LaTeX 文档化解决方案

水

2021 年 1 月 21 日

摘要

你是否还在为 LaTeX 的排版烦恼？你是否还在为 LaTeX 敲打公式烦恼？你是否还在为 LaTeX 作图烦恼？你是否还在为 LaTeX 表格烦恼？··· 这里为你提供了一整套的解决方案，各位看到就是赚到，看一看，瞧一瞧，吃不了亏，上不了当，花一天的时间阅读摸索，此后你就是 LaTeX 的王者。

一套将 PDF 或者 word 转为 LaTeX 文档的解决方案，方便诸君。

目录

1 利器善事

古人云：“工欲善其事，必先利其器”，一套良好的配置往往事半功倍。TEX 的编译器，单线程工作，对单核性能要求较高，在选购电脑时 CPU 主频不宜过低。硬盘的读写很大程度上影响文档编译的时间，推荐上 nvme 固态硬盘，另外在相同的配置下，Linux 下的 I/O 性能更好，推荐使用 Linux 系统，本套解决方案都是在 Linux 下完成，使用 windows 的用户请自行参考修改。

选择一个多功能的鼠标极大有益于提高工作效率，我使用的是罗技 G502，主要优点如下：

- 共有 11 个按键，方便配置各种不同的功能。
- 板载内存，一次修改，多处使用，不同配置，随时切换。
- 滚轮有极速模式，方便快速浏览。

注意，罗技的鼠标官方没有提供 Linux 下的配置软件，有两种解决方法，一是在 windows 下修改好后再在 Linux 下使用；二是在 Linux 下使用 piper 软件来设置。关于 piper 软件的安装和使用请参看 piper 官方文档的说明。图 1 是 piper 软件的使用截图。

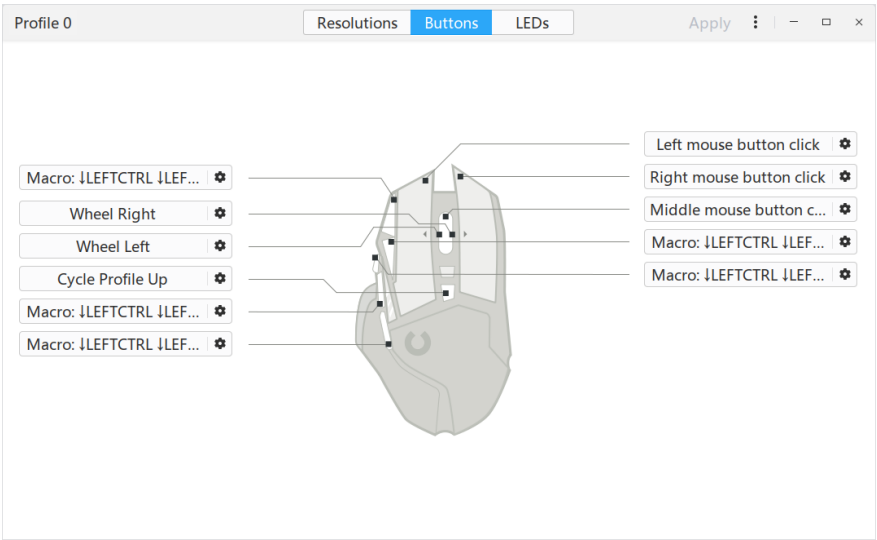

图 1: 罗技鼠标的设置

在各种文档的转化里属扫描版的 PDF 文档最难，网上提供的扫描版质量参差不齐，最好购置一台专业的扫描仪，获得原始的高清文档，一般要求是分辨率 600dpi，彩色扫描。另外，由于 TEX 文档需要不时编译查看效果，显示器需要足够宽大；还有后面由于需要截图，而截图的分辨率基本取决于你显示器的分辨率，故能上 2k 或 4k 的显示器更好。[1]

[1] 当你的显示器分辨率不够，又需要获得高分辨率的截图时，若显卡可以的话，可使用 xrand 修改输出分辨率，再对输出的图像采用缩放操作即可。

2 基本环境

2.1 TEXLive 2020 安装

下载 TEXLive 2020 套件的镜像，可以在各个镜像站中直接下载，例如：

- 清华镜像:https://mirrors.tuna.tsinghua.edu.cn/CTAN/systems/texlive/Images/
- 阿里云镜像：https://mirrors.aliyun.com/CTAN/systems/texlive/Images/

下载好后挂载镜像文件，到挂载的目录执行 sudo ./ install -tl 即可。安装好后，默认目录在 /usr/local/texlive/2020/ ，可执行程序在 bin 目录里。这里不建议修改环境变量，后面需要使用 TEX 时直接指定目录即可。有一些程序运行依赖 TEX，安装的时候直接从源里面安装即可，源里面安装的 TEX 与我们自己安装的 TEX 互不干扰。

安装完成过后，由于 TEXLive 2020 有一些宏包没有收录进去，需要我们手动安装，以 pgfornament 宏包为例，从 CTAN 下载后解压，然后执行如下命令：

```
mkdir -p /usr/local/texlive/2020/texmf-dist/doc/latex/pgfornament
mkdir -p /usr/local/texlive/2020/texmf-dist/tex/latex/pgfornament
mkdir -p /usr/local/texlive/2020/texmf-dist/tex/generic/pgfornament
cp -r doc/* /usr/local/texlive/2020/texmf-dist/doc/latex/pgfornament/
cp -r latex/* /usr/local/texlive/2020/texmf-dist/tex/latex/pgfornament/
cp -r generic/* /usr/local/texlive/2020/texmf-dist/tex/generic/pgfornament
    /
/usr/local/texlive/2020/bin/x86_64-linux/mktexlsr
```

上面最后一句命令是刷新 TEXLive 的缓存，让其识别我们手动安装的宏包。[2]

2.2 TeXStudio 编辑器

后端的环境安装好后，我们前端的编辑器就该登场了。这里推荐 TeXStudio ，基本功能应有尽有还带有强大的宏（Macros）。之前一位印度小哥分享了他在数学课上使用 LATEX and Vim 飞速记笔记的文章 。他速度的提升得益于 vim 中的 snippets 功能插件，然而在 TeXStudio 中，我们借助其中的宏，完全可以实现 snippets 的功能，TeXStudio 的宏可使用 javascript 语言编写，触发条件为正则表达式，也可以自定义快捷键，更为强大。唯一的缺点是没法像 vim 那样使用数字定义优先级，当宏的数量太多时，调整起来较为麻烦。

编译器推荐设置为 xelatex ，对中文更加友好。对应的设置命令为：

```
"/usr/local/texlive/2020/bin/x86_64-linux/xelatex" -synctex=1 -interaction
    =nonstopmode --shell-escape %.tex
```

设置好后，可以在 TeXStudio 里面跑一个例子，file → New From Template 选择其中一个模板，编译一下，再调用 \usepackage{ctex} 宏包，在文档里面输入中文，再编译一下，若没有问题则基本环境准配置就完成了。

[2]若没有刷新 TEXLive，即使我们安装成功，在调用的时候依就会报缺少该宏包的错误。

3 素材准备

基本环境配置好后，整个项目只是处于一个可用状态，想要达到一个好用状态还需更多的配置，下面这些配置是提升我们效率的关键。

3.1 PDF 相关

在将 PDF 文档转化为 LaTeX 文档的过程中，原始 PDF 文档清晰度越高越好，一来利于 OCR 的识别，二来利于图片的获取。一般要求 600dpi 的彩色扫描，可用专业的扫描仪扫描获得，这里推荐拆书扫描，这样图片的失真最少，光照效果更好。条件有限的，可用补光灯和手机搭配，这个需要光线合适均匀，手机成像质量要高，后期强化修剪。

一般扫描好的 PDF 文档我们还不能直接拿来用，后期的处理也是关键。先将我们的 PDF 转变为图片格式，在转成图片的时候最好直接原图提取，不要将其转变为其它的格式，以免质量损失。一般扫描仪得到的 PDF 文档为 jpg 格式，推荐使用 pdfimages 工具，快速高效。

```
pdfimages -j -f 3 -l 12 document.pdf doc #各参数意思可用 --help 参看
```

如果你直接扫描成单张图片的形式，上面那步请忽略。下面我们需要对每张图片进行处理，这里可用 GIMP 处理或者用 ImageMagick 处理，GIMP 是 GUI 操作而 ImageMagick 是命令行操作，编写脚本做批处理首选 ImageMagick 。下面为大家推荐两款处理扫描文档的利器：ScanTailor Advanced 和 comicenhancerpro ，前者是一款开源软件，后者是马同志为处理漫画而编写的，在 Linux 下用 wine 可完美运行。

综合运用上面的工具，完成去水印、洗白、切割、分割等步骤后一款高质量的扫描文档变诞便诞生。再顺便用 OCRmyPDF 工具嵌一个文字层，用 pdftk 添加书签，这样基本就完美了。

3.2 OCR 识别

我们的目标是将其转变为可编辑 LaTeX 的文档。当我们拿到一个 PDF 文档后，若这是一个文字版 PDF，则需要另外特别处理，这里暂且不论。但是一个图片档的 PDF 是，首先需要获取里面的文字内容，也就是 OCR，我这里使用的是百度的 API。用里面的高精度文字识别作为一般文字的 OCR，用里面的表格识别做表格的 OCR。

相关的脚本已写好，分为批量识别和手动识别两种，具体查看项目里面 OCR 的部分。当获得 OCR 的文本文档后，需要对其进行格式化，这里的格式化包括特殊字符、公式、上标、下标、分行、标点、特殊环境镶嵌等等，现阶段对语文、物理、数学、化学、英语采用不同的策略，依靠正则表达式来完成相关任务，相应的脚本已写好，请查看项目中 OCR 格式化相关部分。

这里推荐添加环境变量，这样可以直接在各处使用。直接在家目录下的 .bashrc 文件中加入如下代码即可：

```
if [ -d "$HOME/.config/autokey/myscript/bin" ] ; then
    PATH="$HOME/.config/autokey/myscript/bin:$PATH"
fi
```

3.3 图片处理

一个文档中，往往包含大量的图片，需要讲这些图片嵌入到 LaTeX 文档之中。图片可简单的分为实物图和示意图，其中的实物图亦即是各种实物照片，这一部分的处理简单粗暴直接截图插入，后期的话可用 ImageMagick 批量作处理一下背景，让其透明化。

实际文档中，特别是理工科的文档，包含大量的示意图，对于这一类图片的处理我们的目标是将其变成矢量图片且是可以编辑的。因为对于文档公式中的字母符号之类经常要作改动，而图片中相应的字母和符号也要作改动，能够随时编辑这是刚需。对于各种题目来说，变动题中的已知条件，需要改动相应的图片，图片的编辑要求也是必不可少。另外，将其变成矢量图片，缩放不失真，视觉效果很好。

对于示意图而言，又可以分为黑白和彩色两个，对于黑白和彩色分别编写了不同的脚本，将其转变为 svg 形式的矢量图片，再插入文档中去。各位写插图一般使用 tikz ，tikz 画图不太习惯，日常使用 inkscape 作图，再以 svg 的形式插入文档中。为此有专门的转换脚本、编辑脚本、创建脚本等，具体请查看项目中 svg 相关部份。

作图软件有如下几个推荐：

- tikz
- asymptote
- inkscape

tikz 与 LaTeX 文档结合较好，有很多现成的模块可以直接使用，方便快捷。缺点是编程不便，精度较低，三维不便，函数不便。而 asymptote 是一款以编程的语言设计的矢量数学作图语言，缺点是第三方模块较少，与 LaTeX 文档结合稍有麻烦。inkscape 可视化操作，入门上手简单快捷。

3.4 表格处理

在 LaTeX 中编写表格，特别是复杂表格简直如同噩梦一般，查看了网络上的各种转换脚本，有 excle 中用 vba 写的，有用 Python 写的，有用 Java 写的等等，都不尽人意，最后依靠 Gnumeric 的转换功能实现了复杂表格的编排和与文本的锲合，基本完美[3]。相应的表格处理脚本请查看项目中的表格相关部分。

3.5 文档格式

在 LaTeX 文档的编写过程中，一个好的模板是必不可少的。一本书籍的编排主要涉及到如下部分：

- 封面
- 前沿
- 目录
- 序章
- 正文
- 附录
- 答案
- 参考文献
- 名词索引
- 版本记录
- 试卷生成

每一个部分都有每个一部分的样式，大家可根据自己的喜好逐一定制。这里我们提供了一个模板，这是个大杂烩，综合了大量网上的内容，在此感谢各位网友的分享。另外，这里面有很多写法都不规范，很散，没有做相应功能的封装，后期会逐步完善。其中独立实现了如下内容：

- 选择题（二选、三选、四选、五选、六选、图像选择）
- 对页边注的手动控制
- 页眉页脚（复刻）
- 页眉页脚诗句
- 参考答案
- 常见仪器读数示意图绘制

具体内容请参考项目中的样式控制模板文件。已知问题：虽然可以正常编译使用，但还是有大量的警告信息，代码不够规范，有些问题没有考虑全面，后期有时间了一个个解决。

[3]官方的 Gnumeric 转换功能有点小问题，需要修改源码重新编译一下，各位可自行修改编译或者用我们编译好的。

3.6 辅助部分

3.6.1 公式识别

这里推荐 Mathpix snipping Tool 软件，现阶段增加了中文的识别，亦可以当作 OCR 识别小助手用。此软件为订阅制，需付费使用，若愿折腾可用其提供的 API，每个账号每月有 1000 次的试用。

3.6.2 剪贴板管理

在实际的工作中，需要用到之前粘贴复制的内容，这里一款剪贴板管理软件就尤为必要，这里给大家推荐 CopyQ 。

3.6.3 PDF 查看

我们的源文件是 PDF 版本，一款好的查看器至关重要，这里推荐 llpp 查看器，基于 mupdf ，支持键盘操作，各种自定义。

3.6.4 TeXStudio 宏

根据自己的情况配置自己常用的操作，项目里面的配置文件，是我自己日常使用的，配置规则上百个。建议各位不要拿着别人的用，每个人的操作习惯和思维习惯不一致，只需要看别人是怎么实现某一个功能的就可以了，具体该设置什么条件触发，看自己的喜好，习惯不是适应，而是培养。

3.6.5 选择题格式化

模板里面四个选项的选择题的命令是 \fourchoices{}{}{}{} ，带有四个参数，分别是 A、B、C 和 D 四个选项的内容。我们不可能一个个手动填写，为此写了一个脚本来干这个事，能根据对应的规则自动生成我们需要的格式。详情请查看项目中的选择题部分。

```
\fourchoices
{something}
{something}
{something}
{something}
```

里面的脚本提供了各种选择题（二选、三选、四选、五选、六选）的自动化处理。

3.6.6 快捷键管理

在实际操作的过程中，会有大量的快捷键需要使用，给大家推荐 AutoKey 。这款软件支持 Python 语法，可以利用 Python 完成各种操作，另外更加强大的是支持使用正则进行窗口过滤，同一个快捷键，可以在不同的窗口实现不同的功能。

3.6.7 Rime 输入法引擎配置

可对 Rime 输入法的词库或者快捷短语进行配置，将常用的命令载入。

4 动土开工

将前面的各种东西准备好后，就可以正式开始一个项目了。这里还需要强调一点，我们文件的结构方式，一个项目文件夹下包含四个子文件夹。例如本篇文章的目录结构情况如图 2 。其中 1 放子文件，date 当参考文献和其它，picture 放图片，picture 下放位图，svg 里面放矢量图片，table 下放表格文件。

图 2: 文档结构

其中 svg 和 table 两个目录下的 name.tex 文件，是名称存放文件，可用如下命令批量生成

```
seq -f "name-%04g" 1 9999 > name.txt
```

各个脚本在编写过程中也是按照如此目录设计，若需改动目录结构，请将脚本中的目录一同改动。

至此，即可。Happy LaTeX!