实验报告一

Latex & Git

潘奕霖

https://github.com/PLY1024/system-develop-tools
2024 年 8 月 28 日

目录

1 LaTeX 注意点

LaTeX 相比 markdown 更具有上手难度，从一开始就需要记住代码，而且写错了就直接报错，比较有挫败感。不过比较让人欣慰的是在记住基本代码和折腾好格式之后，LaTeX 基本上不需要在排版上耗费精力，不用像 Word 那样和无形的大手对抗，除了不能实时预览外，基本没什么缺点。而且格式模板的使用十分简单，有模板基本能做到 Word 那样的快速编辑。

在学习中遇到的最大的坑是 Overleaf 作为在线编译器，为编译中文而修改编译方式需要自行寻找一番，而且使用 `https://oi-wiki.org/tools/latex` 中的导入包方式会直接报错，看起来 Overleaf 和 utf-8 有点矛盾。

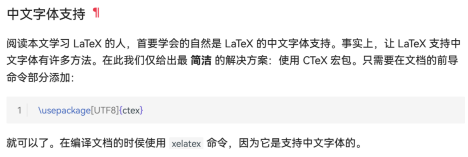

图 1: LaTeX 文档中使用中文的方法

图 2: Overleaf 的报错信息

解决方法就是删掉 [utf-8]，或者文档类型使用 ctexart。

Overleaf 上比较折腾的是换字体。我个人不太喜欢默认的等宽字体，所以想着换成别的字体，但是上传字体之后总是一堆警告[6]，让人很不爽，于是作罢。

LaTeX 的自定义功能还是比较强大的，通过引入 fancyhdr 包能够自定义页眉和页脚。不过要注意在自定义的命令之前要先有`\fancyhead{}`这样的空命令以清空原有的设置，否则会发生冲突。通过指定 R、C 与 L 可以自定义页眉页脚不同位置的信息。我让右上角显示章名，左下角显示笔记仓库，右下角显示页码。值得一提的是显示章名的命令为`\leftmark`，而显示节名的命令为`\rightmark`，不要用错了。

以下是我所使用的包：

```
\fancyhead{}
\fancyhead[R]{\textsl{\leftmark}}
\fancyfoot{}
\fancyfoot[L]{https://github.com/PLY1024/System-develop-tools}
\fancyfoot[R]{\thepage}
```

图 3: 自定义页眉页脚

```
% \usepackage{ctex}
\usepackage{graphicx} % 图片
\usepackage{minted} % 代码
\usepackage{listings} % 代码
\usepackage{enumerate} % 有序列表
\usepackage{float} % 设置位置
\usepackage{hyperref} % 网页链接
\usepackage{fancyhdr} % 页眉页脚
```

图 4: 使用的包

2 LaTeX 知识点

2.1 LaTeX 安装

可以在https://www.tug.org/texlive/下载 TeX Live。LaTeX 有众多的编译器可以使用，我选择 overleaf 作为编辑器，主要是其基于云端，基本开箱即用，并且能够实时编译，极大地省去了安装本地编译器的折腾过程。而且 Overleaf 可以辅助插入图片和列表，极大地省去了写命令的过程。

2.2 源文件的基本结构

分为导言区和正文区。导言区设定文档的格式，语言，导入需要的包。正文区用于显示文字、图表、公式等内容。

2.3 内置文档格式

LaTeX 内置了许多文档格式：

- article 文档
- report 报告

- book 书籍
- beamer 幻灯片

2.4 设置中文

这是中文使用者使用 LaTeX 遇到的最大困难之一，需要一定的操作，且网上的操作大多针对本地编译器，不一定对 Overleaf 生效。

在 Overleaf 中，点击左上角菜单，编译器选择为 XeLatex，这是最为重要的一步，再导入 ctex 宏包，即可对中文进行编译。

若不导入 ctex 宏包，将文档类型设置为 ctexart、ctexrep、ctexbook 等格式也可以对中文进行编译。

注意，编译中文最重要的一步是修改编译器。还要注意的是两种方法会对文档的显示效果产生影响，ctexart 的页眉会显示节名与页码，而普通 article 的页眉不会显示信息且页码位于页面底部。本报告使用的是 ctexart 类型的文档。

2.5 消除“does not contain requested Script “CJK”.”警告

Overleaf 编译中文时会有非常经典的“CJK”警告，原因是 Fandol 字体没有 GSUB 表。通过在设置文档类型时指定编译平台的字符集，LaTeX 就能使用平台自带的字体，避免 Fandol 字体的警告。Overleaf 运行于 Ubuntu，所以指定 [fontset=ubuntu]。

2.6 设置纸张大小和主要字号

`\documentclass[a4paper, 10pt]{article}`将纸张大小设置为 a4，主要字号大小设置为 10pt。

2.7 设置标题，作者，日期与目录

设置标题：

```
\title{}
```

设置作者：

\author

设定日期为今日：

\date{\today}

最后需要\maketitle以使标题生效，且以上的字体均可以改变字形和大小。在\maketitle后加上\newpage可以让标题单独成页。

在使用\tableofcontents后 LaTeX 会自动生成目录，同样可以单独成页。

2.8 设置章、节、小节

- \section{} 设置节
- \subsection{} 设置小节
- \subsubsection{} 设置小小节
- \chapter{} 设置章
- \part{} 设置部

注意，chapter 与 part 需要在 book 类型的文档中使用。

2.9 设置换行

偶然发现\\可以将文本换行，就像
这样。

2.10 设置分段

在文段间空一行可以实现对文字的分段，在文段中回车并不能将文段分段。

2.11 自定义字体

将字体文件上传至.tex 文件所在文件夹，在文档开头添加以下命令：\setmonofont{MapleMono.ttf}就可以自定义等宽字体。但是 MapleMono 字体在报告中显示效果不佳，本报告就使用默认的等宽字体。同样的，将字

体文件上传后在文档开头添加`\setCJKmainfont{}`也可以修改默认字体，但是造成一堆 warning。

图 5: 更换字体的效果

图 6: 一堆警告

2.12 字体类型设置

若不使用自己添加的字体，可以通过改变字形以在 LaTeX 中显示不同的字体。

2.12.1 粗体

`\textbf{}`，**这是粗体**，word.

2.12.2 斜体

\textit{}，这是斜体，*word*.

2.12.3 无衬线字体

\textsf{}，这是无衬线字体，word.

2.12.4 等宽字体

\texttt{}，这是等宽字体，`word`.

2.12.5 罗马字体

\textrm{}，这是罗马字体，word.
可以看到，所谓罗马字体就是默认字体。

2.12.6 使用下划线

\underline{}，这是下划线，在这里非常不恰当地将下划线分类到字形类型设置中。

2.13 字号设置

通过\large等命令来设置字体的大小，这就是随意设置的效果。

2.14 显示 LaTeXLogo

使用 \LaTeX命令显示 LaTeXLogo，类似的，还可以显示 LaTeX 2$_\varepsilon$，TeX 等 Logo。

注意，命令的前后一定要各留有一个空格，否则会有奇怪的报错产生。

2.15 使用代码

\begin{verbatim} \end{verbatim}与\verb|| 命令能够将其内容以纯文本形式显示，因此可以用于显示代码。

通过引入 minted 和 listings 包，也能实现显示代码的效果，不同的是 minted 包通过在指定编程语言后能够支持代码高亮，因而显示效果更佳。

2.15.1 行内代码

- \verb|| print("Man!")
- \mintinline{Latex}|| print("Man!")

2.15.2 代码块

通过\begin{verbatim} \end{verbatim}显示纯文本的代码块:

```
import time
print("Man!")
```

通过以下代码显示带高亮的代码块:

```
\begin{listing}[htb]
    \begin{minted}{python}


    \end{minted}
\end{listing}
```

```python
import time
print("hello")
```

\mint还能显示单独成行的代码。

2.16 列表

2.16.1 无序列表

使用\begin{itemize} \end{itemize}设定列表环境, 在 \item后接列表内容，还可以自定义行标。

- 列表内容 1

* 列表内容 2

2.16.2 有序列表

使用\begin{enumerate} \end{enumerate}设定有序列表，同样可以自定义行标，不过需要引入 enumerate 包。

1) 列表内容 1

2) 列表内容 2

2.17 显示图片

需要引入 graphicx 包，用\begin{figure} \end{figure}设定图片环境，通过\includegraphics[width=0.5\textwidth]{path}显示图片并设置图片大小。可以用\centering设置图片居中，\caption{title}设置图片标题，注意图片的标题需要位于图片的下方。

图 7: 标题需要位于图片的下方

2.18 显示公式

显示行内公式：$equation$，如质能方程 $E = mc^2$。
\begin{equation} \end{equation}设定公式环境或简写为\[\]。

$$E = mc^2$$

复杂公式

$$\frac{d = k\varphi(n) + 1}{e}$$

\frac{}{}也可以显示分数。

可以借助在线公式网站https://editor.codecogs.com/辅助公式的编辑.

2.19 显示表格

\begin{table}[h] \end{table}设定表格环境，\caption{title}设置表格标题，在\begin{tabular}{c c c} \end{tabular}之间设置表格内容。表格格式中 c 代表格内内容居中，l、r 代表左、右对齐，p 可以自由设定宽度，自动换行，c、l、r 间可以添加分割标记，| 与 || 形成不同的分隔。\hline添加横向分割，用法类似 |。表格的内容通过 & 分隔，以双斜杠进行换行。\centering可以居中显示表格。可以对表格进行引用。注意，表格的

表 1: 标题要放置于表格上方

内容 1	内容 2
内容 3	内容 4

标题要放置于表格上方。

2.20 设置引用

用\label{label}设置引用时的标签，用\ref{label}进行引用。

这是一段话。

这是对一段话的引用2.20。

对于有标题的环境类型来说，标签需要设定于标题后。

2.21 显示超链接

导入 hyperref 包：\usepackage{hyperref}，使用url{}命令显示超链接。

笔记仓库：https://github.com/PLY1024/system-develop-tools

2.22 设置浮动位置

在浮动体开始代码处的[]内可以放置!htbp 等字符，用于设置浮动块的位置。

- h 放置此处
- t 放置本页顶部

- b 放置本页底部
- p 放置于某一专门页面

四个字母可以连用，一般使用 htb，！代表覆盖 LaTeX 的放置策略，H 代表将内容准确地放置于代码所在的位置，需要引入 float 包，不过一般不使用。

3 Git 注意点

Git 的概念比较抽象，最开始不太好理解，并且同样是一开始便需要记住命令。个人觉得比较好的方法是找一个视频跟着敲一敲代码，熟悉一下流程。比较麻烦的是 Github 的注册和使用，需要特殊手段，使用 Gitee 虽然是一种解决方案，但感觉上始终不太对。

以下是个人总结的 Git 使用流程：先获取一个仓库，无论是初始化本地文件夹为仓库还是克隆远程仓库；再对本地文件进行跟踪，进行一次提交，这样就对本地文件进行了一次快照；接着就可以对文件进行修改，在修改后暂存文件并进行提交。还可以连接远程仓库，将本地的修改推送到远程仓库。

4 Git 知识点

4.1 安装

4.1.1 Linux

对于基于 Debian 的发行版，使用如下命令安装：

```
sudo apt install git-all
```

我的 WSL 安装的是 Arch Linux，使用如下命令安装：

```
pacman -S git
```

4.1.2 Windows

Windows 可以访问`https://git-scm.com/download/win`下载 Git for Windows 项目，或访问`https://git-scm.com/downloads/guis`下载 Git GUI 客户端。

4.2 初始配置

4.2.1 配置基本信息

设置用户名和邮箱，其中`--globe`选项使得这样的配置成为全局配置，为以后的项目所使用。

```
git config --global user.name "<username>"

git config --global user.email <mailbox>
```

4.2.2 设置文本编辑器

文本编辑器在 Git 需要你输入信息时调用，在未配置时使用的是操作系统的默认文本编辑器。通过以下命令设置 vim 为文本编辑器。

```
git config --global core.editor <editor name>
```

4.3 查看配置

查看所有 Git 能找到的配置：

```
git config --list
```

通过在末尾添加`--show-origin`显示配置所在的文件。

4.4 获得帮助

通过如下三种方法获取 Git 的命令手册：

```
git help <verb>

git <verb> help

man git-<verb>
```

或通过在命令末尾添加`-h`来获得某个命令的简明帮助。

4.5 日志操作

通过以下命令查看提交日志：`git log`，日志中 HEAD 所指向的是当前分支。

一些常用选项：

- `--all` 显示所有分支上的提交，包括远程仓库
- `--patch` 显示每次提交所引入的差异
- `--stat` 显示简略统计信息
- `--pretty` 设置输出格式
- `-n` 显示最近 n 次提交

4.5.1 输出格式选项--pretty

将每个提交信息在一行内显示：`git log --pretty=oneline`

自定义输出格式：`git log --pretty=format:"%h %an %ad: %s"`

- %H 显示提交的完整哈希
- %h 显示提交的简写哈希
- %an 显示作者名字
- %ad 显示修改日期
- %ar 显示距今修改日期
- %s 显示提交说明

形象地显示分支变化：`git log --graph`，可以与`--pretty`连用。

4.6 获取 Git 仓库

4.6.1 在本地目录初始化仓库

在需要初始化的文件夹下执行：`git init`，将该文件夹初始化为 Git 仓库。执行以下命令来追踪指定的文件并进行初始提交。

```
git add <filename>

git commit -m '<message>'
```

4.6.2 克隆远程仓库

通过执行`git clone <url>`，将在当前目录下创建一个与所克隆仓库同名的文件夹，其中包含仓库中的所有项目文件，也可以添加`<dirname>`来自行指定本地文件夹名称。

Git 支持 https://或 SSH 协议与远程仓库进行连接。

4.7 查看当前文件状态

使用`git status`查看仓库内文件所处的状态，添加`-s`选项使得输出更加简洁。对于一个提交后未更改文件的仓库来说，当前的工作目录是“clean”的。`git status`命令会列出当前所在的分支，未跟踪的文件（即未被提交过的文件），与暂存中的文件（即在执行提交命令时会被提交的文件）。通过执行`git add`对文件进行跟踪，意味着将该文件添加到下次提交的内容中。

注意，`git commit -m '<message>'`命令提交的是暂存区中的文件，而非工作目录内的文件。

创建.gitignore 文件可以配置需要忽略提交的文件格式。

`git diff`命令可以查看工作目录中当前文件与暂存区文件的差异，即尚未暂存的改动。

4.8 提交更新

提交命令`git commit`会启动文本编辑器，添加`-m '<message>'`可以直接输入提交说明。

通过添加`-a`选项使 Git 自动将已跟踪文件暂存并提交，略过`git add`步骤。

注意，提交时记录的是暂存区的文件，未暂存的文件不会被提交。

4.9 移除文件

4.9.1 仅从暂存区移除文件

```
git rm --cached <filename>
```

4.9.2 从暂存区移除文件并删除

```
git rm <filename>
```

4.10 链接远程仓库

在 Github 构建一个远程仓库，通过`git remote add <name> <url>`链接该仓库，可以自己设定该仓库在本地显示的名称。执行`git remote`显示远程仓库名，使用`git remote rename <oldname> <newname>`可以修改本地显示的仓库名。

4.11 移除远程仓库

`git remote remove`以移除远程仓库。

4.12 从远程仓库拉取或抓取

4.12.1 fetch

`git fetch <remote>`会拉取远程仓库中本地所没有的数据，在执行完成后会显示远程仓库中所有分支的引用，需要对分支进行手动合并。

4.12.2 clone

`git clone <url>`克隆远程仓库，并默认命名为 origin。

4.12.3 pull

`git pull <remote>`以抓取数据并自动尝试合并到当前分支。

4.13 推送本地分支

`git push origin main`以将本地的 main 分支推送到 origin 仓库。`-u`使得将来使用`git push`即可向仓库推送更改。

4.13.1 使用 token 推送项目

在 Github 的个人主页下，Settings->Developer settings->Personal access tokens 获取一个 token，在`git push`命令要求输入密码时使用。

注意，`git config --global credential.helper store`命令可以保存账号名和 token，下次推送时无需再次输入。

4.13.2 使用 SSH 密钥推送项目

在 shell 运行`ssh-keygen -t rsa -C “<mailbox>”`生成 SSH 密钥, 在 Github 添加公钥进行配对。通过运行 `ssh -T git@github.com`以验证是否成功。

4.14 分支操作

4.14.1 查看分支

`git branch --list`，其中所处的是当前分支.

4.14.2 创建新分支

```
git branch <branchname>
```

4.14.3 切换分支

```
git checkout <branchname>
```

4.14.4 新建分支并直接切换

```
git checkout -b <branchname>
```

4.14.5 合并分支

```
git merge <branchname>
```

4.14.6 删除分支

```
git branch -d <branchname>
```

4.15 将远程仓库中的分支变为本地分支

主要在`git fetch <remote>`后执行，有三种不同的命令，但效果相同:

```
git checkout  <branchname>

git checkout -b <branchname> <remote path>

git checkout  --track <remote path>
```

<remote path>指的是该远程分支在远程仓库中的路径。

4.16 当前工作文件的储存与恢复

git stash命令可以保存当前工作目录下所有正在修改的文件。

git stash apply可以重新加载这些文件。

4.16.1 多次存储

git stash push将文件当前的修改存储为一个副本。

git stach list能列出存储的文件列表，序号由小到大表示存储时间由近到远。

4.16.2 恢复储存的文件

git stash pop将文件恢复为最近一次保存的状态，并删除该次备份。

git stash apply stash@{<number>}则可以恢复某次储存的文件。

4.16.3 删除某一次备份

git stash drop stash@{<number>}删除指定的备份文件。

4.17 复原文件为仓库中保存的状态

git checkout -- <filename>, 即使用库中的文件替换掉本地的文件。

4.18 撤销提交

git reset HEAD~ --soft，在~后跟数字指定撤销过去某一次的提交。不同参数的意义：

- --soft代表撤销提交但仍在暂存区中保留文件
- --hard代表撤销提交且复原回上一次提交时的状态，所有的修改都会丢失
- 不加参数代表撤销提交，且不在暂存区中保留文件，但所做的修改仍然存在

4.19 变基

`git rebase {branchname}`相当于在另一个分支上进行了本分支的修改, 还可以指定对前 n 次提交进行修改: `git rebase -i HEAD~{<number>}`