

LaTeX 文档排版教程 [↗](#)

2018 年 10 月 15 日

---


## 第一部分 · 从零开始学习 LaTeX


吕荐瑞 [🏠 lvjr.bitbucket.io](https://lvjr.bitbucket.io)


暨南大学数学系





 1 开始使用

 2 快速入门

 3 撰写文本

 4 撰写公式

 5 错误处理

 6 命令速查

## 语言简介

LaTeX 是一种流行的科技文档排版语言. 相对于 Word 这种办公排版软件, 它有如下这些优点:

- ▶ 排版出的数学公式美观
- ▶ 自动编号能力强大
- ▶ 稳定性好, 不容易崩溃

## 排版流程

与 C 语言类似，利用 LaTeX 排版文档需要一个编译的步骤。基本的流程如下：

1. 在 `somename.tex` 文件中编写文档
2. 用 `xelatex` 等程序编译上述文档
3. 得到 `somename.pdf` 文档用于阅读或打印

## 软件准备

为方便你使用 LaTeX, [这里](#)提供了相关软件下载.

1. 下载 miktex-2.9-2017.7z 压缩包
2. 将该文件解压到某个路径不含空格的目录
3. 双击 texworks.cmd 打开 TeXworks 编辑器

首次打开 TeXWorks 时可能较慢, 可以耐心多等待几秒.

## 打开模板

在打开 TeXworks 编辑器之后, 我们先从模板文件建立第一个 LaTeX 文档:

1. 点击 File -> New from Template... 菜单项
2. 在窗口中点击打开 Chinese documents 目录
3. 选中 article-demo-cn.tex 再点击 Open 按钮
4. 点击 File -> Save 将该文件保存到某个目录

以后可在 File -> Open Recent 菜单打开 tex 文件.

## 开始编写

现在，我们开始试着编写我们的 LaTeX 文档：

1. 试着在编辑器中修改几个文字
2. 点击工具栏的绿色三角图标将会编译此文档
  - ▶ 如果无错误，右边将会显示生成的 PDF 文档
  - ▶ 如果有错误，下面的输出窗口将会显示错误信息
3. 编译时绿色图标将会变成红色，点击将停止编译
4. 根据错误信息修改文档，然后重新编译

注意：在绿色三角按钮的右边的下拉选择框中可以选择编译程序。对中文文档，务必将它选择到 XeLaTeX 这个程序。

1 开始使用

2 快速入门

3 撰写文本

4 撰写公式

5 错误处理

6 命令速查



## 中文文档

最简单的中文文档如下（其中 % 后面的内容是注释）.

```
\documentclass{article}
```

```
\usepackage{ctex} %中文  
\usepackage{amsmath} %数学
```

```
\begin{document}
```

```
正文内容.  $3^2+4^2=5^2$ .
```

```
\end{document}
```

导言区

正文区

## 文本段落

正文段落之间用空行隔开，比如：

第一个段落.

第二个段落.

第三个段落.

而段落内部的换行用 `\newline` 命令.

## 数学公式

依据其出现位置，公式可分为行内公式和行间公式。

- ▶ 行内公式放在一对  $\$$  中间
- ▶ 行间公式放在一对  $$$$  中间

下面的例子同时出现这两种公式

如果  $p$  是素数， $\gcd(a, p) = 1$ ，则有  
$$a^{p-1} \equiv 1 \pmod{p}$$

如果  $p$  是素数， $\gcd(a, p) = 1$ ，则有

$$a^{p-1} \equiv 1 \pmod{p}$$

## 上标下标

公式中的上标和下标分别用 `^` 和 `_` 表示。比如

$$(x_1 + x_2)^2 = x_1^2 + 2x_1x_2 + x_2^2$$

$$(x_1 + x_2)^2 = x_1^2 + 2x_1x_2 + x_2^2$$


如果上下标中包含多个符号，需要将它们放在一对花括号中。比如前面例子中的  $a^{p-1}$ 。


## 命令参数

用 `\` 加上一个或多个英文字母组成一个 LaTeX 命令。比如之前例子中见到的 `\gcd`, `\equiv` 和 `\pmod`.


有些命令需要带上一个或多个参数, 比如 `\pmod` 和分式命令 `\frac`. 命令的参数要放在一对花括号中, 依次紧接在命令之后. 比如之前我们写的 `\pmod{p}`, 它给出了  $(\text{mod } p)$ . 又如从 `\frac{a+b}{c}` 得到  $\frac{a+b}{c}$ .


有些命令的第一个参数是可选的, 这个可选参数要放在一对方括号中. 比如 `\sqrt{x}` 得到  $\sqrt{x}$ ; 而 `\sqrt[3]{x}` 得到  $\sqrt[3]{x}$ .


 1 开始使用

 2 快速入门

 3 撰写文本

 4 撰写公式

 5 错误处理

 6 命令速查

## 标题目录

要生成文章的标题栏，可以用下面几个命令：

```
\title{文章标题}  
\author{作者姓名}  
\date{写作日期}  
\maketitle
```

其中 `\date` 命令的参数为空时将不显示日期；而 `\date` 命令省略时将使用当前日期。

文章的目录可以用 `\tableofcontents` 命令生成。

## 定理环境

在 LaTeX 中很容易编写自动编号的定理. 首先在导言区中写上

```
\newtheorem{theorem}{定理}  
\newtheorem{corollary}{推论}
```

然后在正文区中写上左侧的代码就得到右侧的结果:

```
\begin{theorem}
```

定理内容.

```
\end{theorem}
```

```
\begin{corollary}
```

推论内容.

```
\end{corollary}
```

定理 1 定理内容.

推论 1 推论内容.

用 `\begin{名称}` 和 `\end{名称}` 包含的内容称为一个环境.





## 制作表格


利用 `tabular` 环境，我们可以制作下面的表格。其中 `|l|ccc|` 中的 `l` 表示第一列左对齐，后面的三个 `c` 表示后面三列居中对齐。

```
\begin{tabular}{|l|ccc|}  
\hline  
数学家 & 费马 & 欧拉 & 高斯 \\  
\hline  
年份 & 1601 & 1707 & 1777 \\  
\hline  
\end{tabular}
```


数学家	费马	欧拉	高斯
年份	1601	1707	1777


 1 开始使用

 2 快速入门

 3 撰写文本

 4 撰写公式

 5 错误处理

 6 命令速查

## 数学阵列

在数学公式中有一个与 `tabular` 用法类似的 `array` 环境. 比如我们可以制作下面的数学阵列.

```
 $\begin{array}{|l|rrrrr|}  
 \hline  
 n & 1 & 2 & 3 & 4 & 5 \\ \hline  
 n^2 & 1 & 4 & 9 & 16 & 25 \\ \hline  
 \end{array} $
```

$n$	1	2	3	4	5
$n^2$	1	4	9	16	25

## 数学阵列

利用 `array` 环境，很容易写出矩阵和行列式等多行多列的公式。

```
\left(\begin{array}{ccc}1 & 2 & 3 \\4 & 5 & 6 \\7 & 8 & 9\end{array}\right)
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

其中的 `\left` 和 `\right` 用于表示一对定界符；定界符（比如这里的圆括号）会依据内容的高度自动调整大小。

## 数学阵列

注意，如果定界符其中一边空缺，则必须用 . 表示，如下例：

```
$|x|=\left\{\begin{array}{ll}x, & \text{if } x \geq 0; \\-x, & \text{if } x < 0. \\ \end{array}\right.$
```

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


## 多行公式


利用 `align*` 环境，很容易写出多行的对齐公式。比如


```
\begin{align*}
(x+y)^2 &= (x+y)(x+y) \\
&= x^2 + xy + yx + y^2 \\
&= x^2 + 2xy + y^2
\end{align*}
```


$$\begin{aligned}(x+y)^2 &= (x+y)(x+y) \\ &= x^2 + xy + yx + y^2 \\ &= x^2 + 2xy + y^2\end{aligned}$$

其中 `&` 指明各行对齐的位置。将 `align*` 改为 `align` 将得到带编号的公式。


 1 开始使用

 2 快速入门

 3 撰写文本

 4 撰写公式

 5 错误处理

 6 命令速查

## 错误显示

在 TeXworks 编译文档时，下面的输出窗口将会显示相关信息。输出窗口有两个页面：

- ▶ “Console output” 页面显示详细的编译信息
- ▶ “Errors, warnings, badboxes” 页面只显示错误和警告
  - ▶ 编译错误用红色标示：这些问题必须改正后重新编译
  - ▶ 编译警告用黄色标示：这些问题初学者暂时可以忽略

如果文档没有错误，编译成功后输出窗口将会自动隐藏。



## 命令未定义

最常见的错误是命令名称输入错误。比如将 `\author{作者}` 错写为 `\autos{作者}`，编译后将会看到如下错误信息：

```
! Undefined control sequence.  
l.12 \autos
```

其中第一行说明错误原因是“命令未定义”，第二行说明错误出现在第 12 行的 `\autos` 这里。

## 缺少 \$ 号

第二常见的错误是忘记将公式放在一对 \$ 里面. 比如将  $2^3=8$  错写为  $2^3=8$ , 编译后将会看到如下错误信息:

```
! Missing $ inserted.  
<inserted text>  
      $  
l.19 2^  
      3=8
```

其中第一行说明错误原因是“缺少 \$ 号”, 后面几行说明错误出现在第 19 行的  $2^3=8$  这里.

## 花括号不配对

第三常见的错误是花括号无法配对. 比如将根号  $\sqrt{2}$  错写为  $\sqrt{2}$ , 编译后将会看到如下错误信息:

```
! Missing } inserted.  
<inserted text>  
      }  
l.17  $\sqrt{2}$ 
```

其中第一行说明错误原因是“缺少 } 号”, 后面几行说明错误出现在第 17 行这里.

1 开始使用

2 快速入门

3 撰写文本

4 撰写公式

5 错误处理

6 命令速查

## 特殊字符

在 LaTeX 中, 有几个特殊字符不能直接输入得到, 而需要用下面对应的命令:

<code>\{</code>	{	<code>\\$</code>	\$	<code>\#</code>	#	<code>\_</code>	—
<code>\}</code>	}	<code>\%</code>	%	<code>\&amp;</code>	&	<code>\^{} </code>	^

要得到反斜杠符号 `\`, 在文本中可以用 `\textbackslash`, 在公式中可以用 `\backslash` 命令.

## 希腊字母

<code>\alpha</code>	$\alpha$	<code>\mu</code>	$\mu$	<code>\epsilon</code>	$\epsilon$	<code>\ varepsilon</code>	$\varepsilon$
<code>\beta</code>	$\beta$	<code>\nu</code>	$\nu$	<code>\theta</code>	$\theta$	<code>\vartheta</code>	$\vartheta$
<code>\gamma</code>	$\gamma$	<code>\xi</code>	$\xi$	<code>\kappa</code>	$\kappa$	<code>\varkappa</code>	$\varkappa$
<code>\delta</code>	$\delta$	<code>\tau</code>	$\tau$	<code>\pi</code>	$\pi$	<code>\varpi</code>	$\varpi$
<code>\zeta</code>	$\zeta$	<code>\upsilon</code>	$\upsilon$	<code>\rho</code>	$\rho$	<code>\varrho</code>	$\varrho$
<code>\eta</code>	$\eta$	<code>\chi</code>	$\chi$	<code>\sigma</code>	$\sigma$	<code>\varsigma</code>	$\varsigma$
<code>\iota</code>	$\iota$	<code>\psi</code>	$\psi$	<code>\phi</code>	$\phi$	<code>\varphi</code>	$\varphi$
<code>\lambda</code>	$\lambda$	<code>\omega</code>	$\omega$				

<code>\Gamma</code>	$\Gamma$	<code>\Lambda</code>	$\Lambda$	<code>\Sigma</code>	$\Sigma$	<code>\Psi</code>	$\Psi$
<code>\Delta</code>	$\Delta$	<code>\Xi</code>	$\Xi$	<code>\Upsilon</code>	$\Upsilon$	<code>\Omega</code>	$\Omega$
<code>\Theta</code>	$\Theta$	<code>\Pi</code>	$\Pi$	<code>\Phi</code>	$\Phi$		

## 函数命令

<code>\sin</code>	<code>sin</code>	<code>\arcsin</code>	<code>arcsin</code>	<code>\det</code>	<code>det</code>	<code>\sup</code>	<code>sup</code>
<code>\cos</code>	<code>cos</code>	<code>\arccos</code>	<code>arccos</code>	<code>\dim</code>	<code>dim</code>	<code>\inf</code>	<code>inf</code>
<code>\tan</code>	<code>tan</code>	<code>\log</code>	<code>log</code>	<code>\hom</code>	<code>hom</code>	<code>\sinh</code>	<code>sinh</code>
<code>\cot</code>	<code>cot</code>	<code>\ln</code>	<code>ln</code>	<code>\gcd</code>	<code>gcd</code>	<code>\cosh</code>	<code>cosh</code>
<code>\sec</code>	<code>sec</code>	<code>\lg</code>	<code>lg</code>	<code>\max</code>	<code>max</code>	<code>\tanh</code>	<code>tanh</code>
<code>\csc</code>	<code>csc</code>	<code>\exp</code>	<code>exp</code>	<code>\min</code>	<code>min</code>	<code>\coth</code>	<code>coth</code>

## 数学符号

<code>\ge</code>	$\geq$	<code>\times</code>	$\times$	<code>\int</code>	$\int$	<code>\leftarrow</code>	$\leftarrow$
<code>\le</code>	$\leq$	<code>\div</code>	$\div$	<code>\iint</code>	$\iint$	<code>\rightarrow</code>	$\rightarrow$
<code>\neq</code>	$\neq$	<code>\pm</code>	$\pm$	<code>\sim</code>	$\sim$	<code>\Leftarrow</code>	$\Leftarrow$
<code>\cdot</code>	$\cdot$	<code>\mp</code>	$\mp$	<code>\approx</code>	$\approx$	<code>\Rightarrow</code>	$\Rightarrow$
<code>\cdots</code>	$\cdots$	<code>\partial</code>	$\partial$	<code>\cong</code>	$\cong$	<code>\leftrightarrow</code>	$\leftrightarrow$
<code> </code>	$ $	<code>\infty</code>	$\infty$	<code>\equiv</code>	$\equiv$	<code>\Leftrightarrow</code>	$\Leftrightarrow$



## 大型算符

公式中常见的大型算符有这几个：极限，求和，连乘。比如：

$$\text{\texttt{\$}\lim_{x\to 0}\sin x = 1\text{\texttt{\$}}}$$

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

$$\text{\texttt{\$}\sum_{k=1}^n f(k)\text{\texttt{\$}}}$$

$$\sum_{k=1}^n f(k)$$

$$\text{\texttt{\$}\prod_{k=1}^n g(k)\text{\texttt{\$}}}$$

$$\prod_{k=1}^n g(k)$$

这些大型算符在行内公式出现时，上下标将显示在右边。

若要让上下标显示在上下方，可在算符命令后加上 `\limits` 命令：

$$\text{\texttt{\$}\sum\limits_{k=1}^n f(k)\text{\texttt{\$}}}$$

$$\sum_{k=1}^n f(k)$$

## 参考资料

纸质书籍 刘海洋,《LaTeX 入门》,第 1 版,2013 年.

电子书籍 黄新刚,《LaTeX 笔记》,第 2 版,2013 年.

LaTeX 文档排版教程 [↗](#)

2018 年 10 月 15 日

## 第二部分 · 学习 LaTeX 正文内容

吕荐瑞 [🏠 lvjr.bitbucket.io](https://lvjr.bitbucket.io)

暨南大学数学系



7 文本段落

8 文本环境

9 文本盒子

10 插入表格

11 插入图片

## 7 文本段落

### 7.1 特殊字符

### 7.2 段落换行

### 7.3 行距调整

## 重音字符

在西欧文字中有一些重音字符，我们可以用下列命令来输入（以 o 为例）：

字符	输入	字符	输入
ò	\'{o}	ó	\b{o}
ó	\'{o}	ô	\c{o}
ô	\^{o}	õ	\d{o}
ö	\"{o}	ö	\H{o}
õ	\~{o}	õ	\u{o}
ō	\={o}	ő	\v{o}
ò	\.{o}	ôo	\t{oo}

## 标点符号

在 LaTeX 中，英文的左右引号是不同的。输入 ‘（键盘上 1 左边的字符）得到左单引号，输入 ’（键盘上 ; 右边的字符）得到右单引号用。而要得到左右双引号，可以连续输入两个左右单引号，即 ‘ ‘ 和 ’ ’。

‘Text’ or ‘ ‘Text’ ’

‘Text’ or “Text”

连字号、起止号和破折号看起来都是横线，但输入方式和横线长度都是不同的。

- ▶ 用 - 得到连接单词的连字号 -
- ▶ 用 -- 得到表示数字范围的起止号 -
- ▶ 用 --- 得到英文的破折号 —

en-dash in 1--9 --- like this

en-dash in 1-9 — like this

## 空格字符

LaTeX 对空格符的处理方式与 Microsoft Word 大不相同. 比如

- ▶ 连续多个空格符等同于一个空格符
- ▶ 各行最前面的任意多个空格符都被忽略
- ▶ 换行符被转换为一个空格符

```
One Two Three  
Four
```

```
One Two Three Four
```

要去掉换行的空格符, 可以用注释符 %; 因为注释符后面的字符将被忽略, 包括换行符.

```
One Two Three%  
Four
```

```
One Two ThreeFour
```



## 7 文本段落

### 7.1 特殊字符

### 7.2 段落换行

### 7.3 行距调整

## 段落换行

用一个空行或者 `\par` 命令可以开始新的段落，同时会有默认的首行缩进。

```
I can eat glass, it doesn't hurt me.
```

```
I can eat glass, it doesn't hurt me.
```

```
\par
```

```
I can eat glass, it doesn't hurt me.
```

```
    I can eat glass, it doesn't hurt me.
```

```
    I can eat glass, it doesn't hurt me.
```

```
    I can eat glass, it doesn't hurt me.
```

## 段落换行

段落缩进量记录在 `\parindent` 命令中, 用 `\setlength` 命令可以修改. 比如

```
\setlength{\parindent}{1em}
```

I can eat glass, it doesn't hurt me.

I can eat glass, it doesn't hurt me.

I can eat glass, it doesn't hurt me.

I can eat glass, it doesn't hurt me.

## 段落换行

用 `\\` 或者 `\newline` 可以强制换行在下一行继续, 且在下一行不会有缩进.

```
I can eat glass, \\ it doesn't hurt me.
```

```
I can eat glass, \newline it doesn't hurt me.
```

```
I can eat glass,  
it doesn't hurt me.
```

```
I can eat glass,  
it doesn't hurt me.
```

## 段落换行

\\ 命令还可以加上可选参数，指明在换行时增加额外的间距。比如

```
I can eat glass, \\[10pt] it doesn't hurt me.
```

```
I can eat glass,
```

```
it doesn't hurt me.
```

另外还有 \\\* 命令，表示在此换行，但不能换页。

## 7 文本段落

### 7.1 特殊字符

### 7.2 段落换行

### 7.3 行距调整

## 行距调整

在 LaTeX 中，默认的行距通常是字体大小的 1.2 倍。如果字体大小为 10pt，默认行距 `\baselineskip` 将为 12pt。若要调整行距，可以直接修改 `\baselineskip` 的值。例如下面将调整行距为双倍行距：

```
\setlength{\baselineskip}{24pt}
```

其中，在 LaTeX 中 `\setlength` 命令用于修改各种长度的值。

## 行距调整

在 LaTeX 中调整行距也可通过修改伸展因子 `\baselinestretch`. 修改后的新行距将等于默认行距乘以伸展因子. 例如下面命令将调整行距为双倍行距:

```
\renewcommand{\baselinestretch}{2}\selectfont
```

```
\linespread{2}\selectfont
```

上面两种修改伸展因子的方法是同样的, 只是第二种方法更简单点. 注意修改伸展因子后不会立即生效, 在下次设定字体大小时才能生效, 因此我们在后面加上了 `\selectfont` 命令. 如果放在导言区则无需此命令, 因为文档开始时自然会设定字体大小.



## 行距调整

在 LaTeX 中修改 `\baselineskip` 和 `\baselinestretch` 都可以改变行距。若同时修改两者，并不会叠加行距，而是以后一次修改为准。这是因为每次设定字体大小时 LaTeX 都保存了默认行距，而修改 `\baselinestretch` 时使用默认行距，而非当前 `\baselineskip` 值，来计算新行距。

在一个段落的各行中，行距 `\baselineskip` 的值保持相同。若在一个段落中多次修改行距，就只有最后一次改动生效。

对于双倍行距，有些人认为它应该指行距等于字体大小的两倍。即如果字体大小为 10pt，双倍行距应该为 20pt。为方便设置这种行距倍数，可以用 `setspace` 宏包。

⑦ 文本段落

⑧ 文本环境

⑨ 文本盒子

⑩ 插入表格

⑪ 插入图片

## 文本环境

在 LaTeX 中，用 `\begin` 和 `\end` 包含起来的特殊段落，即

```
\begin{环境名}...\end{环境名}
```

我们称为环境。不同环境中的内容将用不同的方式来排版。在文本段落中，常见的环境有列表环境，对齐环境和定理环境等。

## 8 文本环境

### 8.1 列表环境

### 8.2 对齐环境

### 8.3 定理环境

### 8.4 抄录环境

## 列表环境

列表环境有三种：无序列表（itemize）、有序列表（enumerate）和描述列表（description）。它们的使用方式见下面例子：

```
\begin{itemize}
```

```
\item 无编号的列表
```

```
\item 带编号的列表
```

```
\end{itemize}
```

```
\begin{enumerate}
```

```
\item 无编号的列表
```

```
\item 带编号的列表
```

```
\end{enumerate}
```

```
\begin{description}
```

```
\item[无序列表] 无编号的列表.
```

```
\item[有序列表] 带编号的列表.
```

```
\end{description}
```

► 无编号的列表

► 带编号的列表

1. 无编号的列表

2. 带编号的列表

无序列表 无编号的列表.

有序列表 带编号的列表.

## 8 文本环境

### 8.1 列表环境

### 8.2 对齐环境

### 8.3 定理环境

### 8.4 抄录环境

## 居中对齐

默认情形，文章段落都是两侧对齐的，但我们也可以排版居中对齐和单侧对齐的段落。在 LaTeX 中，可以用 `center` 环境得到居中的文本段落，其中可以用 `\\` 换行。例如：

```
\begin{center}
The quick brown fox jumps over the lazy dog.\\
The quick brown fox jumps over the lazy dog.
\end{center}
```

The quick brown fox jumps over the lazy dog.  
The quick brown fox jumps over the lazy dog.

如果居中段落在一行放不下，只会在最后一行是居中的，其它行都填满页面宽度。在一个环境内部，用命令 `\centering` 也可以让后面的文本都居中放置。

## 向左对齐

类似地，可以用 `flushleft` 环境得到向左对齐的文本段落。例如：

```
\begin{flushleft}  
The quick brown fox jumps over the lazy dog.  
The quick brown fox jumps over the lazy dog.  
The quick brown fox jumps over the lazy dog.  
The quick brown fox jumps over the lazy dog.  
\end{flushleft}
```

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

类似地，在一个环境内部，用 `\raggedright` 声明同样可以让文本向左对齐。



## 向右对齐

同样地，可以用 `flushright` 环境得到向右对齐的文本段落。例如：

```
\begin{flushright}
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
\end{flushright}
```

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

类似地，在一个环境内部，用 `\raggedleft` 声明同样可以让文本向右对齐。

## 8 文本环境

### 8.1 列表环境

### 8.2 对齐环境

### 8.3 定理环境

### 8.4 抄录环境

## 定理环境

定理命题的撰写的最简单的例子：

```
\newtheorem{dingli}{Theorem}  
\newtheorem{tuilun}{Corollary}  
\begin{dingli}  
This is a theorem.  
\end{dingli}  
\begin{tuilun}  
This is a corollary.  
\end{tuilun}
```

**Theorem 1** This is a theorem.

**Corollary 1** This is a corollary.

这样的输出结果就是各自编号的定理和推论了，定义、命题等等也类似可以这么使用。

## 定理编号

若希望将定理和推论一起编号，前面可以改为这样（表示 corl 也使用 thrm 的编号）：

```
\newtheorem{thrm}{Theorem}  
\newtheorem{corl}[thrm]{Corollary}  
\begin{thrm}  
This is a theorem.  
\end{thrm}  
\begin{corl}  
This is a corollary.  
\end{corl}
```

Theorem 1 This is a theorem.

Corollary 2 This is a corollary.

## 定理编号

最后，如果你希望在新的一节开始时重置定理的编号，可以这么使用：

```
\newtheorem{thm}{Theorem}[section]
\begin{thm}
This is another theorem.
\end{thm}
\renewcommand{\thethm}{\arabic{section}-\arabic{thm}}
\begin{thm}
This is another theorem.
\end{thm}
```

Theorem 8.1 This is another theorem.

Theorem 8-2 This is another theorem.

这个例子也展示了如何修改定理编号的显示方式。后面将介绍如何定制各种编号计数器。

## 8 文本环境

### 8.1 列表环境

### 8.2 对齐环境

### 8.3 定理环境

### 8.4 抄录环境

## 抄录环境

LaTeX 中有一些特殊字符不能直接输入，如果要排版程序代码就不那么方便了。此时可用 `verbatim` 环境。比如：

```
\begin{verbatim}
function test() {
    alert("Hello World!");
}
\end{verbatim}

function test() {
    alert("Hello World!");
}
```

## 抄录命令

类似地还有个抄录命令 `\verb` 用于简短的抄录文本。比如：

```
\verb!\textbf{A}! and \verb=\textit{B}=
\textbf{A} and \textit{B}
```

注意 `\verb` 命令的参数**必须**用两个相同的符号包含起来，**不能用花括号包含参数**。



7 文本段落

8 文本环境

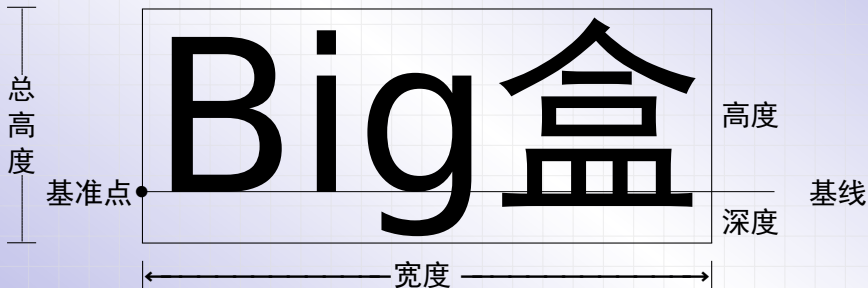
9 文本盒子

10 插入表格

11 插入图片

## 文本盒子

在 TeX 排版文档时，所有内容都会被转换为嵌套的盒子。每个字符是一个盒子，多个字符盒子组成一个行盒子，而多个行盒子组成一个页盒子。



每个盒子有三个尺寸：宽度 (width), 高度 (height) 和深度 (depth)。

## 9 文本盒子

### 9.1 左右盒子

### 9.2 段落盒子

### 9.3 标尺盒子

### 9.4 幻影盒子

### 9.5 盒子变换

### 9.6 盒子变量

## 左右盒子

左右盒子是最简单的盒子，它包括 `\mbox` 和 `\fbox`。例如：

正文 `\mbox{盒子}` 正文 `\fbox{盒子}` 正文 正文 盒子 正文 盒子 正文

可以看出两者的区别在于后者带有边框。

对应的还有另外两种左右盒子 `\makebox` 和 `\framebox`，它们提供更多的选项，比如可以指定盒子的宽度：

正文 `\makebox[8em]{盒子}` 正文 \\  
正文 `\framebox[8em]{盒子}` 正文

正 文          盒子          正  
文  
正 文 盒子 正  
文

## 左右盒子

用 `\makebox` 和 `\framebox` 生成盒子时还可以指定盒子中文本的对齐方式，例如：

```
正文\makebox[10em][l]{靠左对齐}正文\\
正文\framebox[10em][l]{靠左对齐}正文\\
正文\makebox[10em][c]{居中对齐}正文\\
正文\framebox[10em][c]{居中对齐}正文\\
正文\makebox[10em][r]{靠右对齐}正文\\
正文\framebox[10em][r]{靠右对齐}正文
```

正文	靠左对齐	正文
正文	<div>靠左对齐</div>	正文
正文	居中对齐	正文
正文	<div>居中对齐</div>	正文
正文	靠右对齐	正文
正文	<div>靠右对齐</div>	正文

其中 `l`、`c`、`r` 分别表示靠左、居中、靠右对齐。

## 左右盒子

用 `\makebox` 和 `\framebox` 生成盒子时还可以指定盒子中文本的对齐方式，例如：

```
正文\makebox[10em][s]{分散对齐}正文\\
正文\framebox[10em][s]{分散对齐}正文\\
正文\makebox[10em][s]{A Lazy Dog}正文\\
正文\framebox[10em][s]{A Lazy Dog}正文
```

正文	分	散	对	齐	正文
正文	分	散	对	齐	正文
正文	A	Lazy		Dog	正文
正文	A	Lazy		Dog	正文

其中 `s` 表示分散对齐。使用分散对齐选项时，对中文以汉字为单位分散，而对于西文则以单词为单位分散。

## 左右盒子

在定义 `\makebox` 和 `\framebox` 的宽度时，我们可以用 `\width`、`\height`、`\depth` 和 `\totalheight` 这些长度量，它们分别表示盒子内容的宽度、高度、深度和总高度（即高度和深度之和）。比如下面例子设定盒子的宽度等于内容宽度的三倍：

正文 `\framebox[3\width]{盒子}` 正文    正文  正文

对于带框盒子 `\fbox` 和 `\framebox`，我们可以用 `\fboxsep` 调整边框和内容之间的距离，用 `\fboxrule` 调整框线的宽度。两者的默认值分别为 3pt 和 0.4pt。例如：

`\setlength{\fboxsep}{8pt}`  
`\setlength{\fboxrule}{1pt}`  
正文 `\fbox{盒子}` 正文

正文  正文

## 9 文本盒子

### 9.1 左右盒子

### 9.2 段落盒子

### 9.3 标尺盒子

### 9.4 幻影盒子

### 9.5 盒子变换

### 9.6 盒子变量



## 段落盒子

左右盒子中不能包含多行文本或者段落. 此时我们可以改用 `\parbox` 命令或者 `minipage` 环境. 例如:

```
Left\parbox{3em}{One\\ Two\\ Three}Right
\hfill
Left\begin{minipage}{6em}
One\\ Two\\ Three
\end{minipage}Right
```

One  
LeftTwo Right  
Three

One  
LeftTwo Right  
Three

其中必须指定盒子宽度. 两者的用法和参数都一样, 只是在 `\parbox` 中不能包含某些命令或环境, 而 `minipage` 中的内容几乎没有限制.

## 段落盒子

`\parbox` 和 `minipage` 都不含边框，但我们可以将它们放在 `\fbox` 中得到边框，例如：

```
Left\fbox{\begin{minipage}{6em}  
One\\ Two\\ Three  
\end{minipage}}Right
```

Left	One Two Three	Right
------	---------------------	-------

段落盒子有三个可选参数，第一个设定段落盒子和两边内容的对齐。

```
Left\fbbox{\begin{minipage}[t]{4em}  
One\\ Two\\ Three  
\end{minipage}}Right
```

Left 

One
Two
Three

 Right

```
Left\fbbox{\begin{minipage}[c]{4em}  
One\\ Two\\ Three  
\end{minipage}}Right
```

Left 

One
Two
Three

 Right

```
Left\fbbox{\begin{minipage}[b]{4em}  
One\\ Two\\ Three  
\end{minipage}}Right
```

Left 

One
Two
Three

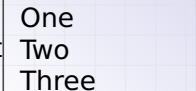
 Right

其中，默认选项 c 表示段落盒子的中线和外部内容的基线对齐，而选项 t 表示盒子顶行的基线和外部内容的基线对齐，选项 b 表示盒子底行的基线和外部内容的基线对齐。

## 段落盒子

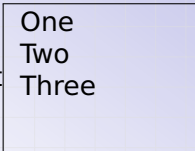
第二个可选参数用于设定段落盒子的高度。例如：

```
Left\fbbox{  
  \begin{minipage}[c][6em]{6em}  
    One\\ Two\\ Three  
  \end{minipage}  
}Right
```

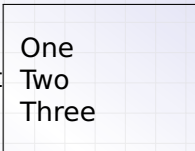
Left  Right

第三个可选参数用于设定段落盒子内部的纵向对齐方式。例如：

```
Left\fbbox{  
  \begin{minipage}[c][5em][t]{6em}  
    One\\ Two\\ Three  
  \end{minipage}  
}Right
```

Left  Right

```
Left\fbbox{  
  \begin{minipage}[c][5em][c]{6em}  
    One\\ Two\\ Three  
  \end{minipage}  
}Right
```

Left  Right

其中默认值 `c` 表示居中对齐，`t` 和 `b` 分别表示顶部对齐和底部对齐。

第三个可选参数用于设定段落盒子内部的纵向对齐方式。例如：

```
Left\fbbox{
  \begin{minipage}[c][6em][s]{3em}
    One\vfill Two\vfill Three
  \end{minipage}
}Right
```

Left 

One
Two
Three

 Right

```
Left\fbbox{
  \begin{minipage}[c][6em][s]{3em}
    \setlength{\baselineskip}{10pt plus 2pt}
    One\\ Two\\ Three
  \end{minipage}
}Right
```

Left 

One
Two
Three

 Right

其中 s 表示分散对齐。在使用分散对齐时，内容中需要有竖直的弹性长度。上面的例子中，分别通过直接使用 `\vfill` 命令和设定弹性行距 `\baselineskip` 得到弹性长度。

## 9 文本盒子

### 9.1 左右盒子

### 9.2 段落盒子

### 9.3 标尺盒子

### 9.4 幻影盒子

### 9.5 盒子变换

### 9.6 盒子变量

## 标尺盒子

LaTeX 中还提供了 `\rule` 命令，用于生成标尺盒子，即不含内容的实心矩形盒子。例如：

Left `\rule{3pt}{14pt}` Right

Left  Right

其中 `\rule` 命令的两个参数分别表示宽度和高度。

宽度为零的标尺盒子虽然无法看到，但可以起到支撑的作用，我们称之为**支架** (strut)。例如：

`\fbox{a} \fbox{A} \quad`  
`\fbox{a\rule{0pt}{7pt}}`  
`\fbox{A\rule{0pt}{7pt}}`




在这个例子中，由于 `a` 和 `A` 的高度不一，左边两个盒子的高度不一样，但加入支架后右边两个盒子的高度就相同了。



## 标尺盒子

标尺盒子默认与该行的基线平行，通过设定 `\rule` 命令的可选参数，可以升高或降低其水平位置。例如：

默认 `\rule{3pt}{14pt}` 默认  
上升 `\rule[5pt]{3pt}{14pt}` 上升  
下降 `\rule[-5pt]{3pt}{14pt}` 下降

默认  默认上升  上升下  
降  下降

## 9 文本盒子

### 9.1 左右盒子

### 9.2 段落盒子

### 9.3 标尺盒子

### 9.4 幻影盒子

### 9.5 盒子变换

### 9.6 盒子变量

## 幻影盒子

L<sup>A</sup>T<sub>E</sub>X 中还有一种幻影盒子，它们的内容看不到，但内容所占据的空间保持不变。幻影盒子可以用 `\phantom` 命令得到，例如：

正文 `\phantom{幻影}` 正文

正文

正文

类似地还有 `\hphantom` 和 `\vphantom` 命令。`\hphantom` 用于生成水平幻影，其宽度等于内容的宽度，而高度和深度为零。`\vphantom` 用于生成竖直幻影，其高度和深度分别等于内容的高度和深度，而宽度为零。例如：

正文 正文 `\`

正文 `\hphantom{水平幻影}` 正文 `\`

正文 `\vphantom{\Huge 竖直幻影}` 正文

正文 正文

正文

正文

正文 正文

在这个例子中，竖直幻影的高度较大，撑开了第二行和第三行的间距。

## 9 文本盒子

### 9.1 左右盒子

### 9.2 段落盒子

### 9.3 标尺盒子

### 9.4 幻影盒子

### 9.5 盒子变换

### 9.6 盒子变量

利用 `\raisebox` 可以升高或者降低盒子的位置。例如：

正文 `\raisebox{6pt}{上升}` 正文 `\` 正文 上升 正文  
正文 `\raisebox{-6pt}{\fbox{下降}}` 正文 正文 下降 正文

`L\raisebox{1pt}{\tiny A}%`  
`T\raisebox{-2pt}{E}X`

L<sup>A</sup>T<sub>E</sub>X

我们也可以在 `\raisebox` 命令中直接指定盒子的最终高度和深度。例如：

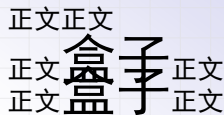
正文 正文 正文 `\` 正文 正文 正文  
正文 `\raisebox{6pt}[30pt][9pt]{\fbox{上升}}` 正文 上升 正文  
正文

其中两个可选参数分别表示所需的高度和深度，深度也可以不指定。

## 盒子变换

利用 `\smash` 命令，可以将已有的盒子拍扁，即保持宽度不变，但将高度和深度变为零。例如：

```
正文正文\\  
正文{\huge 盒子}正文\\  
正文\smash{\huge 盒子}正文
```

正文正文  
正文  正文  
正文

其中第三行的“盒子”的高度已经变为零，所以  $\text{T}_{\text{E}}\text{X}$  排版时不会自动增加第二行和第三行的间距，使得两行部分重叠。

## 盒子变换

若载入 graphics 或 graphicx 宏包，还可以使用这些盒子变换命令：`\resizebox`、`\scalebox`、`\reflectbox` 和 `\rotatebox`。利用 `\resizebox` 命令，可以对盒子进行伸缩变换，例如：

正常	<code>\resizebox{30pt}{5pt}</code>	{矮胖}	正常	正常	矮胖	正常
正常	<code>\resizebox{11pt}{13pt}</code>	{瘦高}	正常	正常	瘦高	正常
正常	<code>\resizebox{30pt}{!}</code>	{胖高}	正常	正常	胖高	正常

该命令的两个参数分别表示给该盒子指定的宽度和总高度。如果其中一个参数取为 `!`，则表示保持宽高比例不变且按照另一参数进行伸缩。

## 盒子变换

利用 `\scalebox` 命令，同样可以对盒子进行伸缩变换，例如：

```
正常\scalebox{1.6}[0.7]{矮胖}正常\\  
正常\scalebox{0.7}[1.6]{瘦高}正常\\  
正常\scalebox{1.6}{胖高}正常
```

正常矮胖正常  
正常瘦高正常  
正常胖高正常

该命令的两个参数分别表示水平和竖直伸缩因子。如果省略竖直伸缩因子，则表示它和水平伸缩因子相同。



## 盒子变换

`\scalebox` 命令的伸缩因子也可以是负数，水平伸缩因子为负数表示伸缩后作水平翻转，而竖直伸缩因子为负数表示伸缩后作竖直翻转。例如：

正常 `\scalebox{-1.6}[0.7]{矮胖}` 正常 \\  
正常 `\scalebox{0.7}[-1.6]{瘦高}` 正常

正常 半 矮 正常  
正常 瘦 正常  
通 回 早

特别地，当水平和竖直伸缩因子分别为  $-1$  和  $1$  时，就表示仅作水平翻转，这等同于 `\reflectbox` 命令。例如：

正常 `\scalebox{-1}[1]{翻转}` 正常 \\  
正常 `\reflectbox{翻转}` 正常

正常 转 正 正常  
正常 转 正 正常

`X\raisebox{-2pt}{\reflectbox{E}}%`  
`T\raisebox{-2pt}{E}X`

X<sub>2</sub>TEX

## 盒子变换

利用 `\rotatebox` 命令，可以对盒子进行旋转变换，例如：

正常 `\rotatebox{45}{逆时针}` 正常 \\  
正常 `\rotatebox{-30}{顺时针}` 正常

正常 逆时针  
正常 顺时针  
正常

其中的参数指明旋转角度，大于零表示逆时针旋转，小于零表示顺时针旋转。

## 9 文本盒子

### 9.1 左右盒子

### 9.2 段落盒子

### 9.3 标尺盒子

### 9.4 幻影盒子

### 9.5 盒子变换

### 9.6 盒子变量

## 定义盒子变量

要将某些复杂内容保存起来，以后多次使用，可以定义一个盒子变量。  
比如

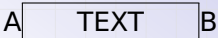
```
\newsavebox{\myboxa}  
\sbox{\myboxa}{TEXT} TEXT or TEXT  
\usebox{\myboxa} or \fbox{\usebox{\myboxa}}
```

- ▶ `\newsavebox` 将 `\myboxa` 定义为一个盒子变量
- ▶ `\sbox` 将内容存入 `\myboxa` 盒子里
- ▶ `\usebox` 从 `\myboxa` 盒子取出内容

## 定义盒子变量

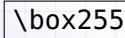
若要设定盒子的宽度及盒子内部的对齐方式，可以将 `\sbox` 命令换成 `\savebox` 命令：

```
\newsavebox{\myboxb}  
\savebox{\myboxb}[6em][c]{TEXT}  
A\fbbox{\usebox{\myboxb}}B
```



若要在盒子中存放抄录文本，可以将 `\sbox` 命令换成 `\lrbox` 环境：

```
\newsavebox{\myboxc}  
\begin{\lrbox}{\myboxc}  
\verb!\box255!  
\end{\lrbox}  
\fbbox{\usebox{\myboxc}}
```



⑦ 文本段落

⑧ 文本环境

⑨ 文本盒子

⑩ 插入表格

⑪ 插入图片

## 10 插入表格

### 10.1 基本表格

### 10.2 跨列表格

### 10.3 跨行表格

### 10.4 浮动表格

### 10.5 表格行距

我们从下面的例子来看看如何用 tabular 环境插入表格：

```
\begin{tabular}{|l|c|r|}  
\hline  
左列    & 中列    & 右列  \\  
\hline  
第二行  & 第二行  & 第二行  \\  
\hline  
左三    & 中三    & 右三  \\  
\hline  
\end{tabular}
```

左列	中列	右列
第二行	第二行	第二行
左三	中三	右三

首先，tabular 环境的参数 `|l|c|r|` 指明了各列的对齐方式，`l`、`c` 和 `r` 分别表示左对齐、居中对齐和右对齐。中间的竖线 `|` 指明各列之间有竖线分隔，如果在某些地方不需要竖线，去掉相应位置的 `|` 即可。表格各行的元素之间用 `&` 号分隔，两行内容用 `\\` 分隔。`\hline` 表示两行之间的横线；你可以用连续两个 `\hline` 得到双横线，或者去掉 `\hline` 以不显示该横线。



对左中右这三种列格式，在其中的单元格中用 `\newline` 无法换行。如果需要在单元格中填写多行内容，可以将内容放在段落盒子里面（比如 `\parbox` 盒子）；或者使用 `p` 列格式直接指定宽度，比如：

```
\begin{tabular}{|l|c|p{3em}|}  
\hline  
左列    & 中列    & 右列  \\  
\hline  
第二行  & 第二行  & 第二行\newline 换行啦  \\  
\hline  
左三    & 中三    & 右三  \\  
\hline  
\end{tabular}
```

左列	中列	右列
第二行	第二行	第二行 换行啦
左三	中三	右三

## 10 插入表格

### 10.1 基本表格

### 10.2 跨列表格

### 10.3 跨行表格

### 10.4 浮动表格

### 10.5 表格行距

在 tabular 环境中，我们可用 \multicolumn 命令得到跨列表格。  
比如：

```
\begin{tabular}{|c|c|c|}  
\hline  
第一行 & 第一行 & 第一行 \\  
\hline  
\multicolumn{2}{|c|}{跨列} & 第二行 \\  
\hline  
第三行 & 第三行 & 第三行 \\  
\hline  
\end{tabular}
```

第一行	第一行	第一行
跨列		第二行
第三行	第三行	第三行

上面的 \multicolumn 命令的第一个参数指明要横跨的列数，第二个参数指明对齐和边框线，第三个参数指明该单元格的内容。

## 10 插入表格

### 10.1 基本表格

### 10.2 跨列表格

### 10.3 跨行表格

### 10.4 浮动表格

### 10.5 表格行距

要使用跨行表格需要先载入 multirow 宏包. 下面是一个例子:

```
\begin{tabular}{|c|c|c|}  
\hline  
\multirow{2}{*}{第一行} & 第一行 & 第一行 \\  
\cline{2-3}  
      & 第二行 & 第二行 \\  
\hline  
第三行 & 第三行 & 第三行 \\  
\hline  
\end{tabular}
```

第一行	第一行	第一行
	第二行	第二行
第三行	第三行	第三行

上面的 `\multirow` 命令的第一个参数指明要横跨的行数, 第二个参数指明宽度 (用 \* 则表示自然宽度), 第三个参数指明该单元格的内容. 另外 `\cline` 命令可以画出限制在某些列之间的边框线.

## 10 插入表格

### 10.1 基本表格

### 10.2 跨列表格

### 10.3 跨行表格

### 10.4 浮动表格

### 10.5 表格行距

## 浮动表格

在前面插入表格的例子中，表格是在 `tabular` 环境对应的位置排版出来的。如果表格高度大于当前页剩余高度，表格就会被放置到下一页中，造成这一页下部留出很大空白。

大部分时候我们并不需要严格限定表格出现的位置，而只要求表格在该段正文的附近出现。此时，我们可以用 `table` 浮动环境来达到自动调整位置的效果。

例如下面的代码得到的是自动浮动的表 1:

```
\begin{table}[htbp!]  
\centering  
\begin{tabular}{|lccr|}  
\hline  
第一行 & 第一行 & 第一行 & 第一行 \\  
第二行 & 第二行 & 第二行 & 第二行 \\  
\hline  
\end{tabular}  
\caption{浮动表格例子}  
\label{tab:float}  
\end{table}
```

第一行	第一行	第一行	第一行
第二行	第二行	第二行	第二行

**表 1:** 浮动表格例子



## 浮动表格

其中的可选参数里，h（here，当前位置）、t（top，页面顶部）、b（bottom，页面底部）、p（page，单独一页）表明允许将表格放置在哪些位置，而！表示不管某些浮动的限制。用 table 浮动环境，还可以用 \caption 命令指明表格的名称，并得到表格的自动编号。

在 \caption 后面用 \label 命令，可以给浮动表格加上标签。在文档其它地方可以用 \ref 命令引用该表格。

## 10 插入表格

### 10.1 基本表格

### 10.2 跨列表格

### 10.3 跨行表格

### 10.4 浮动表格

### 10.5 表格行距

## 表格行距

表格中行距默认等于 `\baselineskip`. 通过修改 `\arraystretch` 因子的值, 我们可以单独调整表格的行距. 下面例子将表格行距修改为原来的 1.5 倍.

```
\renewcommand\arraystretch{1.5}
```

7 文本段落

8 文本环境

9 文本盒子

10 插入表格

11 插入图片

## 插入图片

在  $\text{\LaTeX}$  文档中插入现有的图形，可以使用 `graphics` 或 `graphicx` 宏包，其中 `graphicx` 宏包是对 `graphics` 宏包的改进。我们这里只介绍 `graphicx` 宏包。这一节中的例子都是在载入 `graphicx` 宏包之后才能使用的：

```
\usepackage{graphicx}
```

## 11 插入图片

### 11.1 图文并排

### 11.2 图文分开

### 11.3 浮动图片

首先我们看看小图片的插入.用 `\includegraphics[选项]{图形文件}` 命令来插入图形. 例如

```
文本句子文本句子. 文本句子文本句子. 文本句子文本句子.  
文本句子文本句子. 文本句子文本句子. 文本句子文本句子.  
\includegraphics[scale=0.1]{example-image}  
文本句子文本句子. 文本句子文本句子. 文本句子文本句子.  
文本句子文本句子. 文本句子文本句子. 文本句子文本句子.
```

文本句子文本句子. 文本句子文本句子. 文本句子文本句子. 文本句子文本句子.




文本句子文本句子. 文本句子文本句子. 文本句子文本句子. 文本句子文本句子.  
文本句子文本句子. 文本句子文本句子. 文本句子文本句子. 文本句子文本句子.  
文本句子.

`\includegraphics` 命令有许多选项, 上面的选项 `scale=0.1` 指明了整体的伸缩因子, 常用的选项还有宽度值和高度值选项, 例如 `width=64mm` 和 `height=48mm` 等等. 如果宽度值和高度值只指明一项, 将按同比例对另一项作伸缩.

## 图文并排

上面的图文混排的例子中，图片是和正文的基线对齐的，当图片高度比行距大时，结果不是很美观。你可以用 `\raisebox` 命令稍微降低图片的位置，例如

```
文本句子文本句子。 文本句子文本句子。 文本句子文本句子。  
文本句子文本句子。 文本句子文本句子。 文本句子文本句子。  
\raisebox{-2mm}{\includegraphics[scale=0.1]{example-image}}  
文本句子文本句子。 文本句子文本句子。 文本句子文本句子。  
文本句子文本句子。 文本句子文本句子。 文本句子文本句子。
```

文本句子文本句子。 文本句子文本句子。 文本句子文本句子。 文本句子文本句子。  
文本句子文本句子。 文本句子文本句子。  文本句子文本句子。 文本句子文  
本句子。 文本句子文本句子。 文本句子文本句子。 文本句子文本句子。 文本句子  
文本句子。



## 11 插入图片

### 11.1 图文并排

### 11.2 图文分开

### 11.3 浮动图片

## 图文分开

如果你要插入的是大图片，一般不会和正文混排，而是需要独立居中显示。这可以通过把插入的图片放在 center 环境中来实现。例如

```
文本句子文本句子。 文本句子文本句子。 文本句子文本句子。  
\begin{center}  
\includegraphics[scale=0.2]{example-image}  
\end{center}  
文本句子文本句子。 文本句子文本句子。 文本句子文本句子。
```

文本句子文本句子。 文本句子文本句子。 文本句子文本句子。



文本句子文本句子。 文本句子文本句子。 文本句子文本句子。

## 图文分开

图片大小可以用伸缩因子指定，也可将它设为页芯宽度 `\textwidth` 的某个因子。例如

```
文本句子文本句子。 文本句子文本句子。 文本句子文本句子。  
\begin{center}  
\includegraphics[width=0.15\textwidth]{example-image}  
\end{center}  
文本句子文本句子。 文本句子文本句子。 文本句子文本句子。
```

文本句子文本句子。 文本句子文本句子。 文本句子文本句子。



文本句子文本句子。 文本句子文本句子。 文本句子文本句子。

## 11 插入图片

### 11.1 图文并排

### 11.2 图文分开

### 11.3 浮动图片

## 浮动图片

和插入表格的情形类似，有时候我们也需要自动调整图片的位置。此时，我们可以用 figure 浮动环境来达到这个效果。例如下面的代码得到的是自动浮动的图 1：

```
\begin{figure}[htbp!]  
\centering  
\includegraphics[scale=0.2]{example-image}  
\caption{ 图片样例 }  
\label{fig:float}  
\end{figure}
```



图 1: 图片样例

## 浮动图片

其中的可选参数和浮动表格的 `table` 环境的一样，而且同样可以用 `\caption` 命令指明图片的名称，并得到图片的自动编号。

在 `\caption` 后面用 `\label` 命令，可以给浮动图片加上标签。在文档其它地方可以用 `\ref` 命令引用该图片。

LaTeX 文档排版教程 [↗](#)

2018 年 10 月 15 日

---


## 第三部分 · 学习 LaTeX 文档结构


吕荐瑞 [🏠 lvjr.bitbucket.io](https://lvjr.bitbucket.io)


暨南大学数学系



12 标题摘要

13 章节目录

14 参考文献

15 全文索引



## 标题摘要


用下面的代码可以加入文章的标题、作者和日期信息：

```
\title{Notes On LaTeX Typesetting}  
\author{Some One}  
\date{November 13, 2011}  
\maketitle
```


如果 `\date{}` 命令的参数为空，则不显示日期信息。如果不出现 `\date` 命令，则默认显示当前的日期。


用下面的代码可以加入文章的内容摘要：

```
\begin{abstract}  
some abstract...  
\end{abstract}
```

12 标题摘要

13 章节目录

14 参考文献

15 全文索引

## 章节目录

在 book 和 report 文档类中，可使用下面这些章节命令

```
\part \chapter \section \subsection \subsubsection  
\paragraph \subparagraph
```

在 article 文档类中，除了 \chapter 不能用，其它的都可以用。例如下面的代码

```
\part{部分标题}  
\chapter{章标题}这一章我们介绍这些内容。  
\section{节标题}这一节我们介绍这些内容。  
\subsection{小节标题}这一小节我们介绍这些内容。  
\subsubsection{子节标题}这一子节我们介绍这些内容。  
\paragraph{段标题}这一段我们介绍这些内容。  
\subparagraph{小段标题}这一小段我们介绍这些内容。
```

## 章节目录

用 `\tableofcontents` 命令可以自动从各章节标题生成目录，但此时需要编译两次。上面各个章节命令都对应带有带 \* 号的命令（例如 `\section*`，`\subsection*` 等等），这些带 \* 号的章节标题将不会自动编号，也不会出现在目录中。

在导言区中用下面的命令载入 `hyperref` 宏包

```
\usepackage{hyperref}
```

就可以让生成的文章目录有链接，点击时会自动跳转到该章节。而且也会使得生成的 pdf 文件带有目录书签。

## 13 章节目录

### 13.1 添加目录项

### 13.2 添加书签项

### 13.3 替换书签项

### 13.4 目录页格式

## 添加目录项

利用 `\addcontentsline` 命令，我们可以在目录中添加一项。例如：

```
\addcontentsline{toc}{section}{章节描述}  
\addcontentsline{lof}{figure}{插图描述}  
\addcontentsline{lot}{table}{表格描述}
```

其中，`\addcontentsline` 命令的第一个参数表示添加到哪个目录，`toc`、`lof`、`lot` 分别表示章节目录、插图目录和表格目录。第二个参数表示该项的类型，对于章节目录可以是 `chapter`、`section`、`subsection` 等，而对于插图目录或者表格目录只能是 `figure` 或 `table`。第三个参数表示该项在目录中的文本。

对于章节目录，TeX 程序第一次编译文档时将生成 `.toc` 临时文件，而第二次编译时将从读取 `.toc` 文件内容并生成目录。插图目录和表格目录的情形类似。利用 `tocloft` 宏包，我们还可以生成新的目录，比如例子目录、练习目录等。

## 添加目录项

利用 `\addcontentsline` 命令，我们可以把“参考文献”和“索引”也添加到目录中：

```
\addcontentsline{toc}{chapter}{\bibname}  
\begin{thebibliography}{123456}  
.....  
\end{thebibliography}
```

```
\addcontentsline{toc}{chapter}{\indexname}  
\printindex
```

但参考文献和索引一般都是在新一页开始的，因此这样得到的目录页码是上一页的。在前面加上 `\clearpage` 或者 `\cleardoublepage`，我们可以得到正确的页码。例如：

```
\clearpage
\addcontentsline{toc}{chapter}{\indexname}
\printindex
```

这样页码显示正确，但在 PDF 中点击目录项跳转到的页码还是错误的。可以用 `hyperref` 包的 `\phantomsection` 命令处理此问题：

```
\clearpage
\phantomsection
\addcontentsline{toc}{chapter}{\indexname}
\printindex
```

如果仅要在章节目录添加“目录”，“参考文献”和“索引”这几项，用 `tocbibind` 宏包可能更加方便。



## 13 章节目录

### 13.1 添加目录项

### 13.2 添加书签项

### 13.3 替换书签项

### 13.4 目录页格式

## 添加书签项

利用 `\addcontentsline` 命令添加的目录项，同时也会在 PDF 书签中显示。有时候，例如对于“目录”，我们只希望在 PDF 中添加书签项，而不出现在文档目录中。此时可以用 `hyperref` 宏包提供的 `\pdfbookmark` 命令。例如：

```
\clearpage  
\pdfbookmark{全书目录}{Contents}
```

和添加目录项的做法一样，`\clearpage` 用于保证得到正确的页码。

## 添加书签项

`\pdfbookmark` 命令的完整参数如下：

```
\pdfbookmark[级别]{显示文本}{名称}
```

其中，可选参数“级别”默认为 0，“名称”可以自己选择，但不能互相冲突。

在 `hyperref` 宏包的文档中，也推荐用 `bookmark` 宏包来增添书签项。`bookmark` 宏包的优点在于大多数时候只需要编译一次，而且书签的颜色和字体也可以定制。

## 13 章节目录

### 13.1 添加目录项

### 13.2 添加书签项

### 13.3 替换书签项

### 13.4 目录页格式

## 替换书签项

如果章节名称中包含复杂的数学字符，在 PDF 文档的书签中将无法显示出来。我们看下面的例子：

```
\section{$\Gamma$ 函数}  
\section{$L^2$ 空间}
```

这个例子编译后，其中的  $\Gamma$  将不会在书签中显示，而  $L^2$  将显示为 L2. `hyperref` 宏包提供了 `\texorpdfstring` 命令，用于替换书签的特殊字符。例如：

```
\section{\texorpdfstring{$\Gamma$ 函数}{ $\Gamma$  函数}}  
\section{\texorpdfstring{$L^2$ 空间}{ $L^2$  空间}}
```

这个例子中用文本形式的  $\Gamma$  和  $^2$  分别替换数学中的  $\Gamma$  和上标 2.

## 替换书签项

文本形式的上标数字, `textcomp` 包中只包含 `\textonesuperior` <sup>(1)</sup>、`\texttwosuperior` <sup>(2)</sup> 和 `\textthreesuperior` <sup>(3)</sup> 这三个.

最新的 `hyperref` 和 `xeCJK` 宏包定义了从 0 到 9 的文本形式上下标数字, 并且**部分**定义了文本形式的下标字母. 如 `\textfoursuperior` <sup>(4)</sup>、`\textfiveinferior` <sub>(5)</sub>、`\texteinferior` <sub>(e)</sub> 等等.

如果使用 `xelatex` 编译, 设置合适的字体之后, 直接把这些 Unicode 的文本上下标字符粘贴到 LaTeX 代码中, 也是可行的.

## 13 章节目录

### 13.1 添加目录项

### 13.2 添加书签项

### 13.3 替换书签项

### 13.4 目录页格式

## 目录页格式

要调整章节标题在目录页中的格式，可以用 `titletoc` 宏包。该宏包的用法如下

```
\titlecontents{标题层次}[左间距]{整体格式}  
               {标题序号}{标题内容}{指引线和页码}[下间距]
```

其中“标题层次”参数可以取为 `part`、`chapter`、`section` 等标题名以及 `figure` 和 `table` 浮动图表名。“左间距”指的是“标题内容与页芯左侧的距离。因为“标题序号”一般在“标题内容”的左侧，所以除非标题居中放置或者标题无序号，“左间距”一般需要取大于 0 的值。实际上，该参数总是不可省略的。



## 目录页格式

我们通过例子来看看其它参数的使用. 比如我们要将章 (chapter) 标题用大号粗体居中放置, 同时保留序号, 则可以用下面的命令达到:

```
\titlecontents{chapter}
  [0em]
  {\filcenter\Large\bfseries}
  {\contentslabel{3em}}
  {}
  {}
```

其中“标题序号”参数的 `\contentslabel` 命令指代该部分的序号内容, 如果不写上就没有序号. 而后面的 `3em` 表示“标题序号”左侧和“标题内容”左侧的距离, 这个长度值一般要大于 0, 否则两者就重合了.

## 目录页格式

接下来看看，假如我们需要将节（section）标题去掉序号，在离页芯左侧 4em 处对齐，并且用居中的点（\cdot）来画出指引线，可以用下面的命令达到：

```
\titlecontents{section}
  [4em]
  {}
  {}
  {}
  {\titlerule*[1em]{${\cdot$}}\contentspage}
```

其中 \titlerule\*[1em]{\${\cdot\$}} 命令用于画指引线，1em 表明指引线的各个点的距离。而 \contentspage 命令表示页码。

## 目录页格式

如果用 ctex 宏包来撰写中文文档，需要先载入 titletoc 宏包在载入 ctex 宏包，否则中文设置将会覆盖掉。正确的例子如下：

```
\documentclass{article}
\usepackage{titletoc}
\usepackage{ctexcap}
\begin{document}
...
\end{document}
```

12 标题摘要

13 章节目录

14 参考文献

15 全文索引

## 参考文献

在 LaTeX 中使用参考文献很容易，例如

```
\begin{thebibliography}{123456}  
\bibitem[Kn1]{DK1} D. Knuth, T.A.O.C.P. , Vol. 1,  
Addison-Wesley, 1997.  
\bibitem[Kn2]{DK2} D. Knuth, T.A.O.C.P. , Vol. 2,  
Addison-Wesley, 1997.  
\bibitem[Kn3]{DK3} D. Knuth, T.A.O.C.P. , Vol. 3,  
Addison-Wesley, 1998.  
\end{thebibliography}
```

## 参考文献

上一页的例子所得例子如下：


### 参考文献


- [Kn1] D. Knuth, T.A.O.C.P. , Vol. 1, Addison-Wesley, 1997.
- [Kn2] D. Knuth, T.A.O.C.P. , Vol. 2, Addison-Wesley, 1997.
- [Kn3] D. Knuth, T.A.O.C.P. , Vol. 3, Addison-Wesley, 1998.


其中方括号里的可选参数是结果中显示的各条文献记号，如果省略则使用数字编号。它后面大括号里的必选参数是引用的名称，要引用前两条文献，在前面可以这样使用：

```
D. Knuth wrote some books, e.g. \cite{DK1, DK2}.
```

在 `\begin{thebibliography}` 后面的必选参数指明各文献记号的最大长度，即它的字符长度应该等于各文献记号的最大长度。

12 标题摘要

13 章节目录

14 参考文献

15 全文索引

利用 makeindex 宏包可以建立全文索引。最简单的例子如下：

```
\documentclass{article}
\usepackage{makeidx}
\makeindex
\begin{document}
...
Plain TeX \index{Plain TeX}
...
LaTeX \index{LaTeX}
...
ConTeXt \index{ConTeXt}
...
\printindex
\end{document}
```

此时编译三次才能得到带全文索引的文档：先用 xelatex/pdflatex 编译，再用 makeindex 编译，最后再用 xelatex/pdflatex 编译。



LaTeX 文档排版教程 

2018 年 10 月 15 日

---


## 第四部分 · 学习 LaTeX 版面设计


吕荐瑞  [lvjr.bitbucket.io](https://lvjr.bitbucket.io)


暨南大学数学系



16 文档选项

17 版面设置

18 页眉页脚

19 多栏排版

## 文档类别

在撰写 LaTeX 文档时，通常需要在最前面用 `\documentclass` 命令指定文档类。

```
\documentclass[选项]{文档类}
```

标准文档类有 `article`（文章），`report`（报告），`book`（书籍），`letter`（书信）这几种。从它们的名称可以看出各自的用途；由于用途不同，不同文档类默认的页面布局以及所提供的命令也有一些差别。

文档类的选项涉及这些方面：页面大小，字号大小，纸张方向，草稿定稿，单面双面，新章开始，等等。接下来我们逐一介绍这些选项。

## 文档选项之一

页面大小选项确定了页面的宽度和长度，它的取值有如下这些：

选项	毫米大小	英寸大小
a4paper	210mm × 297mm	8.27in × 11.7in
a5paper	148mm × 210mm	5.83in × 8.27in
b5paper	176mm × 250mm	6.93in × 9.84in
letterpaper	216mm × 279mm	8.5in × 11in
legalpaper	216mm × 356mm	8.5in × 14in
executivepaper	184mm × 267mm	7.25in × 10.5in

默认值为美国常用的 letterpaper 页面。而国内常用的是 a4paper 页面，尺寸稍异。

要设定页面边距的大小，可以用 geometry 宏包（见 第 153 页）。

## 文档选项之二

字号大小选项确定了正文字号，它的取值有如下这些：

选项	正文字号	选项	正文字号
8pt	8pt	12pt	12pt
9pt	9pt	14pt	14.4pt
10pt	10pt	17pt	17.28pt
11pt	10.95pt	20pt	20.74pt

其中 10pt, 11pt, 12pt 三种字号是 LaTeX 的标准字号，其他字号大小是由 `extsizes` 包提供的。要使用这些额外的字号，你需要安装 `extsizes` 包，但无需在导言区载入它。

注意有些字号不是整数，从 10pt 开始，不同字号之间是以 1.2 或  $\sqrt{1.2}$  的比例变化的。


## 文档选项之三

类别	选项	说明
纸张方向	landscape	横向, 若不指定此选项则默认为纵向排版
草稿定稿	draft	草稿, 页面内容有溢出时会显示粗黑条
	final	定稿, 默认值
单面双面	oneside	单面, 页码总在右侧, article 和 report 默认值
	twoside	双面, 奇数页码在右侧, 偶数页码在左侧, book 默认值
新章开始	openright	每章都从奇数页开始, book 默认值
	openany	每章可从任意页开始, article 和 report 默认值


很奇怪的是, LaTeX 没有提供与 landscape (横向) 相对应的 portrait (纵向) 选项.


## 文档选项之四

类别	选项	说明
标题格式	titlepage	标题单独成页, book 和 report 默认值
	notitlepage	标题不单独成页, article 默认值
分栏设置	onecolumn	单栏排版, 默认值
	twocolumn	双栏排版
公式编号	leqno	公式编号在左边, 默认在右边
公式位置	fleqn	公式固定缩进左对齐, 默认为居中对齐
参考文献	openbib	每条文献从第二行起开始缩进, 默认为不缩进

 16 文档选项

 17 版面设置

 18 页眉页脚

 19 多栏排版



## 页面尺寸

利用 `geometry` 宏包，我们可以方便地设置版面。比如设定 B5 页面大小：

```
\usepackage[b5paper]{geometry}
```

也可以直接指定页面的宽度和高度。比如：

```
\usepackage[paperwidth=180mm,paperheight=240mm]  
{geometry}
```

其中页面宽度和高度将会记录在 `\paperwidth` 和 `\paperheight` 命令中。

## 页面尺寸

实际上，geometry 也提供了 \geometry 命令来设置版面。比如前面的例子

```
\usepackage[paperwidth=180mm,paperheight=240mm]  
{geometry}
```

等价于下面的写法：

```
\usepackage{geometry}  
\geometry{paperwidth=180mm,paperheight=240mm}
```

本文档后面的例子都只用 \geometry 命令来设置版面。

## 版心尺寸

接着看如何设置版心的大小，即正文区域的宽度和高度。例如：

```
\geometry{b5paper,text={125mm,195mm},centering}
```

其中的 `text={width,height}` 选项指明了版心的宽度和高度，而后面的 `centering` 选项表示将正文区域自动居中（即上下边距相等，而且左右边距也相等）。上面的代码也可以有另一种写法：

```
\geometry{b5paper,textwidth=125mm,  
           textheight=195mm,centering}
```

其中版心宽度和高度分别记录在 `\textwidth` 和 `\textheight` 命令中。

## 边距尺寸

也可直接设定边距尺寸，让 geometry 包自动计算版心的尺寸。比如

```
\geometry{b5paper,margin=20mm}
```

上述例子设定四周边距均为 20mm。也可以设定不同的边距，比如：

```
\geometry{b5paper,left=20mm,right=25mm,  
          top=22mm,bottom=22mm}
```

若在载入 geometry 宏包时加上 showframe 选项，将会在文档的第一页画出版面布局：

```
\usepackage[showframe]{geometry}
```

这样可以更直观地观察版面，并作相应的调整。

## 页眉页脚边注的尺寸


设定页眉，页脚以及边注的尺寸也类似。比如


```
\geometry{b5paper,margin=25mm,footskip=10mm,  
          headheight=7mm,headsep=5mm,  
          marginparwidth=16mm,marginparsep=4mm}
```

其中


- ▶ footskip 表示页脚底线与版心的距离
- ▶ headheight 表示页眉的高度, headsep 表示页眉底线与版心的距离
- ▶ marginparwidth 表示边注的宽度, marginparsep 表示边注左侧与版心的距离

这些名称对应的几个命令 `\headheight`, `\headsep`, `\footskip`, `\marginparwidth`, `\marginparsep`, 分别表示这些尺寸的大小。

16 文档选项

17 版面设置

18 页眉页脚

19 多栏排版

## 页眉页脚

页眉页脚的样式，在 LaTeX 中称为页面样式（page style）。LaTeX 中默认提供了四种页面样式：

**empty** 没有页眉和页脚

**plain** 没有页眉，页脚是居中的页码，article 和 report 的默认样式

**headings** 没有页脚，页眉两侧分别是章节名称和页码，book 的默认样式

**myheadings** 没有页脚，页眉的外侧是页码，内侧可由用户定义

要切换页面样式，可以用 `\pagestyle` 命令，比如

```
\pagestyle{headings}
```

也可以用 `\thispagestyle` 修改当前页的样式（不影响其他页面）：

```
\thispagestyle{empty}
```

## 页眉页脚

LaTeX 的页面样式是基于 TeX 提供的 `\mark` 命令实现的，要定制它得使用很难理解的 `\leftmark` 和 `\rightmark` 命令。而且可能和其他 mark 冲突。

后来广泛使用的 fancyhdr 宏包虽然简化了页面样式的定制，但仍然无法摆脱上述缺点。

因此这里推荐使用 titleps 宏包来定制页面样式。首先这个宏包将上述难以理解的命令封装起来，用户可以方便地定义页眉页脚。其次，这个宏包是基于 eTeX 提供的 `\marks` 命令实现的，不容易与其他宏包冲突。

此 titleps 宏包由 Javier Bezos 开发，且与另两个宏包 titletoc, titlesec 打包在一起，因此只要安装 titlesec 包就可以了。



## 页眉页脚

要使用 `titles` 宏包设置页面样式，首先需要载入此宏包：

```
\usepackage{titles}
```

接下来我们开始添加新页面样式，并设定页眉：

```
\newpagestyle{main}{  
  \sethead[\thepage][\chaptertitle][\thesection] %偶  
           {\thesection}{\sectiontitle}{\thepage} %奇  
}  
\pagestyle{main}
```


其中 `\newpagestyle` 表示添加新页面样式，该命令里面是新样式的设置。`\sethead` 用于设定页眉。它的前三个可选参数设定偶数页的左、中、右三部分内容，后三个必选参数设定奇数页的左、中、右三部分内容。若前三个参数省略，则奇偶页样式一样。


## 页眉页脚


类似地，在页面样式定义中可以用 `\setfoot` 设定页脚。另外，也可以用 `\headrule` 和 `\footrule` 命令分别表示添加页眉线和页脚线。比如：

```
\newpagestyle{main}[\small\bfseries]{  
  \sethead[\thepage][\thesection] % even  
    {\sectiontitle}{\thepage} % odd  
  \setfoot{}{-\quad\thepage\quad-}{} % odd and even  
  \headrule\footrule  
}  
\pagestyle{main}
```

在这个例子可以看出，`\newpagestyle` 命令的两个必选参数之间有个可选参数，可以用于统一设定页眉和页脚的字体。当然，在 `\sethead` 和 `\setfoot` 命令中也可以单独修改某一部分的字体。

16 文档选项

17 版面设置

18 页眉页脚

19 多栏排版

## 双栏排版

前面已经讲到，在文档类命令中有选项可以设定为双栏排版，比如：

```
\documentclass[twocolumn]{article}
```

实际上还可以在文档中间切换单双栏。比如

```
\twocolumn  
双栏文本  
\onecolumn  
单栏文本
```

需要注意的是，用上述命令切换单双栏后，会自动另起一页。

## 多栏排版

要避免单双栏切换后另起一页,以及使用多栏排版,可以用 `multicol` 宏包. 在载入此宏包后, 用下面的例子就可以实现三栏排版.

```
\begin{multicols}{3}  
\lipsum[4]  
\end{multicols}
```

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse	platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis o-	dio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
--	---	--

其中 `\lipsum` 命令用于生成大段拉丁语文字, 需要先载入 `lipsum` 宏包.

LaTeX 文档排版教程 

2018 年 10 月 15 日


## 第五部分 · 学习 LaTeX 基础知识


吕荐瑞  [lvjr.bitbucket.io](https://lvjr.bitbucket.io)


暨南大学数学系



 20 命令环境

 21 长度间距

 22 编号计数

 23 交叉引用

## 20 命令环境

### 20.1 定义命令

### 20.2 定义环境



## 自定义命令

LaTeX 提供了 `\newcommand` 命令, 用于将常用代码块定义为一个新命令. 比如我们要输入文件路径:

```
D:\textbackslash miktex\textbackslash tex
```

```
D:\miktex\tex
```

这里的 `\textbackslash` 命令太长, 输入不方便, 我们可以用简短的命令代替它:

```
\newcommand{\tbs}{\textbackslash}  
D:\tbs miktex\tbs tex
```

```
D:\miktex\tex
```

## 自定义命令

`\newcommand` 定义的新命令也可以包含参数，比如

```
\newcommand{\myblue}[1]{\textcolor{blue}{#1}}  
\myblue{Some Text}
```

Some Text

其中可选参数里面的 1 表示新命令的参数个数，最多可以有 9 个。这些参数在定义时分别用 #1 到 #9 表示。

还有一个类似的 `\newcommand*` 命令。这个命令与 `\newcommand` 命令的区别是，前者所定义的命令不能包含多个段落（比如里面不能出现空行或者 `\par` 命令）。

## 20 命令环境

### 20.1 定义命令

### 20.2 定义环境

## 自定义环境

利用 `\newenvironment` 命令可以定义一个新环境。比如

```
% define a new environment
\newenvironment{myenv}
  {\hrule\vspace{1em}$\square$\hfill}
  {\hfill$\square$\vspace{1em}\hrule}
% use this new environment
\begin{myenv}Some Text\end{myenv}
```



Some Text



## 自定义环境

同样地，用 `\newenvironment` 定义的新环境也可以有参数。比如

```
% define a new environment
\newenvironment{mycolorenv}[1]
  {\color{#1}$\square$}{$\square$}
% use this new environment
\begin{mycolorenv}{blue}
  Some Text
\end{mycolorenv}
```

□ Some Text □

20 命令环境

21 长度间距

22 编号计数

23 交叉引用

## 21 长度间距

### 21.1 长度单位

### 21.2 弹性长度

### 21.3 长度变量

## 长度单位

在 LaTeX 中会碰到各种长度单位，这里统一介绍如下。

长度单位	换算关系
cm (厘米)	$1\text{cm}=10\text{mm}$
pt (点)	$1\text{pt}=0.351\text{mm}$
bp (大点)	$1\text{bp}=0.353\text{mm}$
pc (pica)	$1\text{pc}=12\text{pt}=4.218\text{mm}$
in (英寸)	$1\text{in}=72.27\text{pt}=72\text{bp}=25.4\text{mm}$
sp (scaled point)	$65536\text{sp}=1\text{pt}$ (TeX 系统最小长度单位)
ex	当前字体中 x 的高度
em	当前字体中 M 的宽度



## 21 长度间距

### 21.1 长度单位

### 21.2 弹性长度

### 21.3 长度变量

## 弹性长度

除了固定长度，TeX 中还有一种弹性长度。行距 `\baselineskip` 实际上就是一个弹性长度。例如我们设置弹性行距如下

```
\setlength{\baselineskip}{12pt plus 2pt minus 1pt}
```

这就表示行距正常值为 12pt，伸长值为 2pt，收缩值为 1pt。实际上，对于弹性长度，TeX 排版时所减少的长度总不会超过其收缩值，而所增加的长度却会按照各伸长值的比例平均分配，从而可能超过其伸长值。

在排版长文档时弹性行距会导致各页行距不一，但在排版单页文档时这样却可以让 TeX 排满整页。

弹性长度的伸长值和收缩值都是可以省略的。如果两者都省略，弹性长度就成为一个固定长度。

## 21 长度间距

### 21.1 长度单位

### 21.2 弹性长度

### 21.3 长度变量

## 修改长度变量

LaTeX 预先定义了一些控制排版的长度变量，我们可用 `\setlength` 命令修改它们：

<code>First \par Second</code>	First
<code>\setlength{\parskip}{12pt}</code>	Second
<code>\par Third</code>	Third

这个例子在中间修改了段间距。另外也可以用 `\addtolength` 命令增加长度变量的值：

<code>First \par Second</code>	First
<code>\addtolength{\parskip}{6pt}</code>	Second
<code>\par Third</code>	Third
<code>\addtolength{\parskip}{6pt}</code>	
<code>\par Fourth</code>	Fourth

## 定义长度变量

用 `\newlength` 命令可以定义一个新的长度变量。比如：

```
\newlength{\mylengtha}  
\setlength{\mylengtha}{10pt}  
\addtolength{\mylengtha}{1cm}  
\the\mylengtha
```

38.45274pt

其中 `\the` 命令用于获取长度变量的值。

## 获取盒子尺寸

要将盒子变量的尺寸保存到长度变量中，我们可以用 `\settowidth`, `\settoheight` 和 `\settodepth` 命令：

```
\newsavebox{\myboxe}  
\sbox{\myboxe}{FancyBox}  
\newlength{\myboxheight}           盒子高度是 9.96498pt  
\settoheight{\myboxheight}{\myboxe}  
盒子高度是 \the\myboxheight
```

这几个命令的参数也可以是一般文本：

```
\newlength{\myboxwidth}  
\settowidth{\myboxwidth}{\fbox{FancyBox}}  
盒子宽度是 \the\myboxwidth  
  
盒子宽度是 49.3249pt
```

20 命令环境

21 长度间距

22 编号计数

23 交叉引用

## 预定义的计数器

LaTeX 的自动编号依赖于各种计数器。在各种预先定义的计数器，常见的有这些：

计数器类别	计数器名
页码计数器	page
章节计数器	part, chapter, section, subsection
公式计数器	equation
图表计数器	figure, table
列表计数器	enumi, enumii, enumiii, enumiv

因为有序列表可以嵌套，所以需要多个计数器记录各层次的编号。



## 计数器的值

节编号依赖于 section 计数器，用 `\arabic{section}` 可显示当前节计数器的数值：

当前段落位于第 `\arabic{section}` 节当前段落位于第 22 节。

实际上，计数器的数值有多种显示方式：

<code>\arabic</code>	<code>\alph</code>	<code>\Alph</code>	<code>\roman</code>	<code>\Roman</code>
1, 2, 3, ...	a, b, c, ...	A, B, C, ...	i, ii, iii, ...	I, II, III, ...

计数器也可以用于整数比较，此时要用 `\value` 命令。比如

```
\ifnum \value{section}=2 Yes \else No \fi No
```

## 计数器的编号样式

在节标题中，节编号的显示方式记录在 `\thesection` 命令中。比如要得到类似 1.3 的节编号样式，可以用如下命令

```
\renewcommand{\thesection}{%  
  \arabic{chapter}.\arabic{section}%  
}
```

每个名称为 `xyz` 计数器都对应一个 `\thexyz` 命令，用于设定编号样式。比如 `page` 计数器对应一个 `\thepage` 命令，`equation` 计数器对应一个 `\theequation` 命令。

## 自定义计数器

用 `\newcounter` 命令可以定义一个新计数器. 新计数器初始值为 0, 但可以修改:

```
\newcounter{mycount}  
\setcounter{mycount}{8}  
\addtocounter{mycount}{-3}  
\stepcounter{mycount}  
计数器当前值为\arabic{mycount}
```

计数器当前值为 6

命令	说明
<code>\setcounter</code>	设定计数器的值
<code>\addtocounter</code>	增加计数器的值, 可以为负数
<code>\stepcounter</code>	计数器的值加 1
<code>\refstepcounter</code>	计数器的值加 1, 并作为之后 <code>\label</code> 命令的值

## 自定义计数器

利用自定义计数器，我们可以定义带编号的环境。比如：


```
\newcounter{mycnt}  
\newenvironment{mytheorem}{%  
  \par\stepcounter{mycnt}Theorem~\arabic{mycnt}\quad  
}{\quad$\square$}  
\begin{mytheorem}  
  Some Text.  
\end{mytheorem}  
\begin{mytheorem}  
  More Text.  
\end{mytheorem}
```


Theorem 1    Some Text.     $\square$

Theorem 2    More Text.     $\square$

若将上面的 `\stepcounter` 改为 `\refstepcounter`，则可在该环境里面使用 `\label` 命令记录计数器编号，并在其它地方用 `\ref` 命令引用此编号。

 20 命令环境

 21 长度间距

 22 编号计数

 23 交叉引用

## 交叉引用

在 LaTeX 中，很多计数器编号都需要在其它地方引用，比如章节编号 chapter、section、subsection，公式编号 equation，浮动图形表格编号 figure、table，以及各种自定义的定理环境的编号。

利用 `\label` 命令，可以将这些编号记录下来，从而在其它地方可以引用该编号。比如

```
\begin{equation}\label{eq:name}
```

```
1+1=2
```

```
\end{equation}
```

前面的公式 (`\ref{eq:name}`) 在第 `\pageref{eq:name}` 页。

$$1 + 1 = 2 \tag{1}$$

前面的公式 (1) 在第 190 页。

其中 `\ref` 引用该公式的编号，而 `\pageref` 引用该公式所在页码。

## 交叉引用

如果该编号是由命令生成的，一般将 `\label` 放在该命令之后；如果该编号是由环境生成的，一般将 `\label` 放在该环境里面。因此浮动图形表格中的 `\label` 命令要放在 `\caption` 命令后面。

```
\begin{figure}[htbp!]  
  \includegraphics[scale=0.2]%  
    {example-image}  
  \caption{ 图片样例 }  
  \label{fig:name}  
\end{figure}  
图~\ref{fig:name}  
在第~\pageref{fig:name}~页.
```



图 2: 图片样例

图 2 在第 191 页.

由于 `\label` 标签有多种类型，推荐使用类型缩写:内容缩写的命名方式，比如前面两个例子的 `eq:name` 和 `fig:name`。

LaTeX 文档排版教程 

2018 年 10 月 15 日

## 第六部分 • 玩转 LaTeX 数学公式


吕荐瑞  [lvjr.bitbucket.io](https://lvjr.bitbucket.io)


暨南大学数学系






 24 数学模式

 25 数学结构

 26 数学符号

 27 多行公式

 28 精细调整

## 行内公式

数学公式分为行内公式 (inline formula) 和行间公式 (displayed formula) 两种. 行内公式和正文在同一行中显示, 可以用下面三种方式来表示:

From \$ 1+1=2 \$, we have ...

From  $1 + 1 = 2$ , we have ...

From \ ( 1+1>2 \ ), we have...

From  $1 + 1 > 2$ , we have...

From \begin{math}1+1>2\end{math}, we have

From  $1 + 1 > 2$ , we have

## 行间公式

而行间公式在单独一行居中显示，可以用如下三种不同的方法表示：

First way

```
$$ x + y > z, $$
```

Second way

```
\[ x + y > z, \]
```

Third way

```
\begin{displaymath}  
x + y > z,  
\end{displaymath}
```

First way

$$x + y > z,$$

Second way

$$x + y > z,$$

Third way

$$x + y > z,$$

## 两种公式

对于行内公式，我们常用简短的  $\dots$  形式。而对于行间公式，我们可以使用  $\backslash[\dots\backslash]$  形式或者  $$$\dots$$$  形式。

用  $$$\dots$$$  形式有一个特有的好处，就是可以用  $\backslasheqno$  命令指定公式的编号，比如

$$$ x + y \leq z \backslasheqno{(1)} $$$

$$x + y \leq z \quad (1)$$

## 编号公式

另外, 我们可以用 `equation` 环境来得到自动编号的行间公式. 例如:

```
\begin{equation}  
x + y \geq z  
\end{equation}
```

$$x + y \geq z \quad (2)$$

## 插入文本

要在数学公式中插入文本，可以利用 `amsmath` 宏包提供的 `\text` 命令。比如：

```
$p=1 \pmod{4} \text{\quad 或 \quad} p=3 \pmod{4}$
```


$p = 1 \pmod{4}$  或  $p = 3 \pmod{4}$

## 数学模式


文本模式用于排版文本段落，而数学模式用于排版数学公式.

当 TeX 进入数学模式时，它的表现和文本模式时有很多区别.


首要区别是在公式里所有空格都会被忽略，TeX 程序会自动给出合适间距.

 24 数学模式

 25 数学结构

 26 数学符号

 27 多行公式

 28 精细调整



## 25 数学结构

### 25.1 上标下标

### 25.2 数学分式

### 25.3 数学根号

### 25.4 数学阵列

## 上标下标

在数学公式里面，用 `_` 来表示下标，而用 `^` 来表示上标（或幂次）：

`$ y_{n+1} = y_n^2 + 1 $`

$$y_{n+1} = y_n^2 + 1$$

要在符号左侧添加上下标，可用 `mathtools` 包提供的 `\prescript` 命令。比如

`$\prescript{a}{b}{W}_c^d$`

$${}_b^a W_c^d$$

## 25 数学结构

### 25.1 上标下标

### 25.2 数学分式

### 25.3 数学根号

### 25.4 数学阵列

## 数学分式

数学分式用 `\frac` 命令给出。比如

```
\frac{1}{2}+\frac{1}{3}=\frac{5}{6}  
$$\frac{1}{2}+\frac{1}{3}=\frac{5}{6}$$
```

$$\frac{1}{2} + \frac{1}{3} = \frac{5}{6}$$

$$\frac{1}{2} + \frac{1}{3} = \frac{5}{6}$$

可以看出，行内公式和行间公式的分式尺寸是不一样的。也可以用 `\tfrac` 和 `\dfrac` 手工选择这两种尺寸的分式。

```
\frac{1}{2+\frac{3}{4}} and  
$$\dfrac{1}{2+\tfrac{3}{4}}$$
```

$$\frac{1}{2+\frac{3}{4}} \text{ and } \frac{1}{2+\frac{3}{4}}$$

## 25 数学结构

### 25.1 上标下标

### 25.2 数学分式

### 25.3 数学根号

### 25.4 数学阵列

## 数学根号

数学根号用 `\sqrt` 命令，默认为二次根号。比如：

`$\sqrt{g}\cdot\sqrt[4]{h}=1$`

$$\sqrt{g}\cdot\sqrt[4]{h}=1$$

要让同一行的根号保持对齐，可以用 `\mathstrut` 命令。比如

`$\sqrt{g\mathstrut}\cdot\sqrt[4]{h\mathstrut}=1$`

$$\sqrt{g}\cdot\sqrt[4]{h}=1$$

## 25 数学结构

### 25.1 上标下标

### 25.2 数学分式

### 25.3 数学根号

### 25.4 数学阵列

## 数学阵列

数学阵列，即矩阵、行列式等有多行多列的数学结构，可用 `amsmath` 包提供的这些环境：

```
\[ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad
\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad
\begin{Bmatrix} a & b \\ c & d \end{Bmatrix} \quad
\begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad
\begin{Vmatrix} a & b \\ c & d \end{Vmatrix} \]
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \begin{Bmatrix} a & b \\ c & d \end{Bmatrix} \quad \begin{vmatrix} a & b \\ c & d \end{vmatrix} \quad \begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$$

其中各列之间用 `\\` 隔开，各行元素之间用 `&` 隔开。



## 数学阵列

用 `amsmath` 提供的阵列环境时，各列元素都是居中对齐的。比如

```
$\begin{pmatrix}  
1 & 1 & -1 \\  
1 & -1 & 1 \\  
-1 & 1 & 1  
\end{pmatrix}$
```

$$\begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix}$$

此类矩阵我们通常喜欢各列元素右对齐。此时可以用更一般的 `array` 环境。此环境用法与前面介绍的 `tabular` 环境类似，也用 `l`, `c`, `r` 分别表示左、中、右对齐。比如

```
$\left(\begin{array}{rrr}  
1 & 1 & -1 \\  
1 & -1 & 1 \\  
-1 & 1 & 1  
\end{array}\right)$
```

$$\left(\begin{array}{rrr} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{array}\right)$$

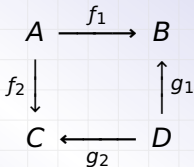
## 绘制交换图

数学公式中的交换图，是一种特殊的数学阵列。方形交换图可用 `amscd` 包绘制，首先需要载入这个宏包：

```
\usepackage{amscd}
```

然后就可以用该宏包提供的 `CD` 命令绘制交换图。比如：

```
\[\begin{CD}
A @>f_1>> B \\
@Vf_2VV @AAg_1A \\
C @<<g_2< D \\
\end{CD}\]
```



其中 `@>>>`，`@<<<`，`@AAA`，`@VVV` 分别表示向左，向右，向上，向下的箭头。

## 绘制交换图

含斜线的交换图可以用 xy-pic 包绘制, 首先需在导言区载入此宏包:

```
\usepackage[all]{xy}
```

这个宏包的原理是先以矩阵方式画出各个元素, 然后画出各个元素之间的箭头. 比如:

```


$$\begin{array}{ccc} X & & Y \\ & \searrow & \\ & & Z \end{array}$$


```

X

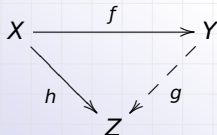
Y

Z

```


$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ & \searrow h & \\ & & Z \end{array}$$


```



## 绘制交换图


注意 xy-pic 交换图既可放在数学环境内部作为公式的一部分，也可作为单独的公式。


```
\xymatrix{
X \ar[rr]^{\text{f}}\ar[dr]_{\text{h}} & & Y \ar@{->}[dl]^{\text{g}}\\
& & Z &
}
```

代码中的 `\ar` 命令 (arrow) 后方括号里的字符串指明了箭头的方向, u、d、l、r 分别指上下左右, 比如 rr 表示指向右边第二个元素, dl 表示指向左下角那个元素。

`\ar` 后的 `@{->}` 表示箭头类型, 还有 `@{=>}`, `@{.>}`, `@{~>}`, `@{-}` 等类型, 不加上这个即使用默认箭头。


`\ar` 后的 `^{\text{f}}` 和 `_{\text{h}}` 表示箭头上的标记, `^` 表示放在箭头前进方向的左侧, `_` 表示放在箭头前进方向的右侧。另外还可用 `|` 表示放在箭头中间。

 24 数学模式

 25 数学结构

 26 数学符号

 27 多行公式

 28 精细调整

## 26 数学符号

### 26.1 各种字母

### 26.2 数学算符

### 26.3 配对括号

## 希腊字母

在数学公式里面，各种希腊字母可以根据它们的英文名对应的命令来得到。例如：

```
\[ \alpha\beta\gamma\delta\epsilon\zeta\eta\theta\iota\kappa\lambda\mu\nu\xi\omicron\pi\rho\sigma\tau\upsilon\phi\chi\psi\omega  
  \lambda\kappa\omega\sigma\zeta  
\]  
\[ \Delta\Gamma\Lambda\Phi\P\Psi\S\Theta\Omega\Upsilon\Xi  
  \Sigma\Theta\Omega\Upsilon\Xi  
\]
```

$\alpha\beta\delta\eta\gamma\iota\kappa\omega\sigma\zeta$   
 $\Delta\Gamma\Lambda\Phi\P\Psi\S\Theta\Omega\Upsilon\Xi$

其中，首字母大写的命令表示大写的希腊字母。

## 数学字母

要输入各种黑板体和花体字母，需要先载入 `amssymb` 宏包。载入后就可使用下面的例子：

`$\mathbb{Z}$`  and  `$\mathcal{O}$`        $\mathbb{Z}$  and  $\mathcal{O}$



## 26 数学符号

### 26.1 各种字母

### 26.2 数学算符

### 26.3 配对括号

## 数学算符

常见的一些数学算符可以看下面的一些例子：

$$2\sin x \cos x = \sin 2x$$

$$2 \sin x \cos x = \sin 2x$$

$$\lim_{n \rightarrow \infty} \frac{1}{n} = 0$$

$$\int \frac{1}{x} dx = \ln |x| + C$$

$$\int \frac{1}{x} dx = \ln |x| + C$$

## 26 数学符号

### 26.1 各种字母

### 26.2 数学算符

### 26.3 配对括号

## 括号大小

在数学公式中，如果直接使用括号，得到的公式会比较丑陋，例如：

```
$$\lim_x(1+\frac{1}{x})^x = \mathrm{e}$$
```

$$\lim_x(1 + \frac{1}{x})^x = e$$

我们可用 `\left` 和 `\right` 命令来得到自动调整大小的括号，例如

```
$$\lim_x\left(1+\frac{1}{x}\right)^x = e$$
```

$$\lim_x\left(1 + \frac{1}{x}\right)^x = e$$


## 括号大小


自动调整大小的括号有时候效果仍然差强人意，这时候你也可以自己指定括号的大小：


```
\[\Bigg< \bigg\{ \Big[ \big(xyz\big) \Big] \bigg\} \Bigg>\]
```

$$\left\langle \left\{ \left[ (xyz) \right] \right\} \right\rangle$$


注意 { 和 } 是特殊字符，需要用 \{ 和 \} 来表示.

 24 数学模式

 25 数学结构

 26 数学符号

 27 多行公式

 28 精细调整

## 多行公式

要方便地输入多行公式，可以使用美国数学会的 `amsmath` 宏包。本节介绍的这些环境在载入了该宏包后才能使用。

首先来看最简单的多行公式环境，即 `gather` 环境。例如：

```
\begin{gather}  
x + y = 5 \\  
2x + 3y = 8  
\end{gather}
```

$$x + y = 5 \quad (3)$$

$$2x + 3y = 8 \quad (4)$$

其中 `\\` 符号用于分割各行。从这个例子可以看出，在 `gather` 环境的多行公式里面，各行之间是不对齐的。

## 多行公式

如果要得到对齐的公式，可以用 align 环境。例如：

```
\begin{align}
x + y &= 5 \quad \backslash\backslash
2x + 3y &= 8 \\
\end{align}
```

$$x + y = 5 \quad (5)$$

$$2x + 3y = 8 \quad (6)$$

其中 & 符号后面的字符就是各行间对齐的位置。



## 多行公式

前面两个环境中的多行公式都是自动编号的. 如果不要编号, 可以用对应的 `gather*` 和 `align*` 环境. 例如:

```
\begin{gather*}  
x + y = 5 \\  
2x + 3y = 8  
\end{gather*}
```

$$\begin{aligned}x + y &= 5 \\ 2x + 3y &= 8\end{aligned}$$

```
\begin{align*}  
x + y &= 5 \\ 2x + 3y &= 8  
\end{align*}
```

$$\begin{aligned}x + y &= 5 \\ 2x + 3y &= 8\end{aligned}$$



## 对齐次环境

最后, amsmath 宏包还提供 gathered 和 aligned 这两个环境. 这两个环境也是次环境, 必须在数学环境中才能使用. 这两者的最大特点在于, 它们不会占用整个宽度, 因此可以作为整体放置在一个复杂公式里面 (这种环境我们称为块环境). 例如:

```
\begin{equation}
```

```
\left.
```

```
\begin{aligned}
```

```
a+b &= 3 \\
```


```
a-b &= 2
```


```
\end{aligned}
```


```
\ \right\} \Rightarrow a^2 - b^2 = 6
```

```
\end{equation}
```

$$\left. \begin{array}{l} a+b=3 \\ a-b=2 \end{array} \right\} \Rightarrow a^2 - b^2 = 6 \quad (8)$$

 24 数学模式

 25 数学结构

 26 数学符号

 27 多行公式

 28 精细调整

## 28 精细调整

### 28.1 公式字号

### 28.2 公式行距

## 公式字号

公式字号跟随正文字号的变化而变化，比如：

`$a+b=c$` and `\Large $a+b=c$`      $a + b = c$  and  $a + b = c$

公式中各部分的尺寸用下面命令来控制（也可以手工设定）：

尺寸命令	尺寸用途
<code>\displaystyle</code>	行间公式
<code>\textstyle</code>	行内公式
<code>\scriptstyle</code>	上下标
<code>\scriptscriptstyle</code>	二级上下标

`$\displaystyle \lim_{n \rightarrow 0} x_n = 0$`      $\lim_{n \rightarrow 0} x_n = 0$

## 28 精细调整

### 28.1 公式字号

### 28.2 公式行距

## 公式行距

在多行公式中，行距默认等于 `\baselineskip` 加上 `\jot` 长度值。  
`\jot` 默认等于 3pt.

在 `array` 环境，及由它定义的环境（例如 `amsmath` 的 `vmatrix` 等）中，行距默认等于 `\baselineskip`. 但通过修改 `\arraystretch` 因子的值，我们可以调整此类公式的行距. 下面例子将其中的行距修改为原来的 1.5 倍.

```
\renewcommand\arraystretch{1.5}
```

实际上，`tabular` 和 `array` 中的行距同样受 `\arraystretch` 因子的影响.



LaTeX 文档排版教程 

2018 年 10 月 15 日


## 第七部分 • 玩转 LaTeX 字体选用


吕荐瑞  [lvjr.bitbucket.io](https://lvjr.bitbucket.io)

暨南大学数学系



 29 设定字体属性

 30 字体文件类型

 31 选用文本字体

## 字体属性

在 LaTeX 中，字体有如下 5 种属性：

1. 字体编码 (font encoding)
2. 字体族名 (font family)
3. 字体系列 (font series)
4. 字体形状 (font shape)
5. 字体尺寸 (font size)

## 29 设定字体属性

### 29.1 字体尺寸

### 29.2 字体形状

### 29.3 字体系列

### 29.4 字体族名

### 29.5 字体编码

## 字体尺寸

在 LaTeX 中可以用各种命令来改变文本字体的大小，它的实际大小和正常字体大小（即 `\normalsize`）有关。而 `\normalsize` 是在文档类 `\documentclass` 中设定的。

字体命令	排版结果	字体命令	排版结果
<code>\tiny</code>	text	<code>\large</code>	text
<code>\scriptsize</code>	text	<code>\Large</code>	text
<code>\footnotesize</code>	text	<code>\LARGE</code>	text
<code>\small</code>	text	<code>\huge</code>	text
<code>\normalsize</code>	text	<code>\Huge</code>	text

当文档类设定字体大小为 10pt, 11pt, 12pt 时, 下面表格列出对应的字号命令的大小.

字体命令	10pt	11pt	12pt
\tiny	5pt	6pt	6pt
\scriptsize	7pt	8pt	8pt
\footnotesize	8pt	9pt	10pt
\small	9pt	10pt	10.95pt
\normalsize	10pt	10.95pt	12pt
\large	12pt	12pt	14.4pt
\Large	14.4pt	14.4pt	17.28pt
\LARGE	17.28pt	17.28pt	20.74pt
\huge	20.74pt	20.74pt	24.88pt
\Huge	24.88pt	24.88pt	24.88pt

注意 \documentclass[11pt]{文档类} 时 \normalsize 只有 10.95pt, 偏小一点.

## 字体尺寸

利用 `\fontsize` 命令，我们也可以指定任意的字号。例如：

```
TEXT  
{\fontsize{20}{24}\selectfont TEXT}  
{\fontsize{40}{48}\selectfont TEXT}  
TEXT
```

TEXT TEXT **TEXT** TEXT

其中 `\fontsize` 命令的第二个参数是基本行距。英文文档行距一般设为字号的 1.2 倍，中文文档的行距一般设为字号的 1.5 倍。改变字号后需要用 `\selectfont` 才能生效。

## 字体尺寸

在 ctex 宏包中还提供了 `\zihao` 命令, 可以选择与 Microsoft Word 相同的字号.

```
\zihao{0} 初号 \zihao{-0} 小初 \zihao{1} 一号 \zihao{-1} 小一  
\zihao{2} 二号 \zihao{-2} 小二 \zihao{3} 三号 \zihao{-3} 小三  
\zihao{4} 四号 \zihao{-4} 小四 \zihao{5} 五号 \zihao{-5} 小五  
\zihao{6} 六号 \zihao{-6} 小六 \zihao{7} 七号 \zihao{8} 八号
```

初号 小初 一号 小一 二  
号 小二 三号 小三 四号 小四 五号 小五 六号 小六 七号 八号



## 29 设定字体属性

### 29.1 字体尺寸

### 29.2 字体形状

### 29.3 字体系列

### 29.4 字体族名

### 29.5 字体编码

## 字体形状

字体形状主要有这些：直立，斜体，伪斜体和小型大写。可用参数命令或声明命令修改：

<code>\textup{Upright Shape} \\</code>	Upright Shape
<code>\textit{Italic Shape} \\</code>	<i>Italic Shape</i>
<code>\textsl{Slanted Shape} \\</code>	<i>Slanted Shape</i>
<code>\textsc{Small Caps Shape} \\</code>	SMALL CAPS SHAPE
<code>\\</code>	
<code>{\upshape Upright Shape} \\</code>	Upright Shape
<code>{\itshape Italic Shape} \\</code>	<i>Italic Shape</i>
<code>{\slshape Slanted Shape} \\</code>	<i>Slanted Shape</i>
<code>{\scshape Small Caps Shape}</code>	SMALL CAPS SHAPE

## 字体形状

字体形状也可以用 `\fontshape` 命令修改（用 `\selectfont` 命令让修改生效），比如：

```
\fontshape{sl}\selectfont Slanted Shape\  
\fontshape{n}\selectfont Upright Shape
```

*Slanted Shape*  
Upright Shape

## 29 设定字体属性

### 29.1 字体尺寸

### 29.2 字体形状

### 29.3 字体系列

### 29.4 字体族名

### 29.5 字体编码

## 字体系列

字体系列根据字体的粗细和宽度来划分。经常用到的如下这两种（正常和粗体）：

```
\textmd{Medium Series} \\
\textbf{Boldface Series} \\
\\
{\mdseries Medium Series} \\
{\bfseries Boldface Series}
```

Medium Series  
**Boldface Series**

Medium Series  
**Boldface Series**

## 字体系列

字体系列也可用 `\fontseries` 命令修改 (用 `\selectfont` 命令让修改生效), 比如:

```
\fontseries{b}\selectfont Boldface \  
\fontseries{m}\selectfont Medium Series
```

**Boldface**

Medium Series

## 29 设定字体属性

### 29.1 字体尺寸

### 29.2 字体形状

### 29.3 字体系列

### 29.4 字体族名

### 29.5 字体编码

## 字体族类

TeX 排版中的所用字体虽然有许多种，但根据的笔画特征通常可把它们分为三大族类：

**罗马字体** 又称衬线字体，字符笔画起始处有装饰；例如 cmr、ecr 和 lmr 字体。

**无衬线字体** 又称等线字体，字符笔画起始处无装饰；例如 cmss、ecss 和 lmss 字体。

**打字机字体** 又称等宽字体，每个字符的宽度都相同；例如 cmtt、ectt 和 lmtt 字体。



## 字体族类

用下面的参数命令或声明命令可以改变字体族类（其中 rm 表示罗马字体族类，sf 表示无衬线字体族类，tt 表示打字机字体族类）：

<code>\textrm{Roman Family} \\</code>	Roman Family
<code>\textsf{Sans Serif Family} \\</code>	Sans Serif Family
<code>\texttt{Typewriter Family} \\</code>	Typewriter Family
<code>\\</code>	
<code>{\rmfamily Roman Family} \\</code>	Roman Family
<code>{\sffamily Sans Serif Family} \\</code>	Sans Serif Family
<code>{\ttfamily Typewriter Family}</code>	Typewriter Family

## 字体族名

至于 rm、sf、tt 这三个字体族类默认对应哪个字体族名，也是可以修改的。例如

```
\renewcommand{\rmdefault}{\lmr}  
\renewcommand{\sfdefault}{\lmss}  
\renewcommand{\ttdefault}{\lmtt}  
\textrm{Roman Family} \\  
\textsf{Sans Serif Family} \\  
\texttt{Typewriter Family}
```

Roman Family  
Sans Serif Family  
Typewriter Family

## 字体族名

也可用 `\fontfamily` 命令直接选用某个字体族（用 `\selectfont` 命令让修改生效）：

```
\fontfamily{cmtt}\selectfont Typewriter  
\fontfamily{fvm}\selectfont Typewriter
```

Typewriter Typewriter

## 29 设定字体属性

### 29.1 字体尺寸

### 29.2 字体形状

### 29.3 字体系列

### 29.4 字体族名

### 29.5 字体编码


## 字体编码

字体编码确定了字体中包含哪些字符以及这些字符的编号。在 LaTeX 中已经预先定义了一些字体编码，比如


编码名称	编码描述	包含字符
OT1	高德纳原来的计算机现代文本字体的编码	128 个
T1	扩展的文本字体编码	256 个
OML	计算机现代数学斜体字体的编码	128 个
OMS	计算机现代数学符号字体的编码	128 个
OMX	计算机现代数学大型符号字体的编码	128 个

字体编码可用 `\fontencoding` 命令修改（用 `\selectfont` 命令让修改生效），比如：

```
\fontencoding{T1}\selectfont
```

 29 设定字体属性

 30 字体文件类型

 31 选用文本字体

## 字体类型之一

在 TeX 安装目录的 fonts 子目录里，可以看到各种字体文件。这是多年的历史形成的。

20 世纪 70 年代，高德纳开发 TeX 软件时也设计了一套 TeX 字体，即计算机现代字体（Computer Modern Fonts）。这套字体在 LaTeX 中的族名为 cmr, cmss, cmtt 等。

计算机现代字体是 METAFONT 格式的，文件名后缀是 mf. 比如 10pt 计算机现代罗马字体（cmr）的 METAFONT 文件是

```
fonts/source/public/cm/cmr10.mf
```

## 字体类型之一

METAFONT 字体中只描述了字符的轮廓曲线，而字符的尺寸信息记录在字体度量文件（TeX Font Metrics）中。字体度量文件的后缀是 tfm。比如上面字体对应的 tfm 文件是

```
fonts/tfm/public/cm/cmr10.tfm
```

在编译 tex 文件为 PDF 时，会从这两种字体文件生成 PK 字体文件，再嵌入 PDF 文件。



## 字体类型之二

20 世纪 80 年代, Adobe 公司推出了 PostScript 数字排版格式, 同时也设计了一种新的字体类型 Type1. Type1 字体比 METAFONT 字体制作更方便. 后来制作的新 TeX 字体都转为用这种格式.

Type1 字体的文件名后缀是 pfb 或 pfa, 对应的字体度量文件的后缀是 afm. 但是 TeX 程序并不能直接处理 afm 文件, 因此需要将 afm 文件转换为 tfm 文件.

比如对于 10pt 大小的拉丁现代无衬线字体 (lmss), 它的 Type1 格式文件是

```
fonts/type1/public/lm/lmss10.pfb  
fonts/afm/public/lm/lmss10.afm
```

在编译 tex 文件为 PDF 时, 会利用 tfm 文件, 将 Type1 字体直接嵌入 PDF 文件中.

## 字体类型之三


20 世纪 90 年代初，苹果公司和 Adobe 公司推出了 TrueType 字体类型，几年后微软公司和 Adobe 公司推出了字体类型 OpenType.


TrueType 字体的文件名后缀是 ttf 或 ttc，而 OpenType 字体的文件名后缀是 ttf 或 otf. 和之前的字体格式相比，这两种新的字体格式有很多优点，比如

- ▶ 将字符轮廓和字符度量记录在同个文件中，使用更方便.
- ▶ 字体文件中可以包含更多字符，适合作为中文字体文件.

新的 XeTeX 和 LuaTeX 程序可以直接使用这两种字体（包含操作系统所包含的字体），编译时直接嵌入字体文件到 PDF 中.

旧的 PDFTeX 程序要使用这两种字体需要先生成 tfm 文件，并作一定配置. 有了 tfm 文件才能在编译时将这两种字体嵌入 PDF 文件中.

 29 设定字体属性

 30 字体文件类型

 31 选用文本字体

## 31 选用文本字体

### 31.1 传统方式

### 31.2 现代方式

## 定义字体的原始方法

在高德纳的 Plain TeX 中，使用新字体前需要先知道字体的文件名，接着用 `\font` 命令定义新字体，然后才能使用这个字体。例如：

```
\font\newfont=cmr10 at 20pt  
{\newfont hello font!}
```

hello font!

这个例子中，第一行将 `cmr10` 字体缩放到 `20pt` 大小定义为 `newfont` 字体。然后第二行用这个新字体显示 `hello font!` 文本。其中的 `cmr10` 就是之前讲到的 `cmr10.mf` 这个 METAFONT 字体文件。

## 新字体选择框架

随着技术的发展，各种新字体层出不穷，这种原始的定义字体方式就有些落后了。于是，在 20 世纪 90 年代发布的 LaTeX 的 2 $\epsilon$  版本中正式引入了新字体选择方案（NFSS）。

NFSS 是一种方便地选择和使用字体的方案。NFSS 把一个字体的属性分为 5 个：编码、族名、系列、形状和尺寸。这几个属性之前已经介绍过了。

这里我们要介绍如何在 NFSS 中定义新字体。即如何让 TeX 程序从这 5 个属性中找到对应的字体文件，以完成编译。

## 定义字体编码

常见的正文字体的编码有 OT1 和 T1 等，而数学字体的编码有 OML、OMS 和 OMX 等。

每种字体编码都对应一个编码文件，即 `otlenc.def`、`tlenc.def`、`omlenc.def` 等，上述这些编码文件在 TeX 系统的 `tex/latex/base` 目录里可以找到。

每个编码文件里面都包含类似下面的一个命令

```
\DeclareFontEncoding{编码}{文本命令}{数学命令}
```

这个命令定义了一个新编码，并分别指定在文本模式和数学模式中切换到此编码时所要执行的命令。

对绝大多数用户，使用现有的字体编码已经足够，不需要自己定义字体编码。

## 定义字体族名

定义某种编码的字体属性可以用如下命令：

```
\DeclareFontFamily{编码}{族名}{命令序列}  
\DeclareFontShape{编码}{族名}{系列}{形状}{字体定义}{  
命令序列}
```

若当前的字体编码为 T1，下面命令将切换罗马字体为 Palatino 字体

```
\renewcommand{\rmdefault}{ppl}
```

LaTeX 将会去查找 t1ppl.fd 文件，该文件的前面两个命令如下

```
\DeclareFontFamily{T1}{ppl}{}  
\DeclareFontShape{T1}{ppl}{m}{n}{<-> pplr8t}{}  

```

第二个命令表示，ppl 字体族的正常粗细，正常形状的字体，对应到 pplr8t 字体文件。



## 选用字体的传统方式

对于常见的字体，TeX 发行版中都包含了它们的定义文件。因此我们可以利用之前介绍的字体属性命令，直接选用合适的字体。例如：

```
\fontencoding{T1}  
\fontfamily{qpl}  
\fontseries{m}  
\fontshape{sc}  
\selectfont  
TeX Gyre Pagella Font
```

TeX GYRE PAGELLA FONT

实际上 LaTeX 提供了一个简化的命令 `\usefont`，可以一次性设定字体属性：

```
\usefont{T1}{qpl}{m}{sc}  
TeX Gyre Pagella Font
```

TeX GYRE PAGELLA FONT

## 恢复正常字体

对字体的编码、族名、系列和形状这几种属性作了修改之后，可用参数命令 `\textnormal` 或者声明命令 `\normalfont` 来恢复默认字体样式。例如：

```
\textit{\textbf{\textsf{Fancy Text}}}\\  
\textnormal{Normal Text} \\  
\\  
\itshape\bfseries\sffamily Fancy Text\  
\normalfont Normal Text\  

```

***Fancy Text***

Normal Text

***Fancy Text***

Normal Text

## 31 选用文本字体

### 31.1 传统方式

### 31.2 现代方式

## 选择英文字体

新的 XeTeX 和 LuaTeX 排版引擎可以直接使用 TrueType 和 OpenType 矢量字体。再利用 fontspec 宏包，我们可以方便的选用 TeX 发行版或者操作系统的字体。比如

```
\usepackage{fontspec}  
\setmainfont{TeX Gyre Pagella}  
\setsansfont{TeX Gyre Heros}  
\setmonofont{TeX Gyre Cursor}
```

其中 \setmainfont, \setsansfont 和 \setmonofont 命令分别设定文档的罗马字体，无衬线字体和打字机字体。字体名可以是字体族名或者是字体文件名。

旧的 PDFTeX 排版引擎不支持这种现代的选择字体方式。因此它不再是排版文档的首选。

## 定义英文字体

要定义新的字体族，可以用 `\newfontfamily` 命令。比如：

```
\newfontfamily{\courier}{Courier New}  
Text {\courier Courier New Font} Text
```

```
Text Courier New Font Text
```

在这个例子中，我们用 `\courier` 命令指代 Windows 系统中的 Courier New 字体，然后就可以随时切换到此字体。

## 选择中文字体

利用 ctex 宏包（用 XeTeX 编译时它调用了 xeCJK 宏包），可以选择相应的中文字体。

```
\usepackage{fontspec}  
\setCJKmainfont[BoldFont=SimHei,ItalicFont=KaiTi]{  
  SimSun}  
\setCJKsansfont[BoldFont=SimHei]{KaiTi}  
\setCJKmonofont{NSimSun}
```

其中的 BoldFont 和 ItalicFont 选项分别表示在粗体和斜体时用另外字体代替。

这里设定的中文字体只对中文有效，不会影响英文字符。由于中文字体自带的英文字符质量一般不高，所以我们通常分开设置中英文字体。

## 定义中文字体

要定义新的中文字体族，可以用 `\setCJKfamilyfont` 命令。比如：

```
\setCJKfamilyfont{yahei}{微软雅黑}  
\CJKfamily{yahei} 这是微软雅黑字体
```

这是微软雅黑字体

在这个例子中，我们用 `yahei` 字体族指代 Windows 系统中的微软雅黑字体，然后就可以用 `\CJKfamily` 命令随时切换到此字体。

同样地，这里所设定的中文字体只对中文有效，不会影响英文字符。注意这里的中文字体的定义命令和使用方式和英文字体的有些不同。

## 寻找可用字体

在定义和选择字体时，我们需要知道字体名称。可用的字体及其名称可以这样找到：

1. 对于操作系统自带的字体，可以在 Microsoft Word 程序的字体选择框查看。
2. 对于 TeX 发行版自带的字体，可以在下面网页中查看：  
<http://www.tug.dk/FontCatalogue/opentypefonts.html>

动手能力较强者也可以用下面方式找出全部可用字体：

1. 按 Win+R 组合键，输入 cmd 并回车打开 Windows 命令控制台
2. 在命令控制台中输入 `fc-list > d:\fontlist.txt` 并回车
3. 打开 d:\fontlist.txt 文件，里面就列出了全部可用的字体



LaTeX 文档排版教程 [↗](#)

2018 年 10 月 15 日


## 第八部分 • 玩转 LaTeX 图形绘制


吕荐瑞 [🏠 lvjr.bitbucket.io](https://lvjr.bitbucket.io)

暨南大学数学系




 32 开始使用

 33 坐标表示

 34 绘制图形

 35 添加节点

 36 数学运算

## 简要介绍

PGF (Portable Graphics Format) 是 Till Tantau 为 LaTeX 设计的一种绘图语言. 而 TikZ 是这种语言的一种前端.

用 PGF 指令绘图会比较繁琐, 因此大多数用户都直接使用 TikZ 指令来绘制图形.

要绘制 TikZ 图形, 首先需要在导言区中载入 tikz 宏包:

```
\usepackage{tikz}
```

本文档主要介绍 PGF/TikZ 2.10 和 3.0 版本.

## 第一个例子


我们先来画一些线段。例如下面的例子：

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





一个 TikZ 画图环境可以包含多个绘图语句，但每个语句必须以英文分号结束。否则将产生类似下面的错误提示：


```
! Package tikz Error: Giving up on this path. Did  
you forget a semicolon?
```

 32 开始使用

 33 坐标表示

 34 绘制图形

 35 添加节点

 36 数学运算

## 长度单位

再来看看一下之前画线段的例子：

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



TikZ 的绘图命令大多都很直观，我们一看就知道是从坐标 (0, 1) 到 (1, 0) 画个线段。

在 TikZ 中，坐标的默认长度单位为厘米 (cm)，其它长度单位也可以使用，但是需要明确写出，例如：

```
\begin{tikzpicture}  
\draw (0pt,30pt) -- (30pt,0pt);  
\end{tikzpicture}
```



## 相对坐标

我们也可以用相对坐标，例如：

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



也就是说，在坐标前加上 + 号表示相对前一个坐标作偏移。这个例子和本节第一个例子的结果是一样的。再看下一个例子：

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



## 相对坐标

如果相对坐标需要记录下来被其它坐标使用,可以在前边用两个 + 号.  
例如:

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



由于第二个坐标是相对坐标,而且它又需要记录下来被第三个坐标使用,所以要用两个 + 号.

我们可以对比这个例子和上一页第二个例子:不管用 ++(1,-1) 还是用 +(1,-1),第二个坐标都等于 (1,0),但是对于前者第三个坐标等同于 (2,1),对于后者第三个坐标等同于 (1,2).



## 极坐标

当然, TikZ 中也可以用极坐标, 此时角度值和长度值之间用冒号隔开.  
例如:

```
\begin{tikzpicture}  
\draw (90:1) -- (0:1) -- (2,1);  
\end{tikzpicture}
```





可以看到, 直角坐标和极坐标混合使用是没有问题的.


## 坐标运算

若需要对坐标进行运算,可以在导言区用 `\usetikzlibrary{calc}` 命令载入 `calc` 扩展,然后用类似下面的例子:


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




 32 开始使用

 33 坐标表示

 34 绘制图形

 35 添加节点

 36 数学运算

## 34 绘制图形

### 34.1 绘制命令

### 34.2 填充命令

### 34.3 路径命令

## 多个线段

我们已经知道用 `\draw` 命令可以画线段. 实际上可以在一个语句中绘制多个线段. 比如

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

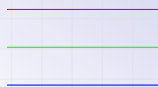


这是前面第一个例子的一种简化写法.

## 线条颜色

要设定 `\draw` 命令所绘制线条的颜色, 可以用 `color` 选项. 与其它 LaTeX 命令一样, TikZ 命令的可选参数也放在方括号中. 例如:

```
\begin{tikzpicture}  
\draw[color=red] (0,2.5) -- (2,2.5);  
\draw[color=green] (0,2) -- (2,2);  
\draw[blue] (0,1.5) -- (2,1.5);  
\end{tikzpicture}
```

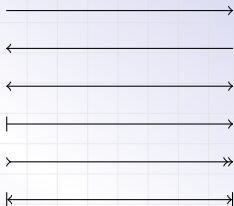


正如上面画蓝线的语句所示, 由于颜色选项经常用到, `color=` 是可以省略的. 实际上, TikZ 将所有不认识的选项视为省略了 `color=` 的颜色选项. 要知道哪些颜色可用, 或者如何定义新颜色, 可以查看 `xcolor` 宏包文档.

## 线段与箭头

类似的我们也可以画带箭头的线段。例如：

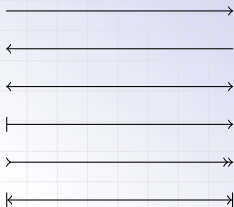
```
\begin{tikzpicture}  
\draw[->] (0,2.5) -- (3,2.5);  
\draw[<-] (0,2) -- (3,2);  
\draw[<->] (0,1.5) -- (3,1.5);  
\draw[|>] (0,1) -- (3,1);  
\draw[>->>] (0,0.5) -- (3,0.5);  
\draw[|<->|] (0,0) -- (3,0);  
\end{tikzpicture}
```



## 线段与箭头

类似的我们也可以画带箭头的线段。例如：

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



和其它 LaTeX 命令一样，TikZ 命令的可选参数也放在方括号中。利用带箭头的线段，画坐标系就毫无问题了。



## 闭合折线

既然我们已经知道怎么画线段，而 TikZ 中多个操作是可以连续指明的，这样画三角形也没问题了。例如：

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



TikZ 默认的线条宽度为 0.4pt。我们可以用 `line width` 选项来加粗这个三角形：

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



## 闭合折线

这时候，问题出来了：我们发现，加粗三角形的左下角有个缺口！这个问题可以用 `cycle` 操作来解决。把前面的例子按如下的方法修改，一切就都正常了：

```
\begin{tikzpicture}[line width=3pt]  
\draw (0,0) -- (1,1) -- (2,0) -- cycle;  
\end{tikzpicture}
```



## 直角折线

再来看怎么画过两点的直角折线：

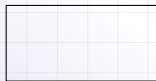
```
\begin{tikzpicture}  
\draw (0,0) |- (1,1);  
\draw (1,0) -| (2,1);  
\end{tikzpicture}
```



其中 `|-` 表示先竖直画线再水平画线，而 `-|` 正好反过来。

将两个直角折线合起来就得到一个矩形。不过我们有更简洁的写法：

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



## 网格线

类似地，我们也可以画网格线。例如：

```
\begin{tikzpicture}  
\draw[step=0.5] (0,0) grid (3,2);  
\end{tikzpicture}
```

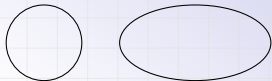


其中的 `step` 参数指明网格的间隙。

## 圆和椭圆

不能总是画最简单的线段，我们接下来来画一种二次曲线：圆和椭圆。  
例如：

```
\begin{tikzpicture}  
\draw (0,0) circle (0.5);  
\draw (2,0) ellipse (1 and 0.5);  
\end{tikzpicture}
```

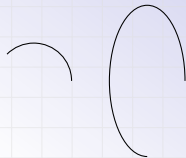


对于圆，我们需要圆心和半径这两个参数，而对于椭圆，我们需要中心，长轴和短轴这三个参数。

## 圆弧与椭圆弧

当然，我们也可以画圆弧和椭圆弧。此时，需要将圆心或中心换成起始点，再指明起始角度和终结角度。例如：

```
\begin{tikzpicture}  
\draw (0.5,0) arc (0: 135: 0.5);  
\draw (2,0) arc (0: 270: 0.5 and 1);  
\end{tikzpicture}
```



注意后面的参数涉及到角度，所以它们之间用冒号分隔。

## 贝塞尔曲线

我们还可以画出贝塞尔 (Bézier) 曲线. 此时需要将 -- 操作改为 .. 操作. 例如:

```
\begin{tikzpicture}  
\draw (0,0) .. controls (1,1) .. (4,0);  
\fill (1,1) circle (1pt);  
\end{tikzpicture}
```

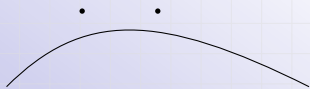


这样画出的是三次贝塞尔曲线, 其中曲线在起点  $(0,0)$  和终点  $(4,0)$  的切线都过点  $(1,1)$ , 因此点  $(1,1)$  称为控制点. 代码中的 `\fill` 是填充命令, 这里用于画一个实心小圆点, 我们后面还会再看到这个命令.

## 贝塞尔曲线

在绘制三次贝塞尔曲线时，也可以提供两个控制点。例如：

```
\begin{tikzpicture}
\draw (0,0) .. controls (1,1) and (2,1) .. (4,0);
\fill (1,1) circle (1pt) (2,1) circle (1pt);
\end{tikzpicture}
```



此时，曲线在起点  $(0,0)$  的切线过第一个控制点  $(1,1)$ ，而在终点  $(4,0)$  的切线过第二个控制点  $(2,1)$ 。



## 贝塞尔曲线

三次贝塞尔曲线的形状可以更加复杂，例如下面的例子：

```
\begin{tikzpicture}  
\draw (0,0) .. controls (1,1) and (2,0) .. (4,1);  
\fill (1,1) circle (1pt) (2,0) circle (1pt);  
\end{tikzpicture}
```



## 连接曲线

在绘制连接两点的曲线时，我们也可指明起始角度和终结角度。例如：

```
\begin{tikzpicture}  
\draw (0,0) to[out=60,in=-90] (4,0);  
\draw (0,0) to (4,0);  
\end{tikzpicture}
```



如果 to 没有任何选项，就和前面的 -- 操作同样画出线段。实际上，to 操作是比较一般的操作，它还有多种选项，这里不再详述。

## 描点折线

在 `\draw` 命令中，我们还可用 `plot` 操作画一般的平面曲线。例如：

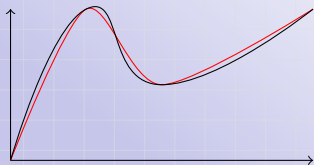
```
\begin{tikzpicture}  
\draw[->] (0,0) -- (4,0);  
\draw[->] (0,0) -- (0,2);  
\draw plot coordinates {(0,0) (1,2) (2,1) (4,2)};  
\end{tikzpicture}
```



## 描点曲线

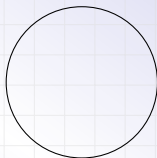
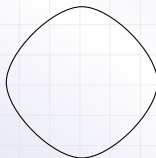
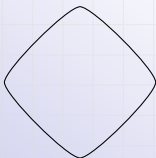
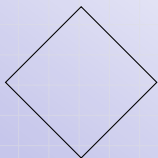
给 plot 操作加上 smooth 选项将用光滑曲线连接各点. 而 tension 选项描述该光滑曲线的绷紧度, 取值范围为从 0 到 1, 默认值为 0.55.

```
\begin{tikzpicture}
\draw[->] (0,0) -- (4,0);
\draw[->] (0,0) -- (0,2);
\draw[color=red] plot[smooth]
    coordinates {(0,0) (1,2) (2,1) (4,2)};
\draw plot[smooth,tension=.9]
    coordinates {(0,0) (1,2) (2,1) (4,2)};
\end{tikzpicture}
```



如果将选项 `smooth` 改为 `smooth cycle`, 将绘制一条闭合曲线:

```
\begin{tikzpicture}[smooth cycle]
\draw plot[tension=0] coordinates {(0,1) (1,0) (2,1) (1,2)};
\draw[xshift=3cm] plot[tension=0.3]
    coordinates{(0,1) (1,0) (2,1) (1,2)};
\draw[xshift=6cm] plot[tension=0.7]
    coordinates{(0,1) (1,0) (2,1) (1,2)};
\draw[xshift=9cm] plot[tension=1]
    coordinates{(0,1) (1,0) (2,1) (1,2)};
\end{tikzpicture}
```

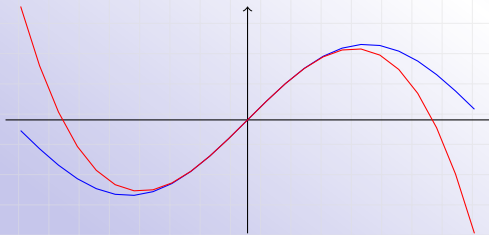


在这个例子可以看到, 对于正方形的四个顶点, `tension=0` 将画出正方形, 而 `tension=1` 将画出圆形.

## 函数曲线

利用 `\draw` 命令的 `plot` 操作，也可以画一般的函数曲线。例如：

```
\begin{tikzpicture}[domain=-3:3]
\draw[->] (-3.2,0) -- (3.2,0) (0,-1.5) -- (0,1.5);
\draw[color=blue] plot (\x,{sin(\x r)});
\draw[color=red] plot (\x,{\x-(1/6)*(\x)^3});
\end{tikzpicture}
```



## 函数曲线

我们仔细观察前面例子中包含函数表达式的这两行：

```
\draw[color=blue] plot (\x,{sin(\x r)});  
\draw[color=red] plot (\x,{\x-(1/6)*(\x)^3});
```

在函数的表达式中，乘号用  $*$  表示，而且不能省略。

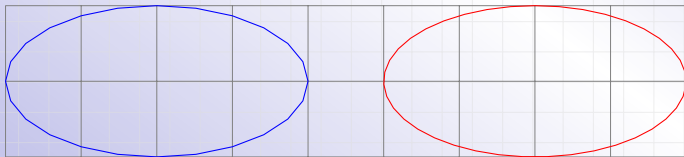
如果表示式不是简单的  $\backslash x$ ，我们一般需要将其中的变量  $\backslash x$  用圆括号括起来，并且将整个表达式用花括号括起来；否则 PGF/TikZ 在解析表达式时多半会出错。

注意代码中的  $\sin$  函数需指明长度类型  $r$ （表示弧度，默认为角度）。在后面内容中，将详细介绍数学函数和数学运算的用法。

## 参数曲线

实际上，我们用 `plot` 是可以画出参数曲线的，其中的 `\x` 就是参数。  
例如：

```
\begin{tikzpicture}[domain=0:2*pi]
\draw[very thin,color=gray] (-2,-1) grid (7,1);
\draw[color=blue] plot ({2*sin(\x r)},{cos(\x r)});
\draw[color=red,xshift=5cm,samples=40]
    plot ({2*sin(\x r)},{cos(\x r)});
\end{tikzpicture}
```



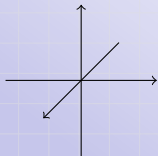
比较图形中的两个椭圆，可以看到第二个比第一个光滑。区别在于取样点数 `samples` 不同。`samples` 选项的默认取值为 25。



## 参数曲线

在 TikZ 中，也可以绘制空间参数曲线，此时我们用三个坐标来表示空间中的点。例如：

```
\begin{tikzpicture}
  [x={(-0.5cm,-0.5cm)},y={(1cm,0cm)},z={(0cm,1cm)}]
  \draw[->] (-1,0,0) -- (1,0,0);
  \draw[->] (0,-1,0) -- (0,1,0);
  \draw[->] (0,0,-1) -- (0,0,1);
\end{tikzpicture}
```



## 参数曲线

在绘制三维图形时，三个坐标轴单位向量的默认值为（注意需要将坐标放入花括号中）

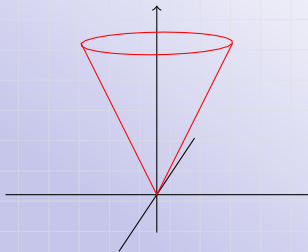
$$x=\{(1\text{cm},0\text{cm})\}, \quad y=\{(0\text{cm},1\text{cm})\}, \quad z=\{(-.385\text{cm},-.385\text{cm})\}$$

在上一页的例子，我们通过重新设定  $x$ ,  $y$  和  $z$  选项，将  $x$  轴指向左下角， $y$  轴指向右边，而  $z$  轴指向上边。

设定了三个坐标轴的指向后，PGF/TikZ 绘制时将根据坐标轴计算空间图形的投影。

在下面例子中，我们画出了空间中的一个圆以及两个线段，以此轮廓图表示一个圆锥面。

```
\begin{tikzpicture}
    [x={(-0.1cm,-0.15cm)},y={(1cm,0cm)},z={(0cm,1cm)}]
\draw[->] (-5,0,0) -- (5,0,0) (0,-2,0) -- (0,2,0)
    (0,0,-0.5) -- (0,0,2.5);
\draw[color=red] plot[domain=0:2*pi]
    ({sin(\x r)},{cos(\x r)},2);
\draw[color=red] (0,0,0) -- (0,1,2) (0,0,0) -- (0,-1,2);
\end{tikzpicture}
```



## 34 绘制图形

### 34.1 绘制命令

### 34.2 填充命令

### 34.3 路径命令

## 单色填充

现在来看填充命令 `\fill`. 先来看如何画实心圆盘和实心矩形. 例如:

```
\begin{tikzpicture}  
\fill (0,0) circle (0.5);  
\fill[blue] (1,-1) rectangle (2,1);  
\end{tikzpicture}
```

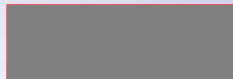


默认的填充颜色是黑色, 当然我们可以自己设定颜色.

## 单色填充

注意 `\draw` 仅绘制区域边界，而 `\fill` 仅填充区域内部，不包括边界。结合两者就可以绘制边界和内部颜色不同的圆盘或者矩形。例如：

```
\begin{tikzpicture}  
\draw[red] (0,0) rectangle (3,1);  
\fill[gray] (0,0) rectangle (3,1);  
\end{tikzpicture}
```



或者，我们可以将 `\draw` 和 `\fill` 合并起来写成 `\filldraw` 命令。此时，这个例子就可以改为（注意我们用 `fill` 和 `draw` 选项分别指定填充和绘制颜色）

```
\begin{tikzpicture}  
\filldraw[fill=gray,draw=red] (0,0) rectangle (3,1);  
\end{tikzpicture}
```

## 单色填充

我们也可以填充自己绘制的封闭曲线，例如：

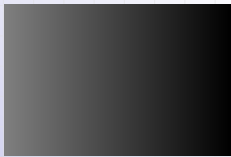
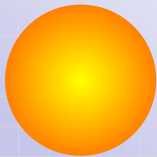
```
\begin{tikzpicture}  
\fill (0,0) -- (1,1) -- (2,0) -- cycle;  
\end{tikzpicture}
```



## 渐变填充

与 `\fill` 命令类似的还有 `\shade` 这个渐变命令。例如：

```
\begin{tikzpicture}  
  \shade[inner color=yellow,outer color=orange]  
    (1,1) circle (1);  
  \shade[left color=gray,right color=black]  
    (3,0) rectangle (6,2);  
\end{tikzpicture}
```





## 34 绘制图形

### 34.1 绘制命令

### 34.2 填充命令

### 34.3 路径命令

## 路径命令

之前介绍的 `\draw`, `\fill`, `\shade` 和 `\filldraw` 命令, 实际上都是带某些选项的 `\path` 命令的简写:

命令	统一表示
<code>\draw</code>	<code>\path[draw]</code>
<code>\fill</code>	<code>\path[fill]</code>
<code>\shade</code>	<code>\path[shade]</code>
<code>\filldraw</code>	<code>\path[fill,draw]</code>

## 路径命令

比如下面四种写法的效果是一样的，它们都画出半径为 1cm 的圆：

```
\draw (0,0) circle (1cm);  
\path [draw] (0,0) circle (1cm);  
\path (0,0) [draw] circle (1cm);  
\path (0,0) circle (1cm) [draw];
```

由于 TikZ 是构造完路径后才绘制或者填充路径的，所以大多数绘图选项都对整个路径有效，不管它们出现在路径的哪个位置。

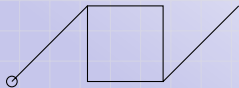
## 路径命令


TikZ 的路径命令，将之前介绍的多种命令统一起来了。路径命令的一般形式如下：


```
\path <specification>;
```


即路径命令后面需要写上路径描述 <specification>，然后以分号结尾。而路径描述可以包含之前我们介绍的各种操作，比如：

```
\begin{tikzpicture}  
\path[draw] (0,0) circle (2pt)-- (1,1)  
             rectangle (2,0) -- (3,1);  
\end{tikzpicture}
```




 32 开始使用

 33 坐标表示

 34 绘制图形

 35 添加节点

 36 数学运算

## 35 添加节点

### 35.1 节点命令

### 35.2 节点位置

### 35.3 连接节点

## 节点命令

如果想在图形上加上文本，就要用到节点命令 `\node` 了。比如绘制直角坐标系：

```
\begin{tikzpicture}
\draw[>-] (0,0) -- (2,0);
\draw[>-] (0,0) -- (0,2);
\node[left] at (0,0) {$0$};
\node[right] at (2,0) {$x$};
\node[below left] at (0,2) {$y$};
\end{tikzpicture}
```



## 节点命令

节点命令的一般形式如下：

```
\node[<options>] (<name>) at (<coordinate>) {<text>};
```

其中

<text> 是节点文本

<coordinate> 是节点坐标

<name> 是节点名称

<options> 是节点选项

在前面例子中，选项 left, right 和 below left 分别表示节点文本相对于节点坐标的几种位置。倘若不指明位置，节点文本的中心将和节点坐标重合。后面将详细介绍节点位置的各种选项。



## 节点操作

节点命令和 `\draw` 等绘制命令实际上是可以写在一起的。例如本节最开始画坐标轴的例子就可以改写如下：

```
\begin{tikzpicture}
\draw[>-] (0,0) node[left]{$0$}
           -- (2,0) node[right]{$x$};
\draw[>-] (0,0) -- (0,2) node[below left]{$y$};
\end{tikzpicture}
```

其中的 `node` 操作的一般形式如下：

```
node[<options>] (<name>) {<text>;}
```

相比 `\node` 命令，`node` 操作不含 `at (<coordinate>)` 部分。

## 节点操作

实际上 `\node` 等同于 `\path node`. 因此下面这两种写法是一样的, 它们都定义了一个节点:

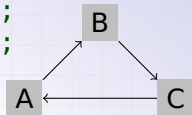
```
\node [fill=gray] (a) at (1,1) {X};  
\path (1,1) node[fill=gray] (a) {X};
```

注意节点并不属于路径的一部分. 实际上, 在路径绘制完毕后, 才会添加节点到该路径.

## 节点名称

至于节点名称的用法，可以先看下面这个例子：

```
\begin{tikzpicture}  
\node[fill=lightgray] (a) at (0,0) {A};  
\node[fill=lightgray] (b) at (1,1) {B};  
\node[fill=lightgray] (c) at (2,0) {C};  
\draw[->] (a) -- (b);  
\draw[->] (b) -- (c);  
\draw[->] (c) -- (a);  
\end{tikzpicture}
```

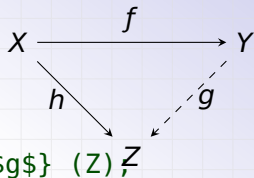


可以看到，知道了节点名称，就可以把它看成坐标那样，作各种线段和曲线的连接。这就是示意图或者流程图的一般绘制方法。

## 画交换图

数学交换图通常是用 `amscd` 包或者 `xy-pic` 包绘制的. 但利用 `TikZ` 的节点命令来绘制会更加自然. 比如:

```
\begin{tikzpicture}[-stealth]
\node (X) at (0,1.5) {$X$};
\node (Y) at (3,1.5) {$Y$};
\node (Z) at (1.5,0) {$Z$};
\draw (X) -- node[above]{$f$} (Y);
\draw (X) -- node[left]{$h$} (Z);
\draw[dashed] (Y) -- node[right]{$g$} (Z);
\end{tikzpicture}
```

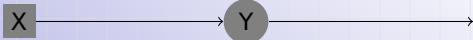


例子中的 `-stealth` 选项设定统一的箭头类型, 也可以在各个 `\draw` 命令中单独设定. 从这个例子我们可以看出, 节点也可以添加到路径的中间. 此时可用 `pos` 指明添加到路径的哪个地方. 默认是 `pos=0.5`, 即添加到路径正中间.

## 节点形状

预先定义好的节点形状有三种：矩形，圆形和点形，但可以自己定义新的形状。节点形状用 `shape` 选项来确定（不指定则默认为矩形），其中的 `shape=` 也可以省略。例如：

```
\begin{tikzpicture}
\node[shape=rectangle,fill=gray] (a) at (0,0) {X};
\node[shape=circle,fill=gray] (b) at (3,0) {Y};
\node[shape=coordinate,fill=gray] (c) at (6,0) {Z};
\draw[->] (a) -- (b);
\draw[->] (b) -- (c);
\end{tikzpicture}
```



可以看到，由于点形节点的面积为零，所以填充颜色和标注文本都没有任何效果。

## 点形节点

这种点形节点还可以用简化的记号来表示. 例如上面例子的点形节点也可以这样表示:

```
\coordinate (c) at (6,0);
```

实际上 `\coordinate` 等同于 `\path coordinate`. 因此上面写法也等同于:

```
\path coordinate (c) at (6,0);
```

## 节点颜色

在 node 操作中可以用 draw, fill 和 text 选项分别指定节点边框, 节点内部和节点文本的颜色. 若不指定颜色, 节点默认使用路径的 draw, fill 和 color 选项的颜色. 比如:

```
\begin{tikzpicture}
\draw[fill=teal] (0,0) -- (1,0)
node[right,draw,fill] {$A$};
\draw[color=teal] (2,0) -- (3,0)
node[right,draw,fill] {$B$};
\draw[color=teal] (4,0) -- (5,0)
node[right,draw=blue,fill=darkgray] {$C$};
\draw[color=teal] (6,0) -- (7,0)
node[right,draw=blue,text=red] {$D$};
\end{tikzpicture}
```



## 35 添加节点

### 35.1 节点命令

### 35.2 节点位置


### 35.3 连接节点



## 节点位置


节点位置可用 `anchor` 选项指定, 不指定时默认为 `anchor=center`, 即居中显示.

```
\begin{tikzpicture}
\draw[fill=blue] (0,0) circle (2pt);
\node[anchor=center] at (0,0) {CENTER};
\end{tikzpicture}
```



也可以指定四个方向, 表示定位节点使得它某个方向与节点坐标重合.

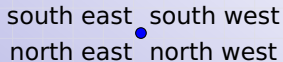
```
\begin{tikzpicture}
\draw[fill=blue] (0,0) circle (2pt);
\node[anchor=east] at (0,0) {east};
\node[anchor=west] at (0,0) {west};
\node[anchor=south] at (0,0) {south};
\node[anchor=north] at (0,0) {north};
\end{tikzpicture}
```



## 节点位置

类似地，也可以指定四个斜向方向，表示定位节点使得它的某个斜角与节点坐标重合。

```
\begin{tikzpicture}
\draw[fill=blue] (0,0) circle (2pt)
  node[anchor=south east] {south east}
  node[anchor=south west] {south west}
  node[anchor=north east] {north east}
  node[anchor=north west] {north west};
\end{tikzpicture}
```



south east south west  
north east north west

注意在一个坐标处是可以同时放置多个节点的。

## 节点位置

位置选项 `anchor` 还有三种取值 `mid`, `mid east` 和 `mid west`, 表示定位节点使得节点文本的中间水平线经过节点坐标. 这里的中间水平线以小写 `x` 字母为标准.

```
\begin{tikzpicture}
\draw (-1,0) -- (10,0);
\draw[fill=blue] (0,0) circle (2pt) node[anchor=mid] {mid x};
\draw[fill=blue] (4,0) circle (2pt) node[anchor=mid east] {mid east x};
\draw[fill=blue] (7,0) circle (2pt) node[anchor=mid west] {mid west x};
\end{tikzpicture}
```



## 节点位置

位置选项 `anchor` 还有三种取值 `base`, `base east` 和 `base west`, 表示定位节点使得节点文本的基线经过节点坐标. 文本的基线通常是小写字母 `abcd` 等的底线.

```
\begin{tikzpicture}
\draw (-1,0) -- (10,0);
\draw[fill=blue] (0,0) circle (2pt)
                 node[anchor=base] {base g};
\draw[fill=blue] (4,0) circle (2pt)
                 node[anchor=base east] {base east g};
\draw[fill=blue] (7,0) circle (2pt)
                 node[anchor=base west] {base west g};
\end{tikzpicture}
```



## 节点位置

节点位置也可以用上下左右 (above, below, left, right) 来表示。这分别表示将节点放在节点坐标的上边、下边、左边、右边，这样表示更加直观，容易理解：

```
\begin{tikzpicture}
\draw[fill=blue] (0,0) circle (2pt);
\node[above] at (0,0) {above};
\node[below] at (0,0) {below};
\node[left]
at (0,0) {left};
\node[right] at (0,0) {right};
\end{tikzpicture}
```



## 节点位置

还有 above left, above right, below left, below right 选项，分别表示将节点放在节点坐标的左上角、右上角、左下角、右下角。

```
\begin{tikzpicture}
\draw[fill=blue] (0,0) circle (2pt)
  node[above left] {above left}
  node[above right] {above right}
  node[below left] {below left}
  node[below right] {below right};
\end{tikzpicture}
```

above left   above right  
below left   below right

## 节点位置

从前面的例子可以看出，节点位置的上下左右选项和东西南北选项是可以一一等同的：

---

above	anchor=south
below	anchor=north
left	anchor=east
right	anchor=west
above left	anchor=south east
above right	anchor=south west
below left	anchor=north east
below right	anchor=north west

---

## 节点位置

用上下左右方式来表示节点位置，除了更加直观外，还有另外一个优点，即可以指定节点的偏移量。例如：

```
\begin{tikzpicture}
\draw[fill=blue] (0,0) circle (2pt);
\node[left=4mm] at (0,0) {left};
\node[below right=8mm] at (0,0) {below right};
\end{tikzpicture}
```

left ●

below right



## 节点位置

如果需要设置更精细的节点位置. 可以使用 positioning 库, 这个库可以这样载入:

```
\usetikzlibrary{positioning}
```

用这个库可以给 above left 等选项分别指定两个方向的偏移量:

```
\begin{tikzpicture}  
\draw[fill=blue] (0,0) circle (2pt);  
\node[below right=8mm and 4mm] at (0,0) {below right};  
\end{tikzpicture}
```



below right

其中, 两个方向的偏移量用 and 连接起来.

## 节点位置

利用 positioning 库，指定偏移量时还可以使用数学表达式：

```
\begin{tikzpicture}  
\draw[fill=blue] (0,0) circle (2pt);  
\node[below=8mm/2+4pt*2] at (0,0) {below right};  
\end{tikzpicture}
```



below right

## 节点位置

如果偏移量不含单位，即表示使用坐标系的默认单位，通常是 cm，即厘米：

```
\begin{tikzpicture}  
\draw[fill=blue] (0,0) circle (2pt);  
\node[below=1.8/2-0.3*2] at (0,0) {below right};  
\end{tikzpicture}
```

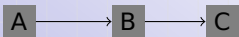


below right

## 节点位置

利用 positioning 库，还可以指定节点的相对位置：

```
\begin{tikzpicture}  
\node[fill=gray] (a) {A};  
\node[fill=gray,right=1cm of a] (b) {B};  
\node[fill=gray,right=0.8 of b] (c) {C};  
\draw[->] (a) -- (b);  
\draw[->] (b) -- (c);  
\end{tikzpicture}
```



同样地，不指明单位时默认单位是厘米。

## 35 添加节点

### 35.1 节点命令

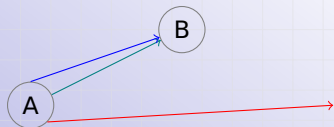
### 35.2 节点位置

### 35.3 连接节点

## 连接节点

在给定节点名称后，可以节点名称连接两个节点，或者节点和某个点.

```
\begin{tikzpicture}
\node[circle,draw=gray] (a) at (0,0) {A};
\node[circle,draw=gray] (b) at (2,1) {B};
\draw[->,teal] (a) -- (b);
\draw[->,blue] (a.north) -- (b);
\draw[->,red] (a.south east) -- (4,0);
\end{tikzpicture}
```

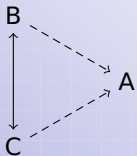


在给定节点名称 a 之后，a.east, a.west, a.south, a.north, a.south east, a.south west, a.north east, a.north west 都可以作为坐标使用.


## 连接节点


连接节点的另一方式是 edge 操作. edge 操作的用法与 to 操作相同. 区别在于 edge 操作所构造的路径将在主路径绘制完后才绘制:


```
\begin{tikzpicture}
\path[densely dashed,->] (0:1) node (a) {A}
  (120:1) node (b) {B} edge (a)
  (240:1) node {C} edge (a) edge [<->,solid] (b);
\end{tikzpicture}
```




不使用 edge 操作, 是没法在一个路径中绘制多个带箭头的线段的.

 32 开始使用

 33 坐标表示

 34 绘制图形

 35 添加节点

 36 数学运算



PGF 的数学引擎支持下面这些数学函数：

abs	cot	hex	neg	rnd
acos	deg	Hex	not	round
add	depth	int	notequal	sec
and	div	ifthenelse	notgreater	sin
array	divide	less	notless	sinh
asin	e	ln	oct	sqrt
atan	equal	log10	or	subtract
atan2	factorial	log2	pi	tan
bin	false	max	pow	tanh
ceil	floor	min	rad	true
cos	frac	mod	rand	veclen
cosec	greater	Mod	random	width
cosh	height	multiply	real	

LaTeX 文档排版教程 [↗](#)

2018 年 10 月 15 日

---


## 第九部分 • 玩转 LaTeX 演示文稿


吕荐瑞 [🏠 lvjr.bitbucket.io](https://lvjr.bitbucket.io)


暨南大学数学系




 37 开始使用

 38 主题选用

 39 局部结构

 40 整体结构

 41 页面结构

## 37 开始使用

### 37.1 简要介绍

### 37.2 基本内容

### 37.3 页面选项

### 37.4 覆盖动画

## 简要介绍

Beamer 是一个用于制作演示幻灯片的 LaTeX 文档类，由德国计算机科学家 Till Tantau 编写。

相对于其它同类工具，Beamer 有如下这些优点：

**功能强大** 各种侧栏、顶栏、底栏，导航栏一应俱全。

**定制灵活** 可以单独改变任何元素的结构，颜色和字体。

**效果多样** 支持各种各样的过渡效果并可以精确控制。

**使用方便** 可以用 pdflatex, xelatex 及 lualatex 编译。

由于这些优点，Beamer 出现之后，很快成为最流行的 LaTeX 演示制作工具。

## 37 开始使用

### 37.1 简要介绍

### 37.2 基本内容

### 37.3 页面选项

### 37.4 覆盖动画

## 英文文档

最简单的 Beamer 英文文档如下：

```
\documentclass{beamer}  
\begin{document}  
  \begin{frame}  
    Hello Beamer!  
  \end{frame}  
\end{document}
```

从这个例子可看出，Beamer 中每帧幻灯片的内容都是放置在一个 frame 环境里面的。

## 中文文档

如果要使用中文，可以用 ctex 宏包，例如：

```
\documentclass{beamer}  
\usepackage[UTF8]{ctex}  
\begin{document}  
  \begin{frame}  
    你好 Beamer!  
  \end{frame}  
\end{document}
```

对于中文文档，建议用 UTF8 编码，然后用 xelatex 程序编译。另外，可以在载入 ctex 宏包时加上 noindent 选项，以取消段落的缩进。



## 页面尺寸

Beamer 幻灯片的页面尺寸默认为 4:3 比例, 128mm 乘以 96mm. 这个尺寸可以在文档类选项中修改 (可用的尺寸选项见下表), 比如要制作宽屏幻灯片可以这样设定:

```
\documentclass[aspectratio=169]{beamer}
```

文档类选项	页面比例	页面尺寸
aspectratio=1610	16:10	160mm x 100mm
aspectratio=169	16:9	160mm x 90mm
aspectratio=149	14:9	140mm x 90mm
aspectratio=141	1.41:1	148.5mm x 105mm
aspectratio=54	5:4	125mm x 100mm
aspectratio=32	3:2	135mm x 90mm

## 左右边距

Beamer 幻灯片的左右边距默认都为 1cm，可用 `\setbeamersize` 命令修改。比如：

```
\setbeamersize{text margin left=8mm}
```

上面例子将左边距改为 8mm。类似地可用 `text margin right` 选项修改右边距。如果有侧栏时，此边距表示正文区与侧栏的距离。

## 字号大小

Beamer 中可使用的全部字号如下: 8pt、9pt、10pt、11pt、12pt、14pt、17pt、20pt, 默认为 11pt. 建议在较大的场合演示时使用大号的字体, 例如:

```
\documentclass[14pt]{beamer}
```

Beamer 中的设置的页面大小比较小, 仅为 128 毫米乘以 96 毫米. 在全屏放映时 PDF 浏览器会自动放大字体, 因此, 同样的大小看起来要比 article 的情形大很多.

## 幻灯片标题

在每帧幻灯片中，可以添加帧标题和帧副标题，例如：

```
\begin{frame}{幻灯片标题}{这是副标题}  
Hello Beamer!  
\end{frame}
```

或者也可以分开来写，如下：

```
\begin{frame}  
\frametitle{幻灯片标题}  
\framesubtitle{这是副标题}  
Hello Beamer!  
\end{frame}
```

## 37 开始使用

### 37.1 简要介绍

### 37.2 基本内容

### 37.3 页面选项

### 37.4 覆盖动画

## 竖直位置

在 Beamer 的每张幻灯片中，正文内容（不包括幻灯片标题）默认都是竖直居中的。这是一种很好的功能，但也许你就喜欢正文竖直居上。没问题，只要在文档类中加上 t 选项就可以了，如下：

```
\documentclass[t]{beamer}
```

如果你只需要让某张幻灯片的正文内容竖直居上、居中或者居下，可以在 frame 环境中分别加上 t、c 或者 b 选项。例如：

```
\begin{frame}[b]  
Hello Beamer from the Bottom!  
\end{frame}
```

## 抄录环境

在 Beamer 演示中使用 `\verb` 抄录命令或者 `verbatim` 抄录环境时，必须在该 frame 中加上 `fragile` 选项，例如：

```
\begin{frame}[fragile]
\frametitle{抄录环境}
这是一段抄录代码：\verb!\frame{hello beamer}!.
\end{frame}
```

这个选项将导致 Beamer 将该 frame 环境的全部内容先写在一个名为 `filename.vrb` 的临时文件里再处理。

## 37 开始使用

### 37.1 简要介绍

### 37.2 基本内容

### 37.3 页面选项

### 37.4 覆盖动画



## 覆盖命令

利用 Beamer 中的覆盖（overlay）命令，我们可以逐步显示幻灯片的内容。常用的覆盖命令有如下这些：

- ▶ `\pause`
- ▶ `\onslide`
- ▶ `\onslide+`
- ▶ `\onslide*`
- ▶ `\uncover`
- ▶ `\visible`
- ▶ `\only`

接下来我们将逐个介绍它们的用法以及区别。

## 覆盖命令之一

最简单的覆盖命令是 \pause. 例如:

一个 \pause 两个 \pause 三个

这些 \pause 命令将把一帧幻灯片分为多页显示.

## 覆盖命令之二

比 `\pause` 高级点的覆盖命令是 `\onslide`. 例如:

`\onslide<1->` 一个 `\onslide<3->` 两个 `\onslide<2->` 三个

这些 `\onslide` 命令同样把一帧幻灯片分为多页逐步显示. 但是用 `\onslide` 我们需要指定在哪几页显示. 例如:

- ▶ `<1->` 表示从第 1 页开始显示;
- ▶ `<2-4>` 表示在第 2 到 4 页显示;
- ▶ `<-5>` 表示在前 5 页显示;
- ▶ `<1,3-5>` 表示在第 1 和第 3 到 5 页显示.

在同一个 frame 中, 最好不要混合使用 `\pause` 和 `\onslide`, 否则覆盖次序可能不是你想要的.

## 覆盖命令之二

如果我们先设置好 `\setbeamercovered` 命令如下：

```
\setbeamercovered{transparent}
```

就可以让 `\pause` 或者 `\onslide` 设定的未显示内容用半透明显示。

## 覆盖命令之二

这里用到的 `\setbeamercovered` 命令，有如下这些选项：

- ▶ `invisible`: 默认值，不显示后面页面的内容
- ▶ `transparent`: 以半透明样式显示后面页面的内容
- ▶ `dynamic`: 同上，但后面页面的内容透明度会逐步变化
- ▶ `highly dynamic`: 同上，但效果更明显

## 覆盖命令之三

与 `\onslide` 类似的还有 `\onslide+` 命令. 这两者的区别在于, `\onslide+` 命令不受 `\setbeamercovered` 命令设定的影响. 用 `\onslide+` 设定的未显示内容, 永远不可能用半透明显示. 例如

`\onslide+<1->` 一个 `\onslide+<3->` 两个 `\onslide+<2->` 三个

## 覆盖命令之四

与 `\onslide` 和 `\onslide+` 类似的还有 `\onslide*` 命令. 用 `\onslide*` 设定的未显示内容, 在幻灯片上不占用空间. 此时

```
\onslide*<1->{壹}\onslide*<3->{贰}\onslide*<2->{叁}
```

### 注意

用 `\onslide*` 命令设定的内容**必须**放在分组括号中. 用 `\onslide` 和 `\onslide+` 设定的内容虽然可以不放在分组括号中, 但建议最好也放在括号中.

## 覆盖命令之五

与 `\onslide`、`\onslide+` 和 `\onslide*` 对应的还有另外三个覆盖命令：`\uncover`、`\visible` 和 `\only`。

1. `\uncover<1-2>\{text\}` 等同于 `\onslide<1-2>\{text\}`
2. `\visible<1-2>\{text\}` 等同于 `\onslide+<1-2>\{text\}`
3. `\only<1-2>\{text\}` 等同于 `\onslide*<1-2>\{text\}`

### 注意

使用覆盖命令 `\uncover`、`\visible` 和 `\only` 时，都**必须**将内容放在分组括号中。



## 取消覆盖

在打印演示文稿时，通常希望将覆盖效果去掉。这时可在文档类中加上 handout 选项。比如

```
\documentclass[handout]{beamer}
```

此外也可将文档类改为 article，然后载入 beamerarticle 宏包，这样 Beamer 演示文稿就变成普通文章（此时 frame 环境，\pause 命令等将不产生任何效果）。比如：

```
\documentclass{article}
\usepackage{beamerarticle}
\begin{document}
\begin{frame}{A frame title}
Normal Text \pause More Text
\end{frame}
\end{document}
```

## 切换动画

Beamer 还支持 PDF 页面的动画切换效果。这些切换动画仅在全屏观看 PDF 时有效。

```
\begin{frame}  
  \transboxout  
  文本内容  
\end{frame}
```

这个例子编译后，当 PDF 阅读器切换到这个页面时，将会从中心到四周以盒状展开内容。

## 切换动画

Beamer 支持的所有 PDF 动画切换效果如下表（第一部分）：


<code>\transblindshorizontal</code>	水平百叶窗
<code>\transblindsvertical</code>	竖直百叶窗
<code>\transboxin</code>	盒状收缩
<code>\transboxout</code>	盒状展开
<code>\transsplithorizontalin</code>	水平收缩
<code>\transsplithorizontalout</code>	水平展开
<code>\transsplitverticalin</code>	竖直收缩
<code>\transsplitverticalout</code>	竖直展开

## 切换动画


Beamer 支持的所有 PDF 动画切换效果如下表（第二部分）：


<code>\transcover</code>	飞入覆盖
<code>\transdissolve</code>	逐渐溶解
<code>\transfade</code>	逐渐显示
<code>\transglitter</code>	溶解擦除
<code>\transpush</code>	推入推出
<code>\transreplace</code>	直接替换
<code>\transwipe</code>	直线擦除
<code>\transduration</code>	定时换页


其中 `\transduration` 的定时换页效果需要指明等待秒数，比如 `\transduration{4}`。

 37 开始使用

 38 主题选用

 39 局部结构

 40 整体结构

 41 页面结构

## 38 主题选用

### 38.1 局部主题

### 38.2 整体主题

### 38.3 主题定制

## 局部主题

Beamer 主题设定了幻灯片的结构，颜色，字体等各方面。每个主题都可分成这四部分：

**外部主题** 可用 `\useoutertheme` 命令设定；

**内部主题** 可用 `\useinnertheme` 命令设定；

**颜色主题** 可用 `\usecolortheme` 命令设定；

**字体主题** 可用 `\usefonttheme` 命令设定。

通过这四种局部主题的选择，将得到一个新的整体主题。

## 局部主题 1 一外部主题

外部主题设定幻灯片是否有顶栏、底栏和侧栏，以及它们的结构，可用下面命令来选择：

```
\useoutertheme{主题名}
```

其中主题名默认为 default，没有顶栏、底栏、侧栏，其他主题名的特点如下：

主题名	顶栏	底栏	侧栏	主题名	顶栏	底栏	侧栏
infolines	单行	有	无	split	单行	有	无
miniframes	多行	可选	无	shadow	单行	有	无
sidebar	无	无	有	tree	多行	无	无
smoothbars	无	无	有	smoothtree	多行	无	无



## 局部主题 2 — 内部主题

内部主题设定幻灯片正文内容（例如标题、列表、定理等）的样式，可用下面命令来选择：

```
\useinnertheme{主题名}
```

其中主题名有如下这些选择：

主题名	描述
default	无序列表为小三角，有序列表为简单数字，默认值
circles	无序列表为小圆盘，有序列表为带圈数字
rectangles	无序列表为小方块，有序列表为方块数字
rounded	无序列表为小圆球，有序列表为球形数字

## 局部主题 3 — 颜色主题

颜色主题设定幻灯片的各部分各结构各元素的配色，可用下面命令来选择：

```
\usecolortheme{主题名}
```

其中主题名有这些选择：

**基本颜色** default、sidebartab、structure；

**完整颜色** albatross（信天翁）、beaver（海狸）、beetle（甲壳虫）、crane（仙鹤）、dove（鸽子）、fly（飞虫）、seagull（海鸥）、wolverine（狼獾）；

**内部颜色** lily（百合）、orchid（兰花）、rose（玫瑰）；

**外部颜色** dolphin（海豚）、seahorse（海马）、whale（鲸鱼）。

## 局部主题 4 — 字体主题

字体主题设定幻灯片的字体，可用下面命令来选择：

```
\usefonttheme{主题名}
```

命令来选择，其中主题名有如下这些选择：

主题名	描述	示例
default	所有文本使用无衬线字体，默认值	Text
serif	所有文本使用罗马字体	Text
structurebold	各种标题和导航栏使用粗体无衬线字体	<b>Title</b>
structureitalicserif	各种标题和导航栏使用斜体罗马字体	<i>Title</i>
structuresmallcapsserif	各种标题和导航栏使用小型大写罗马字体	TITLE

## 局部主题 4 一字体主题

制作投影用的幻灯片时应该选用无衬线字体，这类字体的笔画粗细相同，投影出来容易辨别和阅读。

Beamer 默认无衬线字体（cmss 或者 lmss）笔画太细，而且无衬线数学字体太丑。因此建议在 Beamer 幻灯片中使用 arev 字体。

```
\usefonttheme{professionalfonts}  
\usepackage{arev}
```

其中第一行命令取消 Beamer 对数学字体的改动，在使用专业字体时一般要这样处理。

## 38 主题选用

### 38.1 局部主题





























### 38.2 整体主题

### 38.3 主题定制

## 整体主题

Beamer 的整体主题由外部主题、内部主题、颜色主题和字体主题这四种局部主题组合而成。可用下面命令来选择整体主题（其中可用的主题名及其结构和配色见下面表格）：

```
\usetheme{主题名}
```

 default	 AnnArbor	 Antibes	 Bergen
 Berkeley	 Berlin	 Boadilla	 CambridgeUS
 Copenhagen	 Darmstadt	 Dresden	 EastLansing
 Frankfurt	 Goettingen	 Hannover	 Ilmenau
 JuanLesPins	 Luebeck	 Madrid	 Malmoe
 Marburg	 Montpellier	 PaloAlto	 Pittsburgh
 Rochester	 Singapore	 Szeged	 Warsaw

## 整体主题

Beamer 的整体主题太多了，一个个尝试太费时。你可以在下面这些网址直观地比较这些主题（同时也包含了不同的颜色主题的搭配）：

- ▶ <http://www.hartwork.org/beamer-theme-matrix/>
- ▶ <https://mpetroff.net/files/beamer-theme-matrix/>

Beamer 自带的主题大多非常难看，建议自己仔细定制，或者在互联网上查找新的主题。本文档作者自己定制了一些主题，可以从这里下载它们：

- ▶ <https://github.com/lvjlr/theme>

## 38 主题选用

### 38.1 局部主题

### 38.2 整体主题

### 38.3 主题定制



## 主题定制

Beamer 的各部分的内容都可以自己定制和修改, 和主题的划分类似, 可以从如下这三个方面来定制自己的主题:

定制颜色 可用 `\setbeamercolor` 命令

定制字体 可用 `\setbeamerfont` 命令

定制模板 可用 `\setbeamertemplate` 命令

## 颜色定制

例如，下面的代码修改了正文字体的颜色：

```
\setbeamercolor{normal text}{fg=white,bg=darkgray}
```

其中 fg 和 bg 分别表示前景颜色（即文字颜色）和背景颜色，黑底白字是对比较明显的一种颜色搭配。

另外，用 \usebeamercolor 命令可以使用某元素的颜色。比如：

```
\usebeamercolor[fg]{block title}  
\colorbox{bg}{Block Title}
```

Block Title

其中第一行代码将当前文字颜色设为 block title 的前景颜色，同时将颜色 fg 和 bg 分别指向该元素的前景颜色和背景颜色。因此第二行代码就等同于用 block title 的颜色排版文本了。

## 字体定制

例如，下面的代码修改了脚注文本的字体：

```
\setbeamerfont{footnote}{size=\small,shape=\itshape}
```

其中可用的字体选项如下

字体选项	选项描述
size	设定字体尺寸
shape	设定字体形状
series	设定字体系列
family	设定字体族类
parent	继承其它字体

## 字体定制

下面例子展示了在定制 Beamer 元素的字体时 parent 选项的用法:

```
\setbeamerfont{A}{size=\large}  
\setbeamerfont{B}{series=\bfseries}  
\setbeamerfont{child}{parent={A,B},size=\small}  
\vspace{4em}\normalfont  
Some Normal text.\\[5pt]  
\usebeamerfont{A}  
This text is large.\\[5pt]  
\usebeamerfont{B}  
Large and bold.\\[5pt]  
\usebeamerfont{child}  
Small and bold.
```

Some Normal text.  
This text is large.  
**Large and bold.**  
**Small and bold.**

其中 `\usebeamerfont` 命令表示使用某 Beamer 元素的字体.

## 模板定制

用 `\setbeamertemplate` 命令可定制 Beamer 元素的模板. 比如:

```
\setbeamertemplate{footnote}{%  
  \bfseries\color{blue}%  
  $\Box$;\insertfootnotemark\insertfootnotetext\;$\Box$%  
}
```

正文文本 `\footnote{脚注测试}` 正文文本

正文文本 <sup>a</sup> 正文文本

---

□ <sup>a</sup>脚注测试 □

## 模板定制

为了让模板更容易定制和修改，Beamer 中通常使用格式和内容分离的做法。比如上页的例子可以改写如下：

```
\setbeamercolor{footnote}{fg=blue}  
\setbeamerfont{footnote}{series=\bfseries}  
\setbeamertemplate{footnote}{%  
  $\Box$;\insertfootnotemark\insertfootnotetext\;$\Box$%  
}
```

正文文本 \footnote{脚注测试} 正文文本

正文文本 <sup>a</sup> 正文文本

---

□ <sup>a</sup>脚注测试 □

## 模板定制

将格式和内容分离后. 可用 `\defbeamertemplate` 命令定义多个模板, 随意切换:

```
\defbeamertemplate{footnote}{one}{%
  $\Box$;\insertfootnotemark\insertfootnotetext\;$\Box$}
\defbeamertemplate{footnote}{two}{%
  $\circ$;\insertfootnotemark\insertfootnotetext\;$\circ$}
{\setbeamercolor{footnote}{fg=blue}
 \setbeamertemplate{footnote}[one] 正文\footnote{脚注} 正文}
{\setbeamerfont{footnote}{series=\bfseries}
 \setbeamertemplate{footnote}[two] 正文\footnote{脚注} 正文}
```

正文<sup>a</sup>正文 正文<sup>b</sup>正文

---

□ <sup>a</sup>脚注 □

○ <sup>b</sup>脚注 ○


用 `\setbeamertemplate` 选择已有模板时, 第二个参数用方括号.


## 模板定制

用 `\usebeamertemplate` 命令可以使用某 Beamer 元素的模板。  
比如


```
\setbeamertemplate{my template}{correct}  
...  
Your answer is \usebeamertemplate{my template}.  
... Your answer is correct.
```




 37 开始使用

 38 主题选用

 39 局部结构

 40 整体结构

 41 页面结构

## 39 局部结构

### 39.1 列表环境

### 39.2 区块环境

### 39.3 定理环境

### 39.4 彩色盒子

### 39.5 分栏显示

## 有序列表

在 Beamer 中可以如常使用列表环境，比如有序列表：

```
\begin{enumerate}  
\item 我是第一项  
\item 我是第二项  
\item 我是第三项  
\end{enumerate}
```

1. 我是第一项
2. 我是第二项
3. 我是第三项

## 有序列表

有序列表项的样式可以用下面的代码来设定：

```
\setbeamertemplate{enumerate items}[样式名]
```

其中样式名一共有如下四种选择：

1. default

① circle

① square

① ball

你可以从上面几种样式中任选一种。

## 无序列表

再看看 Beamer 中的无序列表环境，例如：

```
\begin{itemize}  
\item 红色 -- red  
\item 绿色 -- green  
\item 蓝色 -- blue  
\end{itemize}
```

- ▶ 红色 – red
- ▶ 绿色 – green
- ▶ 蓝色 – blue

## 无序列表

无序列表项的样式可以用下面的代码来设定：

```
\setbeamertemplate{itemize items}[样式名]
```

其中样式名一共有如下四种选择（default 和 triangle 一样）：

- ▶ default
- ▶ triangle
- circle
- square
- ball

你可以从上面几种样式中任选一种。

## 描述列表

再看看 Beamer 中的描述列表环境，例如：

```
前面的文本前面的文本前面的文本前面的文本
\begin{description}
\item[红色] 热情、活泼、温暖
\item[绿色] 新鲜、平静、安逸
\item[蓝色] 深远、沉静、寒冷
\end{description}
后面的文本后面的文本后面的文本后面的文本
```

前面的文本前面的文本前面的文本前面的文本

红色 热情、活泼、温暖

绿色 新鲜、平静、安逸

蓝色 深远、沉静、寒冷

后面的文本后面的文本后面的文本后面的文本

## 39 局部结构

### 39.1 列表环境

### 39.2 区块环境

### 39.3 定理环境

### 39.4 彩色盒子

### 39.5 分栏显示



## 区块环境

Beamer 里面定义了一个区块环境，可以用于显示重要的内容。例如：

```
\begin{block}{重要内容}  
2018年5月20日最后截止日期。  
\end{block}
```

重要内容

2018 年 5 月 20 日最后截止日期.

## 提醒环境

与区块环境类似地还有一个提醒环境和例子环境. 先看看提醒环境. 例如:

```
\begin{alertblock}{重要提醒}  
2018年5月20日最后截止日期.  
\end{alertblock}
```

### 重要提醒

2018 年 5 月 20 日最后截止日期.

## 例子环境

再来看看例子环境。例如：

```
\begin{exampleblock}{重要例子}  
2018年5月20日最后截止日期。  
\end{exampleblock}
```

重要例子

2018 年 5 月 20 日最后截止日期.

## 定制区块环境

例如，下面的代码修改了区块环境的样式：

```
\setbeamercolor{block title}{fg=yellow,bg=darkgray}  
\setbeamercolor{block body}{bg=lightgray}  
\setbeamertemplate{blocks}[rounded][shadow=true]
```


### 重要内容

2018 年 5 月 20 日最后截止日期。


其中第一行设定区块环境用圆角带阴影的矩形来表示。


## 39 局部结构

 39.1 列表环境

 39.2 区块环境

 39.3 定理环境

 39.4 彩色盒子

 39.5 分栏显示

## 定理环境

Beamer 中也定义了各种定理环境，而且默认是用区块环境的样式来显示的。例如

```
\begin{theorem}
```

微积分基本公式： $\int_a^b f(x) \mathrm{d}x = F(b) - F(a)$ .

```
\end{theorem}
```

**定理 1** 微积分基本公式： $\int_a^b f(x) \mathrm{d}x = F(b) - F(a)$ .

## 定理环境

各种可用的定理类环境有这些：theorem、corollary、definition、definitions、fact、example 和 examples.

在 Beamer 中定理名默认是英文显示的，如果要改为中文显示，可以在文档开头用类似下面的代码：

```
\documentclass[notheorems]{beamer}
\usepackage[UTF8,noindent]{ctex}
\newtheorem{theorem}{定理}
\newtheorem{example}[theorem]{例子}
\newtheorem*{theorem*}{定理}
\newtheorem*{example*}{例子}
```

其中的 notheorems 选项表示不使用默认的定理类环境.

## 证明环境

Beamer 中也定义了证明环境。例如：

```
\begin{proof}
```

令  $g(x)=e^x-x-1$ 。则当  $x>1$  时，有  $g'(x)=e^x-1>0$ ，  
因此  $g(x)>g(1)=0$ 。即有  $x>1$  时  $e^x>1+x$ 。

```
\end{proof}
```

**证明.** 令  $g(x) = e^x - x - 1$ 。则当  $x > 1$  时，有  $g'(x) = e^x - 1 > 0$ ，因此  $g(x) > g(1) = 0$ 。即有  $x > 1$  时  $e^x > 1 + x$ 。 □



## 证明环境


类似于定理类环境，Beamer 的证明环境中默认也用英文的 “Proof”. 下面的代码


```
\renewcommand{\proofname}{证明}
```

可以将它改为中文的 “证明” 二字.


## 39 局部结构

 39.1 列表环境

 39.2 区块环境

 39.3 定理环境

 39.4 彩色盒子

 39.5 分栏显示

## 彩色盒子

利用 `beamercolorbox` 环境，我们可以生成彩色盒子。比如：

```
\setbeamercolor{myred}{fg=white,bg=red!30!black}  
\begin{beamercolorbox}[wd=\textwidth,ht=2ex,dp=1ex]{myred}  
Hello Color Box!  
\end{beamercolorbox}
```


Hello Color Box!


其中 `red!30!black` 表示用 30% 红色和 70% 黑色混合。盒子选项 `wd`, `ht`, `dp` 分别设定盒子的宽度，高度，深度。


还可用 `left`, `right`, `center` 选项指定盒内文本的三种对齐方式。


Beamer 主题的各种导航栏的彩色盒子，多半是用 `beamercolorbox` 环境绘制的。


## 39 局部结构

 39.1 列表环境

 39.2 区块环境

 39.3 定理环境

 39.4 彩色盒子

 39.5 分栏显示

## 分栏环境

Beamer 中提供了 columns 环境，用于分栏显示。例如：

```
前面\dotfill 文本  
\begin{columns}  
\column{.5\textwidth}  
左栏占一半宽度\par  
左栏占一半宽度  
\column{.5\textwidth}  
右栏占一半宽度\par  
右栏占一半宽度  
\end{columns}  
后面\dotfill 文本
```

```
前面 ..... 文本  
左栏占一半宽度    右栏占一半宽度  
左栏占一半宽度    右栏占一半宽度  
后面 ..... 文本
```

## 分栏对齐


Beamer 的 columns 环境，将会自动在各栏之间留出空隙。


在前面的例子中，各栏总宽度正好等于内容区宽度，加上空隙后就超出内容区宽度。这样，分栏环境就和正文参差不齐。


解决的方法是指定 onlytextwidth 选项。

```
前面\dotfill 文本
\begin{columns}[onlytextwidth]
\column{.5\textwidth}
左栏占一半宽度\par
左栏占一半宽度
\column{.5\textwidth}
右栏占一半宽度\par
右栏占一半宽度
\end{columns}
后面\dotfill 文本
```


```
前面 ..... 文本
左栏占一半宽度 右栏占一半宽度
左栏占一半宽度 右栏占一半宽度
后面 ..... 文本
```

 37 开始使用

 38 主题选用

 39 局部结构

 40 整体结构

 41 页面结构

## 40 整体结构

### 40.1 标题页面

### 40.2 层级结构

### 40.3 目录页面



## 标题页面

在幻灯片中用 `\titlepage` 命令可以生成标题页，一般这是第一张幻灯片。例如：

```
\title{玩转LaTeX幻灯片}  
\author{吕荐瑞}  
\date{2018年5月14日}  
\begin{frame}[plain]  
\titlepage  
\end{frame}
```

其中的 `plain` 选项表示不显示顶栏侧栏底栏等外部元素。

## 标题页面

例如，下面的代码修改了文档标题的字体和颜色：

```
\setbeamerfont{title}{size=\LARGE}  
\setbeamercolor{title}{fg=yellow,bg=gray}
```

其中 fg 和 bg 分别表示文字颜色和背景颜色，某一个不指定就表示用默认颜色。

## 40 整体结构

### 40.1 标题页面

### 40.2 层级结构

### 40.3 目录页面

## 层级结构

在 Beamer 文档中, 可以用 `\part`、`\section`、`\subsection` 等结构命令, 但是不能用 `\chapter`. 例如:

```
\documentclass{beamer}
\begin{document}
\section{One Section}
\begin{frame}First Frame\end{frame}
\begin{frame}Second Frame\end{frame}
\section{The Other Section}
\begin{frame}Third Frame\end{frame}
\end{document}
```

注意这些结构命令**必须**放置在各个 `frame` 环境之间, 放在里面会被覆盖命令重复编号, 也会导致其它编译问题.

## 层级页面

在幻灯片中用 `\sectionpage` 命令可以生成节标题页。例如：

```
\AtBeginSection{  
  \begin{frame}  
    \sectionpage  
  \end{frame}  
}
```

类似地，用 `\subsectionpage` 命令可以生成小节标题页。例如：

```
\AtBeginSubsection{  
  \begin{frame}  
    \subsectionpage  
  \end{frame}  
}
```

## 40 整体结构

### 40.1 标题页面

### 40.2 层级结构

### 40.3 目录页面

## 目录页面

类似于标题页面，我们可在幻灯片中用 `\tableofcontents` 命令生成目录页。例如：

```
\begin{frame}  
\tableofcontents[hideallsubsections]  
\end{frame}
```

其中 `hideallsubsections` 选项表示不显示小节标题。


## 目录页面


例如，下面的代码修改了目录页中节标题的模板和颜色：


```
\setbeamercolor{section in toc}{fg=yellow!80!gray}  
\setbeamertemplate{section in toc}[square]
```


其中第一行将节标题颜色设为 yellow!80!gray（表示 80% 黄色和 20% 灰色混合）。第二行设定在目录中显示方形节标题编号；还有其他选项可以使用：default, circle, ball, ball unnumbered 和 sections numbered.



 37 开始使用

 38 主题选用

 39 局部结构

 40 整体结构

 41 页面结构

## 41 页面结构

### 41.1 页面背景

### 41.2 顶栏底栏

### 41.3 左右侧栏

### 41.4 徽标导航

### 41.5 页面标题

## 页面背景

幻灯片背景由 background canvas 和 background 两部分组成。首先, background canvas 放在最底部,通常是单色背景或者渐变背景。默认是单色背景,其颜色继承了 normal text 的颜色。比如下列代码将背景颜色改为淡灰色:

```
\setbeamercolor{normal text}{bg=lightgray}
```

若要使用渐变背景,可以使用预定义的 vertical shading 模板。比如下面代码设定了蓝黑色渐变背景(其中 \definecolor 定义两种颜色,而最后一行设定渐变背景在底部、中部和顶部的三种颜色):

```
\definecolor{bcolor}{rgb}{0.32,0.3,0.38}  
\definecolor{mcolor}{rgb}{0.08,0.08,0.16}  
\setbeamertemplate{background canvas}%  
    [vertical shading]%  
    [bottom=bcolor,middle=mcolor,top=black]
```

## 页面背景

其次, background 放在 background canvas 元素之前, 且位于其它页面内容之后. 默认的 background 为空, 但可以设定为网格线. 比如:

```
\setbeamertemplate{background}[grid]%  
[step=4mm,color=lightgray]
```

其中 grid 选项指定网格线间隔, 默认为 5mm. color 选项指明网格线颜色, 默认为 10% 前景色.

本文档所用主题的白色网格线和圆角框线背景, 就是在 background 元素中定制的.

## 41 页面结构

### 41.1 页面背景

### 41.2 顶栏底栏

### 41.3 左右侧栏

### 41.4 徽标导航

### 41.5 页面标题

## 顶栏底栏

幻灯片的顶栏和底栏中通常会显示一些信息，用于帮助听众随时了解演示的进度。

Beamer 给顶栏和底栏定义了简单的 text line 模板，可以放置一些文本。比如：

```
\setbeamertemplate{headline}[text line]{  
  \inserttitle\quad\insertsection  
}  
\setbeamertemplate{footline}[text line]{  
  \insertframenumber/\inserttotalframenumber  
}
```

Beamer 还给底栏定义了简单的 page number 和 frame number 模板，分别表示仅在底栏显示页码和帧码。比如

```
\setbeamertemplate{footline}[frame number]
```

## 顶栏底栏

很多外部主题或整体主题会设定带复杂样式的顶底栏，这通常是用 beamercolorbox 环境制作的。我们也可以这样定义，比如：

```
\setbeamercolor{footline}{fg=white,bg=red!30!black}
\setbeamertemplate{footline}{
  \begin{beamercolorbox}%
    [wd=\paperwidth,ht=2.25ex,dp=1ex]{footline}
    \hspace{8mm}
    \insertshorttitle
    \hfill
    \insertshortauthor
    \hfill
    \insertshortdate
    \hspace{8mm}
  \end{beamercolorbox}
}
```

## 41 页面结构

### 41.1 页面背景

### 41.2 顶栏底栏

### 41.3 左右侧栏

### 41.4 徽标导航

### 41.5 页面标题



## 左右侧栏

Beamer 的侧栏可以放在左边或右边，它们分别用 sidebar left 和 sidebar right 元素表示. 但是最简单的添加侧栏的方法是载入 sidebar 外部主题，比如：

```
\useoutertheme[left,width=0.1\paperwidth]{sidebar}
```

此时将在左侧栏显示标题作者以及各节目录，而且当前所在节会高亮显示.

## 41 页面结构

### 41.1 页面背景

### 41.2 顶栏底栏

### 41.3 左右侧栏

### 41.4 徽标导航

### 41.5 页面标题

## 插入徽标

幻灯片的徽标用 logo 元素表示，并可用 \logo 命令定义。比如：

```
\logo{\includegraphics[height=0.5cm]{example-image}}
```

默认徽标是空的，按照上面方式设定后，徽标将被加到各页右下角。

## 导航符号

默认 Beamer 还会在页面右下角添加一系列导航符号，方便演讲者跳转到下一节，下一小节，下一帧，或者返回上页，等等。利用下面的代码可以去掉了这列导航符号：

```
\setbeamertemplate{navigation symbols}{{}}
```

## 41 页面结构

### 41.1 页面背景

### 41.2 顶栏底栏

### 41.3 左右侧栏

### 41.4 徽标导航

### 41.5 页面标题

## 页面标题

幻灯片的帧标题用 frametitle 元素表示，比如下面代码修改了帧标题的样式：

```
\setbeamerfont{frametitle}{size=\large}  
\setbeamercolor{frametitle}{fg=yellow!70!gray}  
\setbeamertemplate{frametitle}{  
    \noindent\insertframetitle\par  
    \noindent\insertframesubtitle\par}
```

其中第一行的设定使得幻灯片的帧标题和正文对齐，这样看起来更加整齐美观。

LaTeX 文档排版教程 [↗](#)

2018 年 10 月 15 日

---


## 第十部分 • 玩转 LaTeX 文档编译


吕荐瑞 [🏠 lvjr.bitbucket.io](https://lvjr.bitbucket.io)


暨南大学数学系



 42 文档拆分

 43 文档合并

 44 辅助文件

 45 编译信息



## 文档拆分

大型文档可以拆成多个文件，然后在主文件中用 `\include` 命令载入主文件。比如

```
% main.tex
\documentclass{article}
\begin{document}
\title{科学论文排版教程}
\author{本文作者}
\maketitle
\include{part-basis.tex}
\include{part-text.tex}
\include{part-math.tex}
\end{document}
```

上面是主文件的代码，而 `part-basis.tex`, `part-text.tex` 和 `part-math.tex` 是三个子文件。使用 `\include` 命令时文件名后缀 `.tex` 可以省略。

## 文档拆分

与 `\include` 命令类似的还有 `\input` 命令. 两者的首要区别是 `\include` 新起一页, 而 `\input` 则不起新页. 因此 `\input` 命令常用来载入导言区的设置文件. 比如

```
\documentclass{article}
\input{style}
\begin{document}
\maketitle
\include{part-basis}
\include{part-text}
\include{part-math}
\end{document}
```

其中 `style.tex` 文件包含了导言区的全部设置. 另外注意 `\include` 不能放在导言区.

## 文档拆分

与 `\include` 命令配套的还有 `\includeonly` 命令，可以用于分段编译，以节省编译时间。比如

```
\documentclass{article}
\includeonly{part-basic,part-math}
\begin{document}
\include{part-basis}
\include{part-text}
\include{part-math}
\end{document}
```

上面例子指定只编译 `part-basic` 和 `part-math` 这两个子文件，因此 `\include{part-text}` 将不会载入子文件 `part-text.tex`。而仅载入其辅助文件 `part-text.aux`，以保证交叉引用仍然能正常显示。


## 文档拆分

利用 docmute 宏包，可让各个子文件也能单独编译。这样比使用 `\includeonly` 更加方便。


```
% main.tex
\documentclass{article}
\input{style}
\usepackage{docmute}
\begin{document}
\include{part-basis}
\include{part-text}
\include{part-math}
\end{document}
```


```
% part-basic.tex
\documentclass{article}
\input{style}
\begin{document}
子文件内容
\end{document}
```

上例左边是主文件内容，其中引入了 docmute 包，保证载入子文件时不会因导言区重复而出错。右边是子文件内容，它是一个完整的文档，因此可以单独编译。

 42 文档拆分

 43 文档合并

 44 辅助文件


 45 编译信息


## 文档合并

与文件放过来的过程是文件合并，此时我们将子文件内容放在主文件开头。因此将文档发给他人时只需提供单个文件，更加方便。比如


```
\begin{filecontents}{style.tex}
子文件内容
\end{filecontents}
\documentclass{article}
\input{style.tex}
\begin{document}
正文区内容
\end{document}
```

在编译时，编译程序将把 filecontents 环境里面的子文件内容写入 style.tex，然后继续编译。注意 filecontents 环境应该放在 \documentclass 命令之前，否则导言区的某些宏包可能会改动某些内部命令，从而导致错误。可以有多个 filecontents 环境。

 42 文档拆分

 43 文档合并

 44 辅助文件


 45 编译信息


## 辅助文件


辅助文件是在编译时自动生成的文件，用于实现各种功能。常见的辅助文件类型如下表：

文件类型	文件用途
aux	生成交叉引用的编号
bbl, blg	生成参考文献
glg, glo, gls	生成术语表
idx, ilg, ind	生成索引
lof	生成插图目录
log	记录编译日志
lot	生成表格目录
out	生成超链接书签
synctex, synctex.gz	用于正向和反向搜索
toc	生成章节目录



 42 文档拆分

 43 文档合并

 44 辅助文件

 45 编译信息

## 编译日志

对于正确的 tex 文档，假如文档名为 hello.tex，编译时显示的信息类似于下面：

```
This is XeTeX, Version 3.1415926-2.2 (MiKTeX 2.9)
entering extended mode
(E:\work\latex\hello.tex
LaTeX2e <2009/09/24>
(D:\CTEX\MiKTeX\tex\latex\base\article.cls
(D:\CTEX\MiKTeX\tex\latex\base\size10.clo))
No file hello.aux.
[1] (E:\work\latex\hello.aux) )
Output written on hello.pdf (1 page).
SyncTeX written on hello.synctex
Transcript written on hello.log.
```

这些信息告诉我们，hello.tex 文档已经编译成功，得到 hello.pdf，同时编译的信息已经写入 hello.log 文件中。

## 45 编译信息

### 45.1 错误信息

### 45.2 警告信息

## 错误信息

如果编译时遇到错误，编译就会暂停，并且显示出错信息等待我们处理。最常见的错误信息如下：

```
! Undefined control sequence.  
l.3 \hello  
      world  
?
```

这个信息表示编译程序遇到了未定义的命令 `\hello`，同时指明这个错误出现在第 3 行。在 `?` 号后面你可以直接按回车键继续编译，或者按 `r` 键让它继续编译且不再显示错误。

对于这种最简单的错误，基本的处理方式如下：首先检查是否这个命令的某个字母写错了；其次检查是否这个命令所依赖的宏包未在导言区载入；如果你确认代码没有问题，有可能是辅助文件的问题，你可以删除 `tex` 文档所在目录的 `.aux`，`.idx`，`.ind`，`.out`，`.toc` 等等文件后重新编译试试。

## 45 编译信息

### 45.1 错误信息

### 45.2 警告信息

## 警告信息

编译时还可能碰到各种警告信息，常见的警告信息类似于下面这些：

```
Overfull \hbox (8.78062pt too wide) in paragraph at  
lines 427--427  
Overfull \vbox (6.56552pt too high) has occurred  
while \output is active []  
Underfull \hbox (badness 10000) in paragraph at  
lines 488--498
```

上面第一行是行溢出警告，即 TeX 程序不能很好的断行，导致超出边界 8.78062pt 的长度。这个警告通常在某行有多个长单词时出现。

第二行是页溢出警告，即 TeX 程序不能很好的分页，导致文本延伸到页面底部。这个警告通常在文档中有大型表格或者图片时出现。

第三行是行稀疏警告，即 TeX 程序在某些行不能放入足够多的单词，导致单词之间空隙太大有碍观瞻。此警告在手工断行时会出现。其中的 badness 值越大表示情况越糟糕，而且超过 10000 后都显示 10000。