操作系统导论

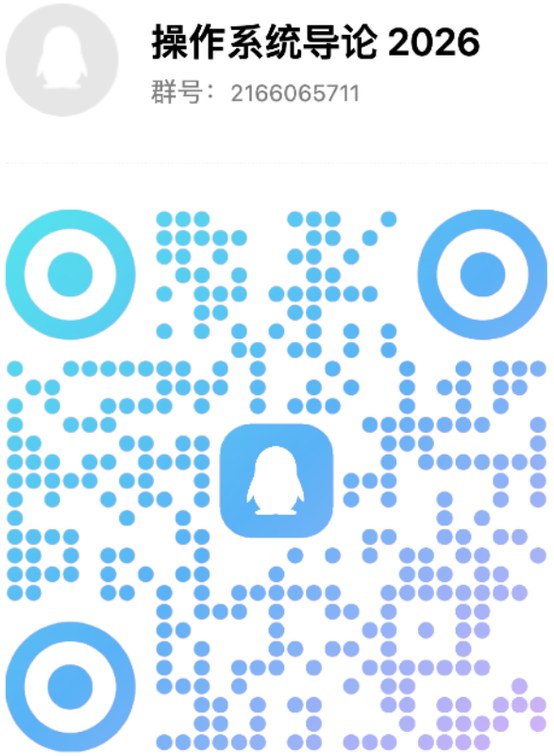
吴化尧

hywu@nju.edu.cn

2026 Spring

课程信息

- 周二 2–4节 @ 逸 B–105
- 课程 QQ 群: 2166065711
  - 群昵称应包括 姓名 和 学号 信息 ⚠️
  - 统计随堂练习 (平时分) 提交的依据



加群问题答案：2026

- 课程网站: https://gist.nju.edu.cn/course-os/
  - 课件下载 (每次课前更新)
  - 课堂示例代码
  - 实验说明

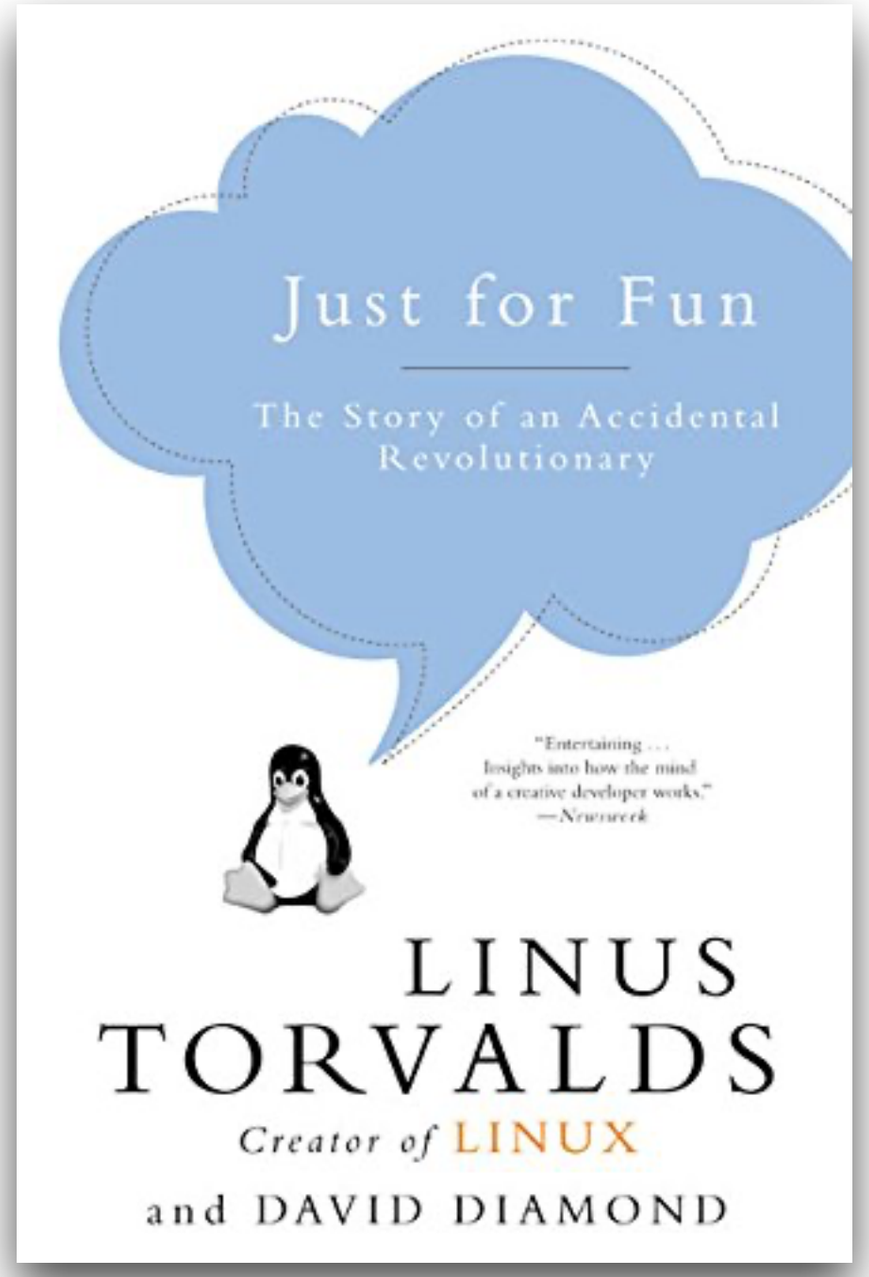
为什么要学操作系统

感受人类文明的伟大

- 计算机世界的 "地基"
  - 支撑我们熟悉的各类应用场景
  - 在原始和混乱的硅基硬件上构建 "秩序和文明"
- 跨越半个世纪的群体智慧
  - 操作系统的历史就是计算机软件和硬件的历史
  - 涵盖了一代又一代顶尖计算机科学家的贡献

为什么要学操作系统

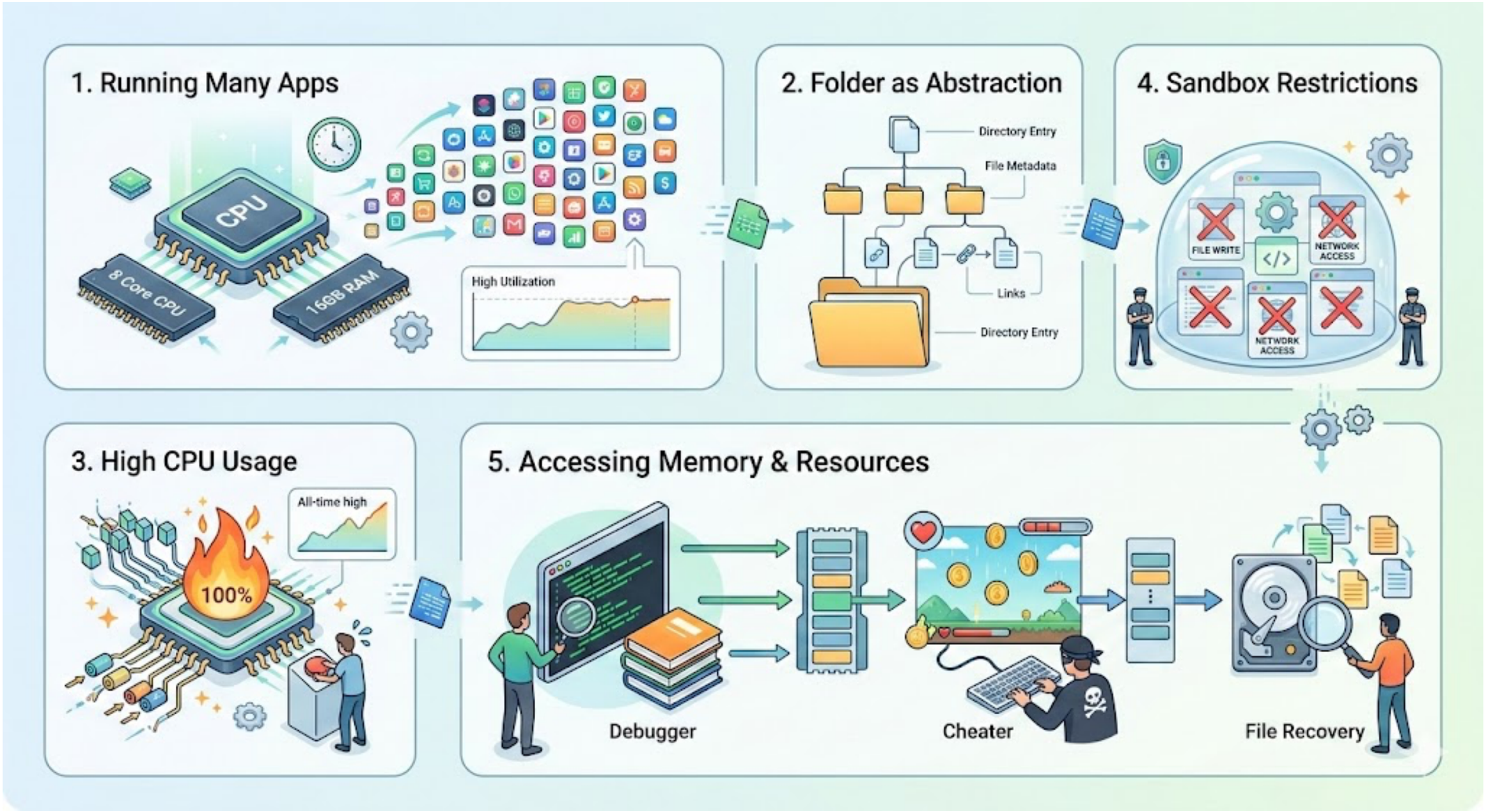
Just for Fun



"Most of the things we do, we do for survival. And some of the things we do, we do for social reasons. But then there are the things we do for fun. And that's the highest stage of all."

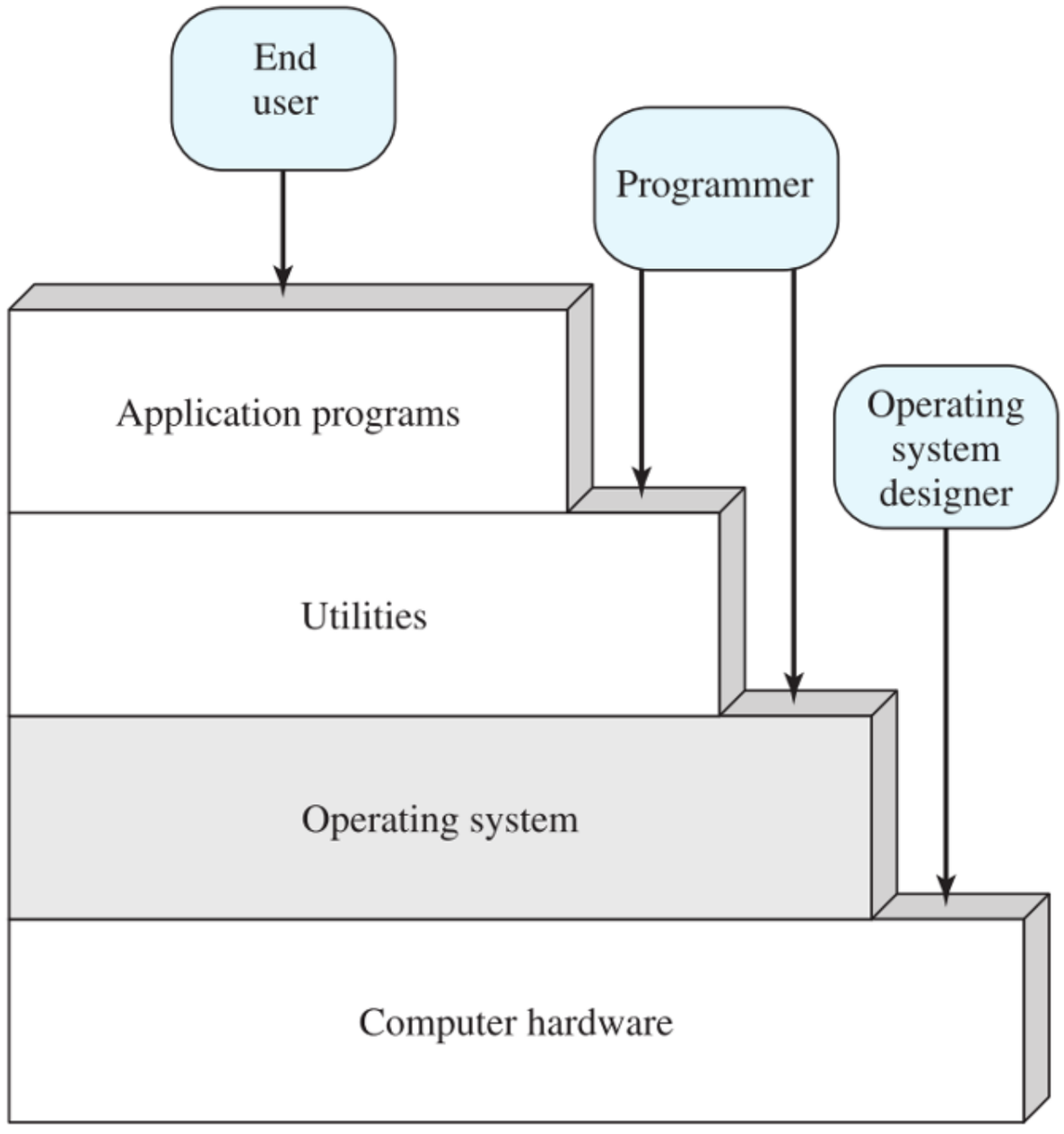
为什么要学操作系统

Just for Fun



为什么要学操作系统

理解一个复杂系统是如何构建和工作的



为什么要学操作系统

理解一个复杂系统是如何构建和工作的

- Make you a more capable programmer 😄
  - 知道系统如何工作、以及如何工作的更好
- Make our lives better (or worse) 😢
  - 未来的世界终将运行在你们编写的软件之上

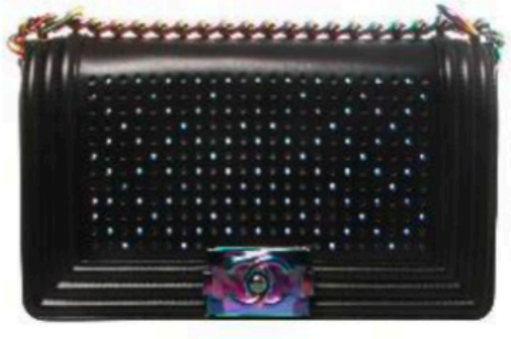
为什么要学操作系统

成为大模型时代的领域专家

- LLM 代替我上大学时代的 "学习"
- 构建知识体系，提高 AI 时代的 "生产力"
  - 是否能问出好的问题 🤦
  - 是否能评估回答的质量 🙅

操作系统的复杂性

横跨很多不同的物理设备

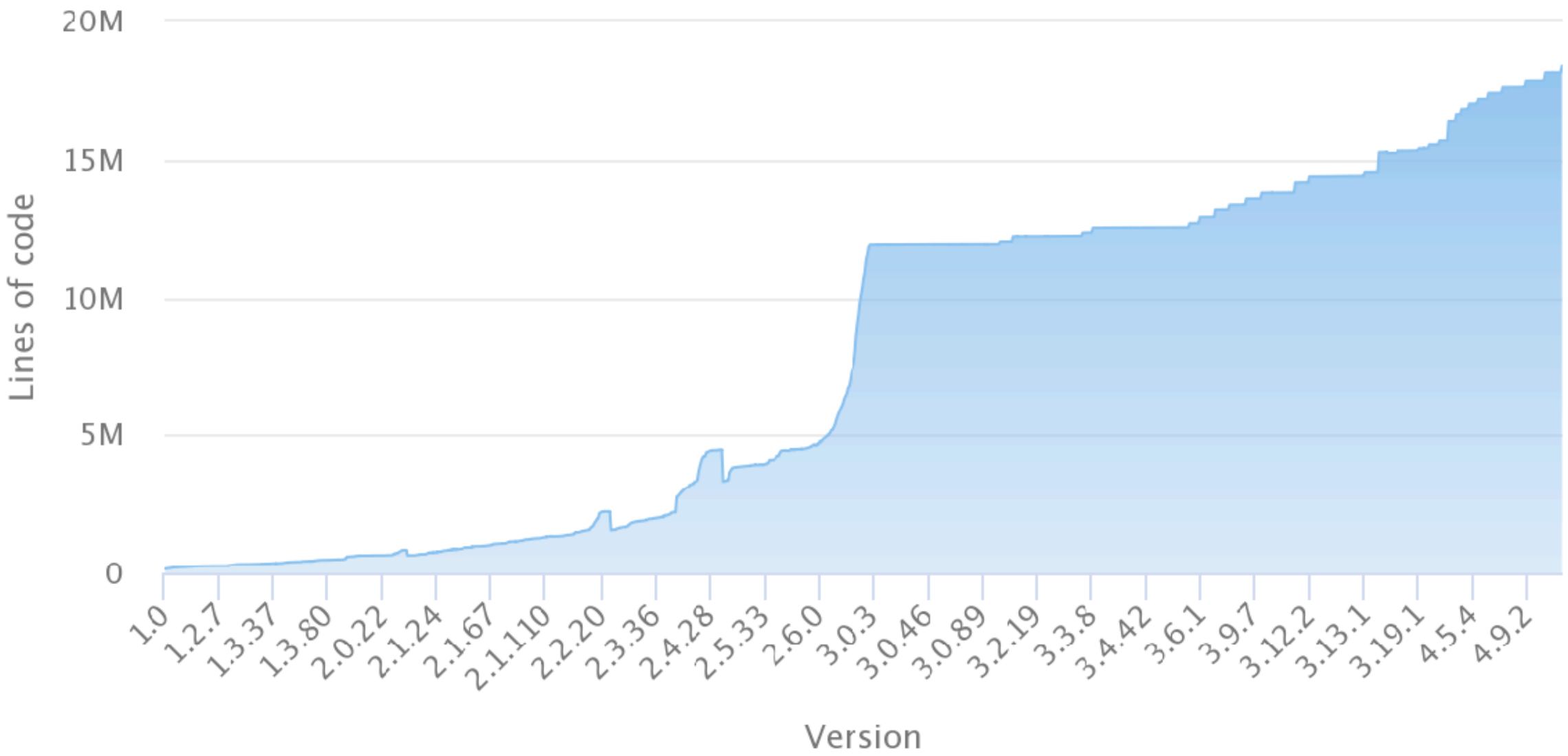


操作系统的复杂性

横跨很多不同的时间尺度

操作系统的复杂性

规模和复杂度急剧增加



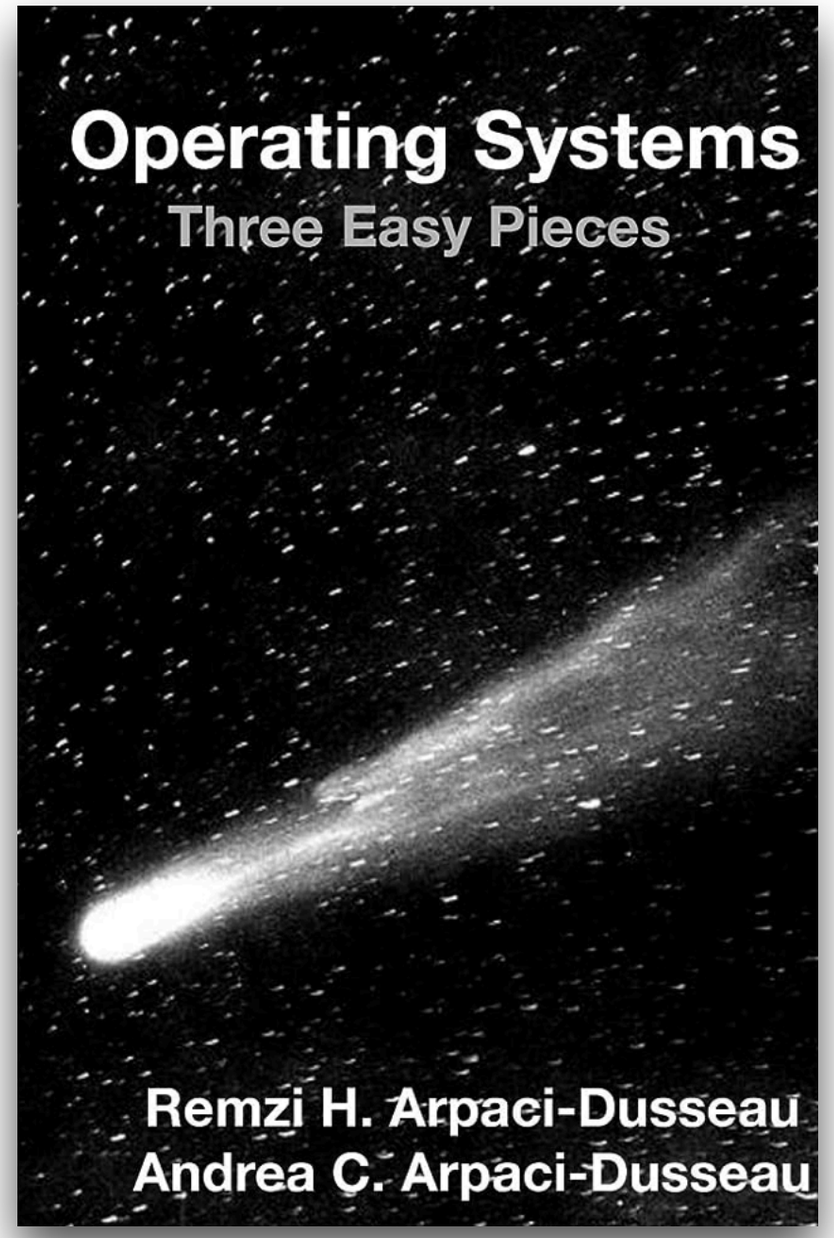
Lines of Code per Linux Kernel

如何理解复杂的系统

聚焦操作系统设计和实现的

- 基本概念：是什么
- 基本动机：为什么
- 基本方法：怎么做

教材



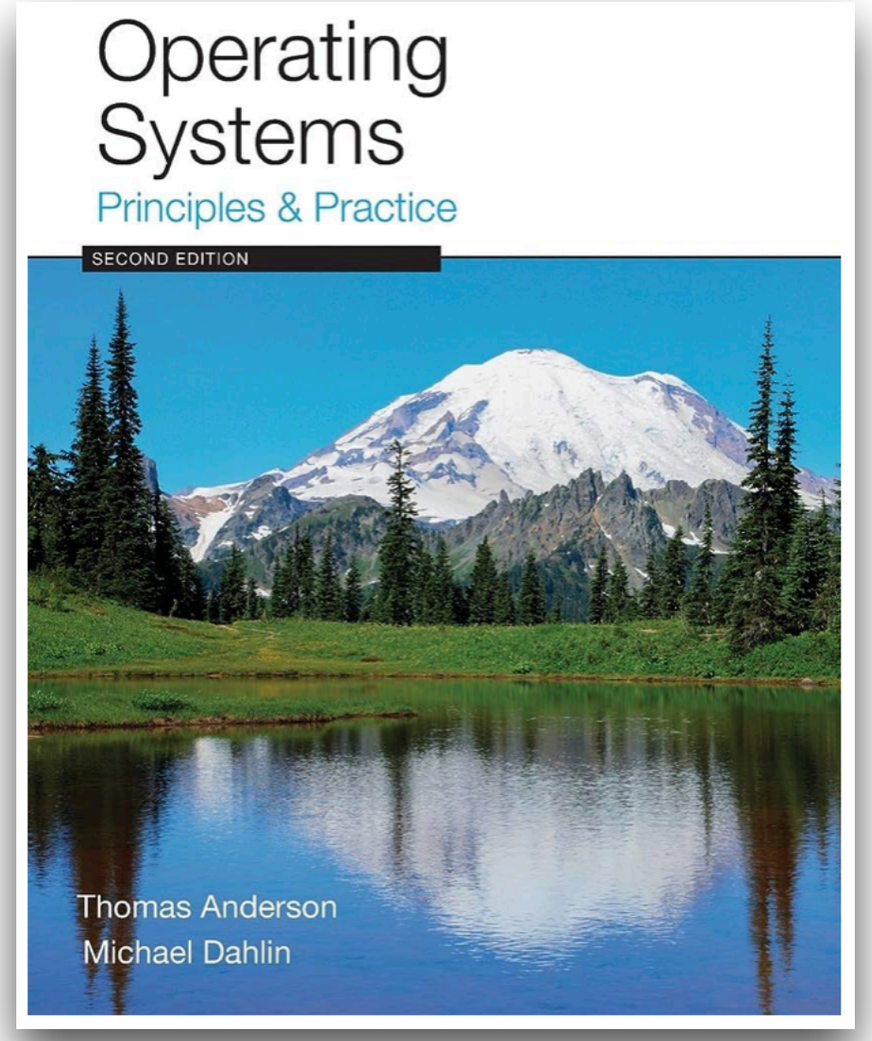
Operating Systems: Three Easy Pieces

Remzi H. Arpaci-Dusseau & Andrea C. Arpaci-Dusseau

https://pages.cs.wisc.edu/~remzi/OSTEP/

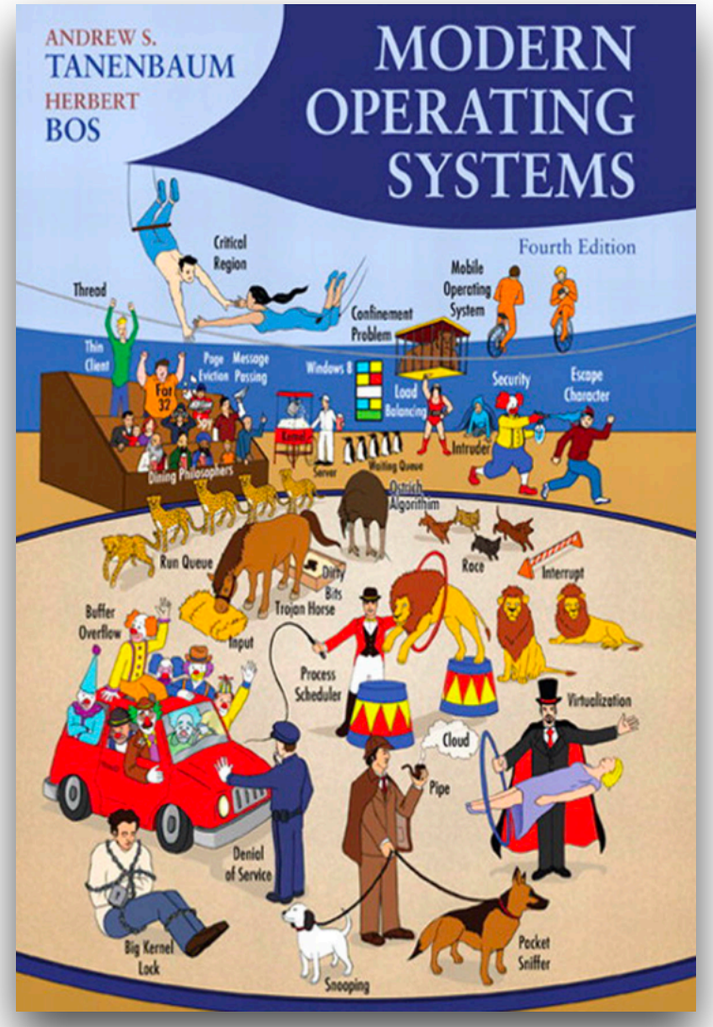
《操作系统导论》人民邮电出版社

参考资料



Operating Systems: Principles & Practice

Thomas Anderson & Mike Dahlin

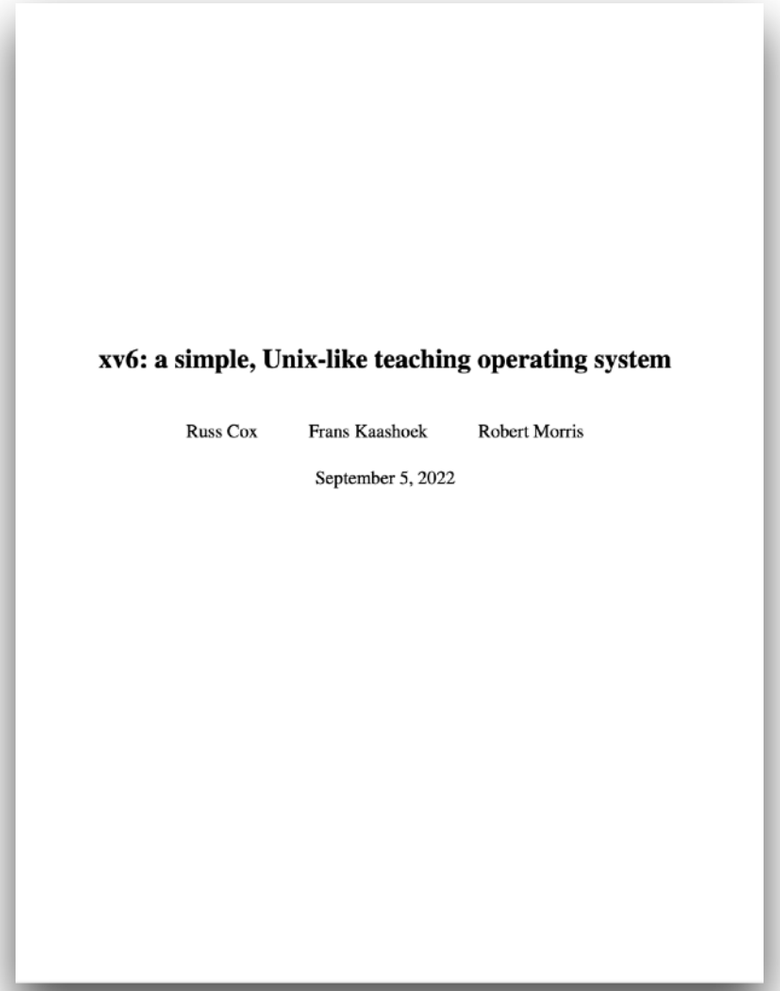


Modern Operating Systems

Andrew Tanenbaum & Herbert Bos

《现代操作系统》机械工业出版社

参考资料



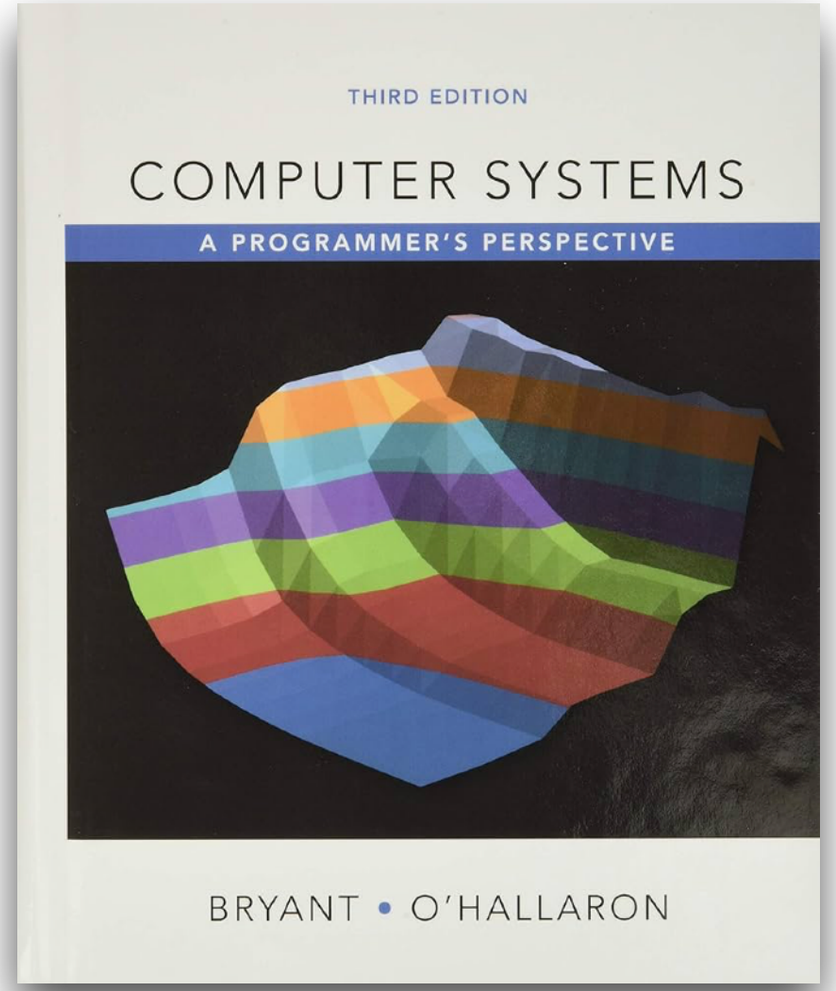
xv6: a simple, Unix-like teaching operating system

Russ Cox Frans Kaashoek Robert Morris

September 5, 2022

xv6: a simple, Unix-like teaching operating system

Russ Cox, Frans Kaashoek & Robert Morris



Computer Systems: A Programmer's Perspective

Randal Bryant & David O'Hallaron

《深入理解计算机系统》机械工业出版社

参考资料

Large Language Model (LLM)

- 理解概念、解释细节、提供示例

```
movq $1,   %rax
movq $1,   %rdi
movq $msg, %rsi
movq $12,  %rdx
syscall

这段汇编代码实现的是什么?
```

这段汇编代码的作用是 **通过 Linux 系统调用 `write` 向标准输出（屏幕）打印一段字符串**。逐行分析如下：

asm 复制 编辑

```asm
movq $1,   %rax     ; 将系统调用号 1 放入寄存器 %rax，对应 Linux 的 sys_write
movq $1,   %rdi     ; 第一个参数：文件描述符 1，表示标准输出（stdout）
movq $msg, %rsi     ; 第二个参数：字符串的起始地址，即要输出的数据
movq $12,  %rdx     ; 第三个参数：要写入的字节数，这里是 12 个字节
syscall             ; 执行系统调用
```

对应的系统调用原型

在 Linux x86-64 下，`write` 的原型是：

c 复制 编辑

```c
ssize_t write(int fd, const void *buf, size_t count);
```

- `fd = 1` → 标准输出
- `buf = msg` → 数据缓冲区的地址
- `count = 12` → 输出的字节数

课程内容安排

- 操作系统概述
- 虚拟化 (Virtualization): 处理器管理、内存管理
- 并发 (Concurrency): 同步和死锁
- 持久化 (Persistence): 文件系统、设备管理
- 预计第 7 周随堂期中测试

课程实验

- 总共 2～3 次课程实验 (Labs)
  - 基于 Linux 的编程实验，利用 OS 提供的 APIs 实现某些能力
  - 每次实验需提交
    - 源代码：Online Judge 评分
    - 实验报告 (1～2页)：人工阅读评分
- 实验截止时间
  - 在 Deadline 前完成至少一次提交将获得 100% 的分数，超过该期限将获得 90% 的分数
  - 在所有实验 Hard Deadline (学期结束) 前可持续提交新版本

学术诚信

⚠️ 独立完成实验是对自己最好的训练

- 将坚守学术诚信 (Academic Integrity) 作为自发的要求
  - 不抄袭别人的代码
  - 不传播自己的代码
  - 鼓励使用 LLM 来帮助澄清概念或解释 API 的使用方式，但不应直接使用 LLM 来生成实验代码

成绩构成

- 课程实验: 30%
- 期中考试 (闭卷) + 平时成绩 (bonus): 20%
- 期末考试 (闭卷): 50%

---

* 重修/免修不免考仍需要按时完成实验、并参加期中和期末考试
* 免修不免考不计算平时成绩
* 补考只更新期末考试成绩

Tips

- Do not panic. 这毕竟是导论课
- 课堂出勤
  - 会尽量 self-contained，争取让前序课程没学好也能听懂
  - 会展示大量示例帮助理解核心概念
- 考试范围以课程 slides 为主

Tips

- Do not panic. 这毕竟是导论课
- 课堂出勤
- 完成实验
  - 根据历史统计，"没来考试" 和 "不交实验" 是挂科的主要原因
  - 很有可能会有和 Lab 内容相关的考试题目
  - 对于完成实验确有困难的同学，提交部分完成的实验代码 (即使是空文件) 也将获得一定的实验分数