\nvDash</code>
$\not\preceq\!\!\!-\!$	<code>\precneqq</code>	$\not\preceq\!\!\!-\!$	<code>\succneqq</code>	$\not\dash$	<code>\nvDash</code>
$\not\preceq\!\!\!-\!$	<code>\precnsim</code>	$\not\preceq\!\!\!-\!$	<code>\succcnsim</code>	$\not\dash$	<code>\nVdash</code>
$\not\approx\!\!\!-\!$	<code>\precnapprox</code>	$\not\approx\!\!\!-\!$	<code>\succcnapprox</code>	$\not\dash$	<code>\nVdash</code>
$\not\subsetneq$	<code>\subsetneq</code>	$\not\supsetneq$	<code>\supsetneq</code>	$\not\triangleleft$	<code>\ntriangleleft</code>
$\not\subsetneq$	<code>\varsubsetneq</code>	$\not\supseteq$	<code>\varsupsetneq</code>	$\not\triangleright$	<code>\ntriangleright</code>
$\not\subsetneq$	<code>\nsubsetneq</code>	$\not\supseteq$	<code>\nsubseteq</code>	$\not\trianglelefteq$	<code>\ntrianglelefteq</code>
$\not\subsetneq$	<code>\subsetneqq</code>	$\not\supsetneqq$	<code>\supsetneqq</code>	$\not\trianglerighteq$	<code>\ntrianglerighteq</code>
$\Leftarrow\!\!\!-\!$	<code>\nleftarrow</code>	$\Rightarrow\!\!\!-\!$	<code>\nrightarrow</code>	$\Leftarrow\!\!\!-\!$	<code>\nleftrightarrow</code>
$\Leftarrow\!\!\!-\!$	<code>\nLeftarrow</code>	$\Rightarrow\!\!\!-\!$	<code>\nRightarrow</code>	$\Leftarrow\!\!\!-\!$	<code>\nLeftrightarrow</code>

表 4.20: \mathcal{AM} S 定界符

\ulcorner	<code>\ulcorner</code>	\urcorner	<code>\urcorner</code>	\llcorner	<code>\llcorner</code>	\lrcorner	<code>\lrcorner</code>
-------------	------------------------	-------------	------------------------	-------------	------------------------	-------------	------------------------

表 4.21: *AMS* 其它符号

\hbar	\hbar	\hbar	\hslash	\mathbb{k}	\Bbbk
\square	\square	\blacksquare	\blacksquare	\circledS	\circledS
\triangle	\vartriangle	\blacktriangle	\blacktriangle	\complement	\complement
\triangledown	\triangledown	\blacktriangledown	\blacktriangledown	\Game	\Game
\lozenge	\lozenge	\blacklozenge	\blacklozenge	\bigstar	\bigstar
\angle	\angle	\measuredangle	\measuredangle	\prime	\backprime
\diagup	\diagup	\diagdown	\diagdown	\varnothing	\varnothing
\nexists	\nexists	\Finv	\Finv	\varnothing	\varnothing
\eth	\eth	\sphericalangle	\sphericalangle	\mho	\mho

排版样式设定

字体和字号

latex字体字号

latex根据文档的逻辑结构来选择默认的字体样式以及字号. 需要更改字体样式或字号的话, 可以使用以下命令.

表 5.1: 字体命令

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

表 5.2: 字号

\tiny	tiny font	\backslash Large	larger font
\scriptsize	very small font	\backslash LARGE	very large font
\footnotesize	quite small font	\backslash huge	huge
\small	small font	\backslash Huge	largest
\normalsize	normal font		
\large	large font		

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

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

需要注意的是，在公式中直接使用 `\textbf` 等命令不会起效，甚至报错。公式中需要使用 `\mathbf` 等命令，详见[数学字母字体](#)小节。

字号命令实际大小依赖于所使用的文档类及其选项。

表 5.3: 标准文档类中的字号大小

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

latex还提供了一个基础的命令 `\fontsize{<size>}{{<base line-skip>}}` 用于设定任意大小的字号。其中 `<size>` 为字号，`<base line-skip>` 为基础行距。默认的字号基础行距都设定为字号的1.2倍。若不是在导言区，`\fontsize` 的设定需要 `\selectfont` 命令才能立即生效。

字体宏包

latex默认字体是由高德纳设计制作的Computer Modern字体。若要使用其他字体，一种方法是利用字体宏包完成的整套配置，下表是常用的字体宏包

表 5.4: 常见的 L^AT_EX 字体宏包

文本/数学字体搭配的宏包	
lmodern	Latin Modern 字体，对 Computer Modern 字体的扩展
cmbright	仿 Computer Modern 风格的无衬线字体
euler	Euler 风格数学字体，也出自于高德纳之手
ccfonts	Concrete 风格字体
txfonts	Times 风格的字体宏包
pxfonts	Palatino 风格的字体宏包
stix	Times 风格的字体宏包
newtxttext,newtxmath	txfonts 的改进版本，分别设置文本和数学字体
newpxtext,newpxmath	pxfonts 的改进版本，分别设置文本和数学字体
mathptmx	psnfss 字体宏集之一，Times 风格，较为陈旧，不推荐使用
mathpazo	psnfss 字体宏集之一，Palatino 风格，较为陈旧，不推荐使用
fourier	Fourier 风格数学字体，配合 Utopia 正文字体
fouriernc	Fourier 风格数学字体，配合 New Century Schoolbook 正文字体
arev	Arev 无衬线字体宏包，Vera Sans 风格
mathdesign	配合 Charter/Garamond/Utopia 正文字体的数学字体宏包
文本字体宏包	
以下字体包括传统的 L ^A T _E X 字体格式以及 TrueType/OpenType 格式。	
cm-unicode	Computer Modern 风格的 Unicode 字体，支持多种西方语言
dejavu	DejaVu 开源字体
droid	Droid 开源字体
inconsolata	Inconsolata 开源等宽字体
libertine	Linux Libertine 和 Linux Biolum 开源字体
roboto	Roboto 开源无衬线字体
sourcesanspro	Source Sans Pro 开源无衬线字体
sourcecodepro	Source Code Pro 开源等宽字体
符号宏包	
mathabx	数学符号宏包之一
MnSymbol	数学符号宏包之一，配合 Minion Pro 文本字体
fdsymbol	数学符号宏包之一
pifont	Zapf Dingbats 符号宏包

另一种方法是使用`fontspec`更改字体（通过 `xelatex` 或 `lualatex` 编译命令直接调用系统和tex发行版中的 `.ttf` 或 `.otf` 格式字体）：

```
\setmainfont{<font name>}[<font features>]
\setsansfont{<font name>}[<font features>]
\setmonofont{<font name>}[<font features>]
```

其中 `` 使用字体的文件名（带拓展名）或字体的英文名称。`` 用来手动配置对应的粗体或斜体，例如使用Windows下的无衬线字体Arial配置粗体和斜体（通常情况下自动检测并配置，无需手动指定）：

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

`` 还能配置字体本身的各种特性，例如字号大小等等。

需要注意的是，*fontspec*宏包会覆盖数学字体设置，需要调用上表中的一些数学字体宏包时，应当在调用*fontspec*宏包时指定 `no-math` 选项。*fontspec*宏包可能被其它宏包或文档类（如`ctex`文档类）自动调用，需要在文档开头的 `\documentclass` 命令中指定 `no-math` 选项。

`ctex`宏包或文档类提供了与*fontspec*宏包非常类似的语法设置中文字体：

```
\setCJKmainfont{<font name>}{<font features>}  
\setCJKsansfont{<font name>}{<font features>}  
\setCJKmonofont{<font name>}{<font features>}
```

由于中文字体少有对应的粗体或斜体，`` 里多用其他字体来配置，比如在Windows中设置基本字体为宋体，并设定对应的 `BoldFont` 为黑体，`ItalicFont` 为楷体：

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

使用*unicode-math*宏包配置Unicode数学字体

Unicode数学字体是一类OpenType字体，包含了Unicode字符集中的数学符号部分，在导言区使用 `\usepackage{unicode-math}` 后，使用 `\setmathfont{}{}` 命令即可。

绝大多数时候，只需要给定字体名称 `` 即可。实际上，[数学字母字体](#)和[加粗的数学符号](#)小节中的内容均已被 *unicode-math*宏包所覆盖，无需调用其他宏包就可以获得统一的字体样式。

文字装饰和强调

强调文字的分式，或是添加下划线等装饰物，或者是改变文字的字体。

`latex`中定义了 `\underline` 命令用来为文字添加下划线：

```
An \underline{underlined} text.
```

|| An underlined text.

`\underline` 命令生成下划线的样式不够灵活，不同的单词可能生成高低各异的下划线，并且无法换行。*ulem*宏包提供了更灵活的 `\uline` 命令：

```
An example of \uline{some long and underlined words.}
```

|| An example of some long and underlined words.

`\emph` 命令将文字变为斜体以示强调，若在已强调的文字中嵌套使用 `\emph` 命令，则变为直立体文字：

```
Some \emph{emphasized words}, including \emph{double-emphasized} words, are shown here.
```

Some *emphasized words*, including double-emphasized words, are shown here.

段落格式和间距

长度和长度变量

长度的数值 `<length>` 由数字和单位组成，常见单位如下：

表 5.6: TeX/LaTeX 中的长度单位

pt	点 (point, 也译作“磅”), $= 1/72.27 \text{ in}$
bp	大点 (big point), $= 1/72 \text{ in}$
in	英寸, $= 2.54 \text{ cm}$
cm	厘米
mm	毫米
em	大致相当于当前字号下大写字母 M 的宽度, 常用于设定水平距离
ex	大致相当于当前字号下小写字母 x 的高度, 常用于设定垂直距离
mu	数学单位 (math unit), $= 1/18 \text{ em}$

还可以使用可伸缩的“弹性长度”，如 `12pt plus 2pt minus 3pt` 表示基础长度为 `12pt`，可以伸展到 `14pt`，也可以收缩到 `9pt`。其中 `plus` 和 `minus` 可以只定义其中一个。

长度的数值还可以用长度变量本身或其倍数来表示，如 `2.5\parindent` (2.5倍段落缩进) 等。

LaTeX 预定义了大量的长度变量用于控制版面格式，如页面宽度和高度等，有以下命令来自定义、赋值或增加长度：

```
\newlength{\<length command>} % 自定义长度变量  
\setlength{\<length command>}{<length>} % 赋值长度  
\addtolength{\<length command>}{<length>} % 增加长度
```

行距

前文提到使用 `\fontsize` 命令为字号设定对应的行距，但实际上更常用的方法是在导言区使用 `\linespread{<factor>}` 命令。其中 `<factor>` 作用于基础行距而不是字号。缺省的基础行距是 1.2 倍字号大小，因此使用 `\linespread{1.5}` 意味着最终行距为 1.8 倍的字号大小。

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

```
{\linespread{2.0}\selectfont  
The baseline skip is set to be twice the normal baseline skip. Pay attention to the \verb|\par| command at  
the end. \par}
```

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

The baseline skip is set to be twice the normal baseline skip. Pay attention to the \par command at the end.

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

字号的改变使立即生效的，而行距的改变直到文字**分段**时才生效。

段落格式

以下长度分别为段落的左缩进、右缩进和首行缩进：

```
\setlength{\leftskip}{<length>}  
\setlength{\rightskip}{<length>}  
\setlength{\parindent}{<length>}
```

它们和设置行距的命令一样，在分段时生效。控制段落缩进的命令为：

```
\indent  
\noindent
```

latex默认在段落开始时缩进，长度为用上述命令设置的 \parindent，如果需要在某一段不缩进，可在段落开头使用 \noindent 命令。相反地， \indent 命令强制开启一段首行缩进的段落。在段落开头使用多个 \indent 可以累加缩进量。

latex默认在 \chapter、\section 等章节标题命令后的第一段不缩进（使用ctex宏包和文档类默认第一段首行缩进）。若不习惯，可以调用 indentfirst 宏包令第一段的首行缩进照常。

段落间的垂直间距为 \parskip，下例设置段落间距在 0.8ex 到 1.5ex 变动：

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

水平间距

\hspace 命令可以在文中插入额外的水平间距：

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

This is a space of 1.5 cm.

\hspace 命令生成的水平间距若在一行的开头或末尾，则有可能因为断行而被舍弃。可以使用 \hspace* 命令代替 \hspace 命令得到不会因断行而消失的水平间距。

命令 \stretch{<n>} 生成一个特殊弹性长度（类似文字的分散对齐），参数 <n> 为权重。基础长度为 0pt，但可以无限延伸，直到占满可用的空间。若一行内出现多个 \stretch{<n>}，这一行的所有可用空间将按照每个 \stretch{<n>} 的权重 <n> 进行分配。

命令 `\fill` 相当于 `\stretch{1}` (注意不能使用 `1.5\fill` 或 `1.5\stretch{<n>}` 这样的用法, 其生成的长度只有基础长度 `0pt`)

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



在正文中使用 `\hspace` 命令生成水平间距时, 往往使用 `em` 作为单位, 生成的间距随字号大小而变. `\quad` 和 `\quad` 命令相当于 `\hspace{1em}` 和 `\hspace{2em}`.

```
{\Large big\hspace{1em}y} \\
{\Large big\quad y} \\
nor\hspace{2em}mal \\
nor\quad mal \\
{\tiny tin\hspace{1em}y} \\
{\tiny tin\quad y}
```



```
big    y
big    y
nor    mal
nor    mal
tin    y
tin    y
```

垂直间距

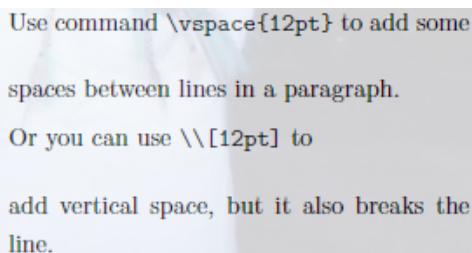
latex中段落、章节标题等元素之间的间距时预设定的, 例如 `\parskip` 默认设置为 `0pt plus 1pt`.

若想人为增加段落之间的垂直间距, 可以在两个段落之间的位置使用 `\vspace{<length>}` 的命令. 与 `\hspace` 相同, `\vspace` 生成的垂直间距在一页的顶端或底端可能被“吞掉”, 可以使用 `\vspace*` 命令产生不会因断页而消失的垂直间距. `\vspace` 也可以用 `\stretch` 设置弹性间距.

在段落的两行之间增加垂直距离, 还可以通过断行命令 `\\\` 增加可选参数, 如 `\\\[6pt]` 或 `*[6pt]`. `\vspace` 也可以在段落内使用, 区别在于 `\vspace` 只引入垂直间距而不断行.

Use command `\verb|\vspace{12pt}|` to add `\vspace{12pt}` some spaces between lines in a paragraph.

Or you can use `\verb|\\\[12pt]|` to `\\\[12pt]` add vertical space, but it also breaks the line.



Use command `\vspace{12pt}` to add some
spaces between lines in a paragraph.
Or you can use `\\\[12pt]` to
add vertical space, but it also breaks the
line.

latex还提供了 `\bigskip`、`\medskip`、`\smallskip` 来增加预定义长度的垂直距离

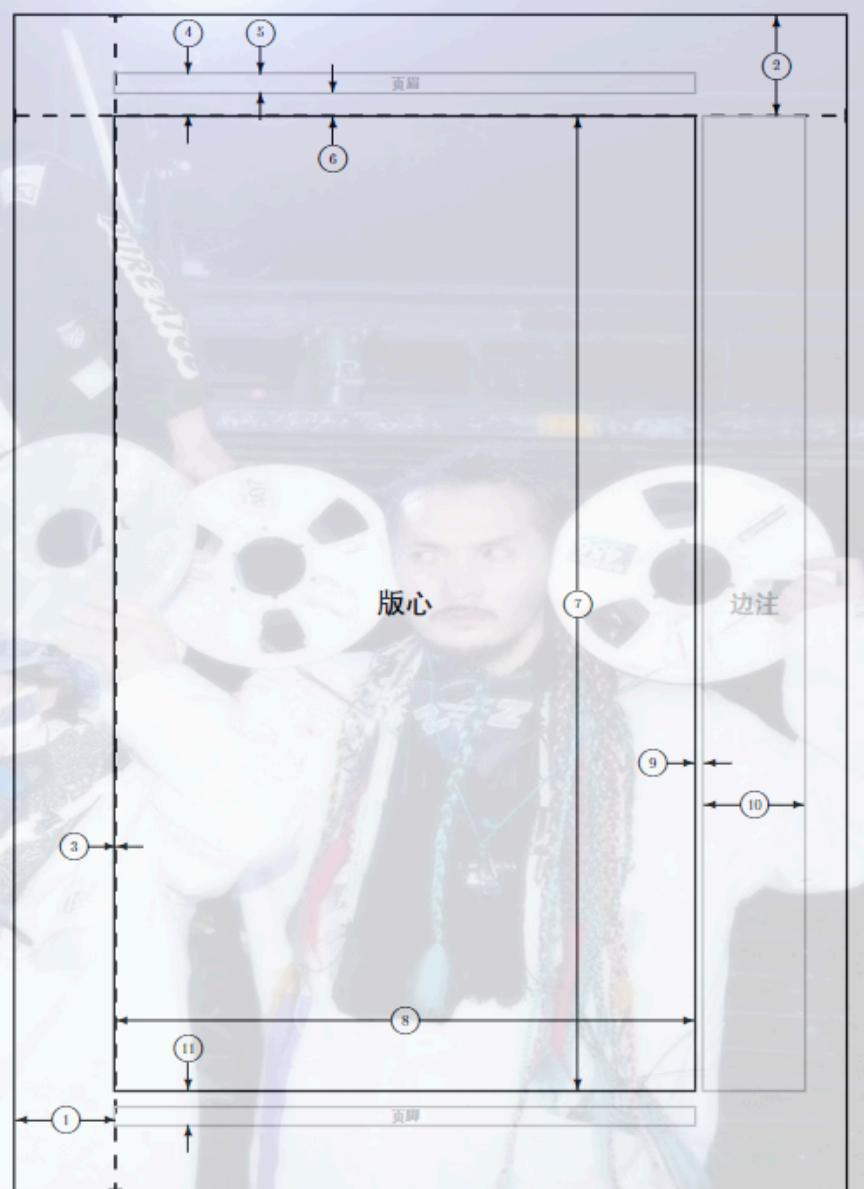
```
\parbox[t]{3em}{TeX\parTeX}
\parbox[t]{3em}{TeX\par\smallskipTeX}
\parbox[t]{3em}{TeX\par\medskipTeX}
\parbox[t]{3em}{TeX\par\bigskipTeX}
```

TeX	TeX	TeX	TeX
TeX	TeX	TeX	TeX

页面和分栏

latex允许用户通过文档类指定选项控制纸张的大小（见[文档类](#)节），包括 `a4paper`、`letterpaper` 等等，并配合字号设置了适合的页边距。

控制页边距的参数由下图给出的各种长度变量控制，可以通过 `\setlength` 命令修改这些参数。也可以使用这些变量决定排版内容的尺寸，如在 `tabularx` 环境或 `\includegraphics` 命令的参数里，设置图片或表格的宽度为 `0.8\textwidth`。



```

1 1in + \hoffset          2 1in + \voffset
3 \oddsidemargin = 0pt    4 \topmargin = -30pt
5 \headheight = 13pt       6 \headsep = 18pt
7 \textheight = 700pt      8 \textwidth = 416pt
9 \marginparsep = 7pt      10 \marginparwidth = 72pt
11 \footskip = 25pt        \marginparpush = 5pt (未显示)
\hoffset = 0pt             \voffset = 0pt
\paperwidth = 597pt         \paperheight = 845pt

```

图 5.1: 本文档的页面参数示意图 (奇数页; 由 layout 宏包生成)。

页边距等比较直观的参数必须间距设置, 以奇数页为例:

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

若要设置合适的 $\langle left-margin \rangle$ 等, 就需要通过以上方程组把 \oddsidemargin 等参数求解出来.

或者利用`geometry`宏包.

利用`geometry`宏包设置页面参数

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

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

也可以直接在宏包选项中设置：

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

其中 `<geometry-settings>` 多以 `<key>=<value>` 的形式组织.

例如设定A4纸张，上下边距1英寸，左右边距1.25英寸，可以使用以下方式中的一种设定：

```
\geometry{a4paper, left=1.25in, right=1.25in, top=1in, bottom=1in}  
% or like this:  
\geometry{a4paper, hmargin=1.25in, vmargin=1in}
```

又例如设定周围边距一致为1.25英寸，可以用更简单的语法：

```
\geometry{margin=1.25in}
```

对于书籍等双面文档，习惯上奇数页右边和偶数页左边留出较大页边距，而靠近书脊一侧的页边距较小：

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

详见帮助文档.

页面内容的垂直对齐

latex默认将页面内容在垂直方向分散对齐，这可能会导致页面排版不匀称，某些页面的垂直间距过宽，甚至报大量 `Underfull \vbox` 警告. latex还提供了另一种策略：将页面内容向顶部对齐，给底部留出高度不一的空白. 以下命令分别令页面在垂直方向向顶部对齐/分散对齐：

```
\raggedbottom  
\flushbottom
```

分栏

latex支持简单的单栏或双栏排版. 标准文档类的全局布局选项 `onecolumn`、`twocolumn` 可控制全文单栏或双栏排版. latex也提供了切换单/双栏排版的命令：

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

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

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

双栏排版时每一栏的宽度为 \columnwidth，由 \textwidth 减去 \columnsep 再除以2得到。双栏之间还有一道竖线，宽度为 \columnseprule， 默认为0，即看不到竖线。

一个比较好用的分栏方案是调用 *multicol* 宏包，它提供了简单的 *multicols* 环境，自动产生分栏，例如以下环境将内容分为3栏：

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

multicol 宏包可以在一页之中切换单栏/多栏，也能处理跨页的分栏，但在 **multicols 环境中无法使用 table 和 figure 等浮动体环境**，浮动体会直接丢失。*multicols* 环境中只能用跨栏的 *table** 和 *figure** 环境，或者用 *float* 宏包提供的 *H* 参数固定浮动体的位置。

页眉页脚

基本的页眉页脚样式

latex 中提供了命令 \pagestyle{<page-style>} 来修改页眉页脚的样式。命令 \thispagestyle{<page-style>} 只影响当页的页眉页脚样式。其中 <page-style> 为样式的名称，在 latex 里预定义了四类样式：

- empty：页眉页脚为空
- plain：页眉为空，页脚为页码。（article 和 report 文档类默认；book 文档类的每章第一页也为 plain 格式）
- headings：页眉为章节标题和页码，页脚为空。（book 文档类默认）
 - article 文档类，twoside 选项 偶数页为页码和节标题，奇数页为小节标题和页码；
 - article 文档类，oneside 选项 页眉为节标题和页码；
 - report 和 book 文档类，twoside 选项 偶数页为页码和章标题，奇数页为节标题和页码；
 - report 和 book 文档类，oneside 选项 页眉为章标题和页码
- myheadings：页眉为页码及 \markboth 和 \markright 命令手动指定的内容，页脚为空。

\pagenumbering{<style>} 命令能够改变页眉页脚中的页码样式。其中 <style> 为页码样式，默认为 arabic（阿拉伯数字），还可修改为 roman（小写罗马数字）、Roman（大写罗马数字）等。注意使用 \pagenumbering 命令后会将页码重置为1。book 文档类的 \frontmatter 和 \mainmatter 内部就使用了 \pagenumbering 命令切换页码样式。对页码格式的详细说明见 [计数器](#) 节。

手动更改页眉页脚的内容

对于 `headings` 或者 `myheadings` 样式, latex 允许用户修改**页眉**上的内容, 特别是因为使用了 `\chapter*` 等命令而无法自动生成页眉页脚的情况:

```
\markright{<right-mark>}
\markboth{<left-mark>}{<right-mark>}
```

在双面排版、`headings` 或 `myheadings` 页眉页脚样式下, `<left-mark>` 和 `<right-mark>` 的内容分别预期出现在偶数页和奇数页. 事实上 `\chapter` 和 `\section` 等章节目录内部也使用了 `\markboth` 或者 `\markright` 生成页眉.

latex 默认将页眉的内容都转为大写字母. 若要保持字母的大小写, 可以尝试以下代码 (但是着不能改变页眉的斜体样式 (`\slshape`), 斜体是定义在 `headings` 样式里的, 可以在 `\markboth` 等命令的参数里先使用 `\normalfont`, 再使用想要的字体样式命令, 或直接使用 `fancyhdr` 宏包) :

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

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

fancyhdr宏包

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

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

```
\fancyhf[<position>]{...}
\fancyhead[<position>]{...}
\fancyfoot[<position>]{...}
```

其中 `<position>` 为 `L` (左) / `C` (中) / `R` (右) 以及与 `o` (奇数页) / `E` (偶数页) 的组合. `\fancyhf` 用于同时定义页眉和页脚, 习惯上使用 `\fancyhf{}` 来情况页眉页脚的设置.

下例给出了 `fancyhdr` 基础的用法, 效果为将章节标题放在与 `headings` 一致的位置, 但使用加粗格式; 页码都放在页脚正中; 修改横线宽度, “去掉”页脚的横线.

```
% 在导言区使用此代码
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]{\markboth{\#1}{}}%
\renewcommand{\sectionmark}[1]{\markright{\thesection \ #1}}%
\fancyhf{}
\fancyfoot[C]{\bfseries\thepage}
\fancyhead[LO]{\bfseries\rightmark}
\fancyhead[RE]{\bfseries\leftmark}
\renewcommand{\headrulewidth}{0.4pt} % 注意不用 \setlength
\renewcommand{\footrulewidth}{0pt}
```

*fancyhdr*还支持用 `\fancypagestyle` 为自定义的页眉页脚样式命令，或者重新定义已有的样式如 `plain` 等：

```
% 自定义 myfancy 样式
\fancypagestyle{myfancy}%
  \fancyhf{}
  \fancyhead{...}
  \fancyfoot{...}
}
% 使用样式
\pagestyle{myfancy}
```

特色工具和功能

参考文献和bibtex工具

基本的参考文献和引用

latex提供的参考文献和引用方式比较原始，较难直接使用。

latex提供了最基本的 `\cite{<citation>}` 命令用于在正文中引用参考文献。`<citation>` 为引用的参考文献的标签，类似 `\ref` 里的参数； `\cite` 带一个可选参数，为引用的编号后加上额外的内容，如 `\cite[page 22]{Paper2013}` 可能得到形如 [13, page 22] 这样的引用。

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

```
\begin{thebibliography}{<widest label>}
  \bibitem[<item number>]{<citation>} ...
\end{thebibliography}
```

其中 `<citation>` 是 `\cite` 使用的文献标签，`<item number>` 自定义参考文献的序号，若省略，则按照自然排序给定序号。`<widest label>` 用以限制参考文献序号的宽度，如99意味着不超过两位数字。通常设定为与参考文献的数目一致。

在article文档类中，`thebibliography` 环境自动生成不带编号的一节，标题默认为 "References"；而在report或book文档类中，则会生成不带编号的一章，标题默认为 "Bibliography"。用户可通过[latex可定制的一些命令和参数](#)节给出的方法

定制参考文献的标题.

以下为一个使用 `\thebibliography` 排版参考文献的例子:

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

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

bibtex数据库

bibtex是最为流行的参考文献数据组织格式之一. 可以通过参考文献样式的支持, 让同一份bibtex数据库生成不同样式的参考文献列表.

bibtex数据库以 `.bib` 为拓展名, 内容是若干个文献条目, 每个条目的格式为:

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

其中 `<type>` 为文献的类型, 如 `article` 为学术论文, `book` 为书籍, `incollection` 为论文集中的一篇等等.

`<citation>` 为 `\cite` 命令使用的文献标签. 在 `<citation>` 之后为条目里的各个字段, 以 `<key> = {<value>}` 的形式组织.

在此简单列举学术论文里使用较多的bibtex文献条目类别:

- `article` 学术论文, 必须字段有 `author`, `title`, `journal`, `year`; 可选字段包括 `volume`, `number`, `pages`, `doi` 等;
- `book` 书籍, 必须字段有 `author/editor`, `title`, `publisher`, `year`; 可选字段包括 `volume/number`, `series`, `address` 等;
- `incollection` 论文集中的一篇, 必需字段有 `author`, `title`, `booktitle`, `publisher`, `year`; 可选字段包括 `editor`, `volume/number`, `chapter`, `pages`, `address` 等;
- `inbook` 书中的一章, 必需字段有 `author/editor`, `title`, `chapter/pages`, `publisher`, `year`; 可选字段包括 `volume/number`, `series`, `address` 等.

例如 `article` 类别的参考文献数据条目写法如下:

```

@article{Alice13,
  title = {Demostration of bibliography items},
  author = {Alice Axford and Bob Birkin and Charlie Copper and Danny Dannford},
  year = {2013},
  journal = {Journal of \TeX perts},
  volume = {36},
  number = {7},
  pages = {114-120}
}

```

所有类别的文献条目格式参考CTAN://biblio/bibtex/base/btxdoc.pdf

bibtex样

参考文献的写法在不同文献里千差万别. bibtex用样式 (style) 来管理参考文献的写法. bibtex提供了几个预定义的样式, 如 plain, unsrt, alpha 等. 若使用期刊模板的话, 可能会提供自用的样式, 样式文件以 .bst 为拓展名.

使用样式文件的方法是在源代码内 (一般在导言区) 使用 `\bibliographystyle{<bst-name>}` 命令. 这里 `<bst-name>` 为 .bst 样式文件的名称, **不带 .bst 拓展名**.

以**bibtex数据库**中给出的数据条目为例, 使用 `\bibliographystyle` 命令选择不同的参考文献样式, 效果大致如下

表 6.1: BibTeX 样式的排版效果

plain

[1] Alice Axford, Bob Birkin, Charlie Copper, and Danny Dannford. Demostration of bibliography items. *Journal of TeXperts*, 36(7):114–120, Mar 2013.

alpha

[ABCD13] Alice Axford, Bob Birkin, Charlie Copper, and Danny Dannford. Demostration of bibliography items. *Journal of TeXperts*, 36(7):114–120, Mar 2013.

abbrv

[1] A. Axford, B. Birkin, C. Copper, and D. Dannford. Demostration of bibliography items. *Journal of TeXperts*, 36(7):114–120, Mar 2013.

amsplain (*AMS* 文档类 *amsart* 等配套的样式)

[1] Alice Axford, Bob Birkin, Charlie Copper, and Danny Dannford, *Demostration of bibliography items*, Journal of TeXperts **36** (2013), no. 7, 114–120.

elsarticle-num (Elsevier 提供的 *elsarticle* 文档类配套的样式)

[1] A. Axford, B. Birkin, C. Copper, D. Dannford, Demostration of bibliography items, Journal of TeXperts 36 (7) (2013) 114–120.

IEEEtran (IEEEtran 模板文档类配套的样式)

[1] A. Axford, B. Birkin, C. Copper, and D. Dannford, “Demostration of bibliography items,” *Journal of TeXperts*, vol. 36, no. 7, pp. 114–120, Mar 2013.

gbt7714-numerical (GB/T 7714—2015 样式, 由 gbt7714 宏包提供)

[1] 陈登原. 国史旧闻: 第 1 卷 [M]. 北京: 中华书局, 2000: 29.

使用bibtex排版参考文献

第一步：准备一份bibtex数据库，假设数据库文件名为 `books.bib`，和latex源代码一般位于同一个目录下。

第二步：在源代码中添加必要的命令。假设源代码名为 `demo.tex`。

1. 首先使用命令 `\bibliographystyle` 设定参考文献的格式。
2. 其次，在正文中引用参考文献。bibtex程序在生成参考文献列表的时候，通常只列出用于 `\cite` 命令引用的那些。若需要列出未被引用的文献，则需要 `\nocite{<citation>}` 命令；而 `\nocite{*}` 则让所有未被引用的文献都列出。
3. 再次，在需要列出参考文献的位置，使用 `\bibliography{<bib-name>}` 命令代替 `\thebibliography` 环境。其中 `<bib-name>` 是bibtex数据库的文件名，**不带 .bib 拓展名**。

```
% demo.tex
\documentclass{article}
\bibliographystyle{plain}

\begin{document}
\section{Some words}
Some excellent books, for example, \cite{citation1}
and \cite{citation2} \ldots

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

注意：`\bibliographystyle` 和 `\bibliography` 命令缺一不可，否则生成参考文献列表时会报错。

第三步：写好以后，可以直接编译（先 `xelatex` 编译一次，再 `bibtex` 编译一次，最后 `xelatex` 再编译两次）

natbib宏包

许多学术期刊喜欢使用人名——年份的引用方式，形如 *(Axford et al, 2013)*。`natbib`宏包提供了对这种引用方式的处理。

除了 `\cite` 之外，`natbib`宏包在正文中支持两个引用方式：

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

分别生成形如 *(Axford et al, 2013)* 和 *Axford et al. (2013)* 的人名——年份引用。正确排版该引用还依赖特定的 bibtex样式。`natbib`提供了latex预定义样式相对于的几个样式，包括 `plainnat`、`abbrvnat` 和 `unsrtnat`。学术论文模板是否支持 `natbib`需要参考帮助文档。

`natbib`宏包也支持数字引用，并且支持将引用的序号压缩，例如：

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

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

*natbib*宏包还有其他选项和用法，例如默认的引用使用小括号包裹的，可指定 `square` 选项改为中括号；再比如 `\citet` 命令也支持可选参数，为引用其后都添加额外内容。详见*natbib*宏包的帮助文档。

biblatex宏包

*biblatex*宏包提供了便捷的格式控制和强大的排序、分类、筛选、多文献表等功能。国内较多*latex*模板都使用*biblatex*，因其对UTF-8和中文参考文献的良好支持。

基于*biblatex*宏包的方式与基于**bibtex**的传统方式有一定区别。

文档结构和*biblatex*相关命令

- 首先在导言区调用*biblatex*宏包。宏包支持以 `<key>=<value>` 形式指定选项，包括参考文献样式 `style`、参考文献著录排序的规则 `sorting` 等。
- 接着在导言区使用 `\addbibrse` 命令为*biblatex*引入参考文献数据库。与基于**bibtex**的传统方式不同的是，这里需要写完整的文件名。
- 在正文中使用 `\cite` 命令引用参考文献。除此之外还可以使用丰富的命令达到不同的引用效果，如 `\citeauthor` 和 `\citeyear` 分别单独引用作者和年份，`\textcite` 和 `\parencite` 分别类似 *natbib*宏包提供的 `\citet` 和 `\citet` 命令，以及脚注式引用 `\footcite` 等。
- 最后需要在排版参考文献的位置使用命令 `\printbibliography`。

编译方式

只需要将基于**bibtex**的编译方式中的 `bibtex` 命令改为 `biber` 即可

*biblatex*样式和其他选项

*biblatex*使用的参考文献样式分为著录样式 (bibliography style) 和引用样式 (citation style)，分别以 `.bbx` 和 `.cbx` 为扩展名。参考文献的样式在调用宏包时使用 `style` 选项指定，或者使用 `bibstyle` 或 `citestyle` 分别指定：

```
% 同时调用 gb7714-2015.bbx 和 gb7714-2015.cbx
\usepackage[style=gb7714-2015]{biblatex}
% 著录样式调用 gb7714-2015.bbx，引用样式调用 biblatex 宏包自带的 authoryear
\usepackage[bibstyle=gb7714-2015,citestyle=authoryear]{biblatex}
```

以下是一些常用的参考文献样式，除*biblatex*宏包自带的样式外，许多样式以单独的宏包发布。

- `authoryear` *biblatex*自带样式，类似*natbib*默认的引用样式效果。
- `authortitle` *biblatex*自带样式，采用作者——题目 (shorttitle字段) 的引用样式。
- `verbose` *biblatex*自带样式，引用样式中包含作者、题目、书名、页码等字段的信息。
- `alphabetic` *biblatex*自带样式，著录样式与**bibtex**的 `alpha` 样式类似。
- `trad-alpha` *biblatex-trad*样式包，移植自**bibtex**默认的 `alpha` 样式，另外还包括 `trad-abbrv`、`trad-plain` 和 `trad-unsrt`。
- `gb7714-2015` 复合中文文献著录标准 GB/T 7714-2015 的样式，著录按顺序编码排版。另外还包括按作者——年份顺序排版著录的格式 `gb7714-2015ay`。
- `caspervector` 以中文文献著录标准 GB/T 7714-2015 为基础的一个样式。
- `ieee` 兼容 IEEEtran 风格的样式，著录按顺序编码排版。另外还包括按作者-年份顺序排版著录的样式 `ieee-alphabetic`。

索引和 makeindex 工具

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

使用 makeindex 工具的方法

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

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

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

第三步：编译过程：

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

索引项的写法

添加索引项的命令为：

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

其中 `<index entry>` 为索引项，写法由下表汇总。其中 `!`、`@` 和 `|` 为特殊符号，若要向索引项直接输出这些符号，需要加前缀 `"`；而 `"` 需要两个引号 `""` 才能输出到索引项。

表 6.2: 索引项的写法列表

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

以下是结合成复杂的索引示例:

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

```
 "Test, 1-2  
 |sub|, see Test
```

使用颜色

LaTeX 原始并不支持各种颜色, 需要使用 `color` 宏包或者 `xcolor` 宏包.

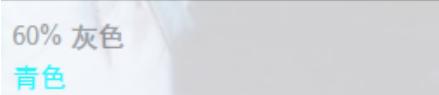
颜色的表达方式

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

```
\color[<color-mode>]{<code>}  
\color{<color-name>}
```

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

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



其二是直接使用名称代替颜色，前提是已经定义了颜色名称（没定义会报错）：

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



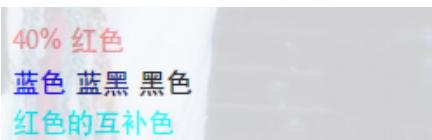
*color*宏包仅定义了8种颜色名称，*xcolor*补充了一些，总共有19种。

表 6.3: *color* 和 *xcolor* 宏包可用的颜色名称

基本的 8 种颜色名称 (后三种颜色为 CMYK 模型)			
black	red	green	blue
white	cyan	magenta	yellow
<i>xcolor</i> 额外可用的颜色名称:			
darkgray	gray	lightgray	
brown	olive	orange	lime
purple	teal	violet	pink

*xcolor*还支持颜色通过表达式混合或互补：

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



还可以通过命令自定义颜色名称，注意这里的 *<color-mode>* 是必选参数：

```
\definecolor{<color-name>}[<color-mode>]{<code>}
```

如果调用`color`或`xcolor`宏包时指定 `dvipsnames` 选项 (`color`宏包实际要指定 `dvipsnames` 和 `usenames` 选项) , 就有额外的68种颜色名称可用. `xcolor`还可以通过指定其他选项载入更多颜色名称, 详见手册.

带颜色的文本和盒子

原始的 `\color` 命令类似于字体命令 `\bfseries` , 使之后的排版内容全部变为指定的颜色, 所以直接使用时通常要加花括号分组. `color`和`xcolor`宏包都定义了一些方便的带颜色元素.

输入带颜色的文本可用用类似 `\textbf` 的命令:

```
\textcolor[<color-mode>]{<code>}{<text>}  
\textcolor{<color-name>}{<text>}
```

以下命令构造一个带背景色的盒子, `<material>` 为盒子中的内容:

```
\colorbox[<color-mode>]{<code>}{<material>}  
\colorbox{<color-name>}{<material>}
```

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

```
\fcolorbox[<color-mode>]{<fcode>}{<code>}{<material>}  
\fcolorbox{<fcolor-name>}{<color-name>}{<material>}
```

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



这里的 `\fcolorbox` 也可以像带框的水平盒子小节里的 `\fbox` 那样调节 `\fboxrule` 和 `\fboxsep` ; 对于 `\colorbox` , 调整 `\fboxsep` 是有效的.

使用超链接

latex中实现超链接功能的是`hyperref`宏包.

`hyperref`宏包

`hyperref`宏包将latex的目录、引用、脚注、索引、参考文献等等都包装成超链接, 但这也使它与其它宏包发生冲突的可能性大大增加. 为减少可能的冲突, 习惯上将`hyperref`宏包放在其它宏包之后调用.

`hyperref`宏包提供了命令 `\hypersetup` 配置各种参数. 参数也可以作为宏包选项, 在调用宏包时指定:

```
\hypersetup{<option1>, <option2>=<value>, ...}
\usepackage[<option1>, <option2>=<value>, ...]{hyperref}
```

当选项值为 `true` 时，可以省略 `=true` 不写。可用参数见下。

表 6.4: `hyperref` 宏包提供的参数设置

参数	默认值	含义
<code>draft=(true false)</code>	<code>false</code>	关闭所有超链接、书签等功能（也可以通过文档类选项指定）
<code>final=(true false)</code>	<code>true</code>	开启所有超链接、书签等功能（也可以通过文档类选项指定）
<code>colorlinks=(true false)</code>	<code>false</code>	设置为 <code>true</code> 为链接文字带颜色，反之加上带颜色的边框
<code>hidelinks</code>		取消链接的颜色和边框
<code>pdfborder={(n) (n) (n)}</code>	<code>0 0 1</code>	超链接边框设置，设为 <code>0 0 0</code> 可取消边框
<code>bookmarks=(true false)[†]</code>	<code>true</code>	是否生成书签
<code>bookmarksopen=(true false)</code>	<code>false</code>	是否展开书签
<code>bookmarksnumbered=(true false)</code>	<code>false</code>	书签是否带章节编号
<code>pdftitle=(string)</code>	空	标题
<code>pdfauthor=(string)</code>	空	作者
<code>pdfsubject=(string)</code>	空	主题
<code>pdfkeywords=(string)</code>	空	关键词
<code>pdfstartview=(Fit FitH FitV)</code>	<code>Fit</code>	设置 PDF 页面以适合页面/适合宽度/适合高度等方式显示，默认为适合页面

[†] 该选项只能作为宏包选项，在调用宏包时指定。

超链接

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

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

`\url` 和 `\nolinkurl` 都像抄录命令 `\verb` 一样输出一个 URL，可以直接输入特殊符号，前者会加上超链接，后者没有（一些 PDF 阅读器会为 URL 文本自动加上超链接）。也可以为一段文本加上超链接：

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

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

使用 `hyperref` 宏包后，文档中的所有引用、参考文献、索引等等都转换为超链接。用户也可对某个 `\label` 命令定义的标签 `<label>` 作超链接（这里的 `<label>` 虽然是可选参数的形式，但通常是**必填的**）：

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

默认的超链接在文字外边加上一个带颜色的边框（在打印时不会打印），可指定 `colorlinks` 参数修改为将文字本身加上颜色，或修改 `pdfborder` 参数调整边框宽度以“去掉”边框； `hidelinks` 参数则令超链接既不变色也不加边框。

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

PDF书签

`hyperref`宏包可以为PDF生成书签。对于章节命令 `\chapter`、`\section` 等，默认情况下会为PDF自动生成书签。与交叉引用、索引等类似，生成书签也要两次编译（先将书签记录写入 `.out` 文件，再正常生成）。

书签的一些属性见上表。使用CJK宏包时，中文书签会出现乱码，需要进行繁琐的设置；但在使用ctex宏包和文档类时无需用户额外设置就可以正确生成。

`hyperref`还提供了手动生成书签的命令：

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

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

章节命令里往往有`latex`命令甚至数学公式，而PDF书签是纯文本，对命令和公式的处理很困难，有出错的风险。

`hyperref`宏包内部处理了很多常见命令，对于未被处理的命令或数学公式，就要在章节标题中使用如下命令，分别提供`latex`代码和PDF书签可用的纯文本：

```
\texorpdfstring{<LATEX code>}{<PDF bookmark text>}
```

例如章节名称中使用公式 $E = mc^2$ ，而书签则使用纯文本形式的 `E=mc^2`：

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

PDF文档属性

`hyperref`宏包还提供了一些参数用于改变PDF文档的属性，部分见上表。

绘图功能

绘图语言简介

`latex`提供了原始的 `picture` 环境，能够绘制一些基本的图形如点、线、矩形、圆、贝塞尔曲线等等，不过`latex`本身绘图功能极为有限，效果也不够美观。

当前较为流行的、用于`latex`的绘图宏包/程序主要有：

- *PSTricks*

以 PostScript 语法为基础的绘图宏包.

- *TikZ & pgf*

*pgf*宏包是在开发`latex`幻灯片文档类 `beamer` 时一并开发的绘图宏包, *TikZ*是在*pgf*基础上封装的一个宏包, 采用了类似 METAPost 的语法.

- *METAPOST & Asymptote*

脱胎于tex配套开发的字体生成程序 METAFONT, 能够调用tex引擎向图片中插入文字和公式. *Asymptote*在

METAPost 的基础上加入了类似C语言的编程能力, 支持三维图形的绘制.

它们作为独立的程序, 通常的用法是将代码写在单独的文件里, 编译生成图片供`latex`引用, 也可以借助特殊的宏包在`latex`代码里直接使用.

下面介绍 *TikZ*绘图宏包内最基本的部分. *TikZ*还支持各种自定义的扩展, 基于 *TikZ*的专门用途的宏包也非常之多.

***TikZ*绘图语言**

在导言区调用 `tikz`宏包, 即可使用:

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

前一种用法为 `\tikz` 带单挑绘图命令, 以分号结束, 一般用于在文字之间插入简单的图形; 后两种用法较为常见, 使用多条绘图命令, 可以在 `figure` 等浮动体中使用.

***TikZ*和路径**

*TikZ*用直角·系或者极·系描述点的位置.

- 直角·下, 点的位置写作 $(\langle x \rangle, \langle y \rangle)$, · $\langle x \rangle$ 和 $\langle y \rangle$ 可以用`latex`支持的任意单位表示, 缺省为 `cm` ;
- 极·下, 点的位置写作 $(\theta):(\langle r \rangle)$, θ 为极角, 单位是度.

还可以为某个点命名: `\coordinate (A) at (<coordinate>)`, 然后就可以使用 `(A)` 作为点的坐标的.

```
\begin{tikzpicture}
  \draw (0,0) -- (30:1);
  \draw (1,0) -- (2,1);
  \coordinate (S) at (0,1);
  \draw (S) -- (1,1);
\end{tikzpicture}
```



坐标的表示形式还包括“垂足”形式（水平和铅锤方向）：

```
\begin{tikzpicture}
\coordinate (S) at (2,2);
\draw[gray] (-1,2) -- (S);
\draw[gray] (2,-1) -- (S);
\draw[red] (0,0) -- (0,0 |- S); % 终点等效于(2,0)
\draw[blue] (0,0) -- (0,0 |- S); % 终点等效于(0,2)
\end{tikzpicture}
```



TikZ最基本的路径为两点之间连线，如 $(x_1, y_1) \text{ -- } (x_2, y_2)$ ，可以连用表示多个连线（折线）。连续使用连线时，可以使用 `cycle` 令路径回到起点，生成闭合的路径。

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



多条路径可用于同一个画图命令中，以空格分割：

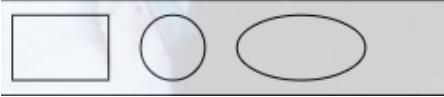
```
\begin{tikzpicture}
\draw (0,0) -- (0,1)
(1,0) -- (1,1) -- (2,0) -- cycle;
\end{tikzpicture}
```



其它常用的路径还包括：

- 矩形、圆和椭圆：

```
\begin{tikzpicture}
\draw (0,0) rectangle (1.5,1); % 指定对角顶点，画出水平铅锤的矩阵
\draw (2.5,0.5) circle [radius=0.5]; % 指定圆心和半径
\draw (4.5,0.5) ellipse [x radius=1, y radius=0.5]; % 指定中心和xy轴半径
\end{tikzpicture}
```



- 直角、圆弧、椭圆弧：

```
\begin{tikzpicture}
\draw (0,0) |- (1,1); % 先铅锤后水平, 等效于 (0,0) -- (0,1) -- (1,1)
\draw (1,0) -| (2,1); % 先水平后铅锤, 等效于 (1,0) -- (2,0) -- (2,1)
\draw (4,0) arc (0:135:1); % 以(4,0)为圆心, 半径为1, 极角从0到135
\draw (6,0) arc (0:135:1 and 0.5); % 以(6,0)为圆心, x半径为1, y半径为0.5, 极角从0到135
\end{tikzpicture}
```



- 正弦、余弦曲线 (1/4周期)：

```
\begin{tikzpicture}
% 以 sin(0:2pi) 为一个周期, 分别为 sin/ cos\ sin\ cos/
\draw (0,0) sin (1,1); % sin/ 为上凸递增 同 sin(0:pi/2)
\draw (0,1) sin (1,0); % sin\ 为下凸递减 同 sin(pi:3pi/2)
\draw (2,1) cos (3,0); % cos\ 为上凸递减 同 cos(0:pi/2) 或 sin(pi/2:pi)
\draw (2,0) cos (3,1); % cos/ 为下凸递增 同 cos(pi:3pi/2) 或 sin(3pi/2:2pi)
\end{tikzpicture}
```



- 抛物线, 用 bend 控制顶点:

```
\begin{tikzpicture}
% \draw <起点> parabola [bend <顶点 (缺省为起点)>] <终点>
\draw (0,0) parabola (1,2);
\draw (2,0) parabola bend (2.25,-0.25) (3,2);
\draw (4,0) parabola bend (4.75,2.25) (5,2);
\end{tikzpicture}
```



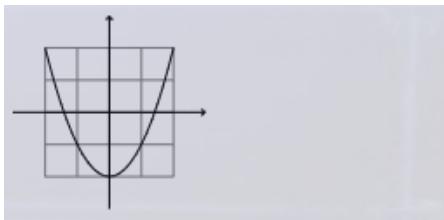
- 二次和三次贝塞尔曲线, 分别使用一个和两个控制点:

```
\begin{tikzpicture}
    % \draw <起点> .. controls <控制点1> [and <控制点2>] .. <终点>
    \draw (0,0) .. controls (2,1) and (3,1) .. (3,0);
    \draw (4,0) .. controls (5,1) .. (5,0);
    \draw[help lines] (0,0) -- (2,1) -- (3,1) -- (3,0)
                    (4,0) -- (5,1) -- (5,0);
\end{tikzpicture}
```



- 网络、函数图像，网格可用 `step` 参数控制网格大小，函数图像用 `domain` 参数控制定义域：

```
\begin{tikzpicture}
    \draw[help lines, step=0.5] (-1,-1) grid (1,1); % 指定对角顶点，画直角坐标系网格，step为网格宽度
    \draw[->] (-1.5,0) -- (1.5,0);
    \draw[->] (0,-1.5) -- (0,1.5);
    \draw[domain=-1:1] plot(\x, {\x*\x*2 -1}); % domain为定义域，plot(\x, {以\x为变量作表达式})为函数图像
\end{tikzpicture}
```



TikZ绘图命令和参数

除了 `\draw` 命令之外，*TikZ*还提供了 `\fill` 命令来填充图形，`\filldraw` 命令则同时填充和描边。除了矩形、圆等现成的闭合图形外，`\fill` 和 `\filldraw` 命令也能够填充人为构造的闭合路径。

```
\draw[...] <path>;
\fill[...] <path>;
\filldraw[...] <path>;
```

绘图参数可作为可选参数用在 `tikzpicture` 环境或 `\tikz` 命令时，参数会影响到所有具体的绘图命令；用在单个绘图命令 `\draw`、`\filldraw` 等时，只对这个命令起效。

以下示例常用的一些绘图参数。

- `color/draw/fill=<color>` 为 `\draw` 或 `\fill` 等命令指定颜色。`draw` 和 `fill` 分别指定描边和填充的颜色，而 `color` 同时指定，也可以省略 `color=` 直接写颜色名称。

```
\begin{tikzpicture}[thick]
\draw[blue] (0,0) rectangle (1,1);
\filldraw[fill=yellow, draw=red] (2,0.5) circle [radius=0.5];
\end{tikzpicture}
```



- `thick=<length>/thin/semithick/...` 指定线条的粗细.

```
\begin{tikzpicture}
\draw[ultra thin] (0,0) -- (0,2);
\draw[very thin] (0.5,0) -- (0.5,2);
\draw[thin] (1,0) -- (1,2);
\draw[semithick] (1.5,0) -- (1.5,2);
\draw[thick] (2,0) -- (2,2);
\draw[very thick] (2.5,0) -- (2.5,2);
\draw[ultra thick] (3,0) -- (3,2);
\end{tikzpicture}
```



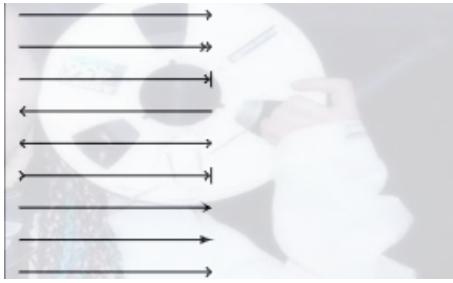
- `solid/dashed/dotted/dash dot/dash dot dot` 指定线条类型 (实线、虚线、点划线等) . 与 `dashed` 对应的有 `densely dashed` 和 `loosely dashed` , 后三种类型同理.

```
\begin{tikzpicture}
\draw[dashed] (0,0) -- (0,2);
\draw[dotted] (0.5,0) -- (0.5,2);
\draw[dash dot] (1,0) -- (1,2);
\draw[dash dot dot] (1.5,0) -- (1.5,2);
\draw[densely dotted] (2,0) -- (3,2) -- (4,0) -- cycle;
\end{tikzpicture}
```



- `<arrow>-<arrow>` 指定线条首尾的箭头形式. 复杂的箭头形式需要在导言区使用 `\usetikzlibrary{arrows.meta}` .

```
\begin{tikzpicture}
\draw[->] (0,4) -- (3,4);
\draw[->>] (0,3.5) -- (3,3.5);
\draw[->|] (0,3) -- (3,3);
\draw[<-] (0,2.5) -- (3,2.5);
\draw[<->] (0,2) -- (3,2);
\draw[>->|] (0,1.5) -- (3,1.5);
\draw[-stealth] (0,1) -- (3,1);
\draw[-latex] (0,0.5) -- (3,0.5);
\draw[-to] (0,0) -- (3,0);
\end{tikzpicture}
```



- `rounded corners[=<radius>]/sharp corners` 将路径转向处绘制成圆角/直角. 可选参数 `<radius>` 控制圆角的半径. 可以对某一段路径直接使用.

```
\begin{tikzpicture}
\draw[rounded corners] (0,0) rectangle (1,1);
\draw (2,0) -- (2,1)
      [rounded corners=.3cm]
      -- (3,1) -- (3.5,0)
      [sharp corners] -- cycle;
\end{tikzpicture}
```



- `scale/xshift/yshift/xslant/yslant/rotate` 设定图形的缩放、位移和旋转.

```
\begin{tikzpicture}
\draw[help lines] (0,0) rectangle (1,1);
\draw[scale=1.5] (0,0) rectangle (1,1); % 以起点缩放1.5倍
\draw[rotate=30] (0,0) rectangle (1,1); % 以起点逆时针旋转30度
\draw[help lines] (2,0) rectangle (3,1);
\draw[yshift=4pt] (2,0) rectangle (3,1); % y轴正方向(向上)移动4pt
\draw[help lines] (4,0) rectangle (5,1);
\draw[xslant=0.4] (4,0) rectangle (5,1); % x轴正方向倾斜0.4倍
\end{tikzpicture}
```



为了重复利用绘图参数、减少代码冗余, *TikZ*引入了“样式”概念, 可以定义一个样式包含绘图参数, 然后将样式作为一个参数用于绘图:

```
\begin{tikzpicture}[myarrow/.style={blue,thick,->}]  
    \draw (0,0) -- (0,1) -- (2,1);  
    \draw[myarrow] (0,0) -- (2,1);  
    \draw[myarrow, dotted] (0,0) -- (2,0) -- (2,1);  
\end{tikzpicture}
```



*TikZ*还提供了 `scope` 环境, 令绘图参数或样式在局部生效:

```
\begin{tikzpicture}  
    \draw (0,0) rectangle (2.5,2.5);  
    \begin{scope}[thick,scale=0.5]  
        \draw (0,0) rectangle (2.5,2.5);  
    \end{scope}  
\end{tikzpicture}
```



TikZ文字结点

*TikZ*用 `\node[<options>] (<name>) at (<coordinate>) {<text>};` 命令绘制文字结点, (`<name>`) 为结点命名, 类似 `\coordinate` ; `at (<coordinate>)` 指定结点的位置, 这两者和前面的 `<options>` 都可以省略, 只有 `<text>` 必填.

```
\begin{tikzpicture}  
    \node (A) at (0,0) {A};  
    \node (B) at (1,0) {B};  
    \node (C) at (60:1) {C};  
    \draw (A) -- (B) -- (C) -- (A);  
\end{tikzpicture}
```



*TikZ*绘图命令和参数小节中的参数可用于 `\node` 命令的配置. 除此之外, `\node` 还有一些特定的参数:

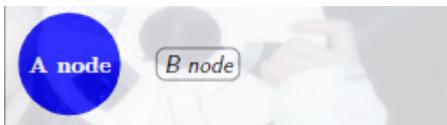
- `anchor=<position>` 令结点的某个角落 `<position>` 与 `<coordinate>` 对应.
- `centered/above/below/left/right/above left/...[=<length>]` 与 `anchor` 等效的选项. 可选的 `<length>` 为结点相对于 `<coordinate>` 的距离.

```
\begin{tikzpicture}
\coordinate (A) at (1,1);
\fill (A) circle [radius=2pt];
\node [draw, anchor=south] at (A) {a};
\node [draw, below right=4pt] at (A) {b};
\end{tikzpicture}
```



- `shape=<shape>` 结点的形状, 默认可用 `rectangle` 和 `circle`, 可省略 `shape=` 直接写. 在导言区使用命令 `\usetikzlibrary{shapes.geometric}` 可用更多形状.
- `text=<color>` 结点文字的颜色.
- `node font={}` 结点文字的字体, 形如 `\bfseries` 或 `\itshape` 等.

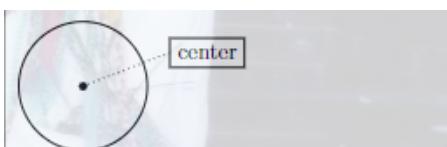
```
\begin{tikzpicture}
\node [circle, fill=blue, text=white, node font=\bfseries] (A) at (0,0) {A node};
\node [rectangle, rounded corners, draw=gray, node font=\sffamily\slshape] (B) at (2,0) {B node};
\end{tikzpicture}
```



- `inner sep=<length> / outer sep=<length>` 结点边界向外和向内的额外距离.
- `minimum size=<length> / minimum height=<length> / minimum width=<length>` 结点的最小大小或最小高度/宽度.

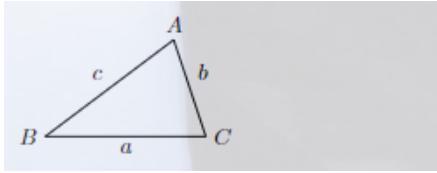
`\node` 命令不仅为文字结点的位置命名, 在 `\draw` 等命令中还可以使用某个结点的相对位置, 以“东南西北”的方式命名:

```
\begin{tikzpicture}
\draw (0,0) circle [radius=1];
\fill (0,0) circle [radius=2pt];
\node [draw] (P) at (15:2) {center};
\draw [dotted] (0,0) -- (P.west);
\end{tikzpicture}
```



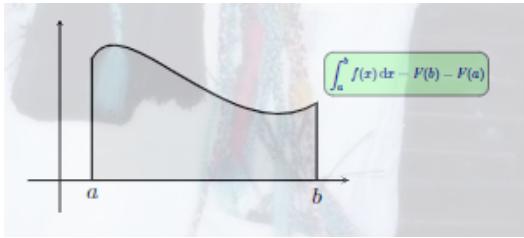
\node 命令的一种等效用法是在 \draw 等命令的路径中使用 node，不仅可用对某个位置标记结点，还能够对线标记：

```
\begin{tikzpicture}
\draw (2,1.5) node[above] {$A$}
      -- node[above left] {$c$}
      (0,0) node[left] {$B$}
      -- node[below] {$a$}
      (2.5,0) node[right] {$C$}
      -- node[below right] {$b$}
      cycle;
\end{tikzpicture}
```



下举较为复杂的例子：

```
\begin{tikzpicture}
\draw[-stealth, line width=0.2pt] (-0.5,0) -- (4.5,0);
\draw[-stealth, line width=0.2pt] (0,-0.5) -- (0,2.5);
\coordinate (a) at (0.5,1.9);
\coordinate (b) at (4,1.2);
\node[below] (a0) at (a |- 0,0) {$a$};
\node[below] (b0) at (b |- 0,0) {$b$};
\filldraw[fill=gray!20, draw, thick] (a0) -- (a) .. controls (1,2.8) and (2.7,0.4) .. (b) -- (b0) -- cycle;
\node[above right, outer sep=0.2cm, rounded corners, fill=green!20, draw=gray, text=blue!60!black, scale=0.6] at (b) {${\displaystyle \int_a^b f(x) dx = F(b) - F(a)}$};
\end{tikzpicture}
```



在TikZ中使用循环

TikZ通过`pgffor`功能宏包实现了简单的循环功能，语法为：

```
\foreach \a in {<list>} {<commands>}
```

上述语法定义了 \a 为变量，在 {<commands>} 中使用 \a 完成循环。

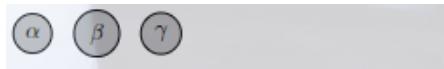
<list> 可以直接将所有值写出来，如 `1,2,3,4`；也可以写成省略形式，如 `1,2,...,10`。

```
\begin{tikzpicture}
\draw (0,0) -- (5,0);
\foreach \i in {0.0,0.1,...,5.0} {
\draw[very thin] (\i,0) -- (\i,0.15);
}
\foreach \I in {0,1,2,3,4,5} {
\draw (\I,0) -- (\I, 0.25) node[above] {\I};
}
\end{tikzpicture}
```



`foreach` 还可以使用变量对参与循环:

```
\begin{tikzpicture}
\foreach \n/\t in {0/\alpha,1/\beta,2/\gamma} {
\node[circle, fill=lightgray, draw] at (\n,0) {$\t$};
}
\end{tikzpicture}
```



自定义 latex 命令和功能

本章帮助制作模板——宏包和文档类，并在其中自定义命令和环境。

自定义命令和环境

latex自带的代码环境是没有背景颜色的，用户可以自己创建一个**宏包**并定义所需要的命令和环境，而非用基础的latex命令来实现。一旦需要修改命令代码的样式，比如更换颜色、加边框等等，可以通过改变环境的命令来实现，而非挨个修改每个命令示例。

定义新命令

使用以下命令：

```
\newcommand{\<name>}[<num>]{<definition>}
```

`\newcommand` 的基本用法需要两个必选参数，第一个参数 `<name>` 是要定义的命令名称（带反斜杠），第二个参数 `<definition>` 是命令的基本定义。方括号的参数 `<num>` 是可选的，用于指定新命令所需的参数数目（最多9个）。如果缺省可选参数，默认就是 `0`，也就是新定义的命令不带任何参数。

```
\newcommand{\tnss}{The not so Short Introduction to \LaTeXe}
This is ``\tnss'' \ldots{} ``\tnss''
```

This is “The not so Short Introduction to L^AT_EX 2 ε ” ... “The not so Short Introduction to L^AT_EX 2 ε ”

在命令的定义中，标记 #1 代表指定的参数，如果想使用多个参数，可以依次使用 #2、……、#9 等标记。

```
\newcommand{\txsit}[1]{This is the \emph{#1} Short Introduction to \LaTeXe}
% in the document body:
\begin{itemize}
  \item \txsit{not so}
  \item \txsit{very}
\end{itemize}
```

latex不允许使用 `\newcommand` 定义一个与现有命令重名的命令。若需要修改目录定义的话，使用 `\renewcommand` 命令。它使用与命令 `\newcommand` 相同的语法。

在某种情况之下，使用 `\providetoggle` 命令是一种比较理想的方案：在命令未定义时，它相当于 `\newcommand`；在命令已定义时，沿用已有的定义。

定义环境

与 `\newcommand` 命令类似，可以用 `\newenvironment` 定义新的环境，语法如下：

```
\newenvironment{<name>}[<num>]{<before>}{<after>}
```

同样地，`\newenvironment` 有一个可选参数。在 `<before>` 中的内容将在此环境包含的文本之前处理，而在 `<after>` 中的内容将在遇到 `\end{<name>}` 命令时处理。

```
\newenvironment{king}
  {\rule{1ex}{1ex}\hspace{\stretch{1}}}
  {\hspace{\stretch{1}}\rule{1ex}{1ex}}
\begin{king}
  My humble subjects \ldots
\end{king}
```

■ My humble subjects ... ■

参数 `<num>` 的使用方法与 `\newcommand` 命令相同。 latex还保证不会不小心新建重名的环境。若确实希望改变一个现有的环境，可以使用 `\renewenvironment`，使用和命令 `\newenvironment` 相同的语法。

xparse宏包

通过 `\newcommand` 和 `\newenvironment` 定义的命令或环境格式比较固定，若需要定义有多个可选参数、或者带星号的命令或环境，可以使用xparse宏包（2020-10-01 版本后已被集成在格式之中）。它提供了：

```
\NewDocumentCommand{<name>}{<arg spec>}{<definition>}
\NewDocumentEnvironment{<name>}{<arg spec>}{<before>}{<after>}
```

相比 `\newcommand` 和 `\newenvironment`, `xparse` 通过 `{<arg spec>}` 来指定参数的个数和格式. 基本的参数格式见下表. 注意 `{<arg spec>}` 中的空格可以忽略.

表 8.1: `xparse` 参数格式

参数格式	说明
<code>m</code>	必选参数, 由 <code>{...}</code> 给出
<code>o</code>	可选参数, 由 <code>[...]</code> 给出; 未给出时返回 <code>-NoValue-</code> 标记
<code>O{(default)}</code>	可选参数, 但在未给出时则返回默认值 <code>(default)</code>
<code>s</code>	可选的星号 *
<code>+</code>	修饰后面的 <code>m</code> 、 <code>o</code> 等, 表示允许在参数中分段

表 8.2: `xparse` 参数示例

参数格式	输入值	#1	#2	#3
<code>m m</code>	<code>{foo}{bar}</code>	<code>foo</code>	<code>bar</code>	
<code>o m</code>	<code>{foo}</code>	<code>-NoValue-</code>	<code>foo</code>	
<code>o o m</code>	<code>[foo]{bar}</code>	<code>foo</code>	<code>-NoValue-</code>	<code>bar</code>
<code>m O{default}</code>	<code>{foo}</code>	<code>foo</code>	<code>default</code>	
<code>m O{default}</code>	<code>{foo}[bar]</code>	<code>foo</code>	<code>bar</code>	
<code>m O{default}</code>	<code>[bar]</code>	报错		
<code>s o m</code>	<code>*[foo]{bar}</code>	<code>\BooleanTrue</code>	<code>foo</code>	<code>bar</code>
<code>s m O{default}</code>	<code>{foo}</code>	<code>\BooleanFalse</code>	<code>foo</code>	<code>default</code>

`-NoValue` 标记可以用 `\IfNoValueTF` 等命令来判断:

```
\IfNoValueTF{<argument>}{{<true code>}}{{<false code>}}
\IfNoValueT{<argument>}{{<true code>}}
\IfNoValueF{<argument>}{{<false code>}}
```

举例如下:

```
% 百分号用于注释掉不必要的空格和换行符
\NewDocumentCommand\hello{o m}{%
  \IfNoValueTF{#1}{%
    {Hello, #2!}%
    {Hello, #1 and #2!}%
  }%
  \hello{Alice}
  \hello[Bob]{Alice}}
```

Hello, Alice! Hello, Bob and Alice!

`\BooleanTrue` 和 `\BooleanFalse` 可以用 `\IfBooleanTF` 等命令来判断:

```
\IfBooleanTF{<argument>}{{<true code>}}{{<false code>}}
\IfBooleanT{<argument>}{{<true code>}}
\IfBooleanF{<argument>}{{<false code>}}
```

举例如下：

```
\NewDocumentCommand\hereis{s m}
{Here is \IfBooleanTF{#1}{an}{a} #2.}
\hereis{banana}
\hereis*{apple}
```

Here is a banana. Here is an apple.

需要注意的是，与命令不同，环境在定义时名字里可以包含 *：

```
\NewDocumentEnvironment{mytabular}{o +m}{...}{...}
\NewDocumentEnvironment{mytabular*}{m o +m}{...}{...}
```

用 s 标记的 * 应该放在 \begin{<env>} 的后面：

```
\NewDocumentEnvironment{envstar}{s}
{\IfBooleanTF{#1}{star}{no star}{}}
\begin{envstar}*
\end{envstar}
```

与 \renewcommand、\providetcommand 等命令类似，xparse 宏包也允许在命令或环境已有定义时做出相应的处理，具体见下表。

表 8.3: xparse 提供的其他命令

定义命令	定义环境	已定义	未定义
\NewDocumentCommand	\NewDocumentEnvironment	报错	给出定义
\RenewDocumentCommand	\RenewDocumentEnvironment	重新定义	报错
\ProvideDocumentCommand	\ProvideDocumentEnvironment	什么也不做	给出定义
\DeclareDocumentCommand	\DeclareDocumentEnvironment	重新定义	给出定义

编写自己的宏包和文档类

编写简单的宏包

若定义了很多新的环境和命令，文档的导言区将要变得很长，在这种情况下，可以建立新的 latex 宏包来存放所有自定义的命令和环境，然后在文档中使用 \usepackage 命令来调用自定义的宏包。

写一个宏包的基本工作就是将原本在导言区的内容拷贝到另一个文件中，这个文件需要以 .sty 作拓展名。还需要加一个宏包专用的命令：

```
\ProvidesPackage{<package name>}

% Demo Package by Tobias Oetiker
\ProvidesPackage{demopack}
\newcommand{\tnss}{The not so Short Introduction to \LaTeXe}
\newcommand{\txsit}[1]{The \emph{#1} Short Introduction to \LaTeXe}
\newenvironment{king}{\begin{quote}}{\end{quote}}
```

这个命令应该放在宏包的最前面，并且一定注意：`<package name>` 需要和宏包的文件名一致。`\ProvidesPackage` 让 latex 记录宏包的名称，从而在 `\usepackage` 命令再次调用同一个宏包时忽略之。

在宏包中调用其它宏包

若想进一步将各种宏包的功能汇总到一个文件里，而不是在文档的导言区罗列一大堆宏包的话， latex 允许在自定义宏包中调用其它宏包，命令为 `\RequirePackage`，用法和 `\usepackage` 一致：

```
\RequirePackage[<options>]{<package name>}
```

编写自己的文档类

若要编写文档类如论文模板等，需要用 `.cls` 作拓展名，开头使用 `\ProvideClass` 命令：

```
\ProvidesClass{<class name>}
```

事实上，诸如 `\chapter`、`\section` 等等许多章节的命令都是在文档类中定义好的，常常只需要调用现有的文档类，下命令用法和 `\documentclass` 十分相像：

```
\LoadClass[<options>]{<class name>}
```

计数器

latex 对文档元素自动计数的能力：章节符号、列表、图表……它们都是依靠 latex 提供的计数器功能完成的。

定义和修改计数器

定义一个计数器的方法为：

```
\newcounter{<counter name>}[<parent counter name>]
```

`<counter name>` 为计数器的名称。计数器可以有上下级的关系，可选参数 `<parent counter name>` 定义为 `<counter name>` 的上级计数器。

以下命令修改计数器的数值，`\setcounter` 将数值设为 `<number>`；`\addtocounter` 将数值加上 `<number>`；`\stepcounter` 将数值加上，并将所有下级计数器归零。

```
\setcounter{<counter name>}{<number>}
\addtocounter{<counter name>}{<number>}
\stepcounter{<counter name>}
```

计数器的输出格式

计数器 `<counter>` 的输出格式由 `\the<counter>` 表示，如页眉页脚节见到的 `\thechapter` 等。这个值默认以阿拉伯数字形式输出，若要更改成其他形式，需要重定义 `\the<counter>`，如将 `equation` 计数器的格式定义为大写字母：

```
\renewcommand{\theequation}{\Alph{equation}}
```

命令 `\Alph` 控制计数器 `<counter>` 的值以大写字母形式形式。下表列出所有可用于修改计数器格式的命令。注意：这些命令只能用于计数器，不能直接用于数字，如 `\roman{1}` 这样的命令为出错。

表 8.4: 计数器输出格式相关命令

命令	样式	范围
<code>\arabic</code>	阿拉伯数字（默认）	
<code>\alph</code>	小写字母	限 0-26
<code>\Alph</code>	大写字母	限 0-26
<code>\roman</code>	小写罗马数字	限非负整数
<code>\Roman</code>	大写罗马数字	限非负整数
<code>\fnsymbol</code>	一系列符号，用于 <code>\thanks</code> 命令生成的脚注	限 0-9

注：`\fnsymbol` 使用的符号顺次为：* † § ¶ || ** †† ††

计数器的输出格式还可以用其它字符，甚至其它计数器的输出格式与之组合。如标准文档类里对 `\subsection` 相关的计数器的输出格式的定义相当于：

```
\renewcommand{\thesubsection}{\thesection.\arabic{subsection}}
```

Latex中的计数器

- 所有章节命令 `\chaper`、`\section` 等分别对应计数器 `chapter`、`section` 等等，而且有上下级关系。而计数器 `part` 是独立的。
- 有序列表 `enumerate` 的各级计数器为 `enumi`、`enumii`、`enumiii`、`enumiv`，也有上下级的关系。
- 图表浮动体的计数器就是 `table` 和 `figure`；公式的计数器为 `equation`。这些计数器在 `article` 文档类中是独立的，而在 `report` 和 `book` 中以 `chapter` 为上级计数器。
- 页码、脚注的计数器分别是 `page` 和 `footnote`。

可以利用之前介绍的公式修改计数器的样式，例如把页码修改成大写罗马数字，左右加横线，或是给脚注加上方括号：

```
\renewcommand{\thepage}{--~\Roman{page}~--}
\renewcommand{\thefootnote}{[\arabic{footnote}]}
```

基本的页眉页脚样式小节中出现的命令 `\pagenumbering` 内部机制就是修改 `page` 计数器的格式 `thepage`，并将计数器值重置为 1。

还有两个有用的计数器：

secnumdepth

latex标准文档类对章节划分了层级：

- 在 article 文档类里 part 为 0 , section 为 1 , 以此类推;
- 在 report 和 book 文档类里 part 为 -1 , chapter 为 0 , section 为 1 , 以此类推.

secnumdepth 计数器控制章节编号的深度, 如果章节的层级大于 secnumdepth , 那么章节的标题、在目录和页眉页脚的标题都不编号 (照常生成目录和页眉页脚) , 章节计数器也不计数.

可以用 \setcounter 目录色湖之 secnumdepth 为较大的数使得层级比较深的章节也编号, 如设置为 4 令 \paragraph 也编号; 或设置一个较小的数以取消编号, 如设置为 -1 令 \chapter 不编号. 后者是生成不编号的章节的一个妙招, 免去了手动使用 \addcontentsline 和 \markboth 的麻烦.

secnumdepth 计数器在 article 文档类里默认为 3 (subsubsection 一级) ; 在 report 和 book 文档类里默认为 2 (subsection 一级) .

tocdepth

tocdepth 计数器控制目录的深度, 如果章节的层级大于 tocdepth , 那么章节将不会自动写入目录项. 默认值同 secnumdepth .

latex可定制的一些命令和参数

- 标题名称/前后缀等. 使用 \renewcommand 修改.

表 8.5: L^AT_EX 可定制的标题名称/前后缀

命令	默认值	含义
\partname	Part	\part 命令生成的标题前缀
\chaptername	Chapter	\chapter 命令生成的标题前缀
\appendixname	Appendix	使用 \appendix 命令生成的附录部分的章标题前缀
\abstractname	Abstract	摘要环境 abstract 的标题名称
\contentsname	Contents	\tableofcontents 命令生成的目录标题
\listfigurename	List of Figures	\listoffigures 命令生成的插图目录标题
\listtablename	List of Tables	\listoftables 命令生成的表格目录标题
\tablename	Table	table 浮动体中 \caption 命令生成的标题前缀
\figurename	Figure	figure 浮动体中 \caption 命令生成的标题前缀
\refname	References	thebibliography 环境或 \bibliography 命令生成的参考文献标题 (article 文档类)
\bibname	Bibliography	thebibliography 环境或 \bibliography 命令生成的参考文献标题 (report 和 book 文档类)
\indexname	Index	\printindex 命令生成的索引标题

注: 形如“第 X 章”和“第 X 部分”的中文章节标题不能直接由修改本表的命令得到, 需要使用 titlesec 等宏包定制。如果使用 ctex 宏包或文档类, 那么标题默认被修改成“第 X 章”和“第 X 部分”的形式, 本表中的其它标题也修改为中文标题。详见 ctex 宏包的帮助文档。

- 长度. 使用 \setlength 修改.

表 8.6: L^AT_EX 可定制的长度参数

命令	默认值	含义
\fboxrule	0.4pt	\fbox 或 \framebox 等带框盒子的线宽
\fboxsep	3pt	\fbox 或 \framebox 等带框盒子的内边距
\arraycolsep	5pt	array 环境的表格项前后的间距 ^{注 1}
\tabcolsep	6pt	tabular 环境的表格项前后的间距 ^{注 1}
\arrayrulewidth	0.4pt	表格线宽
\doublerulesep	2pt	连续两根表格线之间的间距
\abovecaptionskip	10pt	\caption 命令上方的间距 ^{注 2}
\belowcaptionskip	0pt	\caption 命令下方的间距 ^{注 2}
\columnsep	10pt	双栏排版下两栏的间距
\columnseprule	0pt	双栏排版下两栏之间竖线的宽度

注 1: \arraycolsep 和 \tabcolsep 是每个表格项本身前后的间距 (表格线前后无间距; @ 列格式会消除与前后表格项的间距)。两个表格项之间的间距相当于 2\arraycolsep 或 2\tabcolsep。

注 2: 在默认设置下, \caption 命令和位于它下方的图表之间无间距。宏包 caption 改善了这个问题。