LaTeX：优雅高效的写作方式

写在 LaTeX 之前

为什么要使用 LaTeX

- 纯文本
- 跨平台
- 高效
- 优雅
- 强大

什么样的人适合使用 LaTeX

LaTeX 和 Word 各有千秋，有很多Geeks或者LaTeX的狂热粉过分强调了 LaTeX 的一些并非常用的特性，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。

我曾经也受到过这些言论的影响，但当我真正弄明白这个问题(什么样的人适合使用 LaTeX)时，已经不再对这两种工具有太多偏见。因此我想以我自己的方式，向更多的人介绍 LaTeX，客观清晰地认识到 LaTeX 的优缺点。

latex 主要面向的用户是科研工作者和需要高质量排版的用户。不得不承认，它其实并不是一个好学的东西。我们只有通过前期付出一定学习成本才能用它排版出自己心仪的作品。它并不像 Word，打开打字就能用，排版效果也无法直接显示在面前，所以学习的成本还是挺高的。我在研一的时候为了满足某些课程作业的需求开始写 LaTeX，发现成果不错便将所有课程作业都用 LaTeX 排版完成。因此学习和使用 LaTeX 需要一定的毅力去学习，就像学习任何其他的东西一样。

LaTeX 简史

学数学或者学计算机的人都知道一个人，就是高德纳 (Donald E. Knuth)。他在而立之年就写下了巨著《计算机程序设计艺术 (The Art of Computer Programming, TAOCP)》的前三卷，并凭借它成为历史上最年轻的图灵奖获得者。当这部书交付印刷后给他看时，他说：“我不知道怎么办。我花了整整 15 年写这些书，可要是这么难看，我就 再也不写了。我怎么能对这样的作品引以为豪呢？”于是20世纪70年代他就放下了 TAOCP 的写作，用十年的时间写出了 LaTeX 的前身：TeX。同时发布的还有他为 TeX 写的说明书《The TeXBook》。TeX 的版本号从3开始，每次更新都会按照圆周率 π 的小数顺序向后加一位，比如3，3.1，3.14，3.141.....目前我使用的 TeX 版本是3.14159265。

这一系统推出之后对出版业影响巨大。TeX 的核心思想就是源自中国的活字印刷术，使用一系列命令来在纸上的不同位置放下不同的文字，这样完成排版工作。但是最初的 TeX 和手动排版没有太大的本质差别，使用起来还是有些不方便。于是在80年代，美国计算机科学家莱斯利·兰伯特 (Leslie Lamport) 在原来 TeX 的基础上加上了许多功能，使得它变得更加易用。大家在 TeX 的前面加上了兰伯特姓氏的前两个字母，就成为了LaTeX。LaTeX 目前的版本是2ε，表示比2大但是比3小，一般也写作 LaTeX2e。莱斯利·兰伯特也在2014年获得2013年度图灵奖。当时的国际计算机学会 (Association of Computer Machinery, ACM) 和美国数学学会 (American Mathematical Society, AMS) 都采用了这一排版系统，直到今天它仍然是学术排版的不二工具。许多学术期刊都提供 LaTeX 模板，并只接收 TeX 源文件投稿。

LaTeX 是公开源代码的软件，业界许多大牛写出了很多功能各异的宏包来实现各种各样的功能。比如 AMS 官方开发了 amsmath 系列宏包，提供数学排版标准和相应的符号支持；甚至还有专门排版乐谱、化学式、棋谱的宏包。庞大的社区使得 LaTeX 的功能越来越丰富，也让它变得越来越强大。

LaTeX 的安装

安装 LaTeX 并不是一件难事，只要选择合适的发行版就行了。LaTeX 相关的全部资源都在 CTAN (Comprehensive TeX Archive Network) 上，可以登录世界各地的镜像网站看到相关的资源。不过需要手动获取资源的情况其实不多，这里只是作一个介绍。

- Linux 平台可用 TeX Live，建议不要直接用 apt-get 或者 yum 直接安装，那样安装出来的版本功能不全，（加上-full参数同样不全）。最好还是找到最新版本的TeXLive下载安装，省心。
- Windows 平台上比较常用的可能是 CTeX 套装，其中自带的编辑器有 WinEdt 和 TeXWorks。但是CTeX目前更新较慢，而且实际软件包也不全，同样强烈建议安装 TeXLive。在数院研究生会公众号可以获得最新版 TeXLive 的安装包和简明安装教程。TeXLive 同样使用 TeXWorks 编辑器。
- macOS/OS X 墙裂建议使用 MacTeX，安装过程同样简单。目前这个对中文的支持最好。

LaTeX 的工作原理

说了这么多，LaTeX 到底是个什么东西呢？下面我们就来回答这个问题。

LaTeX 是一个排版系统，它是一个引擎。它以文本作为输入，以排版好的文档作为输出。这就像很久以前还没有电脑时候的出版社一样，你把写好的手稿给出版社，出版社帮你出版成书。真正重要的是你写的内容，至于用什么笔、什么纸，都无所谓。所以纯净的 LaTeX 只是一个引擎，它将一个.tex源文件作为输入，把成品文档作为输出。我们写作时其实是在编辑这个.tex源文件。源文件其实只是一个纯文本文件，你可以用任何一种文本编辑器（比如 Windows 下的记事本）来写作。但是由于用 LaTeX 写作不是“所见即所得”的，源文件中会有许多排版命令和标记（类似 HTML），因此方便起见，大家开发了一些专门用于编辑 .tex 文件的编辑器，比如我使用的 VSCode。初学者我推荐 TeXworks (Windows) Vim (Linux) 和 TeXShop (macOS)，在自己有了一定的编辑习惯和定制编辑器的能力之后，可以再换别的编辑器。不过编辑器从来都没有最好的，挑一个趁手的用就行了。

由于一些历史原因，比如打印设备的进步，LaTeX 发行版中带有多个引擎。目前比较常用的引擎是 XeLaTeX，因为它能够兼容的字符集更全面，尤其是在处理东亚文字方面支持得更好。关于字符集和字体的问题我会在后面找机会和大家详细介绍，现在知道这个就好了。所以整个文档的写作过程可以表达为：

> （编写）-> .tex文件 -（输入）-> XeLaTeX -（输出）-> PDF文档 -->发布

这也是我们接下来编写文档的全部过程。

第一个 LaTeX 文档

安装好 LaTeX 的你是不是着急想要开始排版第一个文档了呢？这里我把制作题图的 LaTeX 源文件贴在下面，大家可以按照这个文档写一个 hello.tex，然后自己排版试试看。没有调好中文字体的朋友们也可以尝试，因为这里并没有汉字出现。

```
\documentclass{article}
% the preamble should be here.
% but there's nothing in it.
\begin{document}
    \begin{center}
        Welcome to \\
        \LaTeX
    \end{center}
\end{document}
```

打完了之后用 XeLaTeX 作为引擎排版，然后新鲜出炉的 PDF 文档就会展现在眼前啦！是不是很有成就感！

动手写 LaTeX

LaTeX 文档的基本构成

导言区

导言区是文类定义之后，正文开始之前的部分。它的主要作用是对文档的性质做一些设置，或者自定义一些命令。在前面的例子中，导言区本应该在第1、2行之间，但是因为没有任何内容，所以就省去了。所以含有导言区的文档应该是下面这样的。

```
\documentclass{article}
% the preamble should be here.
% but there's nothing in it.
\begin{document}
    \begin{center}
        Welcome to \\
        \LaTeX
    \end{center}
\end{document}
```

首先第1行中的\documentclass表示的是这个文档的文类(document class)。所谓文类，是指一套预设的排版格式，常见的（英文文类）有 article，report，book 等等。它们之间有一些微妙的差别，比如 book 文类就为了适应书籍的装订，为奇数页和偶数页安排了不同的左右边距；report 和 article 非常相似，但 report 中可以使用更多的章节等级，article 则是几乎最简单的一种。

我们能看到，所有蓝色的部分都是由反斜杠“\”打头的，这些都叫做**命令**(command)。每个命令后可能会有**参数**(argument)，比如\documentclass就接受一个参数，就是用大括号括起来的 article。今后大家可能会遇到有更多参数的命令，那么就在后面继续加大括号就行了，每个参数都应该在一个大括号中。*有的命令没有参数*，比如第4行的“\”表示换行，第5行的“\LaTeX”表示 LaTeX 的 logo，直接使用即可。

正文

在文类的定义之后就是正文部分，每个文档的正文部分都必须在\begin{document}和\end{document}之间。这一对命令组成了一个**环境**(environment)，在不同的环境当中，文本会有不同的格式预设。document 环境是最基本的环境，第3行和第6行之间还有一个 center 环境，这一环境的预设就是其中的内容全部以居中的方式进行排版。常用的环境有很多，例如 equation 环境用于排版公式，figure 环境用于排版图片，itemize 环境表示无序列表，enumerate 环境表示有序列表，诸如此类的环境我们会在后面慢慢提到。

这个文档的全部内容就是 Welcome to LaTeX 。当初为了将它作为题图，我希望它居中，于是放在了 center 环境里。我又希望它分两行呈现，因此在 Welcome to 后面加了一个强制断行命令。这里可能有人会问了，为什么已经按了回车，它却不断行呢？这里涉及到一个 LaTeX 中的基本规则，就是在一般情况下，一个换行、一个空格和多个空格这三种东西表达在输出文档中都是一个空格，而换行则需要连续两个换行符才能实现。与之等价的命令就是“\”，它在排版表格等需要强制换行对齐的环境中非常有用，在正文中也可以使用。在这里，你也可以尝试把它删去，再用两个回车代替，看看排版效果是不是相同。

注释

图中绿色部分就是导言区的位置，当然它并不限于两行，可以有很多。注意到这两行的字以百分号开头，显示为绿色，而且当我们对这个修改后的文档进行排版之后会发现和原来的文档没有任何差别（大家可以自己试试）。这就是我非常喜欢的一个 LaTeX 相对于 Word 的优势，它叫做注释(comment)。注释的内容不参与排版，你可以在里面写自己喜欢的任何东西。任意一行中，从百分号开始以后的部分全部都会在编译时被忽略。

注释有非常大的用途，我来举个例子。你在写完论文的时候发现，之前写的一段文字没有什么用，可以被精简；但是这些文字如果删掉了就没有办法找回来，万一导师让我再把它加上就麻烦了。于是在 LaTeX 中，你可以放心地把这些内容注释掉而不是删掉，这样在修改文档的同时可以保留你原来的文字，就算导师让我加回来我也能立刻完成。

宏包

先前我们提到，导言区的作用之一是设置文档性质。所谓文档的性质有许多方面，比如设定标题、作者、日期、图题格式、表题格式等。其中还一个非常重要的作用就是添加宏包(package)。宏包可以被理解为一些模块，它们提供 LaTeX 本身没有的命令，以实现一些原本不太好实现的效果。比如 ctex 宏包就是一个把文档格式全部中文化的宏包，并提供 ctexart，ctexrep，ctexbook 等与 article，report，book 相对应的文类，方便排版中文文档。

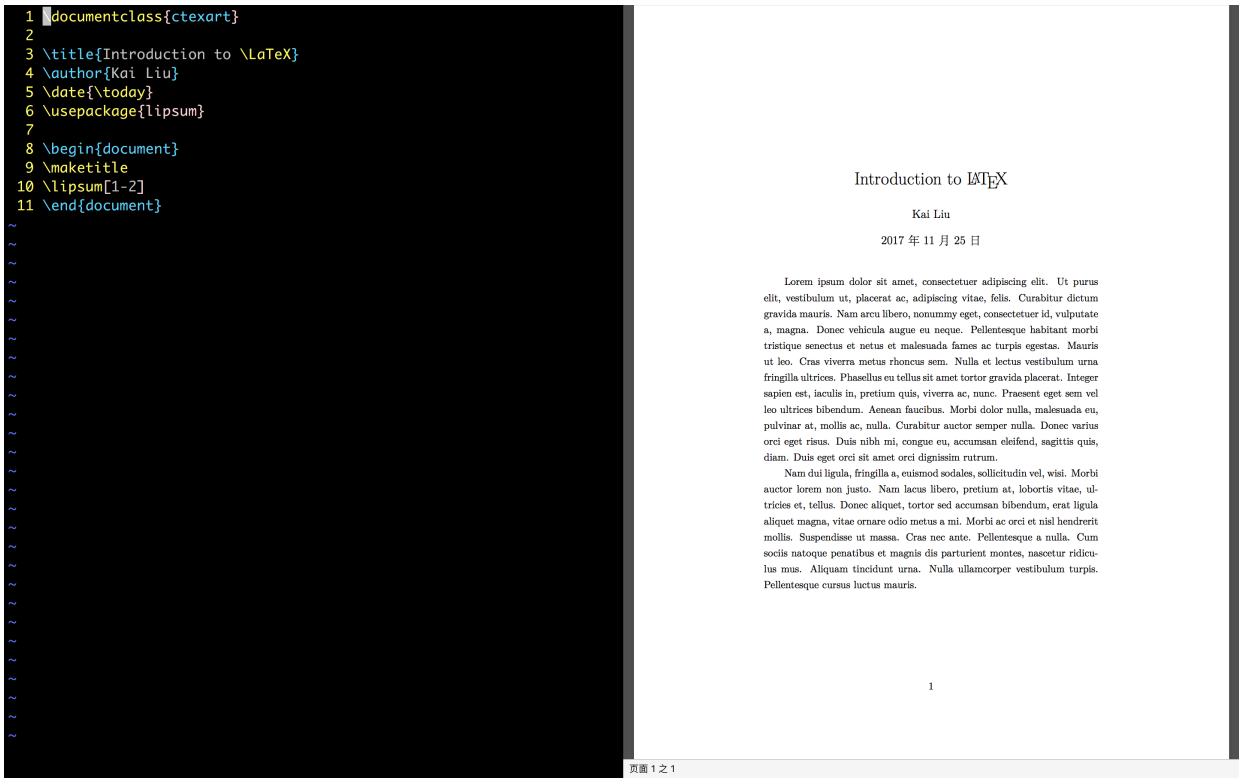
ctexart 文类相比 article，日期变成了中文表达，而且行间距自动增加了，可以说适应汉字的排版。所以在写中文文档的时候，推荐大家使用 ctex 系列文类，可以省心很多。

下面为大家展示一个被 ctex 宏包“汉化”过的文档。

```latex
\documentclass{ctexart}

\title{Introduction to \LaTeX}
\author{Kai Liu}
\date{\today}
\usepackage{lipsum}

\begin{document}
\maketitle
\lipsum[1-2]
\end{document}
```



上面这个文档就是一个相对丰富的文档了。这里要为大家介绍5个命令：

- \title：设置文档标题；有一个参数，用大括号跟在后面，表示标题内容；
- \author：设置文档作者；一个参数；
- \date：设置日期；其中\today表示系统当天日期，也可以手动输入；留空的话则不排版日期；
- \usepackage：调用宏包；此处调用的宏包是 lipsum，用于生成随机文字（上面满篇的洋文其实都是生成的）；第10行的 \lipsum[1-2] 表示插入随机文字的前两段；
- \maketitle：这是这5个命令中唯一一个写在正文区的命令，排版标题；这个命令如果不使用，之前设置的标题、作者、日期都是不会出现在成品中的；使用它之后会排版标题、作者和日期。

有了这些命令，我们就可以顺利地写一篇简单的文章了，大家可以把里面的内容都换成汉字试试看。

字体与字号

现在我们能排版一些基本的文字了，下面我们来介绍一下字体和字号。

与字体相比，字号是相对简单的一个属性。LaTeX 提供了10个命令来声明文字的大小，由小到大分别是：

- \tiny
- \scriptsize
- \footnotesize
- \small
- \normalsize
- \large
- \Large
- \LARGE
- \huge
- \Huge

些命令会由于全局字号的调整而代表不一样的大小，这样相同的内容在调整的过程中的相对大小保持不变，所以全局的修改不会改乱。以 \Large 为例，用法是将这个命令和需要调整的文字一起用大括号包起来，像这样:

```
There are smaller words and {\Large larger words}.
```

那么如果想要指定字号怎么办呢？老师让我们用五号字来写论文，应该怎么办？这时我们就可以用 ctexart 自带的一个参数 c5size 来处理，用法是在文类定义处做一些修改：

```
\documentclass[c5size]{ctexart}
```

这样的话\normalsize就会变成5号字，其他命令也会随之变化。同样地，用cs4size则可以把标准字号定义为小四号。假如想指定字号，而我们熟悉的又是汉字的字号系统（小四、五号之类），我们就可以用 ctex 为我们提供的字号指定命令，用法和前面的命令相同，下面举一些例子大家就明白了：

- 小五：\zihao{-5}
- 六号：\zihao{6}
- 初号：\zihao{0}
- 小四：\zihao{-4}

聪明的你一定看出规律了对不对！

字体是一个相对复杂的话题。在 MS Word 里面，我们可以随意地将文字加粗、倾斜，但其实里面是有一个完整的字体体系的。一般来说，我们用三个东西来定义一个字体：**字体族(font family)**，**字体形状(font shape)**和**字体系列(font series)**。这个字体主要是对英文而言的，字体族中包含罗马(roman)、无衬线(sans serif)、打字机(typewriter)三种，字体形状有直立(upshape)、意大利(italic shape)、倾斜(slanted shape)、小形大写(small capital)等四种，字体系列则有中等和加宽加粗两种。刚刚提到的在 Word 中的加粗其实就是采用一个字体的加宽加粗系列，而倾斜则是改变形状为倾斜。我们看看下面的例子：

```
\documentclass{ctexart}
\begin{document}
\begin{tabular}{|ll|ll|ll|}
\hline
字体族 & 效果 & 字体形状 & 效果 & 字体系列 & 效果 \\
\hline
罗马 & \textrm{Roman} & 直立 & \textup{upright} & 加宽加粗 & \textbf{Bold
extended} \\
无衬线 & \textsf{Sans serif} & 意大利 & \textit{italic} & 中等 &
\textmd{Medium} \\
打字机 & \texttt{Typewritter} & 倾斜 & \textsl{slanted} & & \\
& & 小型大写 & \textsc{SMALL CAPITALS} & & \\
\hline
\end{tabular}
\end{document}
```

字体族	效果	字体形状	效果	字体系列	效果
罗马	Roman	直立	upright	加宽加粗	**Bold extended**
无衬线	Sans serif	意大利	*italic*	中等	Medium
打字机	Typewritter	倾斜	*slanted*		
		小型大写	SMALL CAPITALS		

这就是 LaTeX 中基本的字体。其中无衬线字体主要用于幻灯片展示，因为在距离较远的情况下，衬线对文字形状的识别没有任何用处，所以用无衬线字体来提升观众对文字的识别（所以不要再用宋体和 Times New Roman 做幻灯片了）。打字机字体具有字体等宽的性质，适合排版代码和抄录，方便对齐。意大利和加粗一般用于强调，意大利还可以用于引用他人的文字。通常情况下，罗马字体族就可以满足我们的大部分要求。一个文档中最好不要出现三种以上完全不同的字体，那样会显得非常杂乱。

在汉语中也有对应的字体划分方法。一般罗马就是宋体，意大利是楷体，打字机是仿宋（毕竟汉字全部都等宽），加宽加粗还有无衬线都是黑体。现在大家要明白一点，加粗并不只是把原来的文字加粗，而是用另一种字体来起到你想达到的效果。以上这三种字体的配合基本上就可以满足学术文档中的全部需求了。我们举一个例子：

```
\documentclass{ctexart}
\begin{document}
\begin{tabular}{|ll|ll|ll|}
\hline
字体族 & 效果 & 字体形状 & 效果 & 字体系列 & 效果 \\
\hline
罗马 & \textrm{Roman} & 直立 & \textup{upright} & 加宽加粗 & \text    bf{Bold
extended} \\
无衬线 & \textsf{Sans serif} & 意大利 & \textit{italic} & 中等 & \
textmd{Medium} \\
打字机 & \texttt{Typewritter} & 倾斜 & \textsl{slanted} & & \\
& & 小型大写 & \textsc{SMALL CAPITALS} & & \\
\hline
\end{tabular}
\end{document}
```

加粗的字体并不只是把几个字加粗，用*斜体强调*的时候其实也应该换成另外一种字体，因为汉子倾斜下来真的不好看。而打字机字体族可以让大家找回从前铅笔排版时代的感觉。

另外，如果要给一个词语着重强调，可以使用`\emph`命令。它会把相应的文字改变成意大利形状。例如上图中的“斜体强调”就可以用“\emph{斜体强调}”来表达。

常用排版效果

自此，对于单纯的文字文档，你应该可以像使用 Word 一样熟练地使用 LaTeX 来排版了。这里我们再介绍一些简单常用的排版命令，大家可以自己随便找一些文字来试着排版一下。

1. **脚注**。脚注的命令是 \footnote{脚注内容}，只要跟在需要注的文字后面，就可以在那一页底部产生一个脚注。
2. **强调**。这个之前已经介绍过了，\emph{强调内容}，效果是用 italic 形状排版这些文字。
3. **修改行距**。在导言区使用\linespread{倍数}，可将全文的行距进行调整。默认情况下采用的是基本行距，这一行距指的是从上一行的基线到下一行的基线之间的距离，而不是每一行底部到下一行顶部的距离。
4. **靠左、靠右、居中**。这三个命令都是针对段落的，首先要把需要处理的段落用大括号括起来，然后在开头加上 \raggedright，\raggedleft 和 \centering，分别对应靠左、靠右、居中。

最后再介绍两个环境吧。一个是计数列表，一个是无序列表。计数列表的环境名称是 enumerate，无序列表的环境名称是 itemize。下面是示例：

```
\documentclass{ctexart}
\begin{document}
计数的列表大概长这样：
\begin{enumerate}
\item 这是
\item 一个
\item 计数
\item 列表
\end{enumerate}

无序列表是这样的：
\begin{itemize}
\item 这个
\item 列表
\item 没有
\item 计数
\end{itemize}
\end{document}
```

计数的列表大概长这样：

1. 这是
2. 一个
3. 计数
4. 列表

无序列表是这样的：

- 这个
- 列表
- 没有
- 计数

这两个环境都用 \item 作为新的一行的开始。如果要多级列表的话在里面再包裹一个环境就行了。

文档结构化

我们有时候更需要的是一个结构化的文档。它可以分不同的小节，小节与小节之间还需要有不同的层次，比如第几大点第几小点（像领导讲话一样）。如果我们在写作的时候能有一个提纲，那么写作的思路就会更加清晰，文档也更加简洁易懂。

在 ctexart/article 文类中，LaTeX 提供了如下的小标题级别和对应的命令：

1. section（\section{节标题}）

2. subsection（\subsection{小节标题}）
3. subsubsection（\subsubsection{小小节标题}）
4. paragraph（\paragraph{段落标题}）
5. subparagraph（\subparagraph{小段落标题}）

其中前三个级别是有编号的，后两个则没有编号，效果大概是这样的：

```
\documentclass{ctexart}
\usepackage{lipsum}
\begin{document}
\section{节}
\lipsum[1-2]
\subsection{小节}
\lipsum[3-4]
\subsubsection{小小节}
\lipsum[5-6]
\paragraph{段落}
\lipsum[7-8]
\subparagraph{小段落}
\section{又一节}
\lipsum[9-10]
\subsection{又一小节}
\end{document}
```

1 节

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

1.1 小节

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi.

在一般情况下，三级编号五级结构已经足够大家使用了，太深的组织结构会让读者觉得很乱，也会把文字打得很散，不利于表现主题。在编译的过程中， 引擎会自动为这个节标题进行编号。这是一个非常好用的特性。如果你的文章有十个 section，你想在第二个和第三个之间加入一个 section ，只需要直接往里面加就行了，完全不用考虑编号的事。再比如有几个 section 你想要加深一级变成 subsection，再合并成一个 section，在 LaTeX 这种组织结构下可以很轻松地完成。如果是手打的编号的话，大家可以自行想象一下编号改动的工作量，我反正是不敢想。

在 ctexrep/report 和 ctexbook/book 文类中，会有更高的音节级别，比如 part，chapter 之类。chapter 在学术论文的写作中还算常用，比 section 高一个级别，如果篇幅比较大的话可以更有效地组织文档。写作学术论文的时候一般会有模板可以使用，这时只要按照模板的说明操作就可以了，这里列出仅供参考。

标签与交叉引用

在行文过程中，我们可能需要引用自己前面写过的结果。比如“在第一章第5节我们提到……”“请参见我在第327页做的推导……”之类的。在这种情况下，自动化的需求是非常高的。如果说我手动输入了这些数字，那么一旦文档结构有变动，产生的工作量将非常可观。这些东西你还不能不改，因为你不改的话完全就是错的，本来想引用第3节结果写的是“见第2节”。但这在 LaTeX 中完全不是问题。我们可以用**标签(label)**和**交叉引用(cross reference)**来解决这个问题。

标签和交叉引用的基本思路是这样的。首先把我需要引用的章节用 \label{标签名称} 进行标记，然后在需要引用它的时候用 \ref{标签名称} 进行引用。这样一来，引擎在进行编译的时候就需要编译两次：第一次生成所有章节的编号，第二次再把这些编号填充到相应的位置。所以如果使用了这个命令但只编译了一次，这些引用的地方都将显示为问号。这对所有交叉引用都是一样的。道理很简单，引擎每编译一次就把整个源码读一遍，而只读一遍是没有办法知道被引用的章节编号是多少的（尤其是在前面引用后面的章节）。所以引擎在读第一遍的时候就把需要引用的地方都做上标记，第二次再来处理。下面我们来看一个使用的例子：

```
\documentclass{ctexart}
\begin{document}
\section{前言}
\label{sec:introduction}
这里是说在前面的话。\LaTeX 是一个非常棒的排版工具。我在第 \ref{sec:detail} 节中会有详细的论述。
\section{\LaTeX 介绍}
\label{sec:detail}
我在第 \ref{sec:introduction} 节中提到，\LaTeX 是一个非常好的排版工具，下面进行详细的论述。
\end{document}
```

1 前言

这里是说在前面的话。LaTeX 是一个非常棒的排版工具。我在第 2 节中会有详细的论述。

2 LaTeX 介绍

我在第 1 节中提到，LaTeX 是一个非常好的排版工具，下面进行详细的论述。

如果这个时候我在这两节之间加入了新的一节，LaTeX 介绍就变成了第3节，那么我们不需要复杂的操作，该写什么写什么就行了。例子如下：

```
\documentclass{ctexart}
\begin{document}
\section{前言}
\label{sec:introduction}
这里是说在前面的话。\LaTeX 是一个非常棒的排版工具。我在第 \ref{sec:detail} 节中会有详细的论述。
\section{新的一节}
刚刚忘了这一节，现在把它加上。
\section{\LaTeX 介绍}
\label{sec:detail}
我在第 \ref{sec:introduction} 节中提到，\LaTeX 是一个非常好的排版工具，下面进行详细的论述。
\end{document}
```

1 前言

这里是说在前面的话。LaTeX 是一个非常棒的排版工具。我在第 3 节中会有详细的论述。

2 新的一节

刚刚忘了这一节，现在把它加上。

3 LaTeX 介绍

我在第 1 节中提到，LaTeX 是一个非常好的排版工具，下面进行详细的论述。

大家可以注意到，除了第6行和第7行之外，我什么也没有改。这就是 LaTeX 给我们提供的便利，只要标签打好了，引用的是什么就交给引擎来做吧。

另外这里还有一个需求，就是我在读到这个引用的时候想跳转到被引用的地方去看一看，该怎么办呢？这里我们可以使用 hyperref 宏包，下面举一个例子（这里加上了 colorlinks 选项，方便大家看清楚）：

```
\documentclass{ctexart}
\usepackage[colorlinks=true]{hyperref}
\begin{document}
\section{前言}
\label{sec:introduction}
这里是说在前面的话。\LaTeX 是一个非常棒的排版工具。我在第 \ref{sec:detail} 节中会有详细的论述。
\section{\LaTeX 介绍}
\label{sec:detail}
我在第 \ref{sec:introduction} 节中提到，\LaTeX 是一个非常好的排版工具，下面进行详细的论述。
\end{document}
```

1 前言

这里是说在前面的话。LaTeX 是一个非常棒的排版工具。我在第 2 节中会有详细的论述。

2 LaTeX 介绍

我在第 1 节中提到，LaTeX 是一个非常好的排版工具，下面进行详细的论述。

这个时候红色的标记就可以在阅读时用鼠标点击，会直接跳转到被引用的地方。

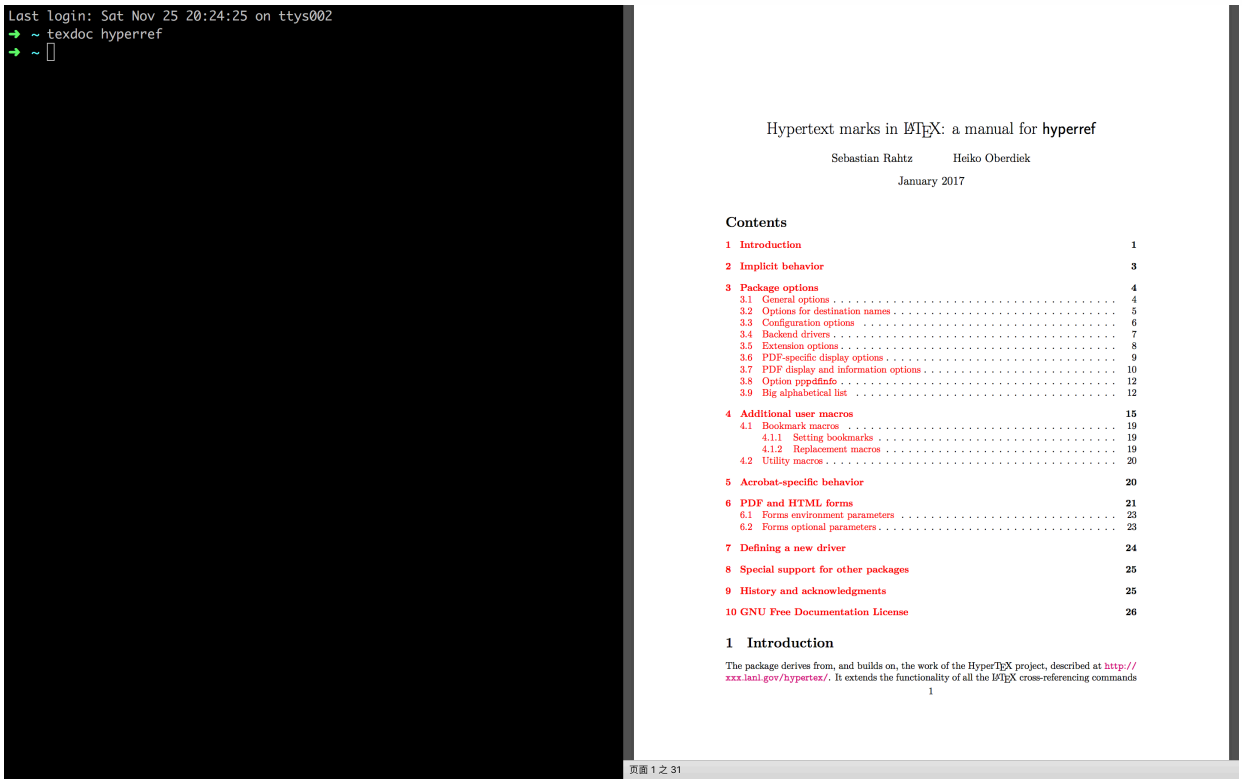
对于标签再多啰嗦两句。标签的命名非常重要，如果还命名成1，2，3这些数字的话，就完全失去了标签的意义了。标签最好要能体现被引段落的内容，这样你在引用的时候就会想起，我要引用的是什么内容，然后打出相应的标签。这是一个非常好的习惯，尤其是在管理特别大的文档的时候。

如何获取帮助

就像你买的洗衣机、电视机、照相机有说明书一样，每个软件也有说明书，我们称之为**文档(documentation)**。对于 LaTeX 的基本命令的使用，我在电脑上装了一个软件 Dash，它专门用于看各种文档，里面就包含 LaTeX 的文档。比如刚刚我们介绍的交叉引用及其相关命令，就可以在其中搜索到。

遇到关于 LaTeX 自带命令不知道怎么使用时，就可以用这个工具来进行一些查询。没有 Dash 也没有关系，直接打开 Google 或者 Bing 之类的搜索引擎把相关的命令打进去一定能找到结果。

还有一些情况下，是你需要使用一个宏包，但是不知道相关的功能怎么使用。这个时候我们就需要命令行工具（Windows 下的 cmd，macOS、OS X、*nix下的 shell）了。在其中输入“texdoc 宏包名”就可以查阅宏包的文档，里面有详细的使用说明，在提出更多问题之前一定要好好阅读。比如我们要查刚刚使用的宏包 hyperref 的文档，就可以在命令行输入“texdoc hyperref”，一个回车下去就会跳出一个 PDF 文档，那就是你要的说明书。



如果你只是想要实现一个功能，既不知道用什么命令，也不知道用什么宏包，那么这个时候你需要 Google。到搜索引擎中去把你想要的功能加上 LaTeX 作为关键词就可以搜到很多有用的资料。在阅读这些资料的时候一定要注意两点：（1）最好阅读英文，更容易找到你要的内容；（2）注意时效性，太旧的文章要辩证接受。当你知道相关的命令和相关的宏包时，你就可以在本地查阅相关的文档了。

如果你找遍了搜索引擎和文档都没有自己想要的，这时就可以选一个社区去提问了。LaTeX 是一个开源软件，拥有完善的文档体系和庞大的社区，只要你有需要，基本都可以得到帮助解决。

另外，如果你手上有一本合适的参考书，也最好先仔细阅读，对 LaTeX 有一个基本的掌握，你阅读文档搜索帮助的效率会更高。

制作目录

上面我们提到了如何在 LaTeX 中安排文档的组织结构，现在我们需要给它们来一个集合，就是制作目录。制作目录其实非常简单，只需要一个命令，就是 \tableofcontents 。这个命令放在哪里，目录就会出现在哪里。和交叉引用相同的一个特点是，目录的排版也需要两次编译。一方面是因为其中涉及到页码，另一方面是涉及到各个章节的标题。我们继续用上次的源代码给大家举例。

```
\documentclass{ctexart}
\usepackage[colorlinks=true]{hyperref}
\begin{document}
\tableofcontents
\section{前言}
\label{sec:introduction}
这里是说在前面的话。\LaTeX 是一个非常棒的排版工具。我在第 \ref{sec:detail} 节中会有详细的论述。
\section{\LaTeX 介绍}
\label{sec:detail}
我在第 \ref{sec:introduction} 节中提到，\LaTeX 是一个非常好的排版工具，下面进行详细的论述。
\end{document}
```

目录

1 前言 1

2 LaTeX 介绍 1

1 前言

这里是说在前面的话。LaTeX 是一个非常棒的排版工具。我在第 2 节中会有详细的论述。

2 LaTeX 介绍

我在第 1 节中提到，LaTeX 是一个非常好的排版工具，下面进行详细的论述。

目录的内容显示为红色，是因为 hyperref 宏包的 colorlinks 选项。我们以后将默认载入这个宏包，告诉大家这些红色的文字都是可以点击跳转的，这也是我非常喜欢的一个特性。

如果我们使用带星号的章节命令，这个章节将不编号也不编目。这个时候对应的标签会被打到上一个章节中去，所以如果你用了这样的章节，就不要用交叉引用啦。效果如下：

```
\documentclass{ctexart}
\usepackage[colorlinks=true]{hyperref}
\begin{document}
\tableofcontents
\section{前言}
\label{sec:introduction}
这里是说在前面的话。\LaTeX 是一个•非常棒的•排版工具。我在第 \ref{sec:detail} 节中会有详细的•论述。
\section*{\LaTeX 介绍}
\label{sec:detail}
我在第 \ref{sec:introduction} 节中提到，\LaTeX 是一个非常好的排版工具，下面进行•详细的论述。
\end{document}
```

目录

1 前言 1

1 前言

这里是说在前面的话。LaTeX 是一个非常棒的排版工具。我在第 1 节中会有详细的论述。

LaTeX 介绍

我在第 1 节中提到，LaTeX 是一个非常好的排版工具，下面进行详细的论述。

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，并编写数学公式了——这并不冲突。

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（用$$包裹） 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，并不推荐。

上下标

实例代码

```
\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}
```

$\sqrt{x}$, $\frac{1}{2}$.

$$\sqrt{x},$$

$$\frac{1}{2}.$$

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 $
\[ \lim_{x\to0}x^2 \quad \int_a^b x^2 dx \]
\[ \lim\nolimits _{x\to0}x^2\quad \int\nolimits_a^b x^2 dx \]
```

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：$\sum_{i=1}^{n} i$ $\prod_{i=1}^{n}$, $\sum\limits_{i=1}^{n} i$ $\prod\limits_{i=1}^{n}$

$$\lim_{x\to 0} x^2 \quad \int_a^b x^2 dx$$

$$\lim\nolimits_{x\to 0} x^2 \quad \int\nolimits_a^b x^2 dx$$

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

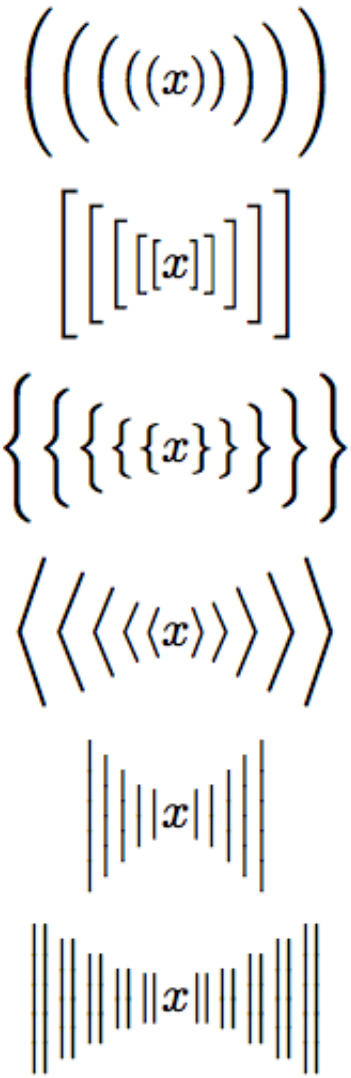
```
\[ \iint\quad \iiint\quad \iiiint\quad \idotsint \]
```

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。

为了调整这些分隔符的大小，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。

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



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。

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

$$x_1,x_2,\dots,x_n \quad 1,2,\cdots,n \quad \vdots \quad \ddots$$

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：

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

使用 smallmatrix 环境，可以生成行内公式的小矩阵。

```
I have a little matrix $ ( \begin{smallmatrix} a&b\\c&d \end{smallmatrix} )
$.
```

效果图:

$$\text{I have a little matrix } \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right).$$

数学符号

首先要跟大家说明的是数学模式中的文字规范。在数学公式中，**只有变量**才使用*意大利体*。而数学常数 e、i，微分符号 d，函数名 sin、cos、log、ln 等都需要用**罗马正体**来表示。另外表示数集的字母 R、C、Q、N 等，需要用**空心字体**来表示。还有一些特殊的**花体**，使用场合比较有限，大家可以查阅相关的资料和书籍来了解使用方式。

一般的拉丁字母大家都可以方便地用键盘输入。而数学中常用的希腊字母就不太好用美式键盘输入了。在 LaTeX 中输入小写希腊字母时，只需要用反斜杠加上字母的读音就可以输入相应的希腊字母了。比如用 \alpha 输入 α，用 \beta 输入 β 等。对应的大写字母只需要将读音的首字母大写即可（LaTeX 中提供的大写希腊字母只有11个，不过已经够用了）。通常我们还会用到希伯来字母 ℵ，可以用 \aleph 来输入。

对于特殊的符号，我们在这里作一个简单的列举，大家可以自己在电脑上尝试输入一下。

- 普朗克常数，\hbar
- 无穷符号，\infty
- 空集符号，\emptyset（也可以调用 amssymb 宏包后使用 \varnothing）
- 偏微分符号，\partial
- 积分符号，\int，\iint，\iiint，\iiiint，分别对应一重、二重、三重、四重积分；更多重积分可以用 \idotsint
- 环路积分符号，\oint
- 求和符号，\sum

- 求积符号，\prod
- 交集符号，\cap；并集符号，\cup
- 乘号，\times；除号，\div
- 不等号，\neq；小于等于，\leq；大于等于，\geq；
- 属于，\in；

另外还有很多函数名，应该用罗马正体书写的，都可以用反斜杠加函数名来实现。比如 \sin，\cos，\log，\ln。还有一些算子也需要这样实现，如 \lim，\max、\min、\gcd 等。

数学符号实在是太多了，变化也十分多样，以上只是一些非常基本的符号，篇幅所限没有办法囊括更多。大家可以去看《LaTeX 入门》的第四章了解更多数学符号的输入方式。对于一般的使用而言，上面的这些应该已经足够了。

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

1. 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：

$$x = a + b + c + \\ d + e + f + g \quad (1)$$

2. 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：

$$x = a + b + c + \\ d + e + f + g$$

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：

$$a = b + c + d \tag{1}$$
$$x = y + z \tag{2}$$

$$a = b + c + d \tag{3}$$
$$x = y + z \tag{4}$$

如果我们是要把一个公式拆分成多行来书写，并给出一个单独的编号，那么上面的两个环境就不那么适用了。在这里为大家介绍 split 环境。split 环境需要嵌套在 equation 环境中使用，也支持像 align 环境中那样的对齐方式（但不能多列对齐），举例如下：

```
\begin{equation}
\begin{split}
\cos 2x &= \cos^2 x - \sin^2 x \\
&= 2\cos^2 x - 1 \\
\end{split}
\end{equation}
```

$$\begin{aligned}\cos 2x &= \cos^2 x - \sin^2 x \\ &= 2\cos^2 x - 1\end{aligned} \tag{1}$$

请注意，永远不要使用 eqnarray 环境。原因如下：

- eqnarray 是糟糕的
- eqnarray 是有害的
- eqnarray 是恼人的
- eqnarray 是邪恶的

Avoid eqnarray!

Lars Madsen

Abstract

Whenever the `eqnarray` environment appears in a question or an example of a problem on the comp.text.tex newsgroup or the TeXhax mailing list there is a large chance that someone will tell the poster not to use `eqnarray`. This article will provide some examples of why many of us consider `eqnarray` to be harmful and why it should not be used.

Lars Madsen holds a master's degree in math from the Department of Mathematical Sciences, University of Aarhus, Denmark, where he is also currently employed as a programmer doing general user support, including quite a lot of LaTeX support.

Lars is a regular on comp.text.tex and the Danish TeX User Group's mailing list. As he witnesses a lot of 'normal' LaTeX users, Lars is a bit concerned on how new users learn LaTeX. As a result of this, he is currently working on the third edition of his 300+ pages Danish LaTeX introduction. He can be contacted at `daleif@imf.au.dk`

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。

```
Otherwise, $f_x(x)=0$. That is,
\[
f_x(x) = \begin{cases}
\dfrac{x+1}{4} & \text{if } 0<y<1 \text{ and } 0<x<2; \\
0 & \text{otherwise}.
\end{cases}
\]
```

Otherwise, $f_x(x) = 0$. That is,

$$f_x(x) = \begin{cases} \dfrac{x+1}{4} & \text{if } 0 < y < 1 \text{ and } 0 < x < 2; \\ 0 & \text{otherwise}. \end{cases}$$

cases 环境有一个重大的缺陷，就是在每个 case 中，公式都会被变成行内公式。所以在这里我们用 \dfrac 命令来强制让分式变成行间公式。如果我们想简单地解决这个问题，就需要使用 mathtools 宏包定义的 dcases 环境，用法和 cases 完全相同，但是可以以行间公式的形式来排版每个 case。

插入图片和表格

图片

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 qrcode.jpeg 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{qrcode.jpeg}
\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[scale=0.5]{qrcode.jpeg}
```

这样图片宽度会被缩放至**原尺寸**的百分之五十，图片的总高度也会按比例缩放。

> \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：

操作系统	发行版	编辑器
Windows	MikTeX	TexMakerX
Unix/Linux	teTeX	Kile
Mac OS	MacTeX	TeXShop
通用	TeX Live	TeXworks

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{qrcode.jpeg}
\caption{交大数学研究生会}
\label{qrcode}
\end{figure}
```

图 1: 交大数学研究生会

“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) 。这四种说明符可以组合使用，自左向右优先级递降。至于最后这个图表出现在哪里，我们就不需要关心了。但是可能出现的一种情况是，如果你的浮动体太多而其他内容太少，它们可能会被堆到整个文档的最后一起出现。

\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。

> 需要注意的是，表的标题应该在表的上方，而图的标题应该在图的下方。这个是排版中的一个默认的规则。 图表同样也可以进行交叉引用，只需要加个一个标签，就可以用 \ref 命令进行引用了。 利用 float 宏包，并将figure的参数换成大写的H，可以禁止浮动，文中放哪就放哪

图 1: 交大数学研究生会

这么好的公众号，你不关注一下吗?

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}
```

1 Introduction

Thanks for inviting me for this LaTeXlecture.

1.1 Graduate Student Union

Sharing activities is strongly advocated by the student union.

1.2 QRcode

This is a fantastic wechat official account!

页面风格

对于一个页面，我们可能还需要页码等页眉、页脚信息。这个时候就需要我们来定制页面风格了。使用 \pagestyle 命令可以调用三种默认的页面风格：

- plain：只有页底中央的页码；
- empty：啥也没有；
- headings：把章节的标题写在页眉处，同时有页码；

其中 headings 是默认的页面风格，依据文类的不同而有所改变，大家可以自己尝试一下。例如调用 plain 风格的命令就是 \pagestyle{plain}，记得要在导言区使用。

页眉页脚

如果需要深度定制各种页眉页脚，可以使用 fancyhdr 宏包。fancyhdr 宏包可以对页眉和页脚的左中右共六个位置显示的内容进行定义，大家可以在这里查看它的说明文档来尝试自己定制一个页面风格。

比如我希望，在页眉左边写上文章标题，右边写上活动方；页脚的正中写上页码；页眉页脚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{An Introduction to \LaTeX}
\chead{}
\rhead{Graduate Student Union}
\lfoot{left foot}
\cfoot{\thepage}
\rfoot{right foot}
\renewcommand{\headrulewidth}{0.4pt}
\renewcommand{\footrulewidth}{0.4pt}
```


1 Section

Random paragraphs Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer 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 odio placerat quam, ac pulvinar elit purus eget enim.

Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque

left foot	1	right foot

首行缩进

ctex 宏包已经处理好了首行缩进的问题（自然段前空两格汉字宽度）。

> 不使用 CTeX 宏集（使用 xeCJK 宏包）的话，请遵照以下提示操作。
>
> 中国人写文章，习惯每一段的段首都空出两个中文汉字的长度。美国人没有这个习惯，他们每一小节的段首都顶格。为了解决这个问题，我们可以在导言区调用 \usepackage{indentfirst}.
>
> 就算是这样，首行缩进的长度，仍然不符合中国人的习惯。我们可以在导言区添加这样的控制序列 \setlength{\parindent}{\ccwd} 来调整首行缩进的大小。这里的 \ccwd 是当前字号下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
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

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：http://liam0205.me/2013/10/17/LaTeX-Linespace/

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}{2em}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

则可以在原有的基础上，增加段间距 2em。如果需要减小段间距，只需将该数值改为负值即可。

模板的使用

在 LaTeX 中使用别人的模板实在是一件非常简单的事情——尤其是很多模板作者已经把文档和教程都做得很好了(丰富的注释和使用说明)。本文不会手把手教大家使用模板，而是给大家讲讲模板应该如何正确打开。

一个好的模板会按照一定的要求或者需求处理好相应内容的排版。可能有很多 LaTeX 初学者都是从模板入手的。想要用 LaTeX 写个东西，但是不会用，就找了个模板下来，套用进去，一切搞定，好像很简单。这样的傻瓜模式做一锤子买卖确实很方便，但其实有许多潜在的问题。一旦出现编译错误，一些初学者就会直接懵掉，不知道怎么处理。所以我还是推荐大家，最好能先有一些 LaTeX 基础再使用模板。这样不仅遇到问题自己心里有底，需要一些修改的时候也能自己动一下手。

寻找模板的时候，大家一定要**注意时效，越新越好**。有的时候年龄太大的模板会在新的引擎下出现一些错误，或者用非常老旧的方式来处理一些问题（比如用 CJK 来处理中文）。另外大家最好能寻找**文档齐全**，能联系上作者的模板。这样在遇到问题自己搞不懂的时候可以直接给作者发邮件询问，相对更有保障一些。

在拿到模板的时候，一定不要着急去开始写作，而一定要先阅读安装说明(README.md)和使用指南。有很多 demo 开头的文件通常都是作者写的示例，这是我们不知从何开始时最好的读物。有的时候你甚至可以直接复制一份 demo 然后在里面添加上自己的内容来作为最后的成品，编译结果不会有太大的差异，但那样无疑会增加一些工作量。

模板的 demo 其实也有很多学习价值。比如我就是前两年在这个模板中看到了非常漂亮的代码排版，然后懂得了使用 minted 宏包来排版代码。当然如果你是为了学习写模板的话，一定要找质量上乘的模板（我相信这个网站上的模板质量都过硬），不然代码质量太差的模板会把初学者引入歧途。不过我并没有这方面的经验，只是提醒想要入坑的朋友们注意。另外，模板 demo 的注释中也有很多关于模板使用的信息，demo 篇幅也不长大家可以仔细阅读一下。

这里提供一个模板下载网站，上面有许多非常实用的模板，上手也十分容易。LaTeX Studio

无关 LaTeX 的一些事情

TeX 家族

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 –）教授愤世嫉俗（大雾；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 Lang）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）。

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。

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。

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 的发行版，他们是许许多多东西的集合。

问题处理

1. 24K纯新手，建议先读完一本入门读物，了解基本的知识；
2. 无论如何，先读文档！绝大部分问题都是文档可以解决的；
3. 利用 Google 等搜索引擎你的问题；
4. 在各个论坛或者 LaTeX 交流群里聪明地提出你的问题。

推荐两本读物：《LaTeX入门》，《LaTeX 2e 完全学习手册》

CTeX论坛提问版：http://bbs.ctex.org/forum.php?mod=forumdisplay&fid=6&page=1

另外知乎专栏关于LaTeX也有不少讨论，经常关注也能学到不少。

编辑器选择

工欲善其事，必先利其器。支持编写 LaTeX 的编辑器有很多，初学者建议先从 TeXWorks 开始，等到有一定基础后可以换一些高度定制化的编辑器。那样不仅可以**大大提高效率**，而且真的会让你**爱上编写纯文本代码**！我曾经尝试过许多编辑器，例如Notepad++, Sublime Text, Atom, VScode, Vim。这里对于编辑器给出一些建议：

1. 编辑器一定要打开行号，打开行号！
2. 多使用编辑器的替换查找功能！
3. 选择一款适合自己的字体，百看不腻有木有！
4. 如果能有代码自动补全，那再好不过。
5. 没有代码补全的最好要有代码高亮。

代码风格

如何编写整洁高效的代码是有梦想的程序员应当思考的。在有一定编写 LaTeX 经验后，请好好思考一下自己的文本代码是否惨不忍睹以及如何改善自己的代码。这里同样给出几点切身建议：

1. 多使用注释！
2. 空行永远不要嫌少！
3. 能简则简，避免长命令以及命令过挤！
4. 多使用自定义命令！
5. 学习模板中高手的代码！