始终

不忘初心

一份其实很短的 LaTeX 入门文档

2014 年 09 月 08 日 | 2019 年 11 月 10 日 | LaTeX | 410194

19k | 35 分钟

优雅的 LaTeX

有很多 Geeks 或者 LaTeX's Fanatical Fans 过分地强调了 LaTeX 的一些并非重点的特性，以至于很多初学者会觉得 LaTeX 很神秘很复杂，从而引发了初学者的畏难情绪甚至是负面情绪。尽管这些 Fans 说得并没有错，我是说在事实上，但是他们的表达方式和内心态度却间接阻碍了 LaTeX 的发展，我想这也是和他们的初衷相悖的。

我曾经也受到过这些言论的影响，但幸运的事，至今为止我已经越过了这些障碍，并更加坚信了他们所言（LaTeX 的优点）的正确性。因此我想以我自己的方式，向更多的人介绍 LaTeX —— **这个优雅，但有着自己高傲，却绝不复杂甚至神秘的东西。**

> 欢迎从校内转移过来的同学们，因为校内的技术原因，文章无法在校内继续更新。尽管非我之故，但还是给各位同学说声抱歉。同时，欢迎各位来到我的博客。：）

你将从这里看到

1. （几乎是）最简洁的 (La)TeX 编辑器——TeXworks——的基本使用方法；
2. 如何使用 (Xe)LaTeX 进行简单的**中英混排**；
3. 简单的文章组织结构；

4. 如何用 (Xe)LaTeX 进行数学公式的排版；
5. 如何在 (Xe)LaTeX 的文档中插入图片/表格；
6. 如何进行简单的版面设置；
7. 几个最常见的带有 TeX 的单词的含义；
8. 出现问题应当如何处理/怎样聪明地提出你的问题——怎样从这里毕业。

你不会从这里看到

1. 如何安装 TeX 发行；
2. 特殊需求（特殊宏包）的解决方法；
3. 关于若干 TeX 发行的好坏比较；
4. 关于各种 TeX 编辑器的好坏比较；
5. 过多的废话。

你应当如何阅读本文

事实上本文在行文过程中，会有相当多的提示帮助你以正确的方式阅读。因此有必要在此先介绍一下最常用的一些标记。

1. 斜体：使用*斜体* 意味着如果忽略掉这些文字，你可能在*逻辑* 上很难理解后面某处的知识；
2. 粗体：使用**粗体**意味如果忽略掉这些文字，你可能在**TeX 的概念**上很难理解后面某处的知识；
3. 粗斜体：使用***粗斜体*** 基本是最重要的部分，是上述两种情况的合并；
4. 引用：使用引用，

   > 表明这些文字在你第一次阅读本文的时候*不需要* 了解，其中的内容可能过于深奥，或者过于琐碎。对于第一次接触 TeX 的你（如果你是 TeX 资深使用者当然不在此列），如果了解到这些内容可能会使你困惑，并且不会从实际上增加你对 TeX 的领悟以及对 TeX 的好感。

关于编辑器的简单介绍

TeX 的源代码是后缀为 `.tex` 的纯文本文件。使用任意纯文本编辑器，都可以修改 `.tex` 文件：包括 Windows 自带的记事本程序，也包括专为 TeX 设计的编辑器（TeXworks，TeXmaker，TeXstudio，WinEdt 等），还包括一些通用的文本编辑器（Sublime Text, Atom, Visual Studio Code 等）。你可以在这些能够编辑纯文本文件的编辑器中任选其一作为你的 TeX 编辑器，也可以使用 TeX 发行自带的编辑器。最流行的两个 TeX 发行（TeX Live 和 MiKTeX）都带有 TeXworks 编辑器。

> 所谓 TeX 发行，也叫 TeX 发行版、TeX 系统或者 TeX 套装，指的是包括 TeX 系统的各种可执行程序，以及他们执行时需要的一些辅助程序和宏包文档的集合。

本文只介绍 TeXworks 的使用，原因有以下一些：

- TeXworks 是 TeX Live 自带的编辑器，而 TeX Live 是 TeX User Group 出品的跨平台发行版，各个操作系统都可以使用；
- 几乎所有 TeX 发行版都带有 TeXworks；
- TeXworks 十分简洁，除了最基本的功能之外，没有其他复杂的东西，能使你将注意力集中在 TeX 的学习上。

启动 TeXworks

启动 TeXworks 很简单，你可以在 Windows 启动对话框中输入 texworks 按回车。具体步骤是：

- 按下键盘上的 Windows 徽标键，同时按下 R 键 —— `<Win> + R`；
- 键入 `texworks`；
- 回车。

如果这样打不开 TeXworks，你可能需要从开始菜单找到 TeXworks 图标以启动；或者进

入 TeX 系统的安装目录找到 TeXworks。

启动之后，TeXworks 的界面，会默认占据你屏幕的左半边，右半边留空。效果如下图：

图中空白的部分，就是输入编辑文本的编辑框；在编辑框的右下角，显示有三个按钮，最左边的是换行符模式，中间是*编码模式*，右边标示当前光标所在位置；编辑框的上方是工具栏，工具栏的右半部分使大家熟悉的功能（新建、打开、保存、撤消、恢复、剪切、复制、黏贴、查找、替换），工具栏的左边则是*编译按钮*（TeXworks 也称其为「排版工具」）；工具栏在往上，则是菜单栏，此处按下不表。

> Windows，Unix 等操作系统对待「换行符」是有不同的。索性 TeXworks 为我们做了足够的提示，方面我们的选择（点击一下那个按钮就知道了）。一般而言，保持默认即可。
>
> 字符（包括英文字符和中文字符）在计算机中，经过编码以二进制的形式存储在计算机中。如果编辑器编码和计算机内部编码不一致，则会导致所谓「乱码」的现象。TeXworks 默认使用 UTF8 编码，在我们的文档中不需要进行任何更改，而对于一些其他的文档可能需要按照要求更改编码。

排版工具

TeXworks 为我们预设了若干排版工具（pdfTeX, pdfLaTeX, XeTeX, XeLaTeX 等），本文主要用到其中的 ***XeLaTeX***。关于这些排版工具的细节，讲解起来会有些复杂。因此此处按下不表，若有兴趣，可以参看后文。当你对 TeX 系统相当熟悉之后，也可以不使用 TeXworks 预设的工具，自己配置排版工具。

TeXworks 默认的排版工具是 pdfLaTeX。如果你希望更改这个默认值，可以在*编辑 - 首选项 - 排版 - 处理工具 - 默认* 中修改。

第一篇文档

Hello, world!

在编辑框中，输入如下内容：

helloworld.tex

```
\documentclass{article}
% 这里是导言区
\begin{document}
Hello, world!
\end{document}
```

将文档保存在你希望的位置，然后在排版工具的下拉选框中选中 **XeLaTeX** 后，按下绿色的编译按钮。一会儿，如果没有意外，屏幕的右边就会出现编译之后结果。如下图：

> 请注意，由于操作系统编码和 TeX 内部实现的限制，在 Windows 平台上，TeX 涉及到的文件（包括 `.tex`，`.jpg` 等各类文件）都不要包含中文名字。否则，在编译时可能会因为编码问题导致稀奇古怪的报错。

很容易发现，输入进编辑框的五行文字，在最终输出的 pdf 档中只显示了 1 行。事实上，交付 TeX 处理的文档内容，并不会全部输出。

此处的第一行 `\documentclass{article}` 中包含了一个控制序列（或称命令/标记）。所谓控制序列，是以反斜杠 `\` 开头，以第一个***空格或非字母***的字符结束的一串文字。它们不被输出，但是他们会影响输出文档的效果。这里的控制序列是 `documentclass`，它后面紧跟着的 `{article}` 代表这个控制序列有一个必要的参数，该参数的值为 `article`。这个控制序列的作用，是调用名为 `article` 的文档类。

请注意，TeX 对控制序列的大小写是敏感的。

> 部分控制序列还有被方括号 `[]` 包括的可选参数。
>
> 所谓文档类，即是 TeX 系统预设的（或是用户自定的）一些格式的集合。不同的文档类在输出效果上会有差别。

此处的第二行以 `%` 开头。TeX 以百分号 `%` 作为注释标记。具体来说，TeX 会忽略从 `%` 开始到当前行末尾的所有内容。这些内容不会被输出，也不影响最终排版效果，只供人类阅读。若要输出 `%` 字符本身，则需要在 `%` 之前加上反斜杠 `\` 进行转义（escape）。例如：

```
1    今年的净利润为 20\%，比去年高。
```

此处 `%` 被当做正常的百分号处理，其后的文字也将被正常输出。

我们继续分析第一篇文档的内容。在注释行之后出现了控制序列 `begin`。这个控制序列总是与 `end` 成对出现。这两个控制序列以及他们中间的内容被称为「环境」；它们之后的第一个必要参数总是**一致的**，被称为环境名。

只有在 `document` 环境中的内容，才会被正常输出到文档中去或是作为控制序列对文档产生影响。也就是说，在 `\end{document}` 之后插入任何内容都是无效的。

从 `\documentclass{article}` 开始到 `\begin{document}` 之前的部分被称为导言区。你可以将导言区理解为是对整篇文档进行设置的区域——在导言区出现的控制序列，往往会影响整篇文档的格式。

> 比如，我们通常在导言区设置页面大小、页眉页脚样式、章节标题样式等等。

实现中英文混排

> 关于 LaTeX 的中文支持，首先要说的是：在现在，一切教你使用 `CJK` 宏包的模板、人、网页、书，都是糟糕的、有害的、恼人的、邪恶的和应该摒弃的。

成功编译输出第一个文档之后，中国 TeX 用户面临的第二个普遍问题大概就是「实现中英文混排」了。众所周知，TeX 系统是高教授开发的。在 TeX 开发当初并没有考虑到亚洲文字的问题。因此早期的 TeX 系统并不能直接支持中文，必须要用其他工具先处理一下（或者是一些宏包之类的）。但是现在，XeTeX 原生支持 Unicode，并且可以方便地调用系统字体。可以说解决了困扰中国 TeX 使用者多年的大问题。至此，我们只需要使用几个简单的宏包，就能完成中文支持了。

所谓宏包，就是一系列控制序列的合集。这些控制序列太常用，以至于人们会觉得每次将他们写在导言区太过繁琐，于是将他们打包放在同一个文件中，成为所谓的宏包（台湾方面称之为「巨集套件」）。`\usepackage{}` 可以用来调用宏包。

除去中文支持，中文的版式处理和标点禁则也是不小的挑战。好在由吴凌云和江疆牵头，现在主要由刘海洋、李清和我维护的 `CTeX` 宏集一次性解决了这些问题。`CTeX` 宏集的优势在于，它能适配于多种编译方式；在内部处理好了中文和中文版式的支持，隐藏了这些细节；并且，提供了不少中文用户需要的功能接口。我们来看如何使用 `CTeX` 宏集来处理中英文混排的文档。

> 请注意，`CTeX` 宏集和 `CTeX` 套装是两个不同的东西。`CTeX` 宏集本质是 LaTeX 宏的集合，包含若干文档类（`.cls` 文件）和宏包（`.sty` 文件）。`CTeX` 套装是一个**过时的** TeX 系统。
>
> 新版 `CTeX` 宏集的默认能够自动检测用户的操作系统，并为之配置合适的字库。对于 Windows 用户、Mac OS X 用户和 Linux 用户，都无需做任何配置，就能使用 `CTeX` 宏集来排版中文。[2015-05-20 更新]

在 TeXworks 编辑框中输入以下内容，**以 UTF-8 编码保存，使用 XeLaTeX 编译**：

EC-mix.tex

```
\documentclass[UTF8]{ctexart}
\begin{document}
你好，world!
\end{document}
```

如果没有意外，你将会看到类似下图的输出结果。

相较于之前的例子，这份代码只有细微的差异：

1. 文档类从 `article` 变为 `ctexart`；
2. 增加了文档类选项 `UTF8`。

> 你也可以直接使用 `xeCJK` 宏包来支持中英文混排。不过大多数情况是不推荐这样做的。因此，如果你能抑制住你小小的好奇心，可以暂时略过这一段，回头再看不迟。 :)
>
> 在 TeXworks 编辑框中输入以下内容，保存，使用 XeLaTeX 编译：
>
> xeCJK-demo.tex
>
> ```
> \documentclass{article}
> \usepackage{xeCJK} %调用 xeCJK 宏包
> \setCJKmainfont{SimSun} %设置 CJK 主字体为 SimSun （宋体）
> \begin{document}
> 你好，world!
> \end{document}
> ```
>
> 如果一切顺利，你将会在屏幕右边的窗口，看见类似下图的输出结果。
>
>
>
> 除了 "document" 环境中同时出现了中文和英文，和最原始的 Hello, world! 不同点在于，导言区中多出了两条控制序列。他们的作用我已经用注释标出了。

`\setCJKmainfont{·}` 是定义在 "xeCJK" 宏包中的控制序列，它可以用来设置 CJK 主字体。

如果你的 TeX 系统提示找不到字体，请按以下提示操作。

Mac OS X 用户请参照这篇博客中的方法，使用系统自带的字体册程序来查看系统字体。

非 Mac OS X 用户请按照如下步骤打开系统命令行（*nix系统请打开终端）：

- 按下 `<win> + R`；
- 键入 `cmd`，回车。

在系统命令行中输入如下命令：

```
fc-list :lang=zh-cn > C:\font_zh-cn.txt
```

（相信使用 *nix 的你，一定知道如何修改上述命令达到你想要的效果）

打开 C 盘根目录下的 `C:\font_zh-cn.txt` 纯文本文档，里面的内容就是你当前系统可用的全部中文字体，形如：

每一个小黑框之间的内容，就对应着一个可用的字体。这些小黑框实际上是换行符，但是由于 Windows 系统的原因，他们没有被正常显示。如果看着不爽，你可以尝试用 TeXworks 打开这个文件查看（放心，能够打开的）。以下是我用 gVim 打开的效果：

其中的每一行，都代表着一个可用的字体。其形式如下：

```
<字体文件路径>: <字体标示名1>, <字体表示名2>:Style=<字体类型>
```

我们可以看到图中的倒数第四行

```
C:/WINDOWS/fonts/simsun.ttc: 宋体,SimSun:style=Regular
```

出现了之前文档里调用的字体 SimSun，此处表明该字体有两个表示名：宋体 和 SimSun，我们在 \setCJKmainfont{·} 中填入任意一个都有同样的效果。

因此，如果之前的文档无法编译通过，请在你的操作系统字体中，选取一个自己喜欢的，将它的字体表示名填入到 \setCJKmainfont{·} 中去。

组织你的文章

作者、标题、日期

保存并用 XeLaTeX 编译如下文档，查看效果：

```
\documentclass[UTF8]{ctexart}
\title{你好，world!}
\author{Liam}
\date{\today}
\begin{document}
\maketitle
你好，world!
\end{document}
```

导言区复杂了很多，但和之前的文档主要的区别只有一处：定义了标题、作者、日期。

在 document 环境中，除了原本的你好，world!，还多了一个控制序列 maketitle。这个控制序列能将在导言区中定义的标题、作者、日期按照预定的格式展

现出来。

> 使用 titling 宏包可以修改上述默认格式。参考TeXdoc。

章节和段落

保存并用 XeLaTeX 编译如下文档，查看效果：

```
\documentclass[UTF8]{ctexart}
\title{你好，world!}
\author{Liam}
\date{\today}
\begin{document}
\maketitle
\section{你好中国}
中国在East Asia.
\subsection{Hello Beijing}
北京是capital of China.
\subsubsection{Hello Dongcheng District}
\paragraph{Tian'anmen Square}
is in the center of Beijing
\subparagraph{Chairman Mao}
is in the center of 天安门广场。
\subsection{Hello 山东}
\paragraph{山东大学} is one of the best university in 山东。
\end{document}
```

在文档类 article / ctexart 中，定义了五个控制序列来调整行文组织结构。他们分别是

- \section{·}
- \subsection{·}
- \subsubsection{·}
- \paragraph{·}

- `\subparagraph{·}`

> 在 report / ctexrep 中，还有 `\chapter{·}`；在文档类 book / ctexbook 中，还定义了 `\part{·}`。

插入目录

在上一节的文档中，找到 `\maketitle`，在它的下面插入控制序列 `\tableofcontents`，保存并用 XeLaTeX 编译**两次**，观察效果：

```
\documentclass[UTF8]{ctexart}
\title{你好，world!}
\author{Liam}
\date{\today}
\begin{document}
\maketitle
\tableofcontents
\section{你好中国}
中国在East Asia.
\subsection{Hello Beijing}
北京是capital of China.
\subsubsection{Hello Dongcheng District}
\paragraph{Tian'anmen Square}
is in the center of Beijing
\subparagraph{Chairman Mao}
is in the center of 天安门广场。
\subsection{Hello 山东}
\paragraph{山东大学} is one of the best university in 山东。
\end{document}
```

> 试试交换 `\maketitle` 和 `\tableofcontents` 的顺序，看看会发生什么，想想为什么。

请注意，在「你好中国」这一节中，两次「中国在East Asia.」中夹有一个空行，但输出

却只有一个换行并没有空行。这是因为 LaTeX 将一个换行当做是一个简单的空格来处理，如果需要换行另起一段，则需要用两个换行（一个空行）来实现。

插入数学公式

首先恭喜你看到这里。如果前面的几个文档你都认真编译过了，那么你已经可以胜任许多文档的排版工作。下面我们进入 LaTeX 最为犀利的部分。

这部分的演示中，为了节省篇幅，将取消导言区中中文支持的部分。在实际使用中，你只需要将导言区中的相关部分加上，就可以同时使用中文和编写数学公式了。

为了使用 AMS-LaTeX 提供的数学功能，我们需要在导言区加载 `amsmath` 宏包：

```
\usepackage{amsmath}
```

数学模式

LaTeX 的数学模式有两种：行内模式 (inline) 和行间模式 (display)。前者在正文的行文中，插入数学公式；后者独立排列单独成行，并自动居中。

在行文中，使用 `$ ... $` 可以插入行内公式，使用 `\[ ... \]` 可以插入行间公式，如果需要对行间公式进行编号，则可以使用 `equation` 环境：

```
\begin{equation}
...
\end{equation}
```

> 行内公式也可以使用 `\(...\)` 或者 `\begin{math} ... \end{math}` 来插入，但略显麻烦。
> 无编号的行间公式也可以使用 `\begin{displaymath} ... \end{displaymath}`

> 或者 `\begin{equation*} ... \end{equation*}` 来插入，但略显麻烦。（`equation*` 中的 `*` 表示环境不编号）
>
> 也有 plainTeX 风格的 `$$ ... $$` 来插入不编号的行间公式。但是在 LaTeX 中这样做会改变行文的默认行间距，不推荐。请参考我的回答。

上下标

示例代码（请保存后，使用 XeLaTeX 编译，查看效果）：

```
\documentclass{article}
\usepackage{amsmath}
\begin{document}
Einstein 's $E=mc^2$.

\[ E=mc^2. \]

\begin{equation}
E=mc^2.
\end{equation}
\end{document}
```

在这里提一下关于公式标点使用的规范。行内公式和行间公式对标点的要求是不同的：行内公式的标点，应该放在数学模式的限定符之外，而行间公式则应该放在数学模式限定符之内。

在数学模式中，需要表示上标，可以使用 `^` 来实现（下标则是 `_`）。**它默认只作用于之后的一个字符**，如果想对连续的几个字符起作用，请将这些字符用花括号 `{}` 括起来，例如：

```
\[ z = r\cdot e^{2\pi i}. \]
```

根式与分式

根式用 `\sqrt{·}` 来表示，分式用 `\frac{·}{·}` 来表示（第一个参数为分子，第二个为分母）。

示例代码（请保存后，使用 XeLaTeX 编译，查看效果）：

```
\documentclass{article}
\usepackage{amsmath}
\begin{document}
$\sqrt{x}$, $\frac{1}{2}$.

\[ \sqrt{x}, \]

\[ \frac{1}{2}. \]
\end{document}
```

可以发现，在行间公式和行内公式中，分式的输出效果是有差异的。如果要强制行内模式的分式显示为行间模式的大小，可以使用 `\dfrac`，反之可以使用 `\tfrac`。

> 在行内写分式，你可能会喜欢 `xfrac` 宏包提供的 `\sfrac` 命令的效果。

> 排版繁分式，你应该使用 `\cfrac` 命令。

运算符

一些小的运算符，可以在数学模式下直接输入；另一些需要用控制序列生成，如

```
\[ \pm\; \times \; \div\; \cdot\; \cap\; \cup\;
\geq\; \leq\; \neq\; \approx \; \equiv \]
```

连加、连乘、极限、积分等大型运算符分别用 `\sum`，`\prod`，`\lim`，`\int` 生成。他们

的上下标在行内公式中被压缩，以适应行高。我们可以用 `\limits` 和 `\nolimits` 来强制显式地指定是否压缩这些上下标。例如：

```
$ \sum_{i=1}^n i\quad \prod_{i=1}^n $
$ \sum\limits _{i=1}^n i\quad \prod\limits _{i=1}^n $
\[ \lim_{x\to0}x^2 \quad \int_a^b x^2 dx \]
\[ \lim\nolimits _{x\to0}x^2\quad \int\nolimits_a^b x^2 dx \]
```

多重积分可以使用 `\iint`，`\iiint`，`\iiiint`，`\idotsint` 等命令输入。

```
\[ \iint\quad \iiint\quad \iiiint\quad \idotsint \]
```

定界符（括号等）

各种括号用 `()`，`[]`，`\{\}`，`\langle\rangle` 等命令表示；注意花括号通常用来输入命令和环境的参数，所以在数学公式中它们前面要加 `\`。因为 LaTeX 中 `|` 和 `\|` 的应用过于随意，amsmath 宏包推荐用 `\lvert\rvert` 和 `\lVert\rVert` 取而代之。

为了调整这些定界符的大小，amsmath 宏包推荐使用 `\big`，`\Big`，`\bigg`，`\Bigg` 等一系列命令放在上述括号前面调整大小。

> 有时你可能会觉得 amsmath 宏包提供的定界符放大命令不太够用。通常这意味着你的公式太过复杂。此时你应当首先考虑将公式中的部分提出去，以字母符号代替以简化公式。如果你真的想要排版如此复杂的公式，你可以参考我这篇博文。

```
\[ \Biggl(\biggl(\Bigl(\bigl((x)\bigr)\Bigr)\biggr)\Biggr) \]
\[ \Biggl[\biggl[\Bigl[\bigl[[x]\bigr]\Bigr]\biggr]\Biggr] \]
\[ \Biggl \{\biggl \{\Bigl \{\bigl \{\{x\}\bigr \}\Bigr \}\biggr \}\Biggr\}
\[ \Biggl\langle\biggl\langle\Bigl\langle\bigl\langle\langle x
\rangle\bigr\rangle\Bigr\rangle\biggr\rangle\Biggr\rangle \]
\[ \Biggl\lvert\biggl\lvert\Bigl\lvert\bigl\lvert\lvert x
```

```
\rvert\bigr\rvert\Bigr\rvert\biggr\rvert\Biggr\rvert \]
\[ \Biggl\lVert\biggl\lVert\Bigl\lVert\bigl\lVert\lVert x
\rVert\bigr\rVert\Bigr\rVert\biggr\rVert\Biggr\rVert \]
```

省略号

省略号用 \dots，\cdots，\vdots，\ddots 等命令表示。 \dots 和 \cdots 的纵向位置不同，前者一般用于有下标的序列。

```
\[ x_1,x_2,\dots ,x_n\quad 1,2,\cdots ,n\quad
\vdots\quad \ddots \]
```

矩阵

amsmath 的 pmatrix，bmatrix，Bmatrix，vmatrix，Vmatrix 等环境可以在矩阵两边加上各种分隔符。

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

效果图：

使用 smallmatrix 环境，可以生成行内公式的小矩阵。

```
Marry has a little matrix $ ( \begin{smallmatrix} a&b\\c&d \end{smallmatrix
```

效果图：

多行公式

有的公式特别长，我们需要手动为他们换行；有几个公式是一组，我们需要将他们放在一起；还有些类似分段函数，我们需要给它加上一个左边的花括号。

长公式

不对齐

无须对齐的长公式可以使用 `multline` 环境。

```
\begin{multline}
x = a+b+c+{} \\
d+e+f+g
\end{multline}
```

效果：

如果不需要编号，可以使用 `multline*` 环境代替。

对齐

需要对齐的公式，可以使用 `aligned` *次环境*来实现，它必须包含在数学环境之内。

```
\[\begin{aligned}
x ={}& a+b+c+{} \\
&d+e+f+g
\end{aligned}\]
```

效果图：

公式组

无需对齐的公式组可以使用 `gather` 环境，需要对齐的公式组可以使用 `align` 环境。他们都带有编号，如果不需要编号可以使用带星花的版本。

```
\begin{gather}
a = b+c+d \\
x = y+z
\end{gather}
\begin{align}
a &= b+c+d \\
x &= y+z
\end{align}
```

效果：

> 请注意，不要使用 `eqnarray` 环境。原因可以参考：
>
> - `eqnarray` 是糟糕的
> - `eqnarray` 是有害的
> - `eqnarray` 是恼人的
> - `eqnarray` 是邪恶的

分段函数

分段函数可以用 `cases` 次环境来实现，它必须包含在数学环境之内。

```
\[ y= \begin{cases}
-x,\quad x\leq 0 \\
x,\quad x>0
```

```
\end{cases} \]
```

效果图：

辅助工具

站在我个人的角度，我建议 LaTeX 用户应当尽可能避免使用辅助工具输入数学公式。但对于急用的初学者而言，适当地使用辅助工具（而不形成依赖）也是有一些收益的。因此这里介绍一些关于数学公式的辅助工具。

- https://mathpix.com/ 能够通过热键呼出截屏，而后将截屏中的公式转换成 LaTeX 数学公式的代码。
- http://detexify.kirelabs.org/classify.html 允许用户用鼠标在输入区绘制单个数学符号的样式，系统会根据样式返回对应的 LaTeX 代码（和所需的宏包）。这在查询不熟悉的数学符号时特别有用。

插入图片和表格

图片

关于 LaTeX 插图，首先要说的是：「LaTeX 只支持 `.eps` 格式的图档」这个说法是错误的。

在 LaTeX 中插入图片，有很多种方式。最好用的应当属利用 `graphicx` 宏包提供的 `\includegraphics` 命令。比如你在你的 TeX 源文件同目录下，有名为 `a.jpg` 的图片，你可以用这样的方式将它插入到输出文档中：

```
\documentclass{article}
\usepackage{graphicx}
```

```
\begin{document}
\includegraphics{a.jpg}
\end{document}
```

图片可能很大，超过了输出文件的纸张大小，或者干脆就是你自己觉得输出的效果不爽。这时候你可以用 `\includegraphics` 控制序列的可选参数来控制。比如

```
\includegraphics[width = .8\textwidth]{a.jpg}
```

这样图片的宽度会被缩放至页面宽度的百分之八十，图片的总高度会按比例缩放。

> `\includegraphics` 控制序列还有若干其他的可选参数可供使用，一般并用不到。感兴趣的话，可以去查看该宏包的文档。

表格

`tabular` 环境提供了最简单的表格功能。它用 `\hline` 命令表示横线，在列格式中用 `|` 表示竖线；用 `&` 来分列，用 `\\` 来换行；每列可以采用居左、居中、居右等横向对齐方式，分别用 `l`、`c`、`r` 来表示。

```
\begin{tabular}{|l|c|r|}
 \hline
操作系统& 发行版& 编辑器\\
 \hline
Windows & MikTeX & TexMakerX \\
 \hline
Unix/Linux & teTeX & Kile \\
 \hline
Mac OS & MacTeX & TeXShop \\
 \hline
通用& TeX Live & TeXworks \\
 \hline
\end{tabular}
```

效果：

浮动体

插图和表格通常需要占据大块空间，所以在文字处理软件中我们经常需要调整他们的位置。 `figure` 和 `table` 环境可以自动完成这样的任务；这种自动调整位置的环境称作浮动体(float)。我们以 `figure` 为例。

```
\begin{figure}[htbp]
\centering
\includegraphics{a.jpg}
\caption{有图有真相}
\label{fig:myphoto}
\end{figure}
```

`htbp` 选项用来指定插图的理想位置，这几个字母分别代表 here, top, bottom, float page，也就是就这里、页顶、页尾、浮动页（专门放浮动体的单独页面或分栏）。 `\centering` 用来使插图居中； `\caption` 命令设置插图标题，LaTeX 会自动给浮动体的标题加上编号。注意 `\label` 应该放在标题命令之后。

> 图片和表格的各种特殊效果，限于篇幅此处无法详叙。请查看最后一章推荐的文档。
>
> 如果你想了解 LaTeX 的浮动体策略算法细节，你可以参考我博客中关于浮动体的系列文章
>
> 如果你困惑于「为什么图表会乱跑」或者「怎样让图表不乱跑」，请看我的回答。

版面设置

页边距

设置页边距，推荐使用 `geometry` 宏包。可以在这里查看它的说明文档。

比如我希望，将纸张的长度设置为 20cm、宽度设置为 15cm、左边距 1cm、右边距 2cm、上边距 3cm、下边距 4cm，可以在导言区加上这样几行：

```
\usepackage{geometry}
\geometry{papersize={20cm,15cm}}
\geometry{left=1cm,right=2cm,top=3cm,bottom=4cm}
```

页眉页脚

设置页眉页脚，推荐使用 `fancyhdr` 宏包。可以在这里查看它的说明文档。

比如我希望，在页眉左边写上我的名字，中间写上今天的日期，右边写上我的电话；页脚的正中写上页码；页眉和正文之间有一道宽为 0.4pt 的横线分割，可以在导言区加上如下几行：

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\lhead{\author}
\chead{\date}
\rhead{152xxxxxxxx}
\lfoot{}
\cfoot{\thepage}
\rfoot{}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\headwidth}{\textwidth}
\renewcommand{\footrulewidth}{0pt}
```

首行缩进

CTeX 宏集已经处理好了首行缩进的问题（自然段前空两格汉字宽度）。因此，使用 CTeX 宏集进行中西文混合排版时，我们不需要关注首行缩进的问题。

> 如果你因为某些原因选择不适用 CTeX 宏集（不推荐）进行中文支持和版式设置，则你需要做额外的一些工作。
>
> - 调用 `indentfirst` 宏包。具体来说，中文习惯于每个自然段的段首都空出两个中文汉字的长度作为首行缩进，但西文行文习惯于不在逻辑节（`\section` 等）之后缩进。使用改宏包可使 LaTeX 在每个自然段都首行缩进。
> - 设置首行缩进长度 `\setlength{\parindent}{2\ccwd}`。其中 `\ccwd` 是 `xeCJK` 定义的宏，它表示当前字号中一个中文汉字的宽度。

行间距

我们可以通过 `setspace` 宏包提供的命令来调整行间距。比如在导言区添加如下内容，可以将行距设置为字号的 1.5 倍：

```
\usepackage{setspace}
\onehalfspacing
```

具体可以查看该宏包的文档。

> 请注意用词的差别：
>
> - 行距是字号的 1.5 倍；
> - 1.5 倍行距。
>
> 事实上，这不是设置 1.5 倍行距的正确方法，请参考这篇博文。另外，RuixiZhang 开发了 zhlineskip 宏包，提供了对中西文混排更细致的行距控制能力。

段间距

我们可以通过修改长度 `\parskip` 的值来调整段间距。例如在导言区添加以下内容

```
\addtolength{\parskip}{.4em}
```

则可以在原有的基础上，增加段间距 0.4em。如果需要减小段间距，只需将该数值改为负值即可。

TeX 家族

恭喜你终于看到了这里。如果你认真完成了上面所有的练习，并琢磨了其中的意义，相信你已经可以用 LaTeX 排版出漂亮的文档了。现在我们说一点历史，帮助你更好地理解 TeX 这个系统。

带有 TeX 的词，仅仅是本文就已经提到了 TeX, LaTeX, XeLaTeX。通常中国学生面对不了解意思的一群形近单词，都会有一种「本能的恐惧」（笑~）。因此，「大神们」在为新手介绍 TeX 的时候，如果互相争论 「XXTeX 比 YYTeX 好」或者是「XXTeX 的 YYTeX 如何如何」，往往会蹦出下面这些带有 TeX 的词汇：
TeX, pdfTeX, XeTeX, LuaTeX, LaTeX, pdfLaTeX, XeLaTeX ...

事实上，这部分的内容太过复杂，我自己的了解也实在有限。所以下面这部分的内容也只能是对我了解到的知识的一个概括，甚至可能有些许谬误。所以大家只需要将这部分的内容当做是一个参考就可以了。

TeX - LaTeX

TeX 是高德纳（Donald Ervin Knuth，1938年1月10日 --）教授~~犊世嫉俗~~追求完美做出来的排版引擎，同时也是该引擎使用的标记语言（Markup Language）的名称。这里所谓的引擎，是指能够实现断行、分页等操作的程序（请注意这并不是定义）；这里的标

记语言，是指一种将控制命令和文本结合起来的格式，它的主体是其中的文本而控制命令则实现一些特殊效果（同样请注意这并不是定义）。

> 你可以在这里找到关于 TeX 引擎的具体描述；
>
> 你可以在这里找到关于标记语言的具体描述。

而 LaTeX 则是 L. Lamport （1941年2月7日 -- ） 教授开发的基于 TeX 的排版系统。实际上 LaTeX 利用 TeX 的控制命令，定义了许多新的控制命令并封装成一个可执行文件。这个可执行文件会去解释 LaTeX 新定义的命令成为 TeX 的控制命令，并最终交由 TeX 引擎进行排版。

> 实际上，LaTeX 是基于一个叫做 plain TeX 的格式的。plain TeX 是高德纳教授为了方便用户，自己基于原始的 TeX 定义的格式，但实际上 plain TeX 的命令仍然十分晦涩。至于原始的 TeX 直接使用的人就更少了，因此 plain TeX 格式逐渐就成为了 TeX 格式的同义词，尽管他们事实上是不同的。

因此在 TeX - LaTeX 组合中，

1. 最终进行断行、分页等操作的，是 TeX 引擎；
2. LaTeX 实际上是一个工具，它将用户按照它的格式编写的文档解释成 TeX 引擎能理解的形式并交付给 TeX 引擎处理，再将最终结果返回给用户。

pdfTeX - pdfLaTeX

TeX 系统生成的文件是 *dvi* 格式，虽然可以用其他程序将其转换为例如 pdf 等更为常见的格式，但是毕竟不方便。

> dvi 格式是为了排版而产生的，它本身并不支持所谓的「交叉引用」，pdfTeX 直接输出 pdf 格式的文档，这也是 pdfTeX 相对 TeX 进步（易用性方面）的地方。

为了解决这个问题，Hàn Thế Thành 博士在他的博士论文中提出了 pdfTeX 这个对 TeX 引擎的扩展。二者最主要的差别就是 pdfTeX 直接输出 pdf 格式文档，而 TeX 引擎则输出 dvi 格式的文档。

> pdfTeX 的信息可以查看wiki.

pdfLaTeX 这个程序的主要工作依旧是将 LaTeX 格式的文档进行解释，不过此次是将解释之后的结果交付给 pdfTeX 引擎处理。

XeTeX - XeLaTeX

高德纳教授在实现 TeX 的当初并没有考虑到中日韩等字符的处理，而只支持 ASCII 字符。这并不是说中日韩字符就无法使用 TeX 引擎排版了，事实上 TeX 将每个字符用一个框包括起来（这被称为**盒子**）然后将一个个的盒子按照一定规则排列起来，因而 TeX 的算法理论上适用于任何字符。ASCII 字符简单理解，就是在半角模式下你的键盘能直接输出的字符。

在 XeTeX 出现之前，为了能让 TeX 系统排版中文，国人曾使用了 天元、CCT、**CJK** 等手段处理中文。其中 天元和CCT 现在已经基本不用，CJK 因为使用时间长且效果相对较好，现在还有人使用。

不同于 CJK 等方式使用 TeX 和 pdfTeX 这两个不直接支持 Unicode 字符的引擎，XeTeX 引擎直接支持 Unicode 字符。也就是说现在不使用 CJK 也能排版中日韩文的文档了，并且这种方式要比之前的方式更加优秀。

XeLaTeX 和 XeTeX 的关系与 pdfLaTeX 和 pdfTeX 的关系类似，这里不再赘述。

使用 XeTeX 引擎需要使用 UTF-8 编码。

> 所谓编码就是字符在计算机储存时候的对应关系。例如，假设有一种编码，将汉字

> 「你」对应为数字「1」；「好」对应为数字「2」，则含有「你好」的纯文本文件，在计算机中储存为「12」（读取文件的时候，将「12」再转换为「你好」显示在屏幕上或打印出来）。
>
> UTF-8 编码是 Unicode 编码的一种，可以参考它的 wiki.

LuaTeX

LuaTeX 是正在开发完善的一个 TeX 引擎，相对它的前辈们还相当的不完善，这里不赘述。

CTeX - MiKTeX - TeX Live

之前介绍了 TeX, LaTeX, pdfTeX, pdfLaTeX, XeTeX, XeLaTeX, LuaTeX 等，他们都是 TeX 家族的一部分。但是作为一个能够使用的 TeX 系统，仅仅有他们还是不够的。CTeX, MiKTeX, TeX Live 都是被称为「发行」的软件合集。他们包括了上述各种引擎的可执行程序，以及一些文档类、模板、字体文件、辅助程序等等。其中 CTeX 是建立在 MiKTeX 的基础之上的。

总结

TeX - pdfTeX - XeTeX - LuaTeX 都是排版引擎，按照先进程度递增（LuaTeX 尚未完善）。

LaTeX 是一种格式，基于 TeX 格式定义了很多更方便使用的控制命令。上述四个引擎都有对应的程序将 LaTeX 格式解释成引擎能处理的内容。

CTeX, MiKTeX, TeX Live 都是 TeX 的发行，他们是许许多多东西的集合。

出现问题应当如何处理/怎样聪明地提出你的问题——怎样从这

里毕业

这篇文章原来的名字叫做**慢慢爱上 LaTeX**，后来行文的过程中发觉 LaTeX 实在是有太多的内容可以讲述，越发地没有把握让大家仅仅通过这一篇短小的文章就喜欢上 LaTeX，于是改成了现在这个名字。

限于篇幅，还有我的精力，这篇文章事实上还有许多的问题没有讲明白。它仅仅是提供了一些，在你学习使用 LaTeX 可能遇到的问题的解决方案，并不完全，而且没有教会你如何处理编译过程中可能遇到的形形色色的错误。
所以这最终只能是一篇小文，而不可能成为一篇正式的文档。

因此，如果通过我这篇小文，体验到了 LaTeX 带给你的一些乐趣，有了对 LaTeX 的兴趣（如果是这样，我就已经能够无比开心了~），请务必去阅读一些正式的 LaTeX 文档。

去读文档

关于 LaTeX 的文档有很多，其中有些内容过时地很快。所以有必要告诉大家，哪些文档应该看，那些文档不应该看。索性，这个问题，刘海洋(milksea)前辈已经叙述得很清楚了。前段时间，我和几个朋友，将现在看起来还未过时的文档打包上传到了我的博客。
参考：ZIP 归档。

遇到问题怎么办

0. 绝对的新手，先读完一本入门读物，了解基本的知识；
1. 无论如何，先读文档！绝大部分问题都是文档可以解决的；
2. 利用 Google 搜索你的问题；
3. 在各个论坛或者 LaTeX 交流群里聪明地提出你的问题。

参 考 ：https://github.com/ryanhanwu/How-To-Ask-Questions-The-Smart-Way/blob/master/README-zh_CN.md

- CTeX 论坛提问版：https://github.com/CTeX-org/forum/issues
- 提供一个 Telegram 交流群：https://t.me/chinesetex
- 提供一个 QQ 交流群：141877998

俗话说，投资效率是最好的投资。 如果您感觉我的文章质量不错，读后收获很大，预计能为您提高 10% 的工作效率，不妨小额捐助我一下，让我有动力继续写出更多好文章。

打赏

Introduction # ctex # TeXworks

‹ 在 Windows 下批量将位图转为 EPS 格式的图档

Matplotlib 教程 ›

说点什么

ⓘ 支持 Markdown 语法

使用 GitHub 登录 预览

头像

rachpt 发表于 大约 1 年前

https://blog.csdn.net/henu111/article/details/81239727

头像

Liam0205 发表于 大约 1 年前

@rachpt 谢谢……被抄袭习惯了……

头像

wzx140 发表于 大约 1 年前

感谢博主的分享

头像

Liam0205 发表于 大约 1 年前

@wzx140 客气

头像

lizongzeshunshun 发表于 12 个月前

问一下博主，使用推荐的 **CTeX** 宏集，如何更改中文和英文字体呢。

头像

Liam0205 发表于 12 个月前

@lizongzeshunshun 使用 CTeX 宏集时：

- 使用 pdfLaTeX/LaTeX 编译，
  - 默认（ `zhmap = true` ）依赖 `CJK + zhmetrics` 支持中文，配置字体需要用 pdfTeX primitives 修改字体 mapping。
  - 配置 `zhmap = false` 选项时，依赖传统 `CJK` 的 Type1 字库支持中文，配置字体需首先生成 Type1 的 CJK 字体，然后使用 `\CJKfamily` 命令指定。
  - 配置 `zhmap = zhmCJK` 选项时，依赖 `zhmCJK` 宏包支持中文，配置字体的方式和 `xeCJK` 宏包接近，使用 `\setCJKmainfont` 等命令。具体参见其宏包文档。不过 `zhmCJK` 宏包目前只支持 TeX Live 发行版，且默认未安装，需要手工安装。
- 使用 XeLaTeX 编译时，底层依赖 `xeCJK` 宏包支持中文，使用 `\setCJKmainfont` 等命令配置中文字体；具体参见其宏包文档。对于英文， `xeCJK` 又依赖 `fontspec` 宏包，使用 `\setmainfont` 等命令配置西文字体；具体参见其宏包文档（英文）。
- 使用 LuaLaTeX 编译时，底层以来 `luatex-ja` 宏包支持中文，使用 `\setmainjfont` 等命令配置中文字体；具体参见其宏包文档（日文）。对于英文， `luatex-ja` 也依赖 `fontspec` 宏包，使用 `\setmainfont` 等命令配置西文字体；具体参见其宏包文档（英文）。

以上详细参见 CTeX 宏集说明文档。

头像

lizongzeshunshun 发表于 12 个月前

@Liam0205 谢谢你的详细回答！

头像

Liam0205 发表于 12 个月前

@lizongzeshunshun 客气。

头像

ceo1207 发表于 11 个月前

哇 这博客真好看，你是自己租的云服务器吗

头像

Liam0205 发表于 11 个月前

> @ceo1207
> 哇 这博客真好看，你是自己租的云服务器吗

你可以了解一下 GitHub Pages 服务。

加载更多