Learn LaTeX by Examples

Kanglong Wu

August 25, 2016

目录 | 0

序 1

第一稿序

其实在之前我是有一稿手册的，开始撰写的日期大概在 2015 年 4 月。但是自己觉得写得太烂，因此索性推倒重写了这一版。这一版的主要特征是：

1 - 我希望能够吸引初学者快速上手，解决手头的问题。因此去掉了枯燥的讲解和无穷无尽的宏包用法介绍，直接使用实例；

2 - 力求突出实用性。当然，也会提点一些可以深入学习的内容，读者可以自行查阅，或者阅读本手册中的扩展阅读章节（即带星号 * 的章节）。

3 - 本手册使用的编辑器为 TeXStudio，而非之前的商业软件 WinEdt. 这使得学习 LaTeX 的门槛更低。当然了，你有权使用任何编辑器，我并不是说 TeXStudio 就是最好的。事实上在你熟练以后，使用记事本 + 命令行的方式生成文档都是没有问题的。

由于工作全部由我一人完成，限于视野，难免存在错漏之处。恳请读者指正。有任何使用中遇到的手册中无法解决的问题，欢迎向我提出。

Mail: wklchris@hotmail.com

Chris Wu
August 25, 2016 于湖北

2 LaTeX 基础

2.1 第一份文稿

编辑器的配置大概是需要讲解一下的，毕竟对于初学者来说是很头疼的事情。本手册就以 TEXStudio 为例进行配置。首先你应该安装一个 TEX Live，它是完全免费的，网址：http://tug.org/texlive/.

虽然它体积较大，但是却是最一劳永逸、最不需要花时间取配置的方法，同时它大概也是功能支持最强的 LATEX 发行版。

打开 TEXStudio 后，选择选项 → 设置 TEXStudio → 构建 → 默认编译器，选择 XELATEX. 这主要是基于中文文档编译的考虑，同时 XELATEX 也能很好地编译英文文档。我建议始终使用它作为默认编译器。

之后你可以在编译窗口输入一篇小文档，并保存为 tex 文件进行测试：

```
\documentclass{article}
\usepackage[slantfont,boldfont]{xeCJK}
\setCJKmainfont[BoldFont=SimHei,ItalicFont=KaiTi]{SimSun}
\begin{document}
        Hello, world!
        你好，世界！
\end{document}
```

点击编译按钮生成，F7 查看。生成的 pdf 在你的 tex 文件保存目录中。具体各行的含义我们会在后文介绍。

2.2 认识 LATEX

2.2.1 命令与环境

LATEX 中的**命令**通常是由一个反斜杠加上命令名称，再加上花括号内的参数构成的。比如：

```
\documentclass{article}
```

如果有一些选项是备选的，那么通常会在花括号前用方括号标出。比如：

```
\documentclass[a4paper]{article}
```

在 LATEX 中，还有一种重要的指令叫做**环境**。它定义了一整个区段的内容都受到环境（即其本身）的控制。这个区段是从 `\begin{environment}` 开始，到 `\end{environment}` 结束的。。比如：

```
\begin{document}
        ...内容...
\end{document}
```

同样的，环境也可以使用备选选项，只需要写在 begin 的后面就行了。

注意：不带花括号的命令后面如果想打印空格，请加上一对内部为空的花括号再键入空格。否则空格会被忽略。例如：`\LaTeX{} Studio.`

2.2.2 保留字符

LaTeX 中有许多字符有着特殊的含义，在你生成文档时不会直接打印。例如每个命令的第一个字符：反斜杠。单独输入一个反斜杠在你的行文中不会有任何帮助，甚至可能产生错误。LaTeX 中的保留字符有：

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

以上除了反斜杠外，均能用前加反斜杠的行驶输出。即你只需要键入：

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

唯独反斜杠的输出比较头痛，你可以尝试：

```
$\backslash$
\texttt{\char92}
```

另外需要说明的是，在中文格式下，波浪线~ 用来输出一个小空格，不能够直接输出。你可以通过命令`$\sim$` 来产生一个。

2.2.3 导言区

任何一份 LaTeX 文档都应当包含以下结构：

```
\documentclass[options]{doc-class}
\begin{document}
        ...
\end{document}
```

其中，在语句`\begin{document}`之前的内容称为**导言区**。导言区可以留空，以可以进行一些文档的准备操作。你可以粗浅地理解为：导言区即模板定义。

文档类的参数 doc-class 和可选选项 *options* 有以下取值：

表 2.1: 文档类和选项

doc-class 文档类	
article	- 科学期刊，演示文稿，短报告，邀请函。
proc	- 基于 article 的会议论文集。
report	- 多章节的长报告、博士论文、短篇书。
book	- 书籍。
slides	- 幻灯片，使用了大号 Scans Serif 字体。
options	
字体	- 默认 10pt
页面方向	- 默认竖向 portrait，可选横向 landscape。
纸张尺寸	- 默认 letterpaper，可选用 a4paper, b5paper 等。
分栏	- 默认 onecolumn，还有 twocolumn。
双面打印	- 有 oneside/twoside 两个选项，用于排版奇偶页。article/report 默认单面。
章节分页	- 有 openright/openany 两个选项，决定是在奇数页开启新页或是任意页开启新页。注意 article 是没有 chapter（“章”）命令的，默认任意页。
公式对齐	- 默认居中，可改为左对齐 fleqn；默认编号居右，可改为左对齐 leqno。

在本文中，多数的文档类提及的均为 report/book 类。如果有 article 类将会特别指明。其余的文档类不予说明。本手册排版即使用了 book 类。

在导言区最常见的是**宏包**的加载工作，命令形如：`\usepackage{package}`。通俗地讲，宏包是指一系列已经制作好的功能“模块”，在你需要使用一些原生 LaTeX 不带有的功能时，只需要调用这些宏包就可以了。比如本文的代码就是利用 `listings` 宏包实现的。

宏包的具体使用将参在各部分内容说明中进行讲解。

2.2.4 错误的查找

在编辑器界面上，下方的日志是显示编译过程的地方。在你编译通过后，会出现这样的字样：

- **Errors 错误**：严重的错误。一般地，编译若通过了，该项是零。
- **Warnings 警告**：一些不影响生成文档的瑕疵。
- **Bad Boxes 坏箱**[1]：指排版中出现的长度问题，比如长度超出（Overfull）等。后面的 Badness 表示错误的严重程度，程度越高数值越大。这类问题需要检查，排除 Badness 高的选项。

你可以向上翻阅控制台记录，来找到 Warning 开头的记录，或者 Overfull/Underfull 开头的记录。这些记录会指出你的问题出在哪一行 (比如 line 1-2) 或者在输出 pdf 的哪一页（比如 active [12]。注意，这个 12 表示页面显示页码或者计数器计数页码，而不是文件打印出来的真实页码）。此外你可以还需要了解：

[1]Box 是 LaTeX 中的一个特殊概念，具体将在第4.2节部分进行讲解。

- 值得指出的是，由于 LaTeX 的编译原理（第一次生成 aux 文件，第二次再引用它），目录想要合理显示**需要连续编译两次**。在连续编译两次后，你会发现一些 Warnings 会在第二次编译后消失。在 TeXStudio 中，你可以只单击一次“构建并查看”，它会检测到文章的变化并自动决定是否需要编译两次。
- 对于大型文档，寻找行号十分痛苦。你需要学会合理地拆分 tex 文件，参阅第2.15节的内容。

2.2.5 文件输出

LaTeX 的输出一般推荐 pdf 格式，由 LaTeX 直接生成 dvi 的方法并不推荐。你在 tex 文档的文件夹下可能看到的其他文件类型：

.sty	宏包文件
.aux	用于储存交叉引用信息的文件。因此，在更新交叉引用（公式编号、大纲级别）后，需要编译两次才能正常显示。
.log	记录上次编译的信息。
.toc	目录文件。
.idx	如果文档中包含索引，该文件用于储存索引信息。
.lof	图形目录。
.lot	表格目录。

有时 LaTeX 的编译出现异常，你需要删除文件夹下除了 tex 以外的文件再编译。此外，在某些独占程序打开了以上的文件时（比如用 Acrobat 打开了 pdf），编译可能出现错误。请在编译时确保关闭这些独占程序。

2.3 标点符号

2.3.1 引号

单引号并不使用两个' 符号组合。左单引号是重音符`（键盘上数字 1 左侧），而右单引号是常用的引号符。英文中，左双引号就是连续两个重音符。

英文下的引号嵌套需要借助\thinspace 命令分隔，比如：

```
``\thinspace`Max' is here.''
```

“‘Max’ is here.”

中文下的单引号和双引号你可以用中文输入法直接输入。

2.3.2 破折与短横

英文的短横分为三种：

- 连字符：输入一个短横：-，效果如 daughter-in-law
- 数字起止符：输入两个短横：--，效果如：page 1–2
- 破折号：输入三个短横：---，效果如：Listen—I’m serious.

中文的破折号你也许可以直接使用日常的输入方式。

至于减号，也许你应该借助于数学环境来输入：$-$.

2.3.3 强调：粗与斜

LaTeX 中专门有个叫做`em {text}`的命令，可以强调文本。对于通常的西文文本，上述命令的作用就是斜体。如果你对一段已经这样转换为斜体的文本再使用这个命令，它就会取消斜体，而成为正体。

西文中一般采用上述的斜体强调方式而不是粗体，例如在说明书名的时候就会使用以上命令。关于字体的更多内容参考字体这一节。

2.3.4 下划线与删除线

LaTeX 原生提供了`underline` 命令简直烂的可以，建议你使用 ulem 宏包下的`uline` 命令代替。ulem 宏包还提供了一系列有用的命令：

```
\uline{下划线} \\
\uuline{双下划线} \\
\dashuline{虚下划线} \\
\dotuline{点下划线} \\
\uwave{波浪线} \\
\sout{删除线} \\
\xout{斜删除线}
```

下划线
双下划线
虚下划线
点下划线
波浪线
删除线
斜删除线

需要注意的是，ulem 宏包重定义了`\emph` 命令，使得原来的加斜强调变成了下划线、原来的两次强调就取消强调变成了两次强调就双下划线，同时也支持换行文本。

2.3.5 其他

角度符号或者温度符号需要借助数学模式`$...$` 输入：

```
$30\,^{\circ}$三角形 \\
$37\,^{\circ}\mathrm{C}$
```

30 ° 三角形
37 °C

西文中还有一种注音符号，比如 ô，也常用于拼音声调，参考注音符号部分的附录内容。

2.4 格式控制

首先了解一下 LaTeX 的长度单位：

pt point，磅。
pc pica。1pc=12pt，四号字大小
in inch，英寸。1in=72.27pt
bp bigpoint，大点。1bp=$\frac{1}{72}$in
cm centimeter，厘米。1cm=$\frac{1}{2.54}$in
mm millimeter，毫米。1mm=$\frac{1}{10}$cm
sp scaled point。TeX 的基本长度单位，1sp=$\frac{1}{65536}$pt

em 当前字号下，大写字母 M 的宽度。
ex 当前字号下，小写字母 x 的高度。

然后是几个常用的长度宏，更多的长度宏使用会在表格、分栏等章节提到。

\textwidth	页面上文字的总宽度，即页宽减去两侧边距。
\linewidth	当前行允许的行宽。

我们通常使用hspace {len}和vspace {len}这两个命令控制特殊的空格，具体的使用方法参考水平和竖直距离这一节。

2.4.1 换行与分段

通常的换行方法非常简单：LaTeX 会自动转行，然后在每一段的末尾，只需要输入两个回车即可完成分段。

在下划线一节的例子中已经给出了强制换行的方式，即两个反斜：\\. 不过这样做的缺点在于下一行段首缩进会消失。你可以用\par 来解决这个问题。

2.4.2 分页

用newpage 命令开始新的一页。
用clearpage 命令清空浮动体队列[2]，并开始新的一页。
用cleardoublepage 命令清空浮动体队列，并在偶数页上开始新的一页。
注意：以上命令都是基于\vfill 的，如果要连续新开两页，请在中间加上一个空的箱子（mbox {}），如newpage mbox {}newpage 。

2.4.3 缩进

英文的段首不需要缩进。但是对中文而言，需要借助 indentfirst 宏包来完成。你可能还需要使用setlength {indent }{2em}这样的命令来完成缩进距离的设置。

2.5 字体

2.5.1 字族、字系与字形

2.5.2 关于中文“斜体”

2.5.3 英文字体

在 XeLaTeX 编译下，一般使用 fontspec 宏包来选择英文字体。注意：fontspec 宏包可能会明显增加编译所需的时间。

```
1 \usepackage{fontspec}
2 \newfontfamily{\lucida}{Lucida Calligraphy}
3 \lucida{This is Lucida Calligraphy}
```

[2]参见浮动体这一节的内容。

2.5.4 中文字体

对于中文，需要参考 XeLaTeX 编译下的 xeCJK 宏包的使用。
比如在导言区：

```
\usepackage[slantfont,boldfont]{xeCJK}
\xeCJKsetup{CJKMath=true}
\setCJKmainfont[BoldFont=SimHei]{SimSun}
% 这里把SimHei直接写成中文“黑体”似乎也可以
```

其中，加载 xeCJK 宏包时使用了 slantfont 和 boldfont 两个选项，表示允许设置中文的斜体和粗体字形。在 setCJKmainfont 命令中，把 SimSun（宋体）设置为了主要字体，SimHei（黑体）设置为主要字体的粗体字形，即 textbf 或者 bfseries 命令的变换结果。你也可以使用 SlantFont 来设置它的斜体字形。

除了 setCJKmainfont，还有 setCJKsansfont（对应 textsf），setCJKmonofont（对应 texttt），以及 setCJKmathfont（对应数学环境下的 CJK 字体，但需要载 xeCJKsetup 中设置 CJKMath=true）。

上面提到的 xeCJKsetup 有下列可以定制的参数，下划线为默认值：

- CJKMath=ture/false：是否支持数学环境 CJK 字体。
- CheckSingle=true/false：是否检查 CJK 标点单独占用段落最后一行。此检查在倒数二、三个字符为命令时可能失效。
- LongPunct={——……}：设置 CJK 长标点集，默认的只有中文破折号和中文省略号。长标点不允许在内部产生断行。你也可以用 += 或者 -= 号来修改 CJK 长标点集。
- MiddlePunct={——•}：设置 CJK 居中标点集，默认的只有中文破折号和中文间隔号（中文输入状态下按数字 1 左侧的重音符号键）。居中标点保证标点两端距前字和后字的距离等同，并禁止在其之前断行。你同样可以使用 += /-= 进行修改。
- AutoFakeBold=true/false：是否启用全局伪粗体。如果启用，在 setCJKmainfont 等命令中，将用 AutoFakeBold=2 这种参数代替原有的 BoldFont=SimHei 这种参数。其中，数字 2 表示将原字体加粗 2 倍实现伪粗体。
- AutoFakeSlant=true/false：是否启用全局伪斜体。仿上。

如果要临时使用一种 CJK 字体，使用 CJKfontspec 命令。其中的 FakeSlant 和 FakeBold 参数根据全局伪字体的启用情况决定；如果未启用则使用 BoldFont、SlantFont 参数指定具体的字体。

```
{\CJKfontspec[FakeSlant=0.2,FakeBold=3]{SimSun} text}
```

对于 Windows 系统，想要获知电脑上安装的中文字体，使用 CMD 命令：

```
fc-list -f "%{family}\n" :lang=zh-cn >c:\list.txt
```

然后到`C:\list.txt` 中进行查看。

2.6 目录与大纲

LaTeX 中，将文档分为

3 数学排版

3.1 行间与行内公式

3.2 数学字体、字号与空格

3.3 基础

3.3.1 上下标与堆叠

3.3.2 微分与积分

3.3.3 分数与根式

3.3.4 累加与累积

3.3.5 矩阵

3.3.6 分段函数与联立方程

3.3.7 多行公式

3.3.8 二项式及其他

3.4 数学符号

3.4.1 希腊字母

3.4.2 二元运算符

3.4.3 二元关系符

3.4.4 箭头

3.4.5 其他

LaTeX 进阶 4

4.1 如何使用自定义命令

4.2 箱子：排版的基础

4.3 水平距离与垂直距离

4.4 字体调用

4.5 自定义章节样式

4.6 自定义目录样式

4.7 自定义图表

4.8 自定义编号列表

4.9 自定义公式编号

4.10 创建参考文献

4.11 附录、图表目录

4.12 编程代码输出环境

5 Tikz 绘图 *

其实 LaTeX 本身是有绘图功能的，可以绘制线段、斜率为整数的直线等基础功能。显然这是不能满足绘图需求的。因此在本手册中，对于 LaTeX 原生的绘图功能不再多做介绍，有兴趣的可以自行了解。本手册主要针对 Tikz 绘图。[1]

PGFPlots/Tikz 是两个 LaTeX 的绘图宏包，功能极其强大。其基本的结构为：

```
\documentclass{doc-class}
\usepackage{tikz}
\usetikzlibrary{...}
\begin{document}
  \begin{tikzpicture}
    ...
  \end{tikzpicture}
\end{document}
```

废话不多说，进入正题。

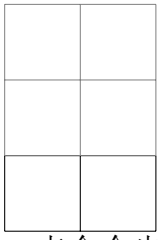
5.1 简单的实例

5.1.1 直线、网格与点

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

注意到 help lines 这个参数使网格绘制更有辅助线的效果，颜色更淡：

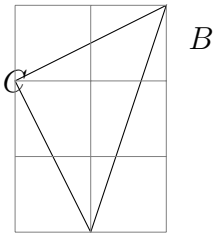
[1] 在 LaTeX 中其实还可以使用 PSTricks 宏包，是非常强力。XYpic 宏包则有时用来画小图。



```
\begin{tikzpicture}
  \draw (0,0) grid (2,1);
  \draw [help lines](0,1) grid (2,3);
\end{tikzpicture}
```

点命令也非常好理解。其中\coordinate 是真正将点的名字和坐标联系起来的命令（相当于“赋值”），而\node 命令仅是一个标注文本的作用。

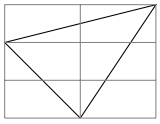
注意到第二行\node 命令，它使得字母 B 标注在 pB 点 315 度的位置。这样的效果往往优于其上一行的\node 命令的效果。注意：不要忘记\node 命令后面的花括号（即使它是空的）！




```
\begin{tikzpicture}
  \coordinate (pA) at (1,0);
  \coordinate (pB) at (2,3);
  \coordinate (pC) at (0,2);
  \node at (pC) {$C$};
  \node[label=315:$B$] at (pB){};
  \draw (pA) -- (pB) -- (pC) -- (pA);
  \draw [help lines](0,0) grid (2,3);
\end{tikzpicture}
```

5.1.2 拉伸

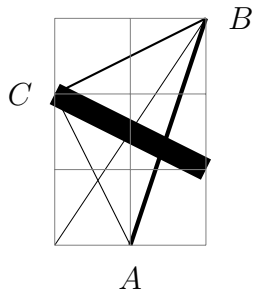
拉伸只需要在 tikzpicture 后添加 xscale/yscale/scale 的可选参数赋值即可。例如：



```
\begin{tikzpicture}[yscale=0.5]
  \coordinate (pA) at (1,0);
  \coordinate (pB) at (2,3);
  \coordinate (pC) at (0,2);
  \draw (pA) -- (pB) -- (pC) -- (pA);
  \draw [help lines](0,0) grid (2,3);
\end{tikzpicture}
```

5.1.3 线宽

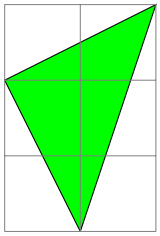
线宽可以在 tikzpicture 后使用 “[line width=5pt]” 之类的参数进行全局调整，也可以在每个 draw 指令下分别地进行调整：




```
\begin{tikzpicture}
  \coordinate (pA) at (1,0);
  \coordinate (pB) at (2,3);
  \coordinate (pC) at (0,2);
  \node[label=270:$A$] at (pA){};
  \node[label=0:$B$] at (pB){};
  \node[label=180:$C$] at (pC){};
  \draw[ultra thick] (pA) -- (pB);
  \draw[thick] (pB)-- (pC);
  \draw[thin] (pC)-- (pA);
  \draw[ultra thin] (pB) -- (0,0);
  \draw[line width=0.3cm] (pC) -- (2,1);
  \draw [help lines](0,0) grid (2,3);
\end{tikzpicture}
```

5.1.4 填色

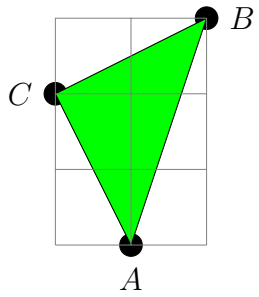
填色的逻辑非常简单，是指向绘制的**闭合**图形内部填色。本例中，draw 命令首尾相接地连接三个点，构成一个闭合图形——三角形，因此可以填色：



```
\begin{tikzpicture}
  \coordinate (pA) at (1,0);
  \coordinate (pB) at (2,3);
  \coordinate (pC) at (0,2);
  \draw[fill=green] (pA) -- (pB) -- (pC) --
     (pA);
  \draw[help lines] (0,0) grid (2,3);
\end{tikzpicture}
```

5.1.5 点样式

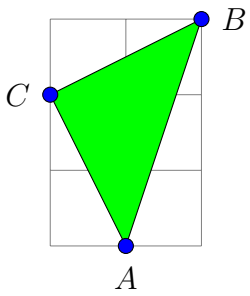
填色之后图形好看了许多，但是还不够。点标在图中似乎也不看清位置，不过这是有方法解决的：




```
\begin{tikzpicture}
  \coordinate (pA) at (1,0);
  \coordinate (pB) at (2,3);
  \coordinate (pC) at (0,2);
  \node[label=270:$A$,circle,draw,fill,inner
      sep=3pt] at (pA){};
  \node[label=0:$B$,circle,draw,fill,inner
     sep=3pt] at (pB){};
  \node[label=180:$C$,circle,draw,fill,inner
      sep=3pt] at (pC){};
  \draw[fill=green] (pA) -- (pB) -- (pC) --
     (pA);
  \draw[help lines] (0,0) grid (2,3);
\end{tikzpicture}
```

但是这一长串简直是太复杂了。大概能看懂 circle 是将点画成圆形，fill 表示填充，inner sep 表示点的大小。而 Tikz 给出的`\tikzstyle` 能够简化步骤。

还有一点就是，绿色的填充似乎位于点的上方？还是乖乖地调整语句顺序吧。

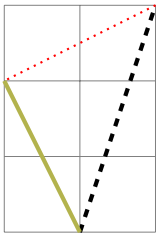



```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (2,3);
  \coordinate (pA) at (1,0);
  \coordinate (pB) at (2,3);
  \coordinate (pC) at (0,2);
  \draw[fill=green] (pA) -- (pB) -- (pC) --
     (pA);
  \tikzstyle{every node} = [circle,draw,fill
     =blue,inner sep=2pt];
  \node[label=270:$A$] at (pA){};
  \node[label=0:$B$] at (pB){};
  \node[label=180:$C$] at (pC){};
\end{tikzpicture}
```

注意：`\tikzstyle{every node}` 控制的是其下方代码的所有 node 的点样式，不影响其上方代码中的点。

5.1.6 线型和颜色

除了点的定义，线当然也可以定义。线型有 dashed/dotted 两种，颜色除了默认的也可以通过叹号的形式控制。



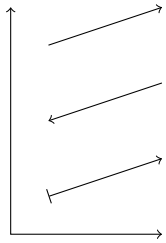
```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (2,3);
  \coordinate (pA) at (1,0);
  \coordinate (pB) at (2,3);
  \coordinate (pC) at (0,2);
  \draw[dashed, ultra thick] (pA) -- (pB);
  \draw[dotted, red, thick] (pB) -- (pC);
  \draw[blue!30!yellow, ultra thick] (pC) --
     (pA);
\end{tikzpicture}
```

主要的颜色包括[2]：red , green , yellow , blue , cyan , magenta , black , gray , darkgray , lightgray , brown , lime , olive , orange , pink , purple , teal , violet , 还有 white .

5.1.7 箭头

箭头也是需要经常绘制的内容。主要也是根据`\draw` 命令的参数控制：

[2]以下色块用`\tikz{\draw[`*color*`,line width=9] (0,0) -- (0.5,0);}` 绘制得到。

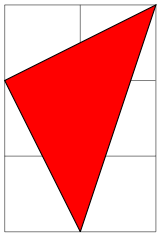


```tex
\begin{tikzpicture}
  \draw[->] (0.5,2.5) -- (2,3);
  \draw[<-] (0.5,1.5) -- (2,2);
  \draw[|->] (0.5,0.5)-- (2,1);
  \draw[<->] (0,3) -- (0,0) -- (2,0);
\end{tikzpicture}
```

5.2 高效书写

5.2.1 变量

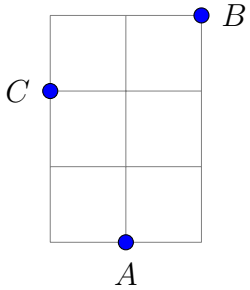
变量申请使用与 LaTeX 相同的语句：\newcommand. 注意：变量需要以反斜杠开头，并与现有命令不重复。



```tex
\begin{tikzpicture}
  \draw [help lines](0,0) grid (2,3);
  \newcommand{\aaa}{1};
  \newcommand{\bbb}{3};
  \newcommand{\ccc}{2};
  \coordinate (pA) at (\aaa,0);
  \coordinate (pB) at (\ccc,\bbb);
  \coordinate (pC) at (0,\ccc);
  \draw[fill=red] (pA) -- (pB) -- (pC) -- (
     pA);
\end{tikzpicture}
```

5.2.2 循环

但是同样的语句书写很多遍是很复杂的，好在 Tikz 提供了循环的实现方法：



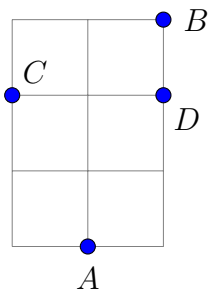

```tex
\begin{tikzpicture}
  \newcommand{\la}{1};
  \newcommand{\lb}{3};
  \newcommand{\lc}{2};
  \draw [help lines](0,0) grid (\lc,\lb);
  \coordinate (pA) at (\la,0);
  \coordinate (pB) at (\lc,\lb);
  \coordinate (pC) at (0,\lc);
  \tikzstyle{every node}=[circle, draw, fill
     =blue,inner sep=2pt];
  \foreach \x/\y in {A/270,B/0,C/180}{
    \node[label=\y:$\x$] at (p\x){};
  }
\end{tikzpicture}
```

注意到：甚至点的名称 pA, pB, pC 中的 A, B, C 也是可以通过 foreach 来指定的！

5.2.3 运算

注意：需要在导言区添加\usetikzlibrary{calc}。文中不再写出。

如果一个绘图工具仅仅能够绘图而不能做运算……它有什么用呢？如果作为科学排版系统下的绘图工具而不支持运算的话，Tikz 岂不是太弱了？




```
\begin{tikzpicture}
  \draw [help lines](0,0) grid (2,3);
  \coordinate (pA) at (1,0);
  \coordinate (pB) at (2,3);
  \coordinate (pC) at (0,2);
  \coordinate (pD) at ($(pB)+(0,-1)$);
  \tikzstyle{every node}=[circle, draw, fill
     =blue,inner sep=2pt];
  \foreach \x/\y in {A/270,B/0,C/45,D/315}{
    \node[label=\y:$\x$] at (p\x){};
  }
\end{tikzpicture}
```

附录 A

注音符号

表 I.1: 注音符号与特殊符号

样式	-	命令	样式	-	命令	样式	-	命令	样式	-	命令
ō	-	\=o	ó	-	\'o	ǒ	-	\v o	ò	-	\`o
ô	-	\^o	ö	-	\"o	ȯ	-	\.o	ő	-	\H o
o̩	-	\d o	ŏ	-	\u o	o̲	-	\b o	o͡o	-	\t oo
	õ		—	$\tilde{o}$			ô		—	$\hat{o}$	
ø	-	\o	Ø	-	\O	ɪ	-	\i	ȷ	-	\j
å	-	\aa	Å	-	\AA	æ	-	\ae	Æ	-	\AE
œ	-	\oe	Œ	-	\OE	¡	-	!`	¿	-	?`