

# **L<sup>A</sup>T<sub>E</sub>X 汉语文档教程**

## **A Brief Chinese Tutorial on L<sup>A</sup>T<sub>E</sub>X**

编程爱好者协会

2020/11/24

本文以 CC BY-NC-SA 4.0 协议随源码发布





# 目录

I 前言：关于 L <sup>A</sup> T <sub>E</sub> X	2
1.1 认识 L <sup>A</sup> T <sub>E</sub> X 和 T <sub>E</sub> X	2
1.2 L <sup>A</sup> T <sub>E</sub> X 能做到的事情，以及选择 L <sup>A</sup> T <sub>E</sub> X 的理由	2
1.3 L <sup>A</sup> T <sub>E</sub> X 的开发工具链及资料指南	3
II 开始使用 L <sup>A</sup> T <sub>E</sub> X	4
2.1 你好，L <sup>A</sup> T <sub>E</sub> X	4
2.2 支持中文的 L <sup>A</sup> T <sub>E</sub> X 文档	4
2.3 初学者会用到的包	5
2.3.1 语言支持包	5
2.3.2 理工科类宏包	5
2.3.3 题注、图表和图片宏包	8
2.3.4 排版布局类宏包	9
2.3.5 杂项类宏包	9
III L <sup>A</sup> T <sub>E</sub> X 基本功能	10
3.1 基本概念	10
3.1.1 控制序列和宏	10
3.1.1.1 控制序列和宏的概念	10
3.1.1.2 定义新宏	10
3.1.2 文档类型	11
3.1.3 导言区和正文	11
3.1.4 计数器	11
3.2 换行和换页	11
3.3 通用格式控制	12
3.4 层次结构和标题	13
3.4.1 标题样式	14
3.4.1.1 编号数字样式	14
3.4.2 改变标题格式	14
IV 附录	15
A 常见问题 (FAQ)	15



## 第 I 部分 前言：关于 L<sup>A</sup>T<sub>E</sub>X

本文使用 L<sup>A</sup>T<sub>E</sub>X 写就，以本文自身使用的代码为例，展示和讲解 L<sup>A</sup>T<sub>E</sub>X 基本语法和在计算机工程领域撰写文章的常见用途用法。本文应当连同源码和一份附带的 img 图片资源文件夹发布，若没有，请向发布者索要，但必须在尊重协议的基础上。

### 1.1. 认识 L<sup>A</sup>T<sub>E</sub>X 和 T<sub>E</sub>X

L<sup>A</sup>T<sub>E</sub>X 和 T<sub>E</sub>X 都是用于排版的语言。T<sub>E</sub>X 及其发展出的语言都是指令驱动的，用于生成 pdf 文件。这些语言的基础都在于宏控制序列，开发者可以通过编写代码来进行精确、高度自动化同时高度可定制的排版任务。无论是字体字号还是段落格式，随笔文集还是论文，都可以用 L<sup>A</sup>T<sub>E</sub>X 中高度集成化的指令设计解决。此外，L<sup>A</sup>T<sub>E</sub>X 可以自动为您的各种题注进行编号，并方便地在后文中引用，这样您就不需要在编号变动后一个个手动改动引用了，这也意味着出错的几率大大降低。

T<sub>E</sub>X 的出现很早，可追溯至 HTML 标记语言出现之前。L<sup>A</sup>T<sub>E</sub>X 继承和发展了 T<sub>E</sub>X，可以简单地认为 T<sub>E</sub>X 相对“朴实无华”而 L<sup>A</sup>T<sub>E</sub>X 内置了众多常用宏。对本文读者而言只需要知道一件事：L<sup>A</sup>T<sub>E</sub>X 相比一般的 T<sub>E</sub>X 更加易用，适合快速上手。本文接下来只讨论 L<sup>A</sup>T<sub>E</sub>X 而不再讨论 T<sub>E</sub>X。

### 1.2. L<sup>A</sup>T<sub>E</sub>X 能做到的事情，以及选择 L<sup>A</sup>T<sub>E</sub>X 的理由

L<sup>A</sup>T<sub>E</sub>X 在排版工作上几乎无所不能。使用社区提供的宏包，它能完成几乎任何常见排版需求。

对选择 L<sup>A</sup>T<sub>E</sub>X 顾虑最大的人群恐怕是 Microsoft Office Word 的用户。诚然 Markdown 和 HTML&CSS 都可用于排版，但是 Markdown 和 HTML&CSS 的业务还是主要集中于网络界面。Markdown 也可以做文档排版，但是就笔者的经验而言，Markdown 使用上虽然简易，但是很多自定义功能实现反而比 L<sup>A</sup>T<sub>E</sub>X 复杂一些。而在离线程序文档中，Markdown 写就的 README 和 L<sup>A</sup>T<sub>E</sub>X 就更没有什么比较意义了。所以，笔者并不认为 Markdown 和 L<sup>A</sup>T<sub>E</sub>X 有太多的业务重叠。也有读者可能会使用方正和 Adobe 的产品，但那些产品通常是收费的（当然，Microsoft Office 也是收费的。您也可以使用某些软件的免费版，但相比之下 L<sup>A</sup>T<sub>E</sub>X 中不存在免费版比收费版更差的情况），而且主要面向专业的排版厂商，个人开发者使用那种体量的企业级产品实在是没有必要。我们这里主要阐述 L<sup>A</sup>T<sub>E</sub>X 相比 Microsoft Office Word 的优劣。

相比于 Microsoft Office Word 的 GUI 为核心的设计，在 L<sup>A</sup>T<sub>E</sub>X 下您一般不需要特别考虑把图片放在哪，L<sup>A</sup>T<sub>E</sub>X 会尝试自己解决这件事。如果对它的工作不满意，也可以通过参数自行指定。相比 Word 下需要点来点去拖来拖去，L<sup>A</sup>T<sub>E</sub>X 只需要您已经写好的代码就能得到最终结果，要修改效果也只需要简单地改写代码，不需要打开额外的属性页面。

L<sup>A</sup>T<sub>E</sub>X 另一大亮点是集成包管理系统，只需要引入几个包敲几行代码就能生成漂亮的数学公式或是化学方程式，物理论文或是生物论文自然也可以。如果需要插图，L<sup>A</sup>T<sub>E</sub>X 也有一些常见的绘图软件，只要需要的示意图不是太复杂，您就能用 L<sup>A</sup>T<sub>E</sub>X 把它画出来。相比于传统讲义设计中繁琐的示意图绘制、公式插入和特殊符号插入，您在 L<sup>A</sup>T<sub>E</sub>X 下只需要几行代码。当然，这些代码需要用户自行编写。

传统计算机工程领域讲义编写中，代码高亮常常是通过截图或第三方工具实现的。这一切在 L<sup>A</sup>T<sub>E</sub>X 社区提供的宏包面前都黯然失色。在 L<sup>A</sup>T<sub>E</sub>X 下，您所需要的只是引入包，设置高亮格式，把



代码复制进来。如果愿意，您甚至可以自己实现一门语言及其在 L<sup>A</sup>T<sub>E</sub>X 下的高亮。哪怕拥有 yacc 工具的用户也需要自己编写代码才能实现这样的自定义语言高亮，但在 L<sup>A</sup>T<sub>E</sub>X 下，用户只需要使用提供好的工具，甚至没有必要真的把这门语言实现出来。

然而，L<sup>A</sup>T<sub>E</sub>X 是诞生于 CLI 时代的工具（当然也有 IDE，但底层仍然是 CLI），对于使用惯了 Windows 提供的图形化操作的用户而言，L<sup>A</sup>T<sub>E</sub>X 可能不是太美好。而且，L<sup>A</sup>T<sub>E</sub>X 并不是所见即所得的工具，所以您需要不断改写和编译您的源码才能得到期望的文档。但是，如果您真的是本文所面向的计算机领域工程师，我有理由相信您是乐于折腾手头的工具和设备的 geek——否则恐怕不会选择这个行业。如果果真如此的话，有理由相信 L<sup>A</sup>T<sub>E</sub>X 能成为您今后的最佳伴侣。

### 1.3. L<sup>A</sup>T<sub>E</sub>X 的开发工具链及资料指南

中文社区最常见的 L<sup>A</sup>T<sub>E</sub>X 开发工具是 C<sub>T</sub><sub>E</sub><sub>X</sub>，但笔者在使用 C<sub>T</sub><sub>E</sub><sub>X</sub> 的过程中遇到了不小的麻烦，主要是附带的 MiK<sub>T</sub><sub>E</sub><sub>X</sub> 版本太旧，无法用于更新宏包。笔者的建议是使用官网的 MiK<sub>T</sub><sub>E</sub><sub>X</sub> 进行 L<sup>A</sup>T<sub>E</sub>X 的编辑。如果您愿意，也可以安装不带 MiK<sub>T</sub><sub>E</sub><sub>X</sub> 的 C<sub>T</sub><sub>E</sub><sub>X</sub> 包，再单独安装 MiK<sub>T</sub><sub>E</sub><sub>X</sub>，这样可以有效避免宏包无法更新的问题。

MiK<sub>T</sub><sub>E</sub><sub>X</sub> 和 C<sub>T</sub><sub>E</sub><sub>X</sub> 本质上是提供了开发环境，实际编译工作是附带的工具链来做的。比较常见的工具链有 pdfL<sup>A</sup>T<sub>E</sub>X、X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X、X<sub>Y</sub>L<sub>T</sub><sub>E</sub><sub>X</sub> 等，笔者建议中文用户使用 X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X。本文就是使用 X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X 编写的。

您可以从清华大学开源软件站<sup>1</sup>获取 C<sub>T</sub><sub>E</sub><sub>X</sub> 套装，而德语 T<sub>E</sub>X 用户组维护的网站 T<sub>E</sub>Xdoc Online<sup>2</sup>处可以获取 L<sup>A</sup>T<sub>E</sub>X 常见宏包的文档。此外，stackexchange 也有一个专门的 T<sub>E</sub>X 论坛<sup>3</sup>，可以在那里查询到常见问题的解决方案，还有一个称为 StackOvernet 的论坛也有大量技术资料。Comprehensive T<sub>E</sub>X Archive Network<sup>4</sup>同样提供大量可供查阅的文档，该网站也能下载到各种宏包。L<sup>A</sup>T<sub>E</sub>X 论坛<sup>5</sup>也有一些常见问题的解答。MiK<sub>T</sub><sub>E</sub><sub>X</sub> 则可以从 MiK<sub>T</sub><sub>E</sub><sub>X</sub> 官网<sup>6</sup>获得。

L<sup>A</sup>T<sub>E</sub>X 包有相当一部分是社区维护的，除非您能联系到包的作者本人，否则不太可能获取客服技术帮助什么的。L<sup>A</sup>T<sub>E</sub>X 宏包因此会出现各种兼容性问题，请注意自行查阅材料确定哪些包可以混用，哪些不可以。此外，MiK<sub>T</sub><sub>E</sub><sub>X</sub> 用户应当了解，安装到计算机上的 MiK<sub>T</sub><sub>E</sub><sub>X</sub> 已经附带了包管理器，可以用于从网络上下载包并自动安装。

---

<sup>1</sup><https://mirrors.tuna.tsinghua.edu.cn/>

<sup>2</sup><http://texdoc.net/>

<sup>3</sup><https://tex.stackexchange.com/>

<sup>4</sup><http://ctan.math.illinois.edu>

<sup>5</sup><https://latex.org/forum/>

<sup>6</sup><https://miktex.org/>




## 第 II 部分 开始使用 L<sup>A</sup>T<sub>E</sub>X

### 2.1. 你好, L<sup>A</sup>T<sub>E</sub>X

L<sup>A</sup>T<sub>E</sub>X 的主要构成十分简单, 第一行声明基本文档类型 (`\documentclass`), 包含宏包 (非必要, 但没有宏包基本上做不了任何事情), 前导区 (`preamble`) 声明必要的东西, 最后开始写文档体。

```
1 \documentclass{article}
2
3 % \begin 表示开始环境, 与\end 配对
4 % document 表示文档体环境
5 % 百分号开始的内容一般是注释, 但也有例外
6 \begin{document}
7 Hello \LaTeX!
8 \end{document}
```

代码 2.1: Hello L<sup>A</sup>T<sub>E</sub>X

这就是一个 L<sup>A</sup>T<sub>E</sub>X 的 Hello world 文档 (不过输出的是 Hello L<sup>A</sup>T<sub>E</sub>X 而不是 Hello world), 在 TeX-works 界面上按下排版键  (默认快捷键 `Ctrl`+`T`) 即可看到编译结果。该例子中的源码只输出一页 pdf, 包含一行 Hello world 字样。它没有引用任何包。从这个例子中可见, L<sup>A</sup>T<sub>E</sub>X 的显示内容是直接写在文档里面就好的, 由控制序列 (控制序列是它的正式名称, 可以简单理解为指令) 指定格式。

上述例子还无法输出中文。如果简单将 Hello world 文本更换为中文, 你会发现, 它什么也不会输出: 编译器没有正确辨认中文字符。

### 2.2. 支持中文的 L<sup>A</sup>T<sub>E</sub>X 文档

给文档加一个包即可正确向 pdf 文件写入中文。以下是 X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X 下的做法:

```
1 \documentclass{article}
2 % 有了这个包, 就能正确输出中文了
3 \usepackage{ctex}
4
5 \begin{document}
6
7 你好, \LaTeX!
8 \end{document}
```

代码 2.2: 你好, 中文 L<sup>A</sup>T<sub>E</sub>X- X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X

使用 pdfL<sup>A</sup>T<sub>E</sub>X, 则需要使用另一个方案:

```
1 \documentclass{article}
2 \usepackage{CJK}
```



```

3
4 \begin{document}
5 \begin{CJK}{UTF8}{song}
6
7 你好，\LaTeX!
8 \end{CJK}
9 \end{document}

```

代码 2.3: 你好，中文 L<sup>A</sup>T<sub>E</sub>X- pdfL<sup>A</sup>T<sub>E</sub>X

以上代码即是令 L<sup>A</sup>T<sub>E</sub>X 支持中文的使用方法<sup>1</sup>。不过，笔者个人喜欢使用 `ctexart` 文档类型。

## 2.3. 初学者会用到的包

L<sup>A</sup>T<sub>E</sub>X 实际上是以宏<sup>2</sup>为基础的语言，所以严格而言，“包”在这里应当叫做宏包。宏以 `\def` 命令定义，此外，`\newenvironment`、`\newcommand` 和 `\renewcommand` 分别可以定义新环境、新宏和修改特定宏的作用。作为一篇简明教程，这里我们只进行“脚本小子”式的教学，讲述常见几个宏包的使用方法，而略去关于宏本身的内容。

### 2.3.1. 语言支持包

对于中文用户，最常见的应该是 CJK 包和 `ctex` 包，这两个是中文支持包。此外，还有 `greektext` 希腊语包等其他语言支持。值得一提的是，有些院校要求子目录使用汉字编号，这可以用 `zhnumber` 包来完成，后面的章节会详细说明。还有一些包提供更多语言学特性，例如生成音标用的 `tipa` 包（IPA 就是国际音标的缩写）。

用 `tipa` 包输出 Hello L<sup>A</sup>T<sub>E</sub>X 的音标为 `[hə'ləʊ][lɛɪtɛk]`<sup>3</sup>。

### 2.3.2. 理工科类宏包

`amsthm` 包可以说是理工类论文最为不可或缺的包，它提供数学环境和符号。常见的还有 `extarrows` 包，它提供特殊的箭头，允许在长箭头上附加文字，可用于书写无机化学方程式或核物理反应方程式。`chemfig` 和 `mhchem` 则常用于书写有机化学方程式，`SIunits` 包提供标准物理学单位。对于计算机行业工作者，最常见的功能大概是引用代码，这经常由 `listings` 包来完成。还有一个常见的叫做 `algorithm2e` 的宏包，用于生成伪代码。

`glossary`、`glossaries`、`nomencl` 这三个宏包都用于绘制术语表。

下面以一些常见用途来举例子。

1.  $\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t} dt$

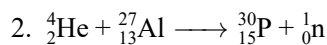
<sup>1</sup>如果您正在比对本源码阅读该份 pdf 文件，您可能已经注意到了：上述代码片中的 `lstlisting` 环境中都加入了一行奇怪的代码，即 `(@*\textcolor{bgcolor__gray}{ }*@)` 而且呈现奇怪的缩进。这里加入一个没有在 pdf 文件中显示出来的控制序列是为了避免错误的换行。总之，读者目前只需要知道一件事：这个控制序列是为了修正 `listings` 包在引用代码片时和中文的不兼容引发的样式问题而加入的，而不是编写中文 L<sup>A</sup>T<sub>E</sub>X 文档所必要的控制序列。后文会简单讲解该控制序列的含义。至于缩进……L<sup>A</sup>T<sub>E</sub>X 是一个不在乎缩进的语言，有些时候空行也无关紧要（但更多的时候，空行有特别意义），这里只好先牺牲一点样式效果，避免更大的样式问题。

<sup>2</sup>宏的概念可以理解为直接原地展开宏内容并执行操作。有程序设计基础的读者可以联想其他语言的宏概念来理解它。宏有利于大量减少代码量，不过相比于其他代码生成技术，宏有个显著缺点：可读性很差。我们建议工程师编写宏时辅以充足的文档说明。

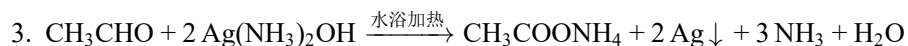
<sup>3</sup>实际上 L<sup>A</sup>T<sub>E</sub>X 不仅这一种读法，详见附录 FAQ。



这是傅里叶变换的基本公式。读者今天能用上电子产品不仅要感谢麦克斯韦等基础电磁理论的奠基人，还要感谢这个您也许看不懂的东西。不过如果您是学电子信息的，那就一定要看懂这个东西，还要学好高等数学和复变函数。

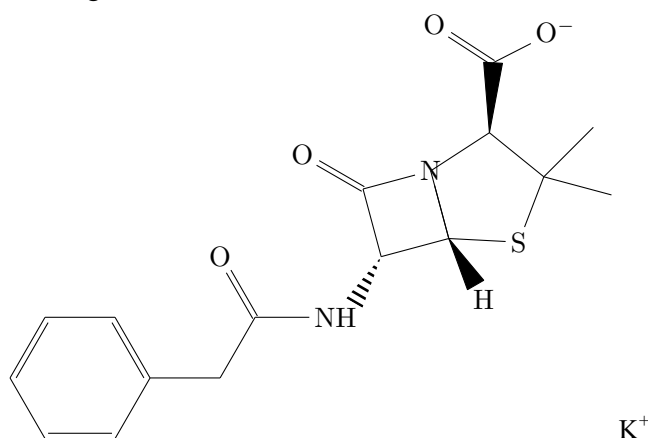


这是一个高中常见的  $\alpha$  粒子轰击铝的核物理方程式。使用  $\alpha$  粒子轰击铝箔后产生不稳定的  ${}^{30}_{13}\text{P}$ ，这种放射性核素很快就会继续衰变。 ${}^{30}_{13}\text{P}$  是最早从实验室制造出来的人工放射性核素之一。



这个是银镜反应的化学方程式，我们在这里用它展示有特殊反应条件的方程式如何用 L<sup>A</sup>T<sub>E</sub>X 书写。

4. chemfig 绘制的结构式：



这种物质叫做青霉素钾，常用于制药。选取这个结构式来绘制是因为它具有许多有机化学结构式中会出现的特性，例如环、环嵌套以及各种分支。<sup>4</sup>

5. algorithm2e 生成的伪代码：

输入：待排序数组  $A[n]$ ，要求其类型强有序

输出：排好序的数组，仍存放在  $A[n]$  中

```

1 Function BUBBLE-SORT( $A[n]$ )
2 begin
3   for  $i \leftarrow 1$  to  $n$  do
4     for  $j \leftarrow 1$  to  $n - i$  do
5       if  $A[j] > A[j + 1]$  then
6         SWAP( $A[j]$ ,  $A[j + 1]$ )
7       end
8     end
9   end
10 end

```

这是一段冒泡排序的伪代码（pseudo-code），仅仅是对算法的描述，不能被实际编译运行。冒泡排序是一个相当经典和简单的排序算法，它每一趟循环把前/后若干个值中最大/小者放在最

<sup>4</sup>说真的，如果您只是用 L<sup>A</sup>T<sub>E</sub>X 来做化学作业，那我建议读者直接导入插图，使用 chemfig 绘制如此简单的一个结构式就花费了我数个小时来查阅资料。



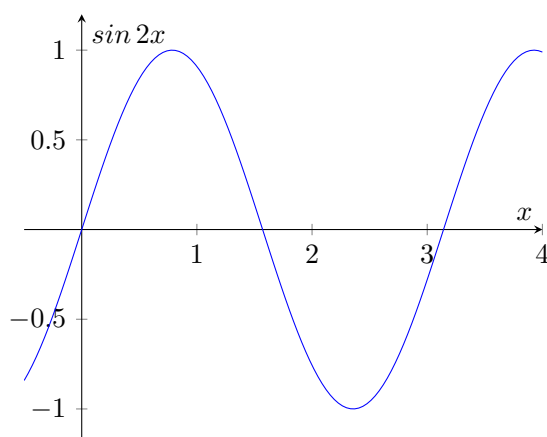
后/前（取决于你要怎样的排序结果，以及如何实现它），所以很显然，每一趟排序后必然已经有至少一个值被放在了它最终该在的位置上。《算法导论》中具体讨论了这个算法。

#### 6. listings 提供的代码片功能：

```
1 // A simple C++ example
2 #include<iostream>
3 using std::cout;
4 using std::endl;
5
6 int main() {
7     cout << "Hello□world" << endl;
8     return 0;
9 }
```

这是一个简单的 C++ 程序，作用就是输出一行 Hello world。一个常见的误解是“std::cin 一定比 scanf 慢”，其实并不一定如此；SysTutorials 上一个题为“C++ 的 cin 真的比 scanf 慢很多吗”的主题讨论了这一点。

#### 7. 函数图象：



这个例子展示了如何在 L<sup>A</sup>T<sub>E</sub>X 下用 pgfplots 绘制一幅笛卡尔坐标系下的函数图象。除此之外，pgf 包还有许多其他绘图功能。

#### 8. 引入术语表：

这里指的术语表是附带定义信息的术语解释表，而不是类似《算法导论》附录那样带页码但没有附带定义的术语表。

## 术语表

**ABI** 即引用程序二进制接口。不同平台、不同操作系统、不同编译器、不同运行时库的二进制接口都有可能不同。

**mangle** C++ 等允许重载的语言中，为了给同名函数分配不同符号而产生的技术。





符号 二进制文件中，用于标记需要使用的标识符时用到的字符串。符号不能产生冲突，否则会引起链接错误。

### 2.3.3. 题注、图表和图片宏包

`caption` 包和 `subcaption` 包用于题注。包含图片时，最常用的包是 `graphicx`。`multirow` 包用于进行单元格合并，`TikZ` 和 `pgf` 用于绘制图样。制表时经常配合 `array` 包制定样式。特别地，`circuitikz` 包主要用于绘制电路图。对于绘制流程图的需求，我们建议使用第三方软件 `graphviz`，由 `graphviz` 生成图片后再利用 `graphicx` 宏包将其导入到文档中。

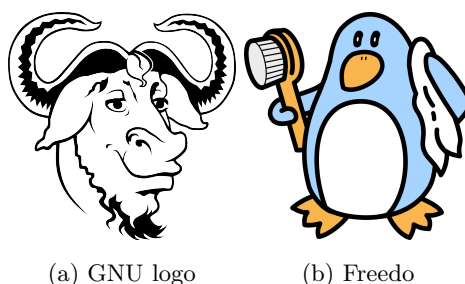


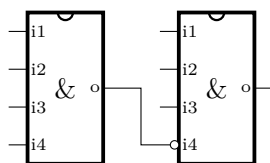
图 2.1: 两个 logos

这个例子中，L<sup>A</sup>T<sub>E</sub>X 为这两张图片分别分配了题注，GNU 的 logo 得到编号 2.1a，小企鹅 Freedo<sup>5</sup>得到编号 2.1b。两张图还有共同题注，编号为 2.1。Aurélio A. Heckert 提供了图 2.1a，Rubén Rodríguez Pérez 等人提供了图 2.1b<sup>6</sup>，这两张图都来自 GNU 画廊<sup>7</sup>。

(0,0)	row 0		
column 0	center		

表 2.1: 一张随意画的 L<sup>A</sup>T<sub>E</sub>X 表格

这个表格例子展示了 L<sup>A</sup>T<sub>E</sub>X 中复杂表格的绘制方法。L<sup>A</sup>T<sub>E</sub>X 下的原生表格实际上做不到这点，但是我们有支持自定义表格的宏包。一般而言，Microsoft Office Word 能做到的表格功能，L<sup>A</sup>T<sub>E</sub>X 都有对应实现。常用的宏包有 `array` 和 `multicol`、`multirow` 等。此外，有一个叫做 `longtable` 的宏包用于绘制可以自动分页的表格。



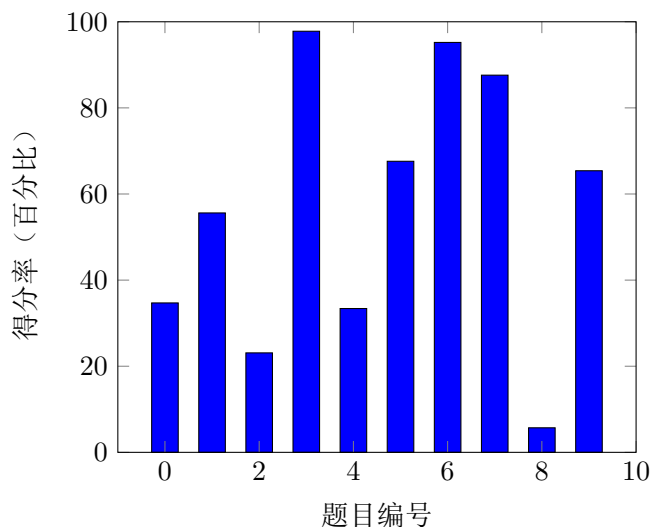
<sup>5</sup>Freedo 是 GNU Linux-libre（一个操作系统内核）的吉祥物。值得一提的是，Linux 最广为人知的吉祥物也是一只企鹅，名为 Tux。然而，许多 Linux 发行版如今选择其他动物作为自身的吉祥物。

<sup>6</sup>这张图是在 <https://svgtopng.com/> 进行了 SVG 格式到 PNG 格式的转码后得到的。

<sup>7</sup>GNU 是一个致力于发扬自由软件精神的组织，知名的 GPL 协议就来自于这个组织，旗下有一个 GNU 操作系统官网并经常在官网上批评一些专有软件（微软和 Adobe 算是最经常被挂起来数落的）。其他一些开源支持者，例如 FFmpeg 的开发者，甚至列出了“耻辱柱”，主要是用于批评那些不遵守开源协议的开发者。



这是一个使用 `circuitTikZ` 包绘制的电路图，包含两个连在一起的四路与门，其中上一级与门的输出反相后连接到下一级与门的第四个输入（注意后一级与门的 `i4` 引脚上的圆圈标记）。这个芯片形式的与门部件是使用 `pgf` 底层命令定义的，并在 `circuitTikZ` 包层面调用封装好的部件。可以在本文随附源码的 `contents/assets/circuittikz/and4b.tex` 看到这个部件的详细定义，包括绘图细节和引脚排布。



#### 2.3.4. 排版布局类宏包

`titlesec`、`titletoc`、`geometry`、`fancyhdr` 等能为用户带来良好的排版体验。这类宏包用于简化页面布局设置操作，例如设置页边距、目录格式、页眉页脚等等。

#### 2.3.5. 杂项类宏包

除了上述与主要功能有关的宏包，L<sup>A</sup>T<sub>E</sub>X 社区还提供了大量排版用的宏包。这些宏包有的仅仅实现了非常简单的功能，例如 `indentfirst` 主要只用于处理缩进。`geometry` 主要用于设置页边距等几何参数。`xcolor` 库用于设置文字色彩。`hyperref` 用于设置超链接。

下文中将会逐步讲解 L<sup>A</sup>T<sub>E</sub>X 内置功能和部分宏包中的常用功能。

**注意** 有许多宏包中的工具需要编译两次才能得到正确结果。所以，遇到编译没出错但效果样式有问题的情况先别着急，试试再编译一次。极少数情况下，这也有助于修正编译错误。发布前也要编译两次再进行审阅。此外，有时候宏包包含顺序也会引发问题，出现宏包内部错误时不妨尝试调整宏包的包含顺序。



## 第 III 部分 L<sup>A</sup>T<sub>E</sub>X 基本功能

### 3.1. 基本概念

#### 3.1.1. 控制序列和宏

##### 3.1.1.1. 控制序列和宏的概念

宏是历史较长的语言中极为常见的一种特性。它诞生于早期编程语言理论不完善、工具不充足的时期，是一种基于文本进行处理的技术，简单说就是对文本进行模式替换或者根据模式重复一定动作。尽管宏在今天的开发者看来相当落后，时至今日，人类社会中仍有巨量软件和文档的生成依赖于宏或者宏驱动的语言。

在 L<sup>A</sup>T<sub>E</sub>X 中，宏和控制序列是很接近的概念，都是以反斜杠 (\) 开始、转义后具有一定特殊含义、完成一定特殊功能的字符串。有些控制序列带有 @ 符号，其语法和功能类似于面向对象编程中对对象内数据域的引用，但普通 L<sup>A</sup>T<sub>E</sub>X 文档中只能通过特殊手段启用这类序列；这类控制序列通常只有在宏包编写内或利用宏包底层接口封装新宏才用得到。宏则一般特指 `\def` 控制序列产生的新的可用宏。

本文中不对宏和控制序列做具体区分。

##### 3.1.1.2. 定义新宏

L<sup>A</sup>T<sub>E</sub>X 中的宏通过 `\def` 或 `\newcommand` 定义。有些语言中同名宏会覆盖之前的定义，但是 L<sup>A</sup>T<sub>E</sub>X 中同名宏通常会引发编译错误，需要使用 `\renewcommand` 覆盖旧有定义。

本文源码中的 `\textlb` 宏 (`lb` 表示 `line break`) 就是通过 `\newcommand` 定义的：

```
\newcommand\textlb{\textbackslash \textbackslash}
```

这是不需要参数的宏，宏名后只需要跟一对花括号指定宏内容。编译器处理对宏的引用时，会先按定义时指定的宏内容对宏进行原地展开（类似于 C，只会展开完整 `token`，而不会只展开半个 `token`），然后进一步处理展开后的内容。

定义需要参数的宏时，宏名后首先跟一对方括号指定参数数目，再跟一对花括号指定宏内容。宏内容中以井号 (#) 对参数按编号进行引用，例如 `#1` 表示第一个参数，`#2` 表示第二个参数。举例而言，有一个这样的宏定义：

```
\newcommand\addbs[1]{\textbackslash #1}
```

那么 `\addbs{foo}` 将会被展开为 `\textbackslash foo`（首个参数就是 `foo`），进一步编译后得到 `\foo`。如果需要多个参数，在引用宏时需要为每个参数都加一对花括号，但定义则与只有一个参数的宏类似，最外层只有一对花括号。例如，假若有这样一个宏定义：

```
\newcommand\foo[2]{$#1_{#2}$}
```

那么使用 `\foo{a}{1}` 对其进行的引用将会被展开为 `$a_1$`，这是 L<sup>A</sup>T<sub>E</sub>X 中的数学环境语法，显示效果为  $a_1$ 。

其他宏包里的宏通常都是使用类似的语法传递参数，但也有些宏可以通过方括号传递参数，或者传递以逗号分割的参数列表。本文仅对 L<sup>A</sup>T<sub>E</sub>X 编写做简要介绍，这类进阶语法按下不表。



### 3.1.2. 文档类型

L<sup>A</sup>T<sub>E</sub>X 支持若干种预设的文档类型。不同文档类型的页面尺寸、默认字体、字号等等都有可能不同，当然，也可以显式指定参数来实现与预设不同的样式效果。很多.tex 主文件第一行就是`\documentclass` 控制序列，我们使用该控制序列来指定当前文档的类型。正常而言，一组用于生成单个 pdf 文件的 L<sup>A</sup>T<sub>E</sub>X 源码应该恰好具有一个`\documentclass` 声明。

不同文档类型支持的控制序列也不同，这点需要格外注意。因此，当您试图在论坛上提问时，应当说明您正在使用或希望使用何种文档类型。

### 3.1.3. 导言区和正文

用于生成 pdf 的源码中总是有一个`\begin{document}` 和`\end{document}` 括起来的区域，这个部分被称为正文。正文前的部分则称为导言区，导言区内通常是不直接参与文本生成的控制序列，或者输出到全局、不能手动直接修改的内容，例如数学环境的定义和提示文字、页眉页脚等，`\documentclass` 也不直接参与文本生成，而只是指定正文的某些样式。

### 3.1.4. 计数器

L<sup>A</sup>T<sub>E</sub>X 主要有两大类计数器：序号计数器和控制计数器，后者主要用于控制每页上浮动结构数量、目录深度等，前者则用于记录和显示序号。本文只阐述序号计数器。

L<sup>A</sup>T<sub>E</sub>X 中有各种各样的控制序列用于产生具有层次的结构，主要是各级标题；下文将会详细介绍多级标题。枚举列表、数学公式、脚注边注等也有自己的多级计数器。序号计数器的初始值通常为 0，也可以用`\setcounter{计数器名称}{数值}` 在任何时候进行赋值（会将跟随其编号的低级计数器置零）。`\stepcounter{计数器名称}` 将指定计数器的值加上 1。多数宏包提供的序号计数器会在通过关联的控制序列被引用后自动步进。因此，如果你将计数器初始化为 1，那么有关序号会从 2 开始编号，而不是 1。

计数器一般附带一个控制序列，即“`\the 计数器名称`”，可以用于获得计数器的值。通过该方式获得的计数器的值可以是罗马数字、阿拉伯数字等，取决于它被如何设置。`\value{计数器名称}` 可以获取计数器值的阿拉伯数字形式，这个特点可以用来将序号输出格式设置为本来不接收计数器参数的数字格式化控制序列。

可以通过`\newcounter{计数器名称}[计数器跟随项]` 设置新的计数器，但这不在本文探讨的范围内。

## 3.2. 换行和换页

很容易发现，在源码下换行后，编译出的 pdf 文件对应位置往往并没有换行（这就是为什么本文源码有那么多空行）。L<sup>A</sup>T<sub>E</sub>X 下最常见的两种换行方式分别是使用换行符号`\`，以及额外空一行。这两种方式都可以用于换行。本文中多数地方使用源码中空行来换行，但细心的读者也许已经发现，本文源码中 document 环境的 title 就用了`\`来换行。

```
\title{\bf \docTitle \ A Brief Chinese Tutorial on \LaTeX}
```

使用`\`换行还有一个技巧，如果附带一对方括号，方括号内部写入一个长度，那么当前换行将会产生额外的空白。本行结尾有一个`\[3mm]` 来展示这个特点。



如你所见，产生了一段竖直方向上的空白。这段空白包含一个空行、两个默认行距外加 3mm 的空白。此外，也有时使用`\newline`进行换行。

换页通常使用`\newpage`或`\clearpage`来实现。不同之处在于，前者只是结束当前页，后者还会把图表等浮动结构输出。`\include`（用于包含另一文件）也会先另起一页再输出被引用文件的内容。

### 3.3. 通用格式控制

L<sup>A</sup>T<sub>E</sub>X 允许控制文字的字体、字号等信息，其中多数功能需要配合宏包提供的附加功能，如表 3.1 所示。数学环境有独立的文字样式设定。

宏	功能	参数	效果	来源
<code>\textup</code>	无效果字体	{要显示的文字}	<b>text</b>	L <sup>A</sup> T <sub>E</sub> X
<code>\textit</code>	意大利斜体字体	{要显示的文字}	<i>text</i>	L <sup>A</sup> T <sub>E</sub> X
<code>\textsl</code>	Slanted 斜体字体	{要显示的文字}	<i>text</i>	L <sup>A</sup> T <sub>E</sub> X
<code>\textsc</code>	小写转大写后输出	{要显示的文字}	TEXT	L <sup>A</sup> T <sub>E</sub> X
<code>\textrm</code>	罗马字体族	{要显示的文字}	text	L <sup>A</sup> T <sub>E</sub> X
<code>\textsf</code>	无衬字体	{要显示的文字}	text	L <sup>A</sup> T <sub>E</sub> X
<code>\texttt</code>	打印字体	{要显示的文字}	text	L <sup>A</sup> T <sub>E</sub> X
<code>\newfontfamily</code>	指定字体格式	{控制序列}{字体名}	N/A	fontspec
<code>\setCJKfamilyfont</code>	导入中日韩 (CJK) 字体，赋予一个别名	{别名}{字体名}	N/A	ctex
<code>\CJKfamily</code>	设置中日韩 (CJK) 语言环境字体	{字体名称}	文字	ctex
<code>\textbf</code>	文字加粗	{要显示的文字}	<b>text</b> 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\underline</code>	添加下划线	{要显示的文字}	<u>text</u> 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\textcolor</code>	彩色文字	[色彩空间]{浮点数色彩值}{要显示的文字}	text 文字	xcolor
<code>\tiny</code>	最小的预设字号	N/A	text 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\footnotesize</code>	脚注标准字号	N/A	text 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\small</code>	小号字体	N/A	text 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\normalsize</code>	默认字号	N/A	text 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\large</code>	较大的字号	N/A	test 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\Large</code>	介于 large 与 LARGE 间的字号	N/A	text 文字	L <sup>A</sup> T <sub>E</sub> X



<code>\LARGE</code>	介于 Large 与 huge 间的字号	N/A	text 文 字	L <sup>A</sup> T <sub>E</sub> X
<code>\huge</code>	介于 LARGE 与 Huge 间的字号	N/A	text 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\Huge</code>	最大的预设字号	N/A	text 文字	L <sup>A</sup> T <sub>E</sub> X
<code>\zihao</code>	中国人习惯的汉字字号	{字号}	三号字	L <sup>A</sup> T <sub>E</sub> X

表 3.1: 文本文字样式控制序列

补充说明几点:

- `\newfontfamily` 本身并不直接参与文本生成。不难注意到该控制序列的第一个参数也是一个控制序列, 我们使用它即可将新控制序列绑定的字体应用到跟随其的文本。您可能已经注意到, 表格中有一个“N/A”项的字体与其他的不太一样。
- 读者或许会发现, `color` 包也能使用 `\textcolor` 宏; 本文中引用的许多包彼此之间具有包含关系, 笔者只在正文中列出笔者最常用的包, 其中有些并不是其原始来源。此外, 对照源码不难发现: 应用字体的控制序列只会影响其后附近的文字, 而不能改变全局字体设置。实际上, 如果把字体设置控制序列放在正文最外层, 其造成的字体改变确实是全局的, 直到重新指定另一个字体; 这里不能影响全局是因为我们是在单元格环境内进行的设置, 离开了这个单元格, 单元格内的设置就会失效。

**注意** 别忘了, 多数字体需要先通过控制序列引入才能正常使用。此外, 有些字体还需要 `fontenc` 包。

### 3.4. 层次结构和标题

L<sup>A</sup>T<sub>E</sub>X 使用的层次结构可以十分多样。不同文档类型常见的搭配也不同, 比如 `article` 类型经常搭配 `part` 而不是 `chapter`, 但无论是哪一个, 其本质是大同小异的。L<sup>A</sup>T<sub>E</sub>X 下任何段落层次结构都以类似方式声明, 即“\结构类型名称 {标题}”的形式。本文仅介绍 `article` 类型, 其他很多类型大致相似, 但一般都有一些特有控制序列。

在 `article` 文档类型下, 常见的结构类型有 `part`、`section`、`subsection`、`subsubsection`。使用 `\section{标题}` 就可以生成一个以“标题”为标题的小节。如果加一个星号即 `\section*{标题}`, 则生成的小节不会带有编号, 多数目录工具也会在目录中跳过不带编号的小节。其他层次也都有类似的特性。不同的是, `part` 是在全局上独立编号的, `section` 在每个 `part` 内独立编号, `subsection` 跟随 `section` 编号, `subsubsection` 跟随 `subsection` 编号。例如, `part 2`, `section 3`, `subsection 4` 的编号默认会显示为 3.4。本文由于采用了一些特殊设定, `section` 跟随 `part` 编号并显示, 所以会显示为 2.3.4。





### 3.4.1. 标题样式

#### 3.4.1.1. 编号数字样式

L<sup>A</sup>T<sub>E</sub>X 支持显示不同形式的数字，包括大小写字母、大小写罗马数字和阿拉伯数字。如果引用了 `ctex` 宏包，还可以显示汉语数字、大写汉语数字、天干地支等。

设置编号数字的样式非常简单，只要使用 `\renewcommand` 重新定义计数器宏即可。重定义宏的内容通常是 `\格式控制序列 {计数器名称}` 的形式，默认的、有关序号格式的控制序列有：

宏	内容
<code>\alph</code>	小写字母
<code>\Alph</code>	大写字母
<code>\arabic</code>	阿拉伯字母
<code>\roman</code>	小写罗马数字
<code>\Roman</code>	大写罗马数字

### 3.4.2. 改变标题格式

本文介绍的改变标题格式的方法以 `titlesec` 的 `\titleformat` 为例，这是最常见的方案。

考虑到引用位置的需求，各级标题中最重要的部分就是编号。这里我们先明确计数器的概念。L<sup>A</sup>T<sub>E</sub>X 中计数器一般指的是 `part`、`section` 等默认计数器（你也可以生成自己的计数器，但本文不讨论这点），它们表示的就是对应结构的当前计数值。例如，本小节当前的计数器（`subsection`）值为 **3.4.2**。你也可以自行修改计数器的值，但本文也不讨论这点。

L<sup>A</sup>T<sub>E</sub>X 下，很多计数器具有一个以 `the` 为前缀的控制序列，用于控制编号输出样式。例如，`\thepart` 表示当前 `part` 序号。本小节的位置可以用 `\thesubsection` 表示<sup>1</sup>。

宏	在该文档本位置的值
<code>\thepart</code>	III
<code>\thesection</code>	3.4
<code>\thesubsection</code>	3.4.2

这里特别强调一点：计数器名称不带反斜杠（`\`），但是对应的宏是有的。L<sup>A</sup>T<sub>E</sub>X 的一切宏都以反斜杠开头以标识出它是个宏。除了特定宏指令参数外，`part` 都不会被当作计数器名称，而是普通的纯文本（`plain text`）。

计数器对应宏的显示样式可以用 `\renewcommand{样式控制序列}` 来改变。如果你希望使用罗马数字作为 `section` 序号，那么可以这样做：

```
\renewcommand{\Roman}{section}}
```

<sup>1</sup> 正在对照源码的读者可能注意到了上文使用了 `\ref`。`\ref` 是用于全文引用某位置的语法，但是 `\thesubsection` 永远只能指示当前位置。这就是二者的区别。



## 第 IV 部分 附录

### 附录 A 常见问题 (FAQ)

问 L<sup>A</sup>T<sub>E</sub>X 到底如何发音?

答 这要追溯到它的词源。T<sub>E</sub>X 一词来源于  $\tau\epsilon\chi$ , 这是一个希腊语中的词根, 与科学和艺术有关, 发音类似 technology 中的 tech 音节 (事实上 tech 这个词根也与这一希腊语词根有关), 可读作 [tek] 或 [tex] (注意不能读作 [teks])。前缀 la-则发 [leɪ] 或 [lɑː]。合起来, 则 L<sup>A</sup>T<sub>E</sub>X 的发音接近于 lay-tek 或 lay-tech。

问 为什么我无法像本文源码中那样使用 \addbs 等宏?

答 源码中前导区内以 \newcommand 声明的宏都不是自带宏。如果您也希望使用类似宏, 需要自己加入自己的文档中。本文中未在前导区以 \newcommand 声明的宏都是文档类型内置宏或来自宏包, 如果是其他宏无法正确编译, 那很可能是宏包安装不正确、文档类型不匹配或宏包引用不正确。

问 如何安装宏包?

答 不同的 L<sup>A</sup>T<sub>E</sub>X 发行版的具体操作不一样, 请查阅您所使用发行版的文档或咨询技术人员。本文使用的 MiK<sub>T</sub>E<sub>X</sub> 则附带包管理器, 遇到未安装的包时可能会自动安装它们或者向您索要许可, 取决于您如何安装 MiK<sub>T</sub>E<sub>X</sub>。

问 为什么生成文档没有乱码, 源码却无法正确显示?

答 本文源码采用 UTF-8 编码, 笔者建议读者们也尽可能采取 UTF-8 编码。Windows 平台所广泛采用的 GBK2312 编码实际上并不是很常见。如果出现了用自己的软件打开 pdf 文件没有乱码, 打开源码却是乱码, 那么很有可能是编码问题, 使用专门工具将源码从 UTF-8 转到你的平台所使用的编码就可以了。

问 为什么我的文本编辑器打开源码后, 产生了多于预期的空行?

答 如果“多余”的空行每处只有一行, 那么其实是正常的。L<sup>A</sup>T<sub>E</sub>X 语法中, 源码需要换两行 (留一行空行) 才会在生成的 pdf 文档里换行。但如果你的文本编辑器显示了每处两个空行, 那就是不太正常的现象, 但也不必过于在意; 这可能是因为源码是 CRLF 格式换行, 但是你的文本编辑器是 LF 换行。

问 为什么我的 L<sup>A</sup>T<sub>E</sub>X 报告错误“无法使用 pdfL<sup>A</sup>T<sub>E</sub>X 编译”?

答 有些宏包不提供对 pdfL<sup>A</sup>T<sub>E</sub>X 的支持, 所以会人为引入这个错误。强行使用 pdfL<sup>A</sup>T<sub>E</sub>X 编译只会让情况更糟, 我们建议您尝试改用 X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X 并尽可能避免使用和 X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X 不兼容的宏包。

问 源码原本应当是引用的部分, 为何出现问号?

答 \ref 引用的部分没有对应的 \label 就会引发这个现象。检查你的 include 结构关系是否正确, 有无遗漏, 以及引用时有没有出现 typo。





问 我在尝试编译这份文档的源码，为什么我的编译器报告找不到图片？

答 该文档原文确实包含了一些图片，放在与 L<sup>A</sup>T<sub>E</sub>X 源码同一目录下的 img 文件夹。如果您编译该文档的位置没有 img 文件夹或 img 文件夹内缺乏应有的文件，就会出现这个问题。

问 我在尝试使用 CJK 搭配 pdfL<sup>A</sup>T<sub>E</sub>X 输出中文页眉并在前导区设置了标题格式，为什么没有效果？

答 如果您确定要使用 pdfL<sup>A</sup>T<sub>E</sub>X，请在全文第一次使用中文之前进行 CJK 环境声明。如果前导区也有中文输出，那么只需要在前导区第一次出现中文字符之前声明一个空的 CJK 环境即可。

问 我的编译器报告“无法打开文件”，这是为什么？

答 有两种可能，其一是没有文件写入权限，其二是文件已被占用打开。前者可以通过文件权限标志来判断，此外您也需要判断文件位置是不是加了写保护。确定不是没有写入权限的话就基本可以确定是另一个进程已经以占用方式打开了该文件，只要把占用者关掉就可以了。通常而言，占用者主要是各类阅读器，比如 Adobe Reader。

问 我的 L<sup>A</sup>T<sub>E</sub>X 编译器报告的错误所在行数比我的文档总行数还多，是为什么？

答 很可能是宏包内部问题，尝试调换宏包包含顺序，或者再编译一次，有时可以解决问题。如果还不行，有可能是宏包彼此之间的兼容性问题，也可能是宏包和 L<sup>A</sup>T<sub>E</sub>X 发行版不兼容。遇到兼容性问题是很难自己解决的，请咨询宏包发布者。

问 L<sup>A</sup>T<sub>E</sub>X 经常报出晦涩难懂的错误报告，是我的问题吗？

答 一般而言，出错是用户的问题，但晦涩难懂的错误报告不是。L<sup>A</sup>T<sub>E</sub>X 是基于宏的语言，相信计算机工程师们都知道一件事情，就是但凡有什么东西跟宏沾上了关系，它的底层原理就会变得极度复杂。L<sup>A</sup>T<sub>E</sub>X 形式上的简便易用正是建立在底层纷繁复杂的宏上面的，这意味着一旦您写的 L<sup>A</sup>T<sub>E</sub>X 文档出现语法问题，L<sup>A</sup>T<sub>E</sub>X 会在宏展开到底层时才能真正发现错误，正是这点导致了 L<sup>A</sup>T<sub>E</sub>X 的编译报错信息往往极其晦涩难懂。如果您无法接受这种违反人类习惯的报错信息，您可以去尝试阅读和书写 C++ 的宏和模板，尤其是 STL 代码和 Boost 库代码。那样您就会明白，L<sup>A</sup>T<sub>E</sub>X 报错的晦涩程度在计算机工程领域真的不值一提（笑）。当然这不是在批评 C++，为了让最终用户得到方便，上层开发者不得不写很多连同代码和错误报告（如果最终用户的使用方式有误的话）都极其晦涩难懂的代码，这就是一门语言为了得到通用性和易用性不得不付出的代价，也就是极其难以读懂的底层错误报告。其实耐着性子读完 L<sup>A</sup>T<sub>E</sub>X 发行版给出的错误报告的话，揪出实际出错位置还是很容易的。说句题外话，C++ 不同编译器提供的错误报告可读性也不同，受不了 GCC 的 yacc 技术提供的报告的话不妨尝试 Clang/LLVM。

问 L<sup>A</sup>T<sub>E</sub>X 报告了大量警告，有没有必要一个一个消除掉？

答 我们建议您尽可能避免任何警告。然而作为一个排版软件，仅仅是字体问题就足够 L<sup>A</sup>T<sub>E</sub>X 报告上百个警告。只要您对最终输出的效果还算满意，那么关于字体的警告也并非十分重要。尽管如此，我们建议您使用专业工具以确定您的 pdf 是否真的使用了您所要求的字体，而不是某些替代品。Underfull \hbox 是另一个常见警告，通常是编译器无法判断合适的换行点造成的，如果您认为当前排版结果合适就可以忽视它。



问 如何给我的 pdf 嵌入字体？

答 通常而言嵌入字体不是 L<sup>A</sup>T<sub>E</sub>X 的工作，它只负责输出一份文档。如果您希望嵌入所使用的字体，一般的解决方案是使用第三方专业工具。如果您使用的字体不是太偏门，那么实际上 MiK<sub>T</sub>E<sub>X</sub> 就会帮您嵌入多数字体。

问 我对计算机工程并不感兴趣，L<sup>A</sup>T<sub>E</sub>X 适用于我吗？

答 只要您不怕麻烦并且追求比 Word 更好的用户体验，那么无论您的工作方向是什么，L<sup>A</sup>T<sub>E</sub>X 都很适合。文学作品的排版，算法著作，化学实验报告，物理学论文，甚至是漫画书，都可以用 L<sup>A</sup>T<sub>E</sub>X 来排版。本文主要面向计算机工程领域进行介绍，但这不代表计算机工程领域是 L<sup>A</sup>T<sub>E</sub>X 的唯一方向。

问 我可以用 L<sup>A</sup>T<sub>E</sub>X 来写作业或论文吗？

答 一般而言当然可以，不过具体看贵单位的要求如何。有些机构会强制要求你使用 L<sup>A</sup>T<sub>E</sub>X 的模板撰写论文，毕竟 L<sup>A</sup>T<sub>E</sub>X 比 Word 更加容易确定格式。有些学校也允许用 L<sup>A</sup>T<sub>E</sub>X 写论文，相信用户很容易体会到，用 L<sup>A</sup>T<sub>E</sub>X 写论文远比用 Word 方便，除非你从来没用过 L<sup>A</sup>T<sub>E</sub>X。笔者个人强烈建议每所学校都引入 L<sup>A</sup>T<sub>E</sub>X 作为标准文档工具。

问 作为个人开发者，我如何确定素材的法律风险？

答 一般而言，我们推荐使用有协议可查的素材，无论是图片、字体、代码、商标还是文字作品。这几类都是常见的高民事诉讼风险素材。对于各类素材，尽量寻找明确标注版权的网站，以及个人或组织的官方网站并确定作品应当如何被使用，同时尽可能标注来源以降低受到诉讼的风险。有些正版软件附带一些素材版权，如果您在使用这些软件，就可以在其中免费使用相应素材。（请注意：您购买了附带素材 A 版权的正版软件 B，不代表您在同一台机器上的软件 C 也能合法使用素材 A。）多数情况下，素材会有附带的使用条款或协议，我们建议您在发布前仔细审阅这些法律文件（特别地，不对第三方发布的文件一般也没有法律风险）；应当特别提醒的是，不同语言的翻译版本含义出现冲突时应以原文为准，因此我们认为有必要完整阅读文件原文，而不是翻译。有些协议尽管法律效力不明确，如尚无相关司法解释和判例的，我们也仍然不建议违反它们，除非您不介意自己和自己的产品被作为反面教材被挂起来批评并被各大主流厂商排斥。最好的选择是只使用开放的材料和软件以规避法律风险，如果一定要使用法律风险高的素材，我们建议您在发布前仔细审阅相关法律文件，并在文件上随附声明。用于商业用途则务必咨询律师，除非您想和大厂商的法务部门来场法庭辩论。