

# CCNUthesis 用户手册

夏康玮 林康益

kangweixia\_xdyy@163.com

2022-01-31 v0.0.1<sup>\*</sup>

## 目录

<b>1</b>	<b>计算机基础知识</b>	<b>3</b>
1.1	编译环境和代码编辑器	3
1.1.1	计算机语言	3
1.1.2	代码编辑器	3
1.1.3	发行版	5
1.2	命令行	5
1.2.1	GUI 与 CLI	5
1.2.2	Windows 10 系统下的命令行基本操作	5
1.2.3	Windows Terminal	7
1.3	重要的环境变量 Path	8
1.3.1	环境变量	8
1.3.2	环境变量 Path	9
1.3.3	发行版 C <sub>T</sub> E <sub>X</sub> 对环境变量的影响	10
1.3.4	添加到 Path	11
1.4	编码	11
1.4.1	编码——数字与文字的一一对应	11
1.4.2	乱码问题与 Unicode 编码	12
1.4.3	L <sup>A</sup> T <sub>E</sub> X 中的编码问题	13
1.4.4	更改文件的编码	13
1.5	PDF 阅读器	14
1.5.1	Adobe Acrobat Reader	14

---

<sup>\*</sup><https://github.com/xkwxdyy/CCNUthesis>

<sup>†</sup><https://gitee.com/xkwxdyy/CCNUthesis>

1.5.2	SumatraPDF . . . . .	14
1.5.3	WPS . . . . .	14
<b>2</b>	<b>L<sup>A</sup>T<sub>E</sub>X 基础知识</b>	<b>15</b>
2.1	安装相关 . . . . .	15
2.1.1	安装 T <sub>E</sub> X Live . . . . .	15
2.1.2	安装外置 PDF 阅读器 . . . . .	15
2.1.3	安装 Visual Studio Code . . . . .	15
2.2	L <sup>A</sup> T <sub>E</sub> X 知识补充 . . . . .	15
2.2.1	表格 . . . . .	15
2.2.2	选择题选项排版 . . . . .	15

# 1 计算机基础知识

开源排版系统  $\text{\LaTeX}$  与大家之前熟悉的软件有较多的不同，在使用过程中很可能遇到各种困难。没有任何编程经验的同学最好先看这个小视频『教程』学编程前必知的 8 个电脑操作，然后阅读此部分，能够帮助你。

## 1.1 编译环境和代码编辑器

常看到这样的提问“为什么我的  $\text{\LaTeX}$  界面和别人不一样？”，“非得安装 TeXstudio 吗？”，“安装完桌面为什么没有图标？”；学习  $\text{LaTeX}$  时也会碰到许多相似的概念如  $\text{\TeX}$  Live,  $\text{CTEX}$ ,  $\text{xelatex}$ ,  $\text{TeXworks}$  等，使人感到迷惑。这些问题的根源在于， $\text{\LaTeX}$  是一种计算机语言，与 Word 等软件并不一样，下面就来一一阐述。

### 1.1.1 计算机语言

人类社会有汉语，英语，法语等多种语言。然而，计算机并不能读懂人类的语言，要想指挥计算机完成各种任务，就要使用计算机语言。计算机能直接识别并运行的只有由 0 和 1 组成的二进制代码，称为**机器语言**，由于其可读性太差，科学家又创造了 C, Java 等采用贴近人类语言的语法格式（主要是英语）描述程序的**编程语言**，并开发了对应的**编译器**，可以将编程语言翻译为计算机能识别的二进制代码来运行，这个翻译的过程称作**编译**，在编译前的文件称为**源代码**。

$\text{\LaTeX}$  是一种宏语言，在排版时，通过各种各样的指令对文档的各种格式（比如字体，行距，居中，编号）进行控制，一份  $\text{\LaTeX}$  源代码中既含有需要输出在文档中的具体内容，也含有控制指令。写完源代码后，通过调用各种程序对源代码进行**编译**，最后得到排版完成的文档。

根据不同的需求， $\text{\LaTeX}$  系统在编译时可供调用的程序有不少（其中有编译器）。最初只有  $\text{\TeX}$ ，通过命令行下的命令  $\text{tex}$  调用，后来又开发了不少扩展， $\text{\TeX}$  程序连同这些扩展称为不同的  $\text{\TeX}$  引擎，常见的有  $\text{\TeX}$ ,  $\text{pdfTeX}$ ,  $\text{XeTeX}$ ,  $\text{LuaTeX}$ 。通过不同的命令，可以调用不同的引擎以不同的方式编译源代码输出不同格式的文档，这些命令称作**编译方式**，常用的有  $\text{pdflatex}$ ,  $\text{xelatex}$ ,  $\text{lualatex}$ ，分别调用了  $\text{pdfTeX}$ ,  $\text{XeTeX}$ ,  $\text{LuaTeX}$  引擎输出 PDF 文件<sup>1</sup>。

### 1.1.2 代码编辑器

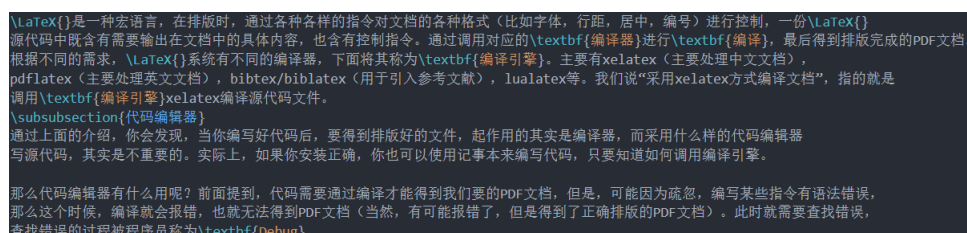
通过上面的介绍，你会发现，当你编写好代码后，要得到排版好的文件，起作用的其实是编译器，而采用什么样的代码编辑器写源代码，其实是不重要的。实际上，

<sup>1</sup>其实最早  $\text{\LaTeX}$  输出格式是 DVI，随着时代发展，后来编译时会调用  $\text{xdvipdmtx}$  程序直接输出 PDF 格式。

如果你安装正确，你也可以使用记事本来编写代码，只要知道如何调用编译引擎，选择恰当的编译方式。

那么代码编辑器有什么用呢？前面提到，代码需要通过编译才能得到我们要的 PDF 文档，但是，可能因为疏忽，编写某些指令有语法错误，那么这个时候，编译就会报错，也就无法得到 PDF 文档（当然，有可能报错了，但是得到了正确排版的 PDF 文档）。此时就需要查找错误，查找错误的过程被程序员称为 **Debug**。

如果使用记事本这种编辑器，你会发现查找错误非常痛苦，因为代码和你要排版的内容直接混在了一起，都是白底黑字。许多代码编辑器都含有**语法高亮**的功能，会把不同的代码自动变成不同的颜色，这样，查找起来就比较方便了，语法高亮的效果如图（以 Visual Studio Code 为例）。



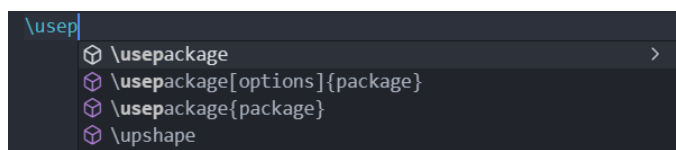
`\LaTeX` 是一种宏语言，在排版时，通过各种各样的指令对文档的各种格式（比如字体，行距，居中，编号）进行控制，一份 `\LaTeX` 源代码中既会有需要输出在文档中的具体内容，也会有控制指令。通过调用对应的 `\textbf{}` 编译器进行 `\textbf{}` 编译，最后得到排版完成的 PDF 文档。根据不同的需求，`\LaTeX` 系统有不同的编译器，下面将其称为 `\textbf{}` 编译器。主要有 `xelatex`（主要处理中文文档），`pdfelatex`（主要处理英文文档），`bibtex/biblatex`（用于引入参考文献），`lualatex` 等。我们说“采用 `xelatex` 方式编译文档”，指的就是调用 `\textbf{}` 编译器 `xelatex` 编译源代码文件。

`\subsubsection{代码编辑器}`

通过上面的介绍，你会发现，当你编写好代码后，要得到排版好的文件，起作用的其实是编译器，而采用什么样的代码编辑器写源代码，其实是不重要的。实际上，如果你安装正确，你也可以使用记事本来编写代码，只要知道如何调用编译引擎。

那么代码编辑器有什么用呢？前面提到，代码需要通过编译才能得到我们要的 PDF 文档，但是，可能因为疏忽，编写某些指令有语法错误，那么这个时候，编译就会报错，也就无法得到 PDF 文档（当然，有可能报错了，但是得到了正确排版的 PDF 文档）。此时就需要查找错误，查找错误的过程被程序员称为 `\textbf{Debug}`。

另一个重要的功能是**语法补全**，在输入各种指令时，非常容易打错，或者掉个把符号比如括号，许多代码编辑器在设定好你的编程语言后，当你输入某些指令的前几个字符时，就会自动帮你联想可能的指令（很多时候就是你要的），这样就不容易打错，并且很多时候，插入（等括号字符时，会自动成对出现，避免漏掉。效果如图



第三个方面是编译运行， $\text{\LaTeX}$  有些代码编辑器如 TeXstudio，不仅可以写代码，也提供了一键编译的按钮，如果你的编辑器没有编译的功能，就需要使用命令行编译（详见 1.2 节），还有一些功能只能通过命令行使用。这样对初学者来说上手起来就比较快。在安装  $\text{\LaTeX}$  时，一般会附带安装一个编辑器 TeXworks，同样提供了高亮，补全，一键编译的效果，但整体比较简陋，没有 TeXstudio 功能那么多，所以就会出现很多人推荐 TeXstudio，导致很多人误解安装  $\text{\LaTeX}$  就是安装 TeXstudio。

个人比较喜欢的代码编辑器是 Visual Studio Code，这是微软开发的开源编辑器，具有很强的自定义性，还有各种各样实用的插件，可根据自己的需求配置各种指令和快捷键，不过编译环境需要自己配置，可以参考这个链接：[使用 VSCode 编写 LaTeX](#)，推荐对  $\text{\LaTeX}$  有一定了解后，再更换代码编辑器提高效率。

### 1.1.3 发行版

发行版指的是 L<sup>A</sup>T<sub>E</sub>X 整个软件包的版本，T<sub>E</sub>X Live，C<sub>T</sub><sub>E</sub>X，MiK<sub>T</sub><sub>E</sub>X 指的就是发行版，通常问安装哪个版本，指的是哪个发行版。有些模版和代码只能特定的发行版下编译，比如本论文模版基于 T<sub>E</sub>X Live 2021，邓老师的旧模板基于 C<sub>T</sub><sub>E</sub>X，不同的发行版通常不能兼容，因此在一般情况下**请勿安装超过一个发行版！**

一个发行版主要有三个部分：各种编译器，许许多多的宏包，宏包的说明文档。宏包可以理解为指令集，提供了更多的排版指令，当你需要对应指令时，只要加载对应的宏包就可以调用许多方便的指令。

因此，你会发现其实安装 L<sup>A</sup>T<sub>E</sub>X，是安装这一门语言的**编译环境**，使得你可以在自己的电脑上编译 L<sup>A</sup>T<sub>E</sub>X 源文件。前面说过，如果编辑器没有编译的功能，就需要使用命令行编译（详见 1.2 节），本质是电脑运行程序的另一种方式。因此，**安装完成后，桌面没有图标！**需要自己尝试编译一个含中文的文件，如果编译成功，才能说明安装成功，或者依照手册《install-latex-guide-zh-cn》<sup>2</sup>，这个手册非常重要，会被多次提到。

## 1.2 命令行

### 1.2.1 GUI 与 CLI

我们现在的计算机操作用户界面采用图形方式显示，允许用户使用鼠标等输入设备操纵屏幕上的图标或菜单选项，这种界面称作**图形用户界面 (Graphical User Interface, 简称 GUI)**。我们平常使用电脑主要都是在这样的界面进行的，通过鼠标打开文件，运行程序，非常方便。

L<sup>A</sup>T<sub>E</sub>X 是由美国计算机学家莱斯利·兰伯特 (Leslie Lamport) 在 20 世纪 80 年代初期开发的。在那个年代，图形界面的运用并不广泛，甚至鼠标也不普及，使用最为广泛的用户界面是**命令行界面 (Command-line Interface, 简称 CLI)**，这种界面通常不支持鼠标，用户通过键盘输入指令，计算机收到指令后，予以执行。

正是因为这样，L<sup>A</sup>T<sub>E</sub>X 是按照命令行界面的用户设计的，这就是为什么无论什么编译方式（包括 *Texstudio* 的一键编译），其本质都是通过命令行运行指令调用相应的编译引擎，因此，我们需要学习一定的命令行知识。实际上，大部分的计算机语言都是类似这样，通过编译得到我们见到的软件。

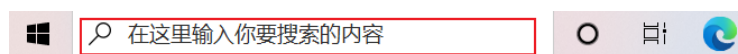
### 1.2.2 Windows 10 系统下的命令行基本操作

下面的操作均在 Windows 10 家庭版上完成，Mac 和 Linux 上的操作类似。

在 Windows 中，命令行程序是“命令提示符”或“Windows Powershell”，可以利用菜单栏的搜索框（如图）查找。

---

<sup>2</sup><https://gitee.com/OsbertWang/install-latex-guide-zh-cn>



运行时，建议右键选择“以管理员身份运行”，两个程序如图



打开命令行窗口后（以“命令提示符为例”），会显示命令提示符（图中的红方框），由当前盘符、目录（即文件夹）和一个大于号 > 组成，图中表示当前目录为 C 盘 WINDOWS 文件夹下的 system32 文件夹。

```
管理员: 命令提示符
Microsoft Windows [版本 10.0.19042.928]
(c) Microsoft Corporation. 保留所有权利。

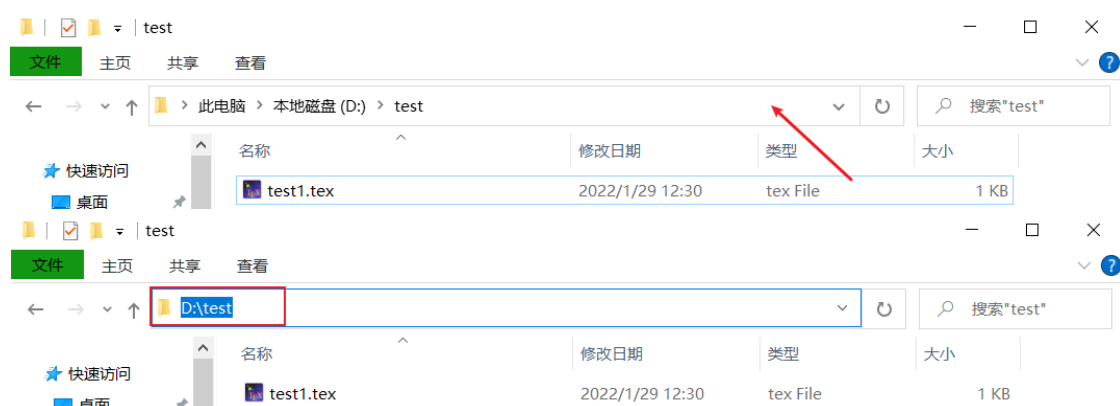
C:\WINDOWS\system32>
```

打开后在 > 的后面会有一个光标闪烁，等待输入命令，Windows 命令行命令和文件名不区分大小写，输入一行命令后按回车键执行，请注意把输入法调成英文半角，尤其是输入: 和\的时候。

下面以“用命令 xelatex -shell-escape 编译 D 盘下一个 tex 文件”为例进行演示

### Step1: 确定要编译的文件路径

首先找到你要编译的文件，在文件的目录的地址栏单击鼠标左键，被选中的部分就是这个文件的路径。



如图所示，源文件的存放路径是D:\test

### Step2: 将命令行的当前目录切换为要编译的文件路径

就像我们打开文件夹一样，只有命令行的当前目录为要编译的文件路径，才能进行编译，不然编译器根本找不到要编译的文件。

首先要从 C 盘更改为 D 盘，输入命令`D:` 后按回车键运行，会发现当前目录变成 D 盘。然后输入命令`cd \test` 即可进入 D 盘下的 test 文件夹（目录）。

```
C:\WINDOWS\system32>D:
D:\>cd \test
D:\test>
```

### Step3: 编译

输入命令`xelatex --shell-escape test1.tex` 后按回车键开始编译，`xelatex` 是调用的编译器的名称；`--shell-escape` 是编译器的设置参数，在调用外接程序时常用；`test1.tex` 是要编译的文件的文件名。请注意输入法和空格！

等到再次出现命令提示符时 (如图所示)，说明编译完成，此时可以回到原来的目录查看编译得到的 PDF 文件。

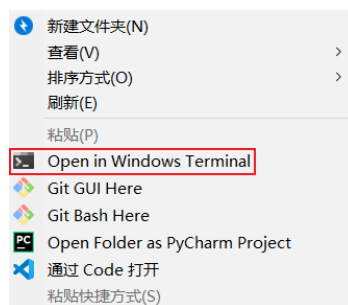
```
Output written on test1.pdf (1 page).
Transcript written on test1.log.
D:\test>
```

如果迟迟没有出现命令提示符，或者没有 PDF 文件，或者排版的效果非常奇怪，说明编写的代码有错误，需要进行排查。可以阅读当前目录下生成的`test1.log` 日志文件，也可以直接阅读命令行窗口里的输出信息进行错误排查。由于报错信息都是英语，需要随时准备使用翻译软件或者搜索引擎。

这个例子展示了命令行的基本用法，通过输入磁盘的字母和`cd` 指令进入需要运行命令的目录，再执行相应的命名，也有很多命令行命令可以直接运行而不需要切换目录。希望大家看到“命令行下运行 xx”这样的字眼时，能够不再害怕，而是尝试上手操作。

### 1.2.3 Windows Terminal

使用命令行时每次都要手动切换目录未免让人感到厌烦，Windows 10 用户可以到微软商店（Microsoft Store）下载软件“Windows Terminal”。安装完成后重新启动电脑，你会发现右键菜单增加了选项“Open in Windows Terminal”。



此时，只要在你需要进入的路径下按右键选择“Open in Windows Terminal”，弹出来的命令行窗口的当前目录就是你进入的路径！但这种方式默认不是“以管理员身份运行”，因此，在一些情况下还是需要采用1.2.2节的方法。

**总结:** 本节的操作方法适用于所有需要在命令行下运行的程序，基本方法就是先用命令 `cd+ 路径` 或者右键菜单切换到文件的路径，然后用程序名 + (设置参数) + (文件名) 的格式运行，加括号是因为有些程序不需要输入文件，或不需要设置参数。

如果你有认真读了 L<sup>A</sup>T<sub>E</sub>X 的安装手册《install-latex-guide-zh-cn》

## 1.3 重要的环境变量 Path

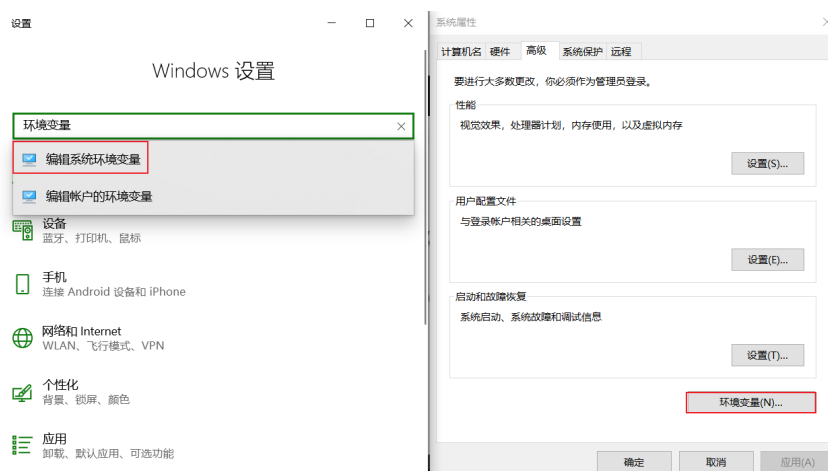
你可能会想，能不能用命令行运行其他程序比如 Word 呢？然而，当你输入 `Word.exe` 时，会得到如下的报错信息：'`Word.exe`' 不是内部或外部命令，也不是可运行的程序或批处理文件。

然而，你会发现，当你成功安装 L<sup>A</sup>T<sub>E</sub>X 后，编译器的调用命令如 `xelatex`, `pdflatex` 可以通过命令行在任意目录下使用；一些教程常常能看见这样的字眼“将 xx 添加到 Path/环境变量”使人摸不着头脑，其中的原因涉及到本节所讲的环境变量。本节内容主要参考这个视频：[【教程】什么是环境变量？](#)

### 1.3.1 环境变量

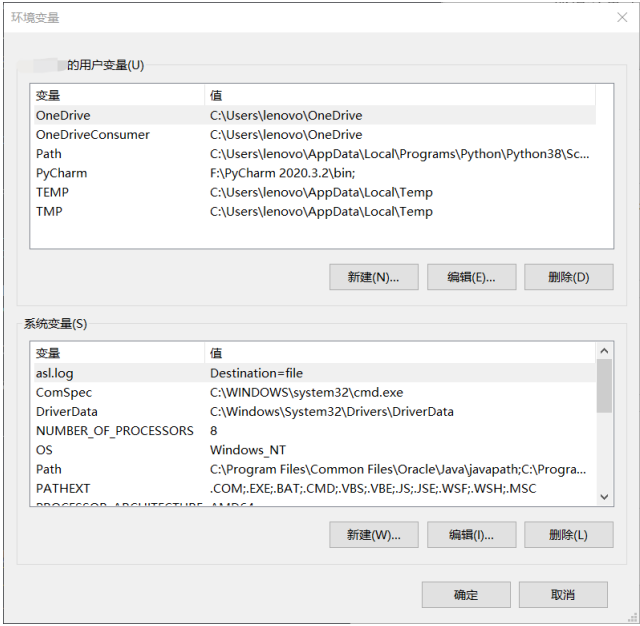
环境变量（environment variables）一般是指在操作系统中用来指定操作系统运行环境的一些参数，如：临时文件夹位置和系统文件夹位置等。它的主要作用，通俗来说就是指明操作系统的重要目录在哪里。比如，环境变量 `SystemRoot` 指明了系统目录所在的位置，在 Windows 10 的地址栏中输入 `%SystemRoot%` 后回车，会发现跳转到 C 盘的 Windows 文件夹，这正是系统的安装目录！

显然，环境变量不止一个，打开设置，搜索“环境变量”，选择“编辑系统环境变量”，在“高级”页签右下角可以看到“环境变量”





这样就打开了环境变量对话框，可以在这里进行编辑，会发现有用户变量和系统变量，用户变量只针对当前登陆的用户，系统变量针对所有使用这台电脑的人，因此一般情况下只要编辑系统变量就行了。但为了保险起见，在添加环境变量时，最好同时在系统变量和用户变量中都添加。



### 1.3.2 环境变量 Path

Path 是非常重要的环境变量，表示系统指定可执行文件的搜索路径，当在“运行”中或者命令行中输入程序的名称时，系统就会在 Path 指明的目录里搜索对应的程序，找到则运行，找不到就会报错。也就是说，如果将程序 A 放到 Path 指定的目录下，这个程序就可以随时随地通过“运行”或命令行运行！

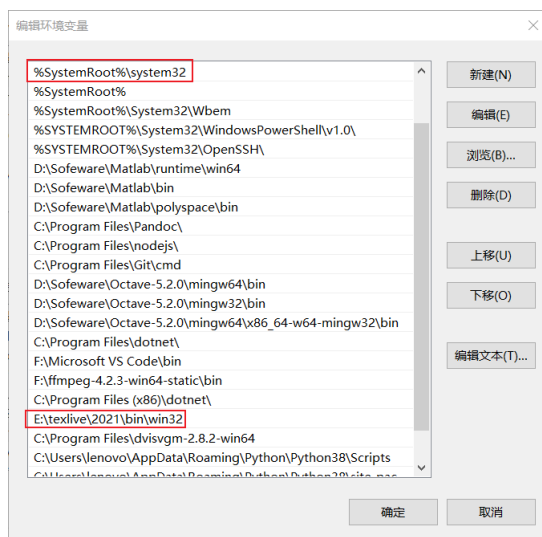


现在让我们看看 Path 的内容，单击系统变量中 Path 那一行后再单击编辑，显示出来的路径就是 Path 指定的目录。

注意图中框起来的两个路径

1. %SystemRoot%\system32 是命令提示符cmd.exe 的存放路径。

2. `E:\texlive\2021\bin\win32` 是  $\text{T}_{\text{E}}\text{X}$  Live 2021 的编译器和编译需要调用的程序的存放路径。安装盘符与安装时的选择有关，默认为 C 盘，我的电脑安装在 E 盘。



在 Path 中含有的路径下所有的可执行文件 (.exe) 都可以在任何地方通过“运行”或命令行直接运行，如果发行版  $\text{T}_{\text{E}}\text{X}$  Live 2021 被正确安装了，那么在电脑的系统变量 Path 条目中就会出现前面的路径 2。此时，在代码编辑器点击“一键编译”或者使用第 1.2.2 节的方法在命令行下编译时，计算机才能正确的到  $\text{T}_{\text{E}}\text{X}$  Live 编译器的安装路径下找到相应的编译器运行，从而编译源代码，生成 PDF 文件。

也就是说，如果环境变量没有路径 2，那么就无法在命令行下直接调用编译器，在编译源代码时会报错，说明  $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$  安装有问题或者根本没有安装。

### 1.3.3 发行版 $\text{C}_{\text{T}}\text{E}_{\text{X}}$ 对环境变量的影响

如果你曾经安装过  $\text{C}_{\text{T}}\text{E}_{\text{X}}$  现在想要更换为  $\text{T}_{\text{E}}\text{X}$  Live 那么请注意，卸载  $\text{C}_{\text{T}}\text{E}_{\text{X}}$  后，环境变量 Path 可能会丢失路径 `%SystemRoot%\system32`，请自行检查，如果丢失，则单击“新建”，手动添加该路径到环境变量 Path 中。此外，如果环境变量中有 mingw 或 jdk 相关的内容，也请暂时删除，安装之后再添加到  $\text{T}_{\text{E}}\text{X}$  Live 的环境变量的后面。

特别提醒，如果电脑里安装了 2345 好压<sup>3</sup>这个压缩软件，也会对安装构成影响，建议卸载并更换其他压缩软件<sup>4</sup>。

本部分详细内容见《install-latex-guide-zh-cn》的开头。

<sup>3</sup>警告！请务必远离所有带“2345”的软件，并且卸载时要时刻小心文字游戏！必要时借助强力卸载工具。

<sup>4</sup>推荐 7-Zip，Bandizip 6.27-6.29 或 WinRAR。

### 1.3.4 添加到 Path

如果我们想让某个程序能够通过“运行”或者在命令行下直接运行，有两种方法。第一种方法是把这个程序添加到环境变量 Path 指明的目录中比如提到过的 system32 文件夹，但这样不方便管理。第二种方法是把这个程序所在的目录加入到环境变量 Path，这就是许多教程中的“添加 xx 到 Path”。

在把一个程序添加到 Path 时，首先要找到那个程序的存放路径，方法同 1.2.2 节的 Step1，然后通过 1.3.1 节开头的操作步骤打开环境变量，并且在 Path 条目里点击“新建”，将路径粘贴进去后，先按回车键再点击确定，这样就添加完成。

有许多强力开源软件或框架只能使用命令行操作，如 Git<sup>5</sup>，Pandoc<sup>6</sup>，FFmpeg<sup>7</sup>，Hexo<sup>8</sup> 在安装时，如果有安装包，运行时就会自动添加相关的路径到 Path，如果只有运行程序，就需要自己把程序的目录添加到 Path，这之后才能通过命令行在任意路径下使用。细心的同学可以发现在 1.3.2 节的第二张图（其实是我的电脑里的 Path 的截图）中有许多的软件，比如 Matlab，Pandoc，Git，Octave，FFmpeg，Python。

实际上，与 L<sup>A</sup>T<sub>E</sub>X 相同，安装其他编程语言比如 C，Python 时，其实主要也是两步，首先把编译器放到一个指定的文件夹，然后把这个文件夹的路径添加到 Path 中，而安装包，实际上是将这个过程包装好方便用户使用！

请特别注意，有些程序的安装包默认不把路径添加到 Path，比如 Visual Studio Code 和 Python，而是在安装时作为可选项，因此安装时 千万不要无脑点击下一步！而是要注意勾选“添加到 Path（Add to the Path）”。

## 1.4 编码

当你打开某些文件时，你可能发现眼前是完全无法阅读的一团乱码（比如用 TeXstudio 打开邓老师的旧模版），这里涉及到了编码的问题，下面的介绍主要参考了这个视频：『教程』文字频频乱码，这背后是显卡的扭曲还是规则的沦丧？

### 1.4.1 编码——数字与文字的一一对应

人类社会不仅有语言，还有各种各样的文字，承载了大量的信息，但计算机内部储存的全是二进制的 0 和 1，如何在计算机中储存文字呢？

计算机最早诞生在美国，因此我们从英语开始。虽然美国人在做研究和说瞎话时用到的单词很多，但所有英语单词都是由 26 个字母组成的，只要通过设计，让数字能够代表每一个字母，也就是建立 26 个字母与 0 和 1 组成的二进制代码的一一对应关系（这是一个双射！），计算机就能处理文字了。

<sup>5</sup> 本手册的编写依靠它进行协作<https://git-scm.com/downloads>

<sup>6</sup> 强力格式转换<https://pandoc.org>

<sup>7</sup> 多媒体视频处理软件，许多软件里都有<https://ffmpeg.org>

<sup>8</sup> 基于 Git 的开源博客框架<https://hexo.io/zh-cn/index.html>

于是，当时的科学家设计了一张表，给每一个字母（区分大小写）或符号分配一个数字，这个数字称为该字符的**编码**。比如，A 在这个表上对应的数字为 65（在计算机里为二进制数 1000001）这张表还有一个名字，叫做**美国信息交换标准代码**，简称 **ASCII**<sup>9</sup>

随着时间的推移，计算机在全世界传播开来，不同的国家和地区针对自己的文字也设计了对应的编码表，比如中国设计了中文的编码表 **GBK**，还有一个近义词叫 **GB 2312**。但与英文不同，汉字的个数非常多，因此这张表很大。在中国大陆使用的 Windows 系统简体中文版，处理字符时默认采用 GBK 编码方式；中国台湾省也有一张编码表叫做 **Big5** 针对繁体中文。

#### 1.4.2 乱码问题与 Unicode 编码

各个国家和地区在设计编码表时都是相互独立进行的，因此，同一段二进制代码在不同的编码表上，几乎不可能对应相同的文字。一段文字在 A 国的电脑上保存时，A 国电脑上的 Windows 系统会默认按 A 国的编码表把每一个字符对应的编码储存到文件里，但是，当这个文件在 B 国的电脑上打开时，B 国电脑上的 Windows 系统读取这些编码时，却会默认按照 B 国的编码表反推这些编码对应的字符，结果自然而然显示出一堆乱七八糟的无意义文字，这种现象就叫做**乱码**！

不同国家和地区间交换文件，不同操作系统间交换文件，都有可能出现类似的乱码情况。**乱码的本质，就是对于相同的数字编码，却用不同的编码表反推对应的字符**。这样得到的结果当然不一样，就像破译密码时拿错了密码本。

显然，乱码问题阻碍了国际交流，一个国际组织另起炉灶，设计了一张特别大的表，叫做 **Unicode**，这张表足够大，可以包含全人类所有的字符，这样，只要采用这张编码表，就不会出现乱码的问题了。但在对字符分配编码的方式上，出现了许多标准，其中有一种方式是目前的主流，叫做 **UTF-8**。

不同的编码还给编程带来了许多麻烦，你也许听过这样的说法“安装 xx 时，路径不能含有中文”，“在编程时，文件夹和源代码都不能含有中文”，“系统用户名不能含有中文”。这些问题的根源都在于简体中文版 Windows 系统默认采用 GBK 编码，而一些程序可能不支持 GBK，这就产生了各种各样的问题！比如在安装 T<sub>E</sub>XLive 2021 时，如果系统用户名含有中文，就可能导致无法安装，解决方法还是参见《install-latex-guide-zh-cn》的 1.1.3 节。

然而，Mac 系统却很少出现上一段的这些麻烦的问题，这是因为 Mac 系统默认采用 UTF-8 编码，兼容性好，因此 Mac 是非常适合编程的电脑。一些厉害的程序员使用 Mac 系统编程，某些大型互联网公司甚至提供 Mac 作为办公电脑。

---

<sup>9</sup>American Standard Code for Information Interchange

### 1.4.3 L<sup>A</sup>T<sub>E</sub>X 中的编码问题

许多编程语言都对源代码采用的编码方式有要求，否则无法正常编译，比如 Python 要求代码采用 UTF-8 编码方式。L<sup>A</sup>T<sub>E</sub>X 比较特别，与发行版，编译引擎以及是否输入中文有关。

最早的 T<sub>E</sub>X 系统只支持 ASCII 编码，但可以通过设置使得扩展 ASCII 编码能正常排版。在排版英文文档时，可以直接使用 pdf<sub>l</sub>atex 方式进行排版。

为了使得 L<sup>A</sup>T<sub>E</sub>X 能够排版中文，Werner Lemberg 开发了 CJK<sup>10</sup> 宏包。由于汉字编码方式 GB 2312, GBK 和 UTF-8 都是兼容 ASCII 的编码，通过 CJK 宏包，可以把多个字符对应到一个汉字上，支持中文的排版，在实际使用时，只要加载了 CJK 宏包，就可以使用 pdf<sub>l</sub>atex 方式进行排版。发行版 C<sub>T</sub>E<sub>X</sub> 支持这种方式排版中文文档，同时也是邓老师的旧模板采用的处理方式。

这种处理方法要求源文件采用 GBK 编码方式，存在许多问题，简单列举几个：

1. CJK 宏包的这种方法是一种黑客手段，没有彻底解决中文排版的问题，采用这种方式编译得到的 PDF 文件，其中的中文字符复制出来全是乱码！许多老师用 C<sub>T</sub>E<sub>X</sub> 做的课件，里面的汉字都无法正常复制。
2. 上面的这个问题，导致 17 级的同学在论文查重时，查重报告乱码！
3. C<sub>T</sub>E<sub>X</sub> 目前只支持 Windows 系统，旧模板无法在 Mac 和 Linux 系统上使用！
4. C<sub>T</sub>E<sub>X</sub> 已经停止更新，论坛已经关闭，许多遗留的 bug 没有修复，新的功能无法使用，网上正确且有效的资料少（英文资源几乎为零）！

更多问题详见 2018 年，为什么不推荐使用 C<sub>T</sub>E<sub>X</sub> 套装了。

新排版引擎 X<sub>Y</sub>T<sub>E</sub>X 和 LuaT<sub>E</sub>X 都直接支持 UTF-8 编码，新的中文排版方式应运而生，孙文昌开发了 xeCJK 宏包，配合 xelatex 方式进行中文排版，后来又出现了功能更强大的 ctex 宏集。新模版使用的就是这种方式，源文件采用 UTF-8 编码方式，利用 ctex 宏集，xelatex 方式进行编译。解决了原有方式存在的许多问题。

因此，新旧模板的源代码的编码方式不同！但主流通用的代码编辑器默认都是 UTF-8 编码，比如 TeXstudio, Visual Studio Code。因此打开旧模板时，会直接乱码，要么采用记事本等文本编辑器，要么更改文件的编码，这就带来了许多麻烦。

### 1.4.4 更改文件的编码

通过上面的内容我们知道，当遇到乱码时或者编程语言对编码有要求时，我们都需要更改文件的编码方式。

---

<sup>10</sup>CJK 实际上是中日韩三国语言英文单词的首字母

## 1.5 PDF 阅读器

PDF, 全称为 Portable Document Format, 意为“可携带文档格式”, 是由 Adobe Systems 用于与应用程序、操作系统、硬件无关的方式进行文件交换所发展出的文件格式。这种格式最大的优点在于, 不论在任何操作系统, 手机还是电脑, 最终的显示效果都是统一的, 这就极大的方便了资料的传阅。 $\text{\LaTeX}$  代码在成功编译后, 得到的便是 PDF 文件, 下面介绍几个 PDF 阅读器, 用于满足两个需求, 一是在写论文时随时编译查看效果, 二是查看最终效果。

### 1.5.1 Adobe Acrobat Reader

Adobe Acrobat Reader 是由 PDF 格式的设计公司 Adobe 推出的一款免费的 PDF 阅读器 (请与收费的 Pro 版区分!)。这个阅读器的显示效果最好, 且支持 PDF 的所有功能 (如 JavaScript 脚本、动画、3D 对象等),  $\text{\LaTeX}$  的一些高级功能的效果只有使用这个阅读器才能完全显示。并且可以查看 PDF 的许多属性比如使用的字体, 在精细排版中会用到。

但有两个缺点, 首先, 安装时会强制安装在 C 盘, 所以要记得腾空间。其次, *Adobe Acrobat Reader* 会锁定 PDF! 在 Windows 中重复编译时, 要先关闭已经由 *Adobe Reader* 打开的 PDF 文档, 否则 PDF 文件会被锁定而不能更新 (同时会报错), 因此一般在写论文途中采用其他的 PDF 阅读器。

下载地址:<https://www.adobe.com/cn/acrobat/pdf-reader.html>

### 1.5.2 SumatraPDF

SumatraPDF 是一个很小的开源 PDF 阅读器, 具有免安装版 (Portable), 可以放在 U 盘随身携带, 并且打开速度非常快, 适合在写作途中查看文档效果, 可以通过设置使得 PDF 可以反向搜索 (定位到对应的代码)。当然, 如果只是预览, Texwork 编译后会自动弹出预览窗口。

下载地址:<https://www.sumatrapdfreader.org/download-free-pdf-viewer> 或<https://sourceforge.net/projects/sumatrapdf-reader.mirror/>

### 1.5.3 WPS

WPS 就不需要过多介绍了, 与 Office 不同, 它不强制安装在 C 盘。WPS 包括了文字 (Word), 表格 (Excel), 演示 (PPT) 和 PDF 四个功能, 安装好之后就可以直接阅读 PDF 文件。顺带一提, WPS PDF 可以给没有书签的 PDF 文档手动加书签 (不需要会员!), 对于习惯使用电子书的同学会方便很多。

下载地址: <https://www.wps.cn/>



## 2 L<sup>A</sup>T<sub>E</sub>X 基础知识

### 2.1 安装相关

因为没有用过 Linux 系统，所以下面仅针对 Windows 系统和 Mac 系统展开，Linux 系统是类似的。

#### 2.1.1 安装 T<sub>E</sub>X Live

安装必读的中文官方文档：[install-latex-guide-zh-cn](#)

这篇文档已经千锤百炼，非常成熟了。其中 Windows 用户遇到的问题是最多的，Windows 用户一定要先把 **Windows** 系统的部分完整认真读完再去跟着安装，要心里有数，自己的电脑是个什么状态。

Mac 用户的就很简单了，pkg 包下载直接一键安装即可。

#### 2.1.2 安装外置 PDF 阅读器

其实不妨可以先试试编辑器自带的阅读器，如果觉得体验不好再下载也不迟。

前面介绍过 SumatraPDF，这是 Windows 用户推荐使用的，而 Mac 用户推荐使用的是 Skim 阅读器。

#### 2.1.3 安装 Visual Studio Code

虽然你安装完 T<sub>E</sub>X Live 后会有一个编辑器，但是个人是非常推荐使用 Visual Studio Code 的，有几个理由：

1. Visual Studio Code 打开速度快，比 T<sub>E</sub>Xstudio 是肉眼可见地快。
2. 配置非常简单，下载一个插件 LaTeX Workshop，把别人弄好的配置文件代码复制粘贴后即可使用。
3. 详细配置教程详见 [使用 VSCode 编写 LaTeX](#)。

### 2.2 L<sup>A</sup>T<sub>E</sub>X 知识补充

#### 2.2.1 表格

#### 2.2.2 选择题选项排版