第一周主要讲解 terminal 和 shell 的使用。如何用命令行控制电脑。命令行应当与计算机共存，所以必然要学。

我记得老师说到过 nvim，也可以用 lazyvim（是叫这个名字吗？）

对于 markdown 来说，它是 html 的一个子集，所以可以用 html 的标签来写 markdown。当然也是 html 的化简。

另外我们也需要 GNU Maker。这东西是用来自动编译的，似乎类似于 makefile？这一点我可能不会在辅学去学而选择在 b 站大学自己自学。

几个注意事项：
1. 电脑信息？
2. 权限？sudo？
3. 环境变量的处理？
- shell 查找指令-> 找到对应二进制文件-> 执行程序
环境变量的作用在第二部分，它需要去查找文件。所以为什么一些编译会失败，是因为没有将编译器加入到环境变量中，导致环境变量找不到编译器进而无法执行编译指令
4. 分盘的作用在于将系统文件分配到执行较快的一部分以提高反应效率。
5. 以课程为单位建立文件夹。
6. 回避中文名。

7. 名言：
计算机一定是对的，所以未经测试的代码一定是错的。
一切结果的产生都是有原因的，所以一切结果都是可以预测的。
重复性工作应当交给计算机，所以应当学会编程。
8. 课程对应：

---

第一部分：组成一个可以执行特定指令集的计算机

1）计算机系统 1-3，学习如何构成计算机，以及他们的指令集
2）计算机逻辑，这部分更像电路，使用数电方面的知识用门电路组建逻辑电路。—可用电子电路基础课程代替。
3）计组，在计逻的基础上，搭建特定功能并组合，实现指令集功能。这就

是一个计算机雏形了。
（可参考南大的内容，这部分有个老师讲的很好。）
4）计算机体系。主要是用于优化计算机，我们会学到很多技巧用于提高计算机性能。承接 3。

---

第二部分：方便的操纵计算机

1）汇编，汇编接近计算机指令集，因此可以先行学习。当然可以并行学习高级程序语言。、、基于汇编可以尝试去切入信安方面的二进制攻防（?）5）操作系统原理与实践，计算机上同时运行多个程序，如何调度，如何分配资源，如何保证安全。好吧他看起来就像是某种操作系统原理及其优化？
6）编译原理，编译器的原理，如何将高级语言转化为汇编语言。这东西是 gpt 自己帮我加上的，但是我觉得很有道理。
7）各类高级语言。这一点可以衔接 6，即所谓“高级程序语言与高级程序语言的编译”。
8）程序设计。
9）程序设计的进阶形态：FDS，ADS，算法分析，三门课并行将实现数据结构和算法的配合。这里或许可以考虑 ACM。
10）面向对象程序设计，主要是面向对象原理，对象的分工。他和 9 应当可以分属不同分支。

晶体管-> 门电路-> 逻辑电路（计逻）
->CPU-> 计算机雏形（计组）
-> 计算机（计算机体系）

-> 汇编语言（汇编语言）
-> 操作系统（操作系统原理）
-> 编译器-> 高级语言（高级语言与编译原理）
-> 算法和数据结构（DS 课程）
-> 面向对象（oop）

领域：
安全，人工智能，系统，CV，PL