目录

第 1 章 这是我测试的第一个章节捏

首先测试一下，能不能自定义一个环境吧

1.1 这是我测试的一个子章节

我将在这个子章节这里进行我书籍的自定义环境的测试
先来一个 mightguy 环境试一试1.1

命题 1.1
duiduidui

定理 1.1
Theorem

引理 1.1
Lemma

推论 1.1
Corollary

迈特凯 1.1
首先请将上面那一条里面提到的自动清理中间文件的语句清除掉，保证你编译出来的报错信息能存下来。

LATEX 的报错很奇葩，有时候它不会把报错信息输出到 VSCODE 的问题工作区里面，它只是单纯的无法编译出来结果然后 P 都不放一个，让人很烦，这时候我们就需要去.log 文件里面搜索关键词 error 然后一个一个的找，看看包含这个单词的错误有多少，得到报错之后再一个一个的搜或者拿去问 AI 得到结果。

需要注意的是，先改正比较靠前的报错在该正比较靠前的错误。因为 LATEX 在报错后并不会停止编译文件，而是直接把整个文档编译完了然后把所有的报错都集中起来。所以说有时候一个报错会因为在原文中反复出现而出现十几一二十次，你先把报错文件最开始出现的报错解决掉，后面的报错可能就连带解决了。

如果你还是找不到问题所在的话，可以把你的项目压缩成一个压缩包然后把整个项目上传到 OVERLEAF 里面，首先这个网站的 LATEX 宏包非常非常的多，至少可以帮你规避掉一点没有安装某个宏包而引发的错误。其次这个网站在编译失败之后会把所有的失败文档分条归总起来，比一个一个的搜索要好很多。

如果你找到了问题所在，最好的解决方法不是直接去原文里面直接进行改正。你也看到了，木叶之秋这本书很长，合计起来有一两万字，如果你一开始就在原文里面进行改正的话，有很大的概率导致报错结果过长而难以找到你改了之后引发的次生灾害。

这时候就要把你的问题提炼出来，在另外一个分支里面写一个小项目进行改正（专注于某一个小 BUG 而写的这个项目其实也叫最小工作流程），在小项目里面解决掉这个问题之后再把解决方案汇总到木叶之秋这个大方案里面，这样才能高效地解决问题。

最后也是最重？要的一步就是，把你遇到的问题写到这个文档里面。毕竟你写下来的话总过是要比放到你那个不太充裕的记忆宫殿里面要有效得多，而且还能拿出来帮助后面的同学们。