第1讲 操作系统概述

2024.02.24

1. 引入

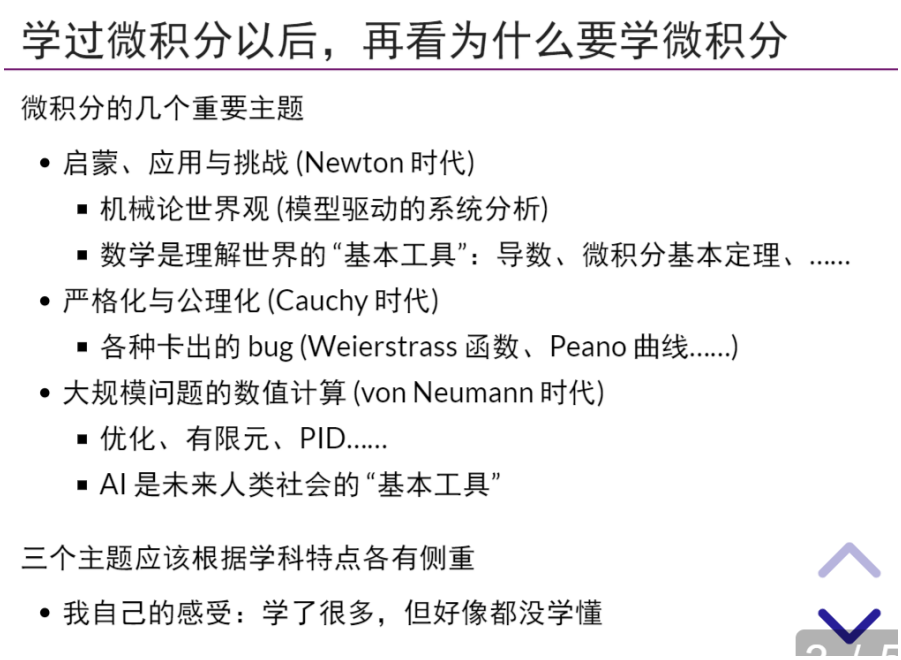
操作系统的发展历史，回答三个问题：

- why：为什么要学操作系统？
- what：到底什么是操作系统？
- how：怎么学操作系统？

2. 为什么要学“任何东西”？

为什么要学操作系统呢？

- 为什么要学微积分/离散数学/XXXX/……？
- 长辈/学长：~~擦干泪不要问为什么~~



为什么要学“任何东西”？

重走从无到有的发现历程

- 基本思想、基本方法、里程碑、走过的弯路
- 最终目的：应用/创新 (做题得分不是目的而是手段)
  - 如果只是记得几个结论，那 ChatGPT 已经做得很好了

学习“任何东西”的现代方法

- 使用辅助工具加速探索
  - 数值/符号计算：numpy, sympy, sage, Mathematica, ...
  - 可视化：matplotlib
    - All-in-one: Jupyter (2017 ACM Software System Award)
    - Life is short; you need Python
- (正好我有一个微积分相关的案例)

4 / 5

为什么学习操作系统？

你体内的“编程力量”尚未完全觉醒

- 每天都在用的东西，你还没搞明白
  - 为什么能创建窗口？为什么 Ctrl-C 有时不能退出程序？
  - 为什么有的程序能把组里服务器的 128 个 CPU 用满？
- 你每天都在用的东西，你实现不出来
  - 浏览器、编译器、IDE、游戏/外挂、杀毒软件、病毒.....

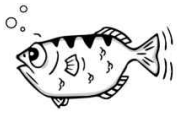
《操作系统》带你补完“编程”的技术体系

- 悟性好：学完课程就在系统方向“毕业”
  - 具有编写一切“能写出来”程序的能力 (具备阅读论文的能力)
- 悟性差：内力大增
  - 可能工作中的某一天想起上课提及的内容

5 / 5

3. 什么是操作系统?

“管理软/硬件资源、为程序提供服务”的程序？

	目标纯粹 必须管理一台计算机	目标中立 管理计算机硬件就行	目标混乱 管理什么都行
对象纯粹 必须服务二进制代码	 Windows/Linux/vxWorks 当然是操作系统	 固件也是操作系统	 gdb也是操作系统
对象中立 服务对象是程序就行	 浏览器/微信/支付宝/JVM 都是操作系统	 Hadoop也是操作系统	 资源管理器也是操作系统
对象混乱 服务谁都可以	 机箱是操作系统	 机房也可以算操作系统	 校长为什么不是操作系统?

理解操作系统

“精准”的教科书定义毫无意义 (但作者得被迫去写)

- 定义是“全部”的一个极简表达
  - 如果只想“了解”，那可以读一下定义
  - 如果想学习操作系统，就必须理解“全部”

操作系统“全部”的 overview：

- 操作系统如何从一开始变成现在这样的？
- 三个重要的线索
  - 硬件 (计算机)
  - 软件 (程序)
  - 操作系统 (管理硬件和软件的软件)

Logisim Demo

数字电路模拟器

- 基本构件：wire, reg, NAND, NOT, AND, NOR
- 每一个时钟周期
  - 先计算 wire 的值
  - 在周期结束时把值锁存至 reg

会编程，你就拥有全世界！

- 同样的方式可以模拟任何数字系统 (包括计算机系统)
- 同时还体验了 UNIX 哲学
  - Make each program do one thing well
  - Expect the output of every program to become the input to another

理解操作系统

本课程讨论狭义的操作系统

- 操作系统：硬件和软件的中间层
  - 对单机 (多处理器) 作出抽象
  - 支撑多个程序执行
- 学术界谈论的 “操作系统” 是更广义的 “System”
  - 例子：对多台计算机抽象 (分布式系统)

理解操作系统

- 理解硬件 (计算机) 和软件 (程序) 的发展历史
- 夹在中间的就是操作系统

1950s 的计算机软件 (cont'd)

Fortran 已经“足够好用”

- 自然科学、工程机械、军事……对计算机的需求暴涨

```
C---- THIS PROGRAM READS INPUT FROM THE CARD READER,
C---- 3 INTEGERS IN EACH CARD, CALCULATE AND OUTPUT
C---- THE SUM OF THEM.
  100 READ(5,10) I1, I2, I3
   10 FORMAT(3I5)
      IF (I1.EQ.0 .AND. I2.EQ.0 .AND. I3.EQ.0) GOTO 200
      ISUM = I1 + I2 + I3
      WRITE(6,20) I1, I2, I3, ISUM
   20 FORMAT(7HSUM OF , I5, 2H, , I5, 5H AND , I5,
     *   4H IS , I6)
      GOTO 100
  200 STOP
      END
```

操作系统：先跑这一堆大控纸带，然后跑下一堆……

1950s 的操作系统

> 库函数 + 管理程序排队运行的调度代码。

写程序 (戳纸带)、跑程序都是非常费事的

- 计算机非常贵 ($\$50,000 - \$1,000,000$)，一个学校只有一台
- 算力成为一种服务：多用户轮流共享计算机，operator 负责调度

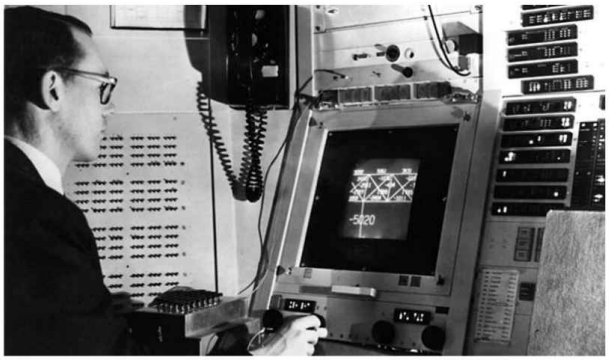
操作系统的概念开始形成

- 操作 (operate) 任务 (jobs) 的系统 (system)
  - “批处理系统” = 程序的自动切换 (换卡) + 库函数 API
  - Disk Operating Systems (DOS)
    - 操作系统中开始出现“设备”、“文件”、“任务”等对象和 API

1960s 的计算机硬件

集成电路、总线出现

- 更快的处理器
- 更快、更大的内存；虚拟存储出现
  - 可以同时载入多个程序而不用“换卡”了
- 更丰富的 I/O 设备；完善的中断/异常机制



1960s 的操作系统

> 能载入多个程序到内存且调度它们的管理程序。

为防止程序之间形成干扰，操作系统自然地将共享资源 (如设备) 以 API 形式管理起来

- 有了进程 (process) 的概念
- 进程在执行 I/O 时，可以将 CPU 让给另一个进程
  - 在多个地址空间隔离的程序之间切换
  - 虚拟存储使一个程序出 bug 不会 crash 整个系统

操作系统中自然地增加进程管理 API

- 既然可以在程序之间切换，为什么不让它们定时切换呢？
- Multics (MIT, 1965)：现代分时操作系统诞生

今天的操作系统

> 通过“虚拟化”硬件资源为程序运行提供服务的软件。

空前复杂的系统之一

- 更复杂的处理器和内存
  - 非对称多处理器 (ARM big.LITTLE; Intel P/E-cores)
  - Non-uniform Memory Access (NUMA)
  - 更多的硬件机制 Intel-VT/AMD-V, TrustZone/~~SGX~~, TSX, ...
- 更多的设备和资源
  - 网卡、SSD、GPU、FPGA...
- 复杂的应用需求和应用环境
  - 服务器、个人电脑、智能手机、手表、手环、IoT/微控制器.....

4. 本课程

课程内容概述

操作系统：软件硬件之间的桥梁

- 本课程中的软件：多线程 Linux 应用程序
- 本课程中的硬件：现代多处理器系统

(设计/应用视角) 操作系统为应用提供什么服务？

- 操作系统 = 对象 + API
- 课程涉及：POSIX + 部分 Linux 特性

(实现/硬件视角) 如何实现操作系统提供的服务？

- 操作系统 = C 程序
  - 完成初始化后就成为 interrupt/trap/fault handler
- 课程涉及：xv6, 自制迷你操作系统

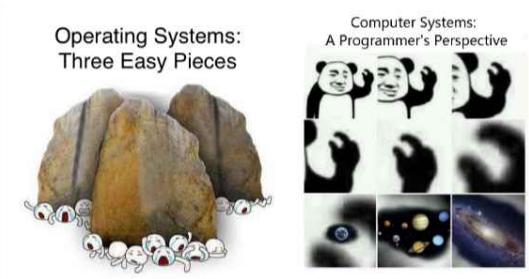
两个视角：

- 从下往上看（下面是硬件上面是软件），硬件是无情的执行指令的机器（fetch，decode，execute），在它看来，os就是一个普通的c程序而已，在硬件启动初始化的时候就开始执行，然后成为一个handler
- 从软件视角来看，操作系统就是一个库，这个库规定了操作系统里有什么样子的对象——有进程，有文件，以及应用程序可以对这些对象做出什么样的操作。

5. 怎么学操作系统

学习操作系统：现代方法

1. 读得懂的教科书和阅读材料
   - Operating Systems: Three Easy Pieces



2. 问题驱动，用代码说话
   - Demo 小程序、各类系统工具 (strace, gdb, ...) 的使用
   - xv6-riscv, AbstractMachine
   - RTFM, STFW, RTFSC (*F* can be a colorful word)

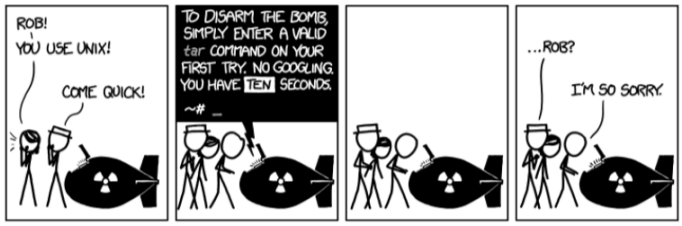
Prerequisites

> 计算机专业学生必须具备的核心素质。

1. 是一个合格的操作系统用户
   - 会 STFW/RTFM 自己动手解决问题
   - 不怕使用任何命令行工具
     - vim, tmux, grep, gcc, binutils, ...
2. 不怕写代码
   - 能管理一定规模 (数千行) 的代码
   - 能在出 bug 时默念“机器永远是对的、我肯定能调出来的”
     - 然后开始用正确的工具/方法调试

给“学渣”们的贴心提示：不要尝试“架空学习”，回头补基础

1. 成为 Power User



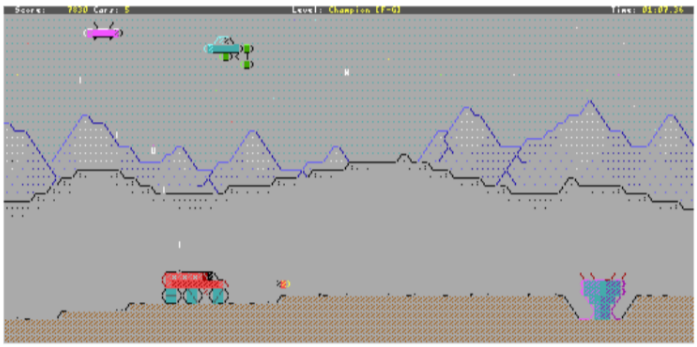
感到 Linux/PowerShell/... 很难用？

1. 没有建立信心、没有理解基本逻辑
   - 计算机科学自学指南
2. 没有找对材料/没有多问“能不能再做点什么”
   - Baidu v.s. Google/Github/SO v.s. ChatGPT
3. 没有用对工具 (man v.s. tldr; 该用 IDE 就别 Vim)
   - 过了入门阶段，都会好起来

2. 学会写代码

写代码 = 创造有趣的东西

- 命令行 + 浏览器就是全世界



  - 我们还有 sympy, sage, z3, rich, ... 呢
- 不需要讲语言特性、设计模式、……
  - 编就对了；你自然而然会需要它们的

最重要的：Get Your Hands Dirty

> 听课看书都不重要。独立完成编程作业即可理解操作系统。

应用视角 (设计)：Mini Labs x 6

- 使用 OS API 实现“黑科技”代码

硬件视角 (实现)：OS Labs x 5

- 自己动手实现一个真正的操作系统

全部 Online Judge

- 代码不规范 → `-Wall -Werror` 编译出错
- 代码不可移植 → 编译/运行时出错：`int x = (int)&y;`
- 硬编码路径/文件名 → 运行时出错：`open("/home/a/b", ...)`

6. 总结

Take-away Messages

操作系统没有传说中那么复杂 (程序视角：对象 + API，硬件视角：一个 C 程序)

- 为什么要学操作系统：解锁 “实现一切” 的系统编程能力
- 什么是操作系统：应用视角 (一组对象 + API)、机器视角 (一个程序)
- 怎么学操作系统：答案就在代码中