操作系统
Operating Systems

钮鑫涛
南京大学
2024春

本门课由蒋炎岩和钮鑫涛共同构建

课程信息

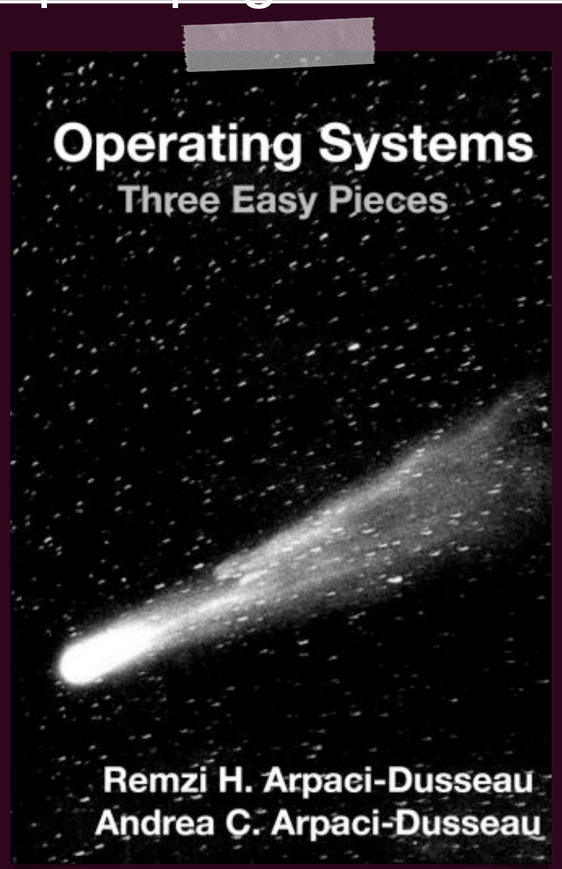
- 讲师: 钮鑫涛 (Email: niuxintao@nju.edu.cn)
- 课程前置: C/C++，计算机组成原理
- QQ群：827387683
  - **加群请注明姓名和学号！**
- 课程主页: https://niuxintao.github.io/courses/2024Spring-OS/
- Online Judge: 敬请期待
- Office hour：周五上午10点到12点，南雍楼223

助教

- 范佳明 (MG21330020@smail.nju.edu.cn)
- 陈宏楠 (MG21330010@smail.nju.edu.cn)

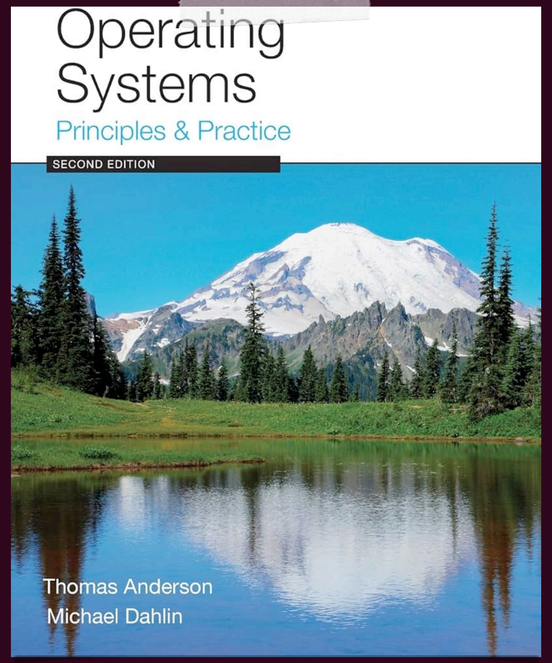
课本

- Operating Systems: Three Easy Pieces (OSTEP), by Remzi and Andrea, available for free online: https://pages.cs.wisc.edu/~remzi/OSTEP

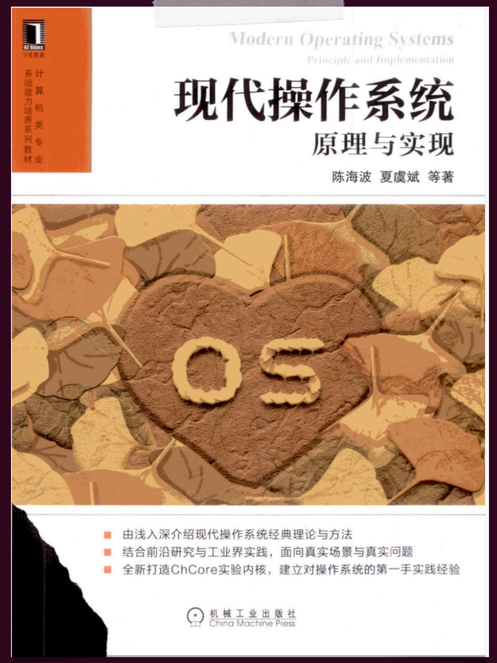


参考教材

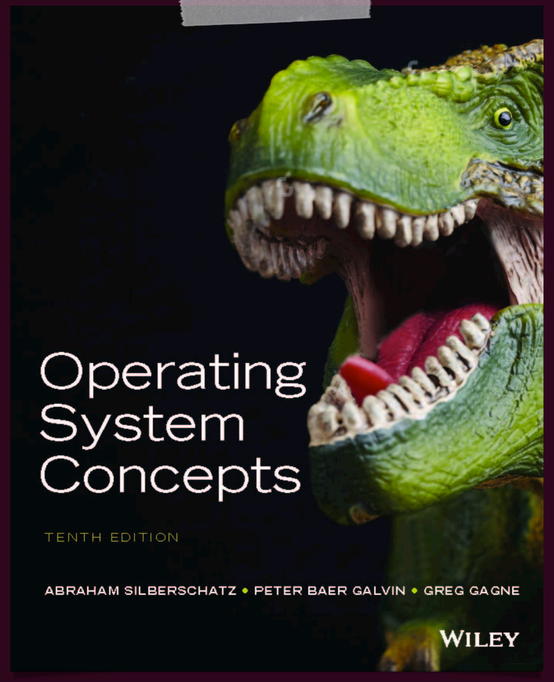
- Operating Systems, Principles and Practice (OSPP), 2nd edition, by Anderson and Dahlin



- 现代操作系统—原理与实现，陈海波/夏虞斌

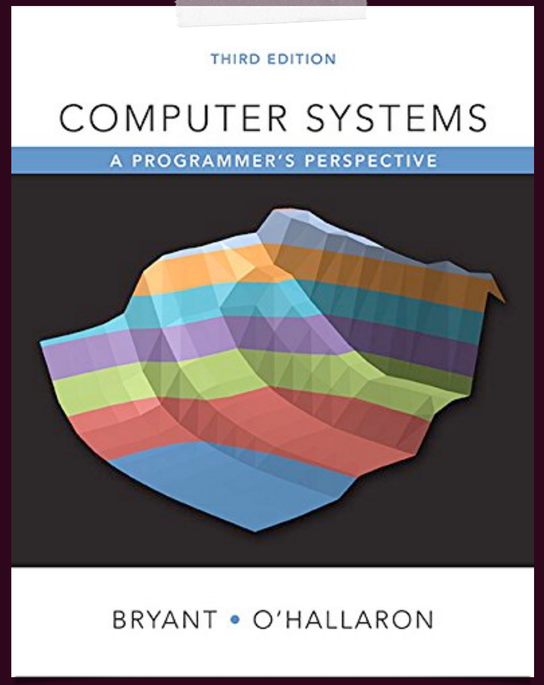


- Operating System Concepts (OSC)， 10th edition, by by Abraham Silberschatz, Peter B. Galvin, Greg Gagne

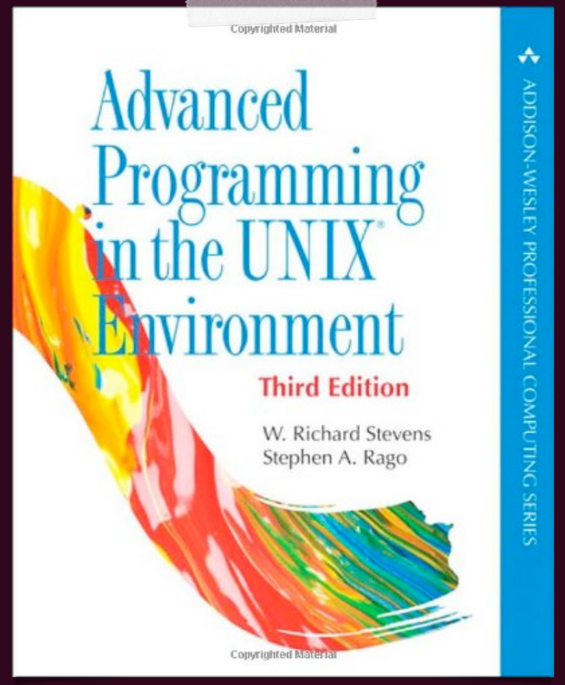


参考资料

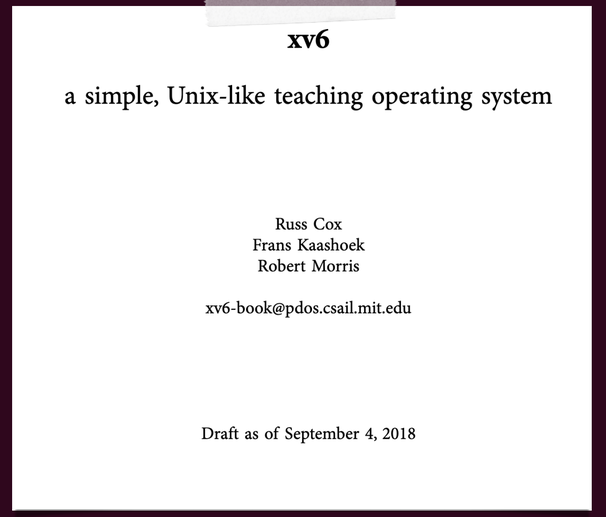
- Computer Systems: A Programmer's Perspective (CSAPP), 3rd edition, by Bryant and O'Hallaron



- Advanced Programming in the UNIX Environment, 3rd edition, by Stevens and Rago



- XV6, a simple Unix-like teaching operating system



- Read The Friendly Manual (RTFM) …
  - Linux manpages (tldr, man…)
  - Bash
  - GCC
  - Strace
  - GDB
  - Make
  - Binutils (ld, as, objdump…)

考核方式

- 编程实验(Mini Labs) +系统实验(OS Labs) + 期中考试 + 期末考试
  - 编程实验 (只需要编辑一个c源文件代码，大致500行以内)，利用系统api解决某个具体问题 （5次） (25%)
  - 系统实验，逐步实现一个操作系统（4次） (25%)，每次实验需要提交一个实验报告（2页以内的pdf）
  - 期中考试 （15%）
  - 期末考试 （35%）

比例可能会调整

Online Judge

- 本课程的实验环境推荐Ubuntu，在命令行执行如下命令将得到实验框架

```
敬请期待
```

- 我们会为选修课程的同学生成了唯一的秘钥，并以邮件形式发送到你的学号@smail.nju.edu.cn 邮箱，有遗漏的请联系助教。配置好 `Makefile` 中的 `TOKEN` 环境变量后，在相应的实验目录中 (而不是项目根目录) 中执行以下命令完成提交：

```
敬请期待
```

- Mini/OS Lab 都在 Online Judge 评测。程序在容器中编译、运行，并由机器自动判定结果是否正确。在此基础上，最终实验的评分仍有一部分由助教决定。

学术诚信

- 不将代码上传到互联网
- 不抄袭别人（包括互联网）完成的实验代码，可以相互讨论，但只限于high level
- 如果发现互相抄袭行为， 抄袭和被抄袭双方的成绩都将被取消
- 实在不会，只要给出相应的说明和解释即可（可酌情给分）

教学大纲

- 绪论 （2周）
  - 概论、硬件视角下的操作系统、应用视角下的操作系统、抽象视角下的操作系统
- 虚拟化（6周）
  - 进程、线程、调度、地址空间
- 并发（4周）
  - 多处理器编程、互斥、同步、真实世界的并发编程
- 持久化 （4周）
  - 输入输出设备、驱动程序与文件系统、数据可靠性

为什么要学这门课?

系统的实现能力

- 理解一个复杂系统是如何层层实现

系统的编程能力

- 熟悉操作系统提供的各种对象，以及如何更好的利用他们

系统的思考能力

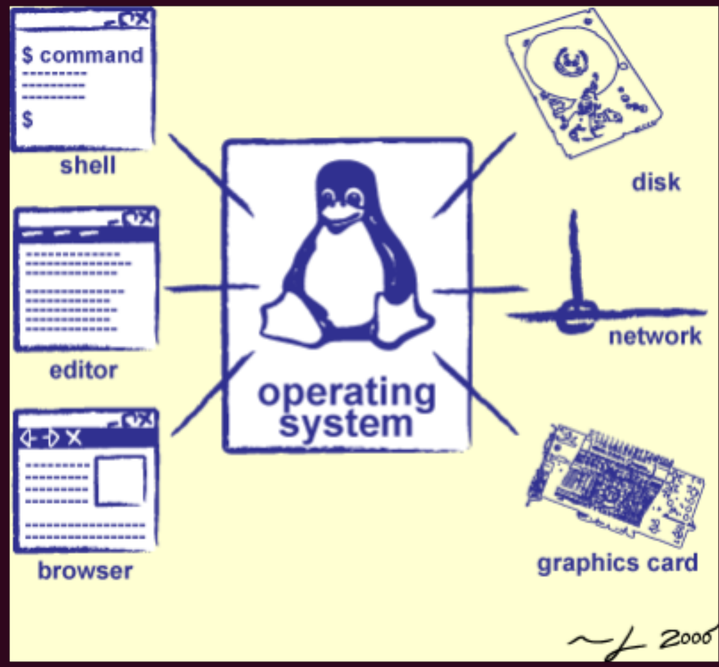
- 操作系统的各种设计理念，以及运用到其他方面!

让我们开始吧！

操作(Operating) 系统(System)

Switchboard Operator

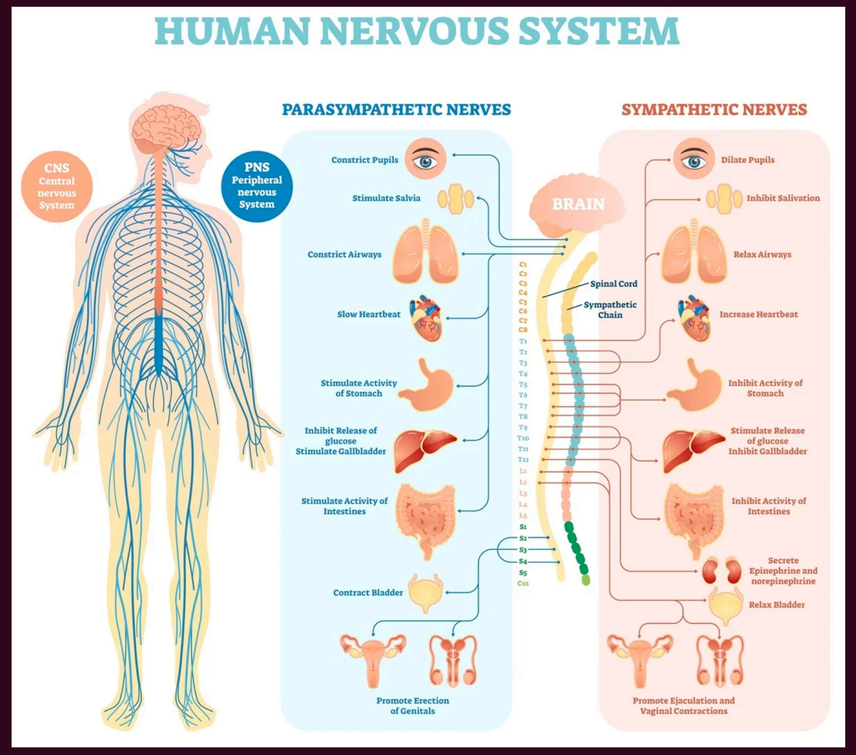
Computer Operators



Operating System

操作(Operating) 系统(System)

- 什么构成一个“系统”?
  - 多个相关联的部分：相互之间可以协作以达到特定目标或执行特定功能



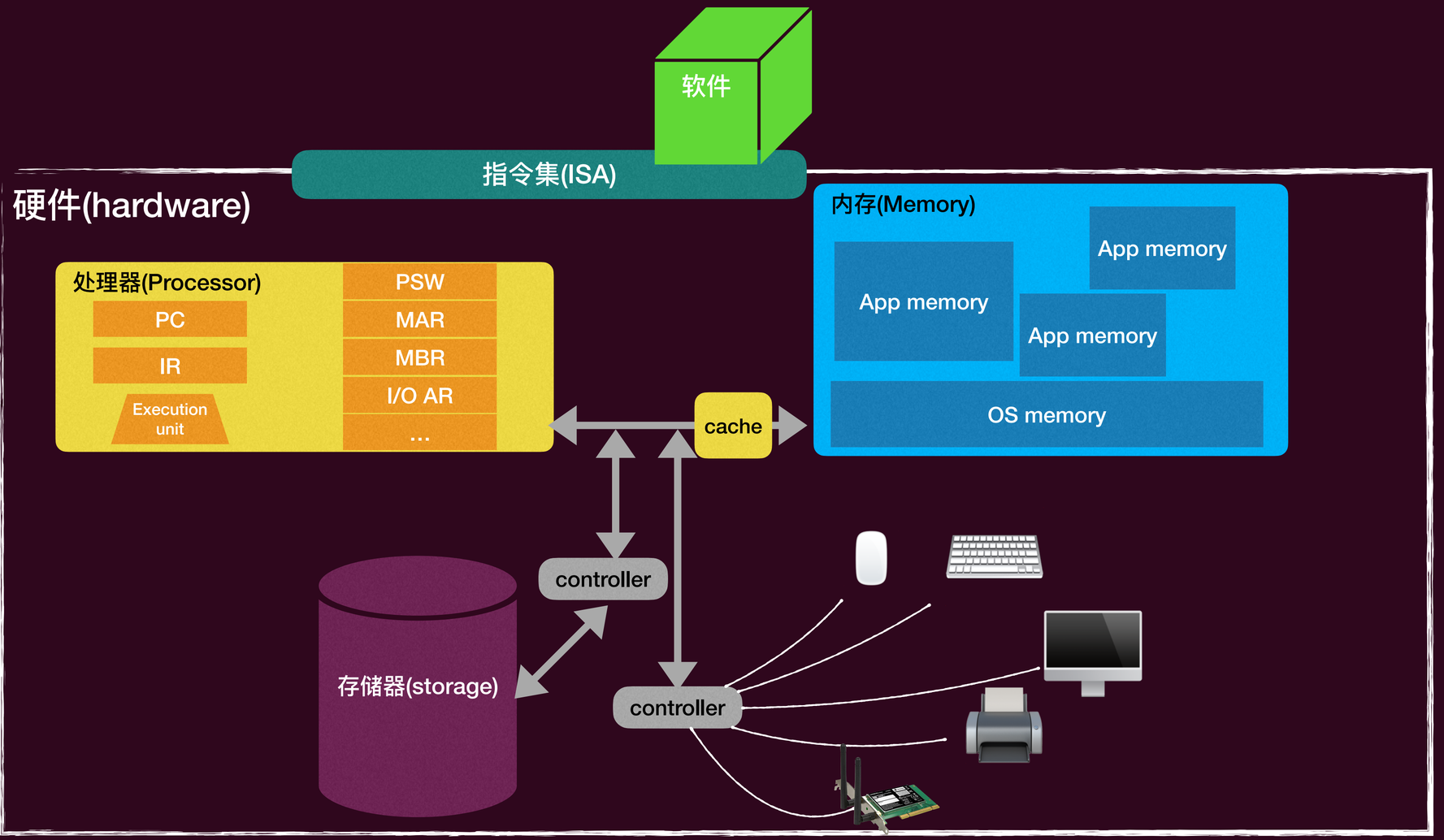
什么是操作系统?

- 没有一个统一的定义
  - A body of software, in fact, that is responsible for making it easy to run programs, allowing programs to share memory, enabling programs to interact with devices, and other fun stuff like that. (OSTEP)
  - An operating system is the layer of software that manages a computer's resources for its users and their applications.(OSPP)
  - A program that acts as an intermediary between a user of a computer and the computer hardware. (OSC)
  - It is the software that runs in kernel mode. (MOS, by tanenbaum)
  - 操作系统是对硬件进行管理和抽象并为应用提供服务并进行管理的程序 (现代操作系统)

什么是操作系统?

- 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）
    - 核心任务或者目标?

软硬件的接口

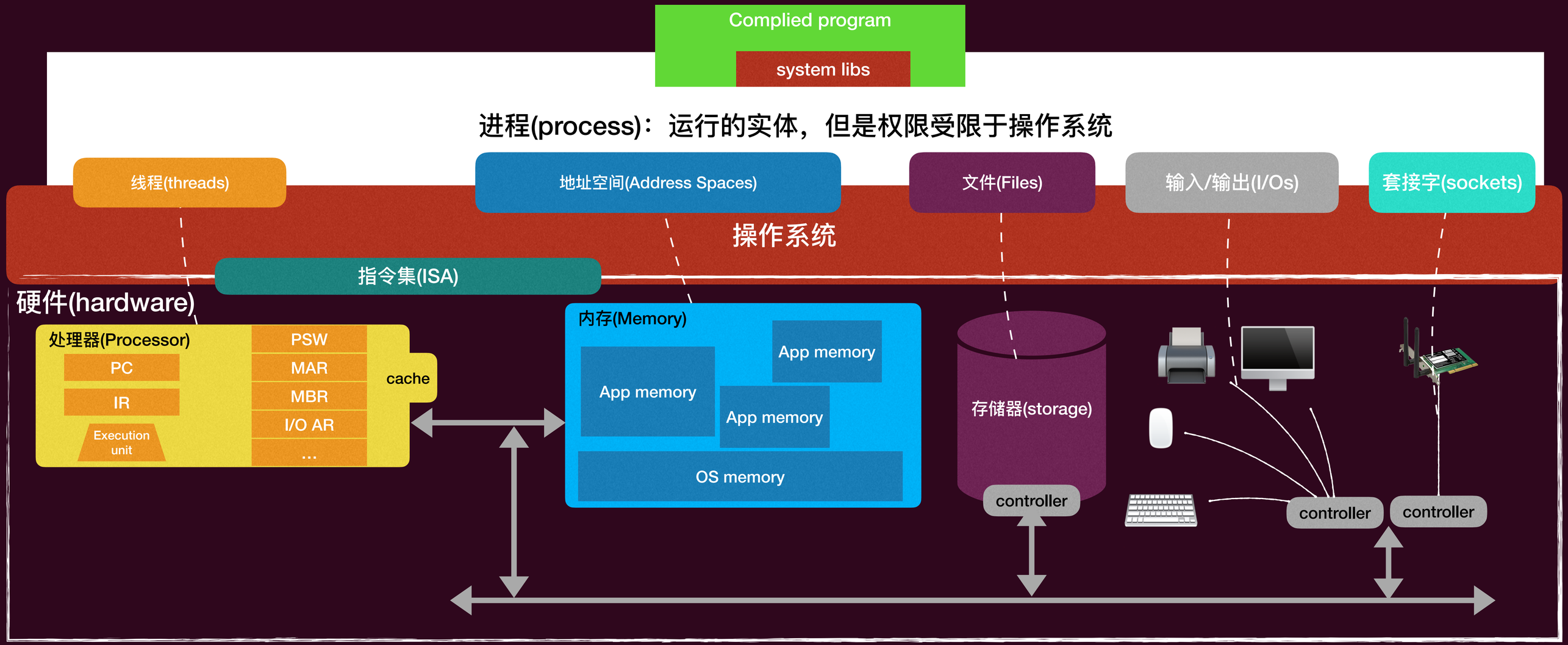


- 没有操作系统，程序将直面底层硬件
  - 非统一接口，含有大量细节
  - 有限制
  - 不安全

操作系统的功能

- 幻术师
  - 提供简洁、易用的物理层抽象
    - 拥有无限的内存，专属的机器
    - 高级的对象：文件、进程、信号...
    - 屏蔽限制

操作系统—幻术师


Complied program
system libs
进程(process)：运行的实体，但是权限受限于操作系统
线程(threads)
地址空间(Address Spaces)
文件(Files)
输入/输出(I/Os)
套接字(sockets)
操作系统
指令集(ISA)
硬件(hardware)
处理器(Processor)
PC
IR
Execution unit
PSW
MAR
MBR
I/O AR
...
cache
内存(Memory)
App memory
App memory
App memory
OS memory
存储器(storage)
controller
controller
controller

对于程序来说，他所认为的世界

用户态 (User)

Complied program

system libs

进程(process)：运行的实体，但是权限受限于操作系统

线程(threads)

地址空间(Address Spaces)

文件(Files)

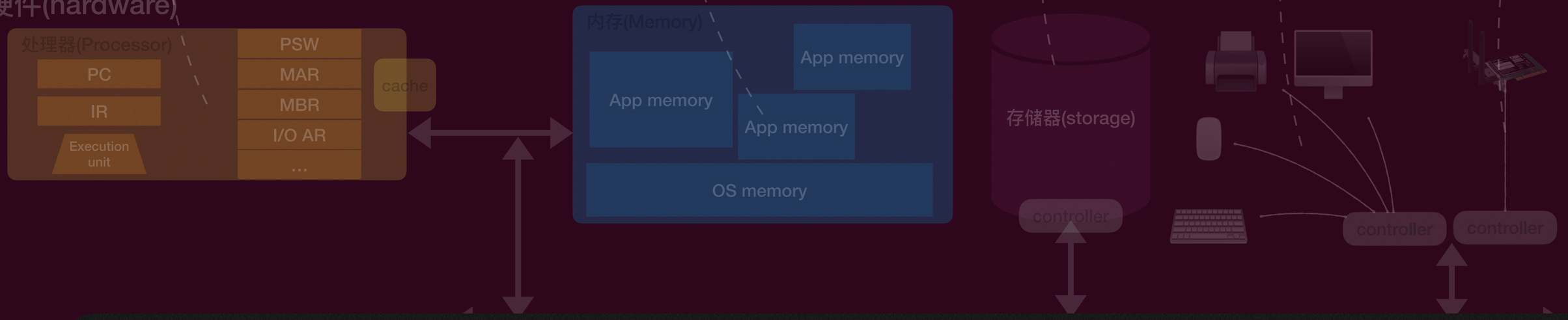
输入/输出(I/Os)

套接字(sockets)

操作系统

受限的指令集(ISA)

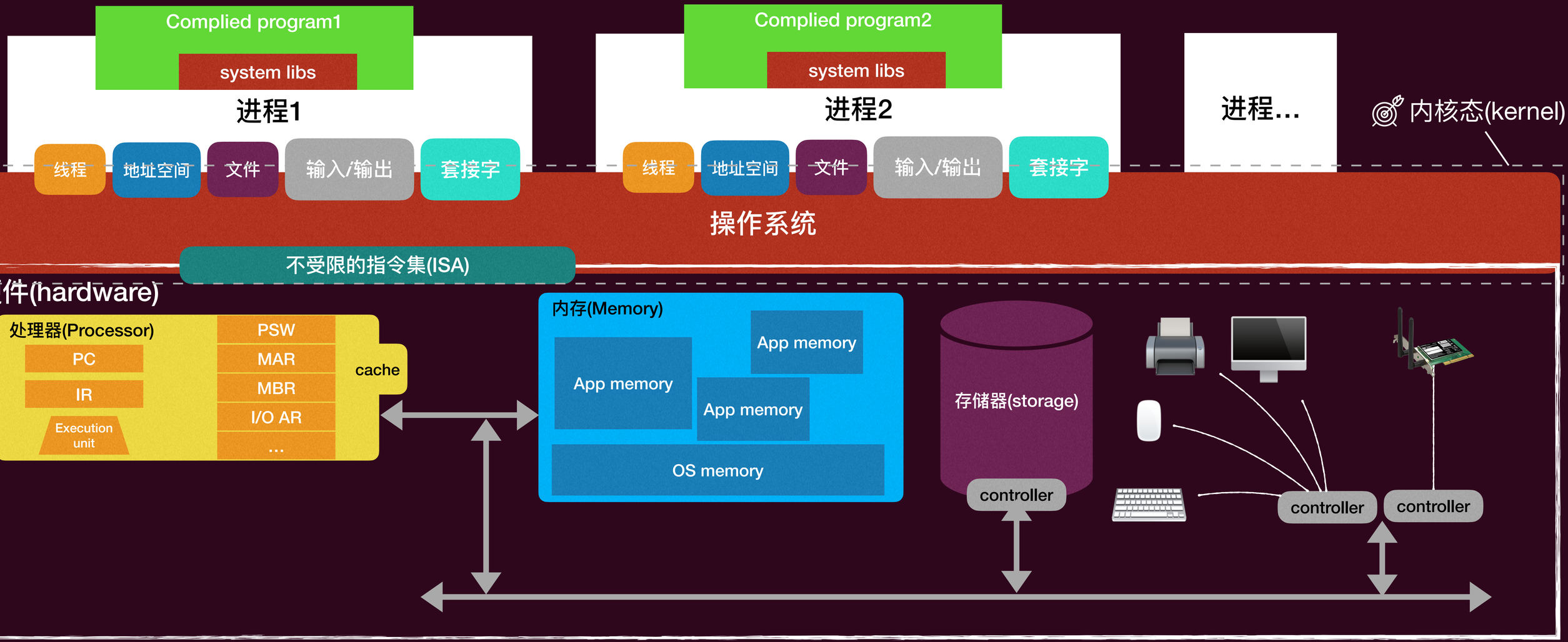
硬件(hardware)



程序的“机器”就是操作系统提供的进程这个抽象，每个程序运行在自己的进程上，进程提供了比原始硬件机器更加优美的接口

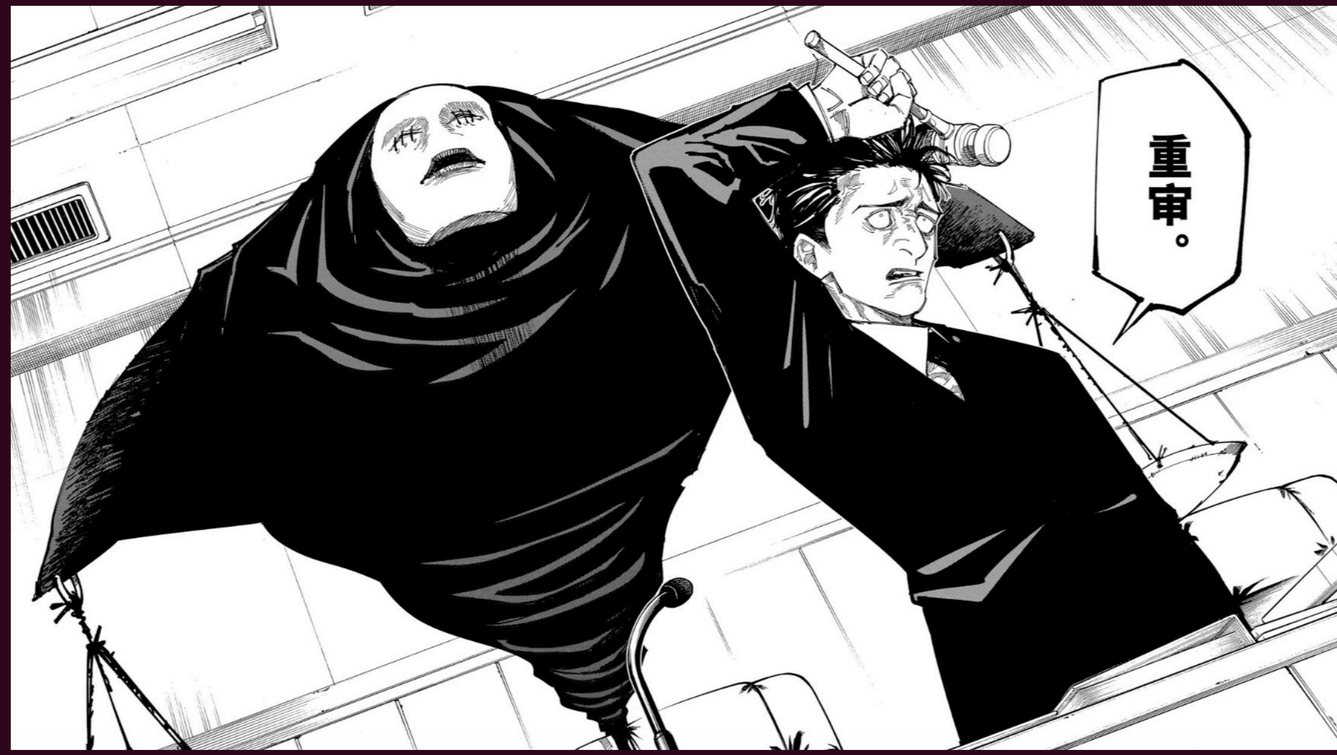
对于操作系统而言的世界

操作系统把“难看”的硬件接口翻译成了友好易用的接口，为每个程序提供他们的进程

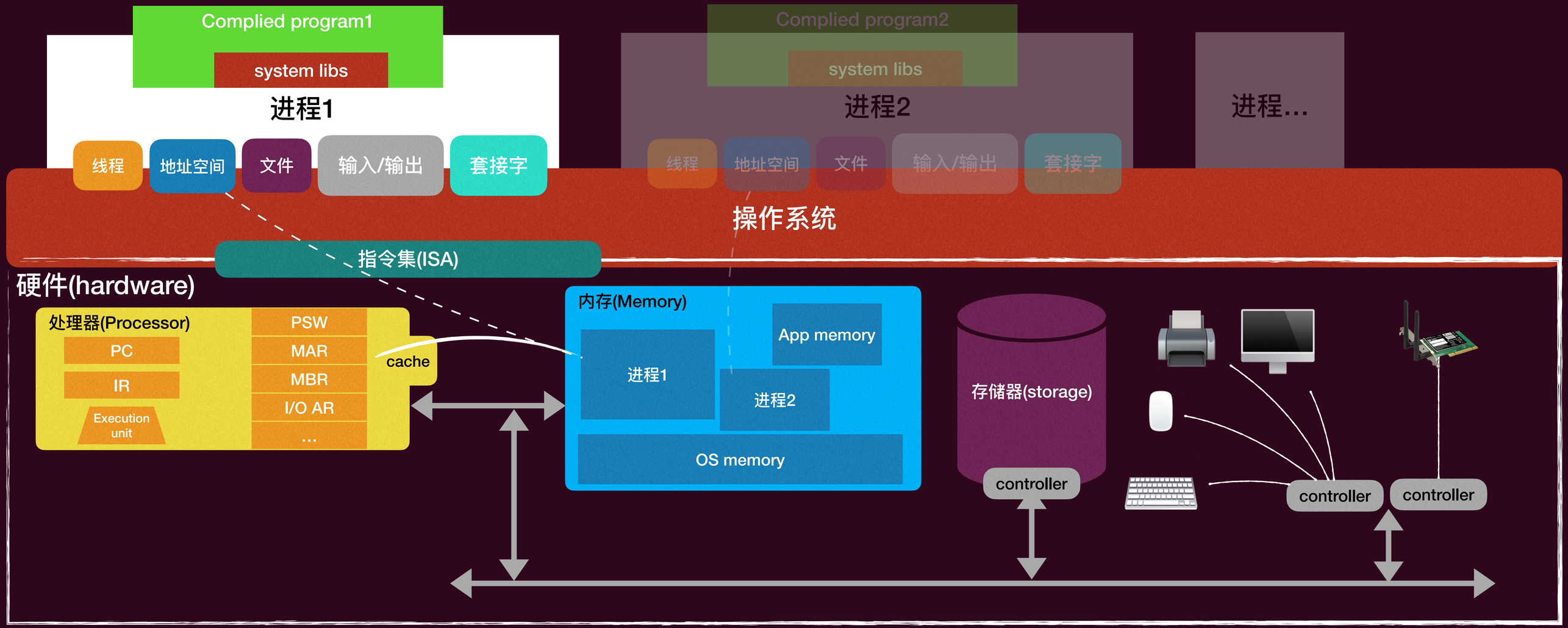


操作系统的功能

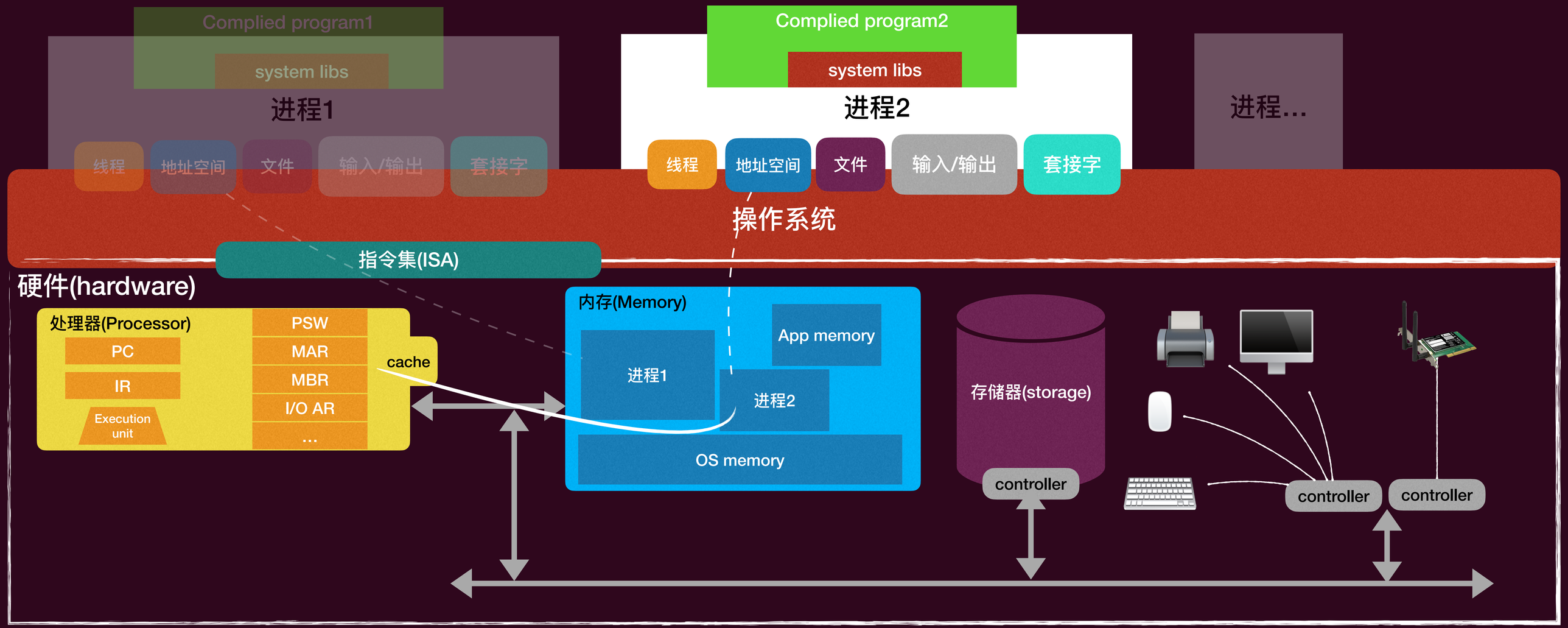
- 裁决人
  - 资源的管理
    - 分配
    - 保护
    - 分享



进程运行


Complied program1
system libs
进程1
线程
地址空间
文件
输入/输出
套接字
Complied program2
system libs
进程2
线程
地址空间
文件
输入/输出
套接字
进程...
操作系统
指令集(ISA)
硬件(hardware)
处理器(Processor)
PC
IR
Execution unit
PSW
MAR
MBR
I/O AR
...
cache
内存(Memory)
进程1
App memory
进程2
OS memory
存储器(storage)
controller
controller
controller

进程切换


Complied program1
system libs
进程1
线程
地址空间
文件
输入/输出
套接字
Complied program2
system libs
进程2
线程
地址空间
文件
输入/输出
套接字
进程...
操作系统
指令集(ISA)
硬件(hardware)
处理器(Processor)
PC
IR
Execution unit
PSW
MAR
MBR
I/O AR
...
cache
内存(Memory)
进程1
App memory
进程2
OS memory
存储器(storage)
controller
controller
controller

进程切换

操作系统将进程彼此隔离，操作系统将自己和其余程序隔离

Segmentation fault (core dumped)

处理

Complied program1

system libs

进程1

线程 地址空间 文件 输入/输出 套接字

Complied program2

system libs

进程2

线程 地址空间 文件 输入/输出 套接字

进程...

操作系统

指令集(ISA)

硬件(hardware)

处理器(Processor)

PC

IR

Execution unit

PSW

MAR

MBR

I/O AR

...

cache

内存(Memory)

进程1

App memory

进程2

OS memory

存储器(storage)

controller

controller

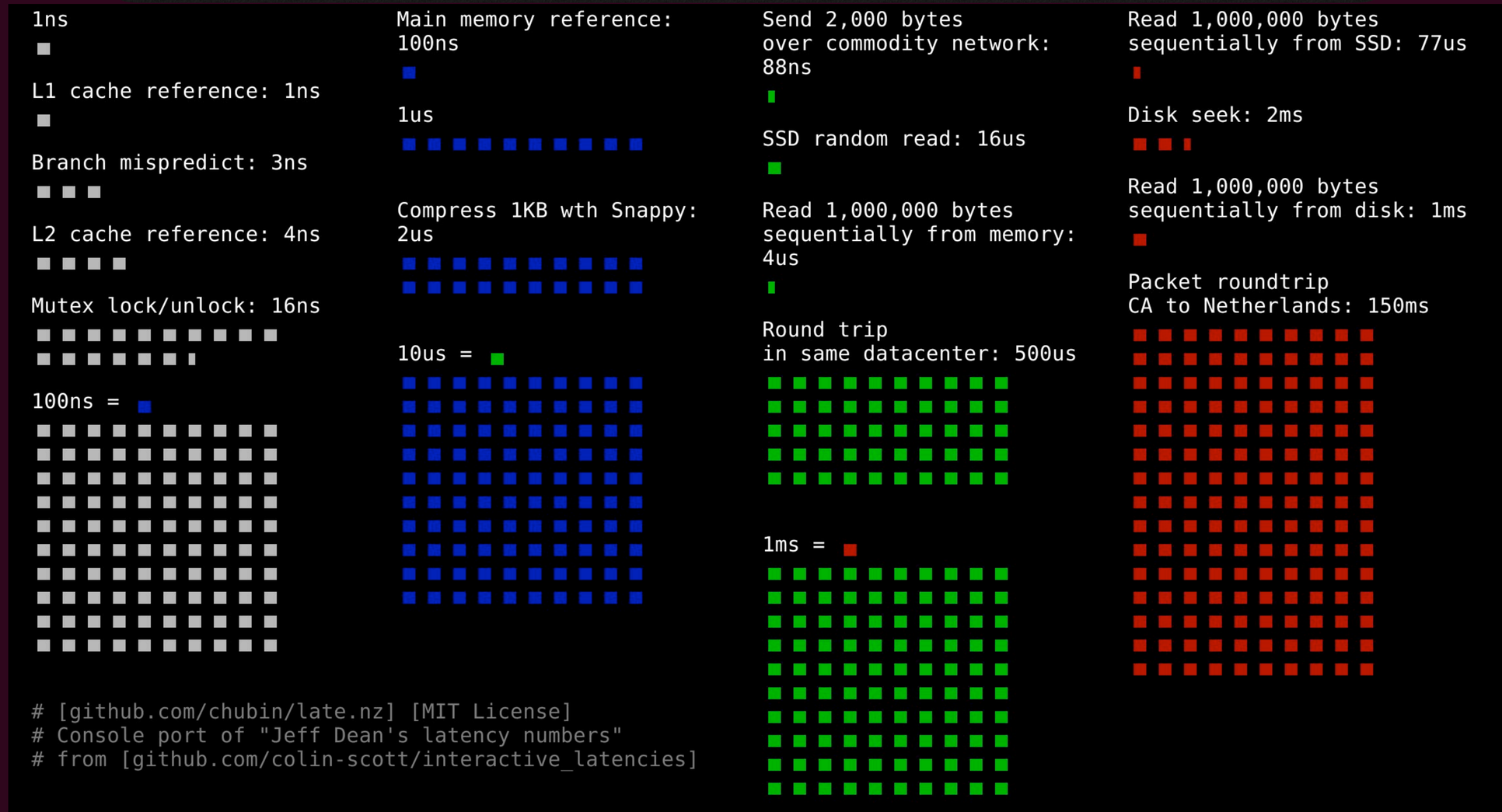
controller

操作系统的功能

- 性能优化员
  - 操作系统的底线是支持应用程序的运行
    - 自身反而次要
    - 理想情况下操作系统自身的资源开支(overhead)应该不大
    - 应用程序运行应该尽可能快

Latency numbers every programmer should know

操作系统不断的在各种存储设备间腾挪转换，以达到应用程序性能最优

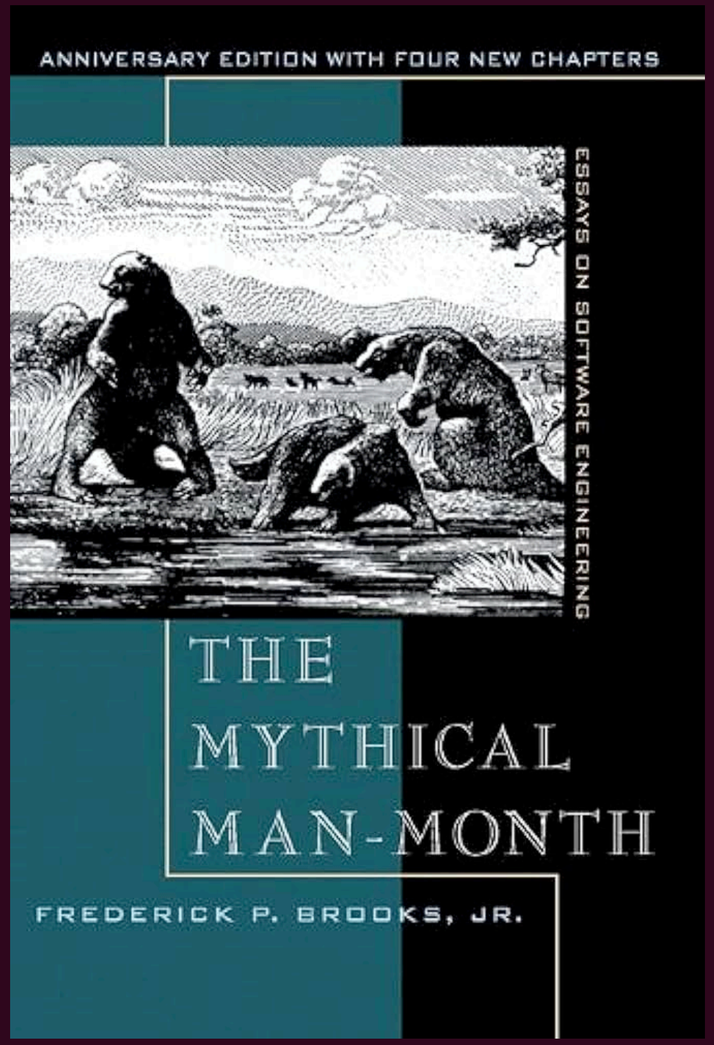


操作系统的功能

- 当然，操作系统还有一些其他的任务：
  - 可靠性：尽可能不崩溃，即使崩溃，尽可能不损坏业务逻辑
  - 安全：对抗恶意的程序
  - 能源：尽可能省电
  - …

操作系统的发展

- 操作系统已经人类目前为止最为复杂的软件系统之一了
  - 《人月神话》这本书专门吐槽了Frederick P. Brooks当年开发IBM System/360操作系统的各种坑
    - 开发复杂系统中**软件工程**思维很重要!
  - 现如今，商业操作系统已经不是一个单个人能够写出来的了
- 当然，操作系统其也不是一开始就这么巨大的。。。。
  - 计算机硬件发展 → 更复杂的应用和 API
  - 更复杂的应用和 API → 计算机硬件发展



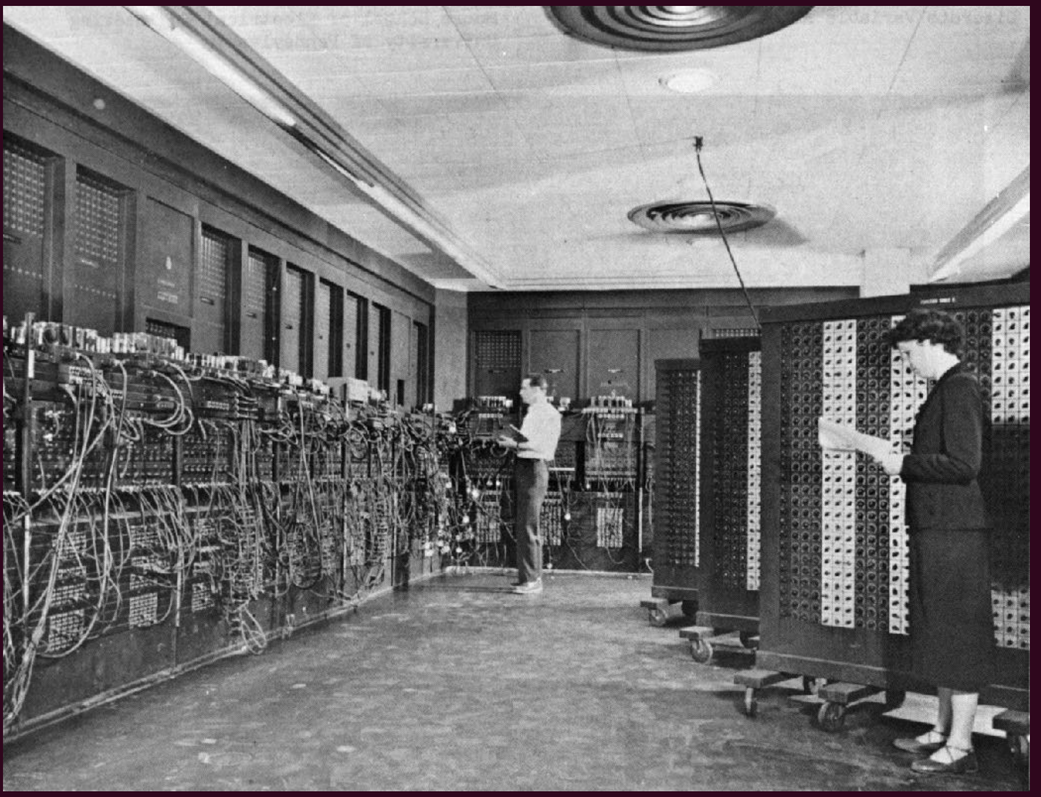
操作系统历史

- The first generation (1945–55) 真空管时代
- The second generation (1955–65) 晶体管和批处理系统
- The third generation (1965–1980) 集成电路和多道处理程序
- The fourth generation (1980–present) 个人计算机时代
- The fifth generation (1990–present) 移动计算机时代

一开始...

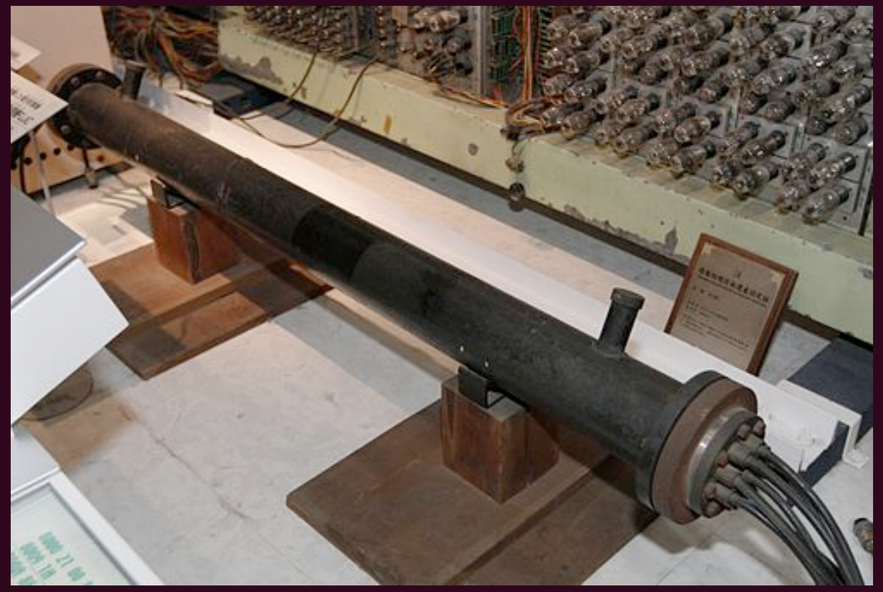
- 只有硬件，没有其他...
  - 没有操作系统
  - 没有系统库函数
  - 没有编译器

1945s-1955s 真空管时代



世界上第一台通用计算机“ENIAC”

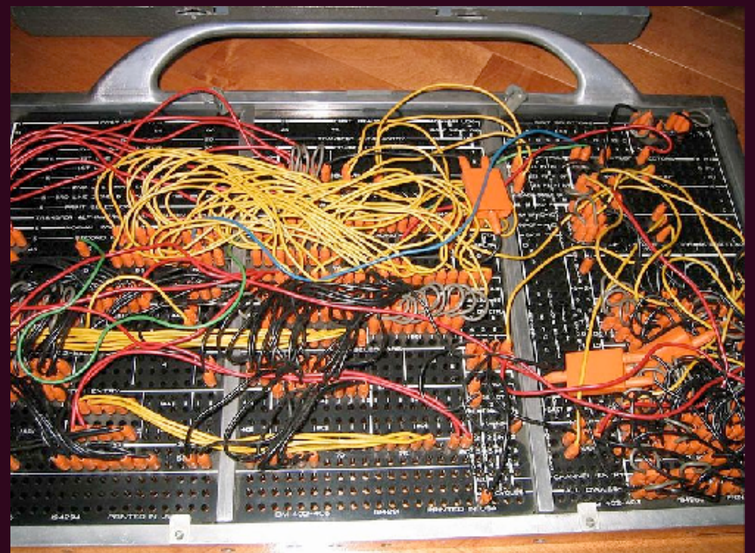
真空管：实现逻辑门

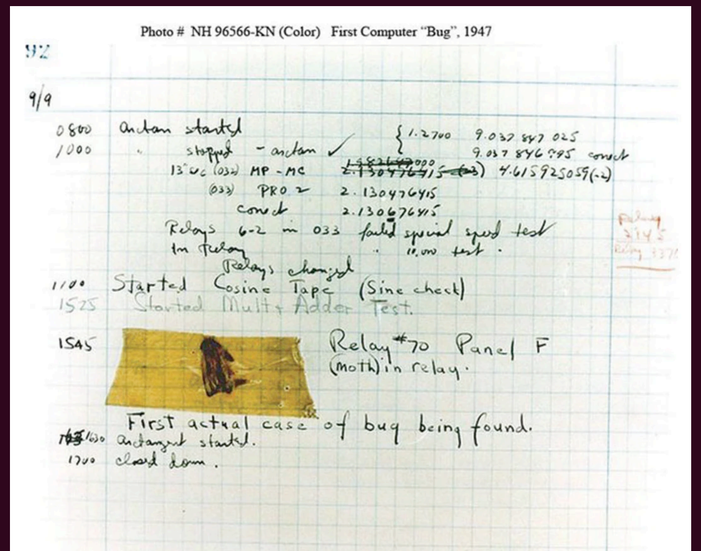


延迟线：实现存储器

1945s-1955s 真空管时代

- 同时期其他计算机: Z3, Colossus (发明者之一：Alan Turing)
- 没有操作系统，运行的程序是一些简单的计算：打印平方数、素数表、计算弹道...
  - 一个程序的运行将独占整个计算机
- 编程是物理线路连接（hard-wire），以及需要解决真正的“bug”!





1945s-1955s 真空管时代

- 在1950s早期，人们终于受够了硬连线，开始使用穿孔卡来表达编码，卡片机的想法最早起源于1801年的纺织机

IBM 026 Printing Card Punch

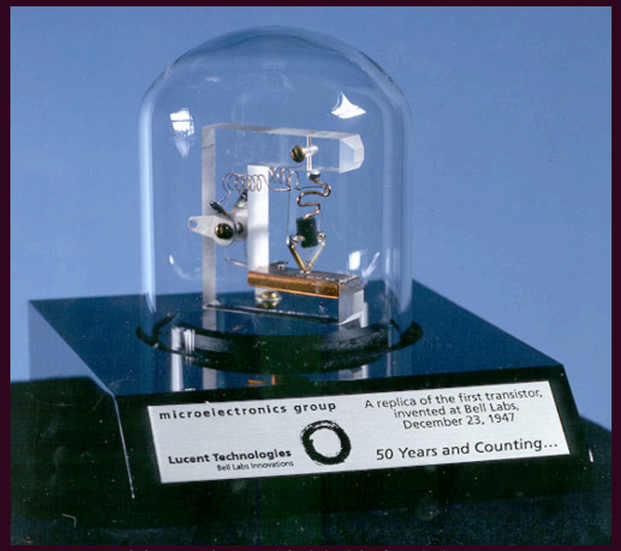
An IBM 026 commercial card code

雅卡尔织布机: 通过打孔卡片控制织出的图形

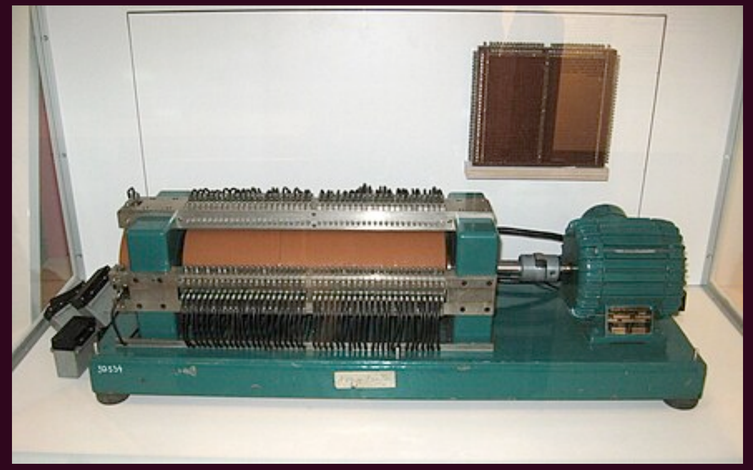
- 编程从此变得容易了…

1955s-1965s 晶体管时代

- 晶体管的出现使得我们可以设计出更小、更快和更加可靠的逻辑门
- 磁芯使得我们可以得到更大的内存



第一个晶体管的复制品

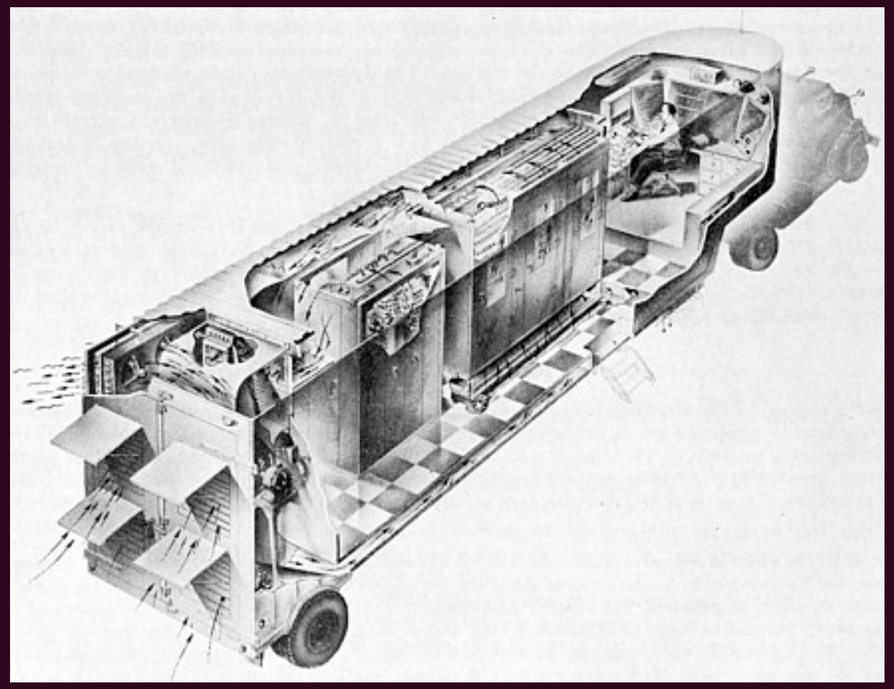


Drum memory from Sweden's first binary computer

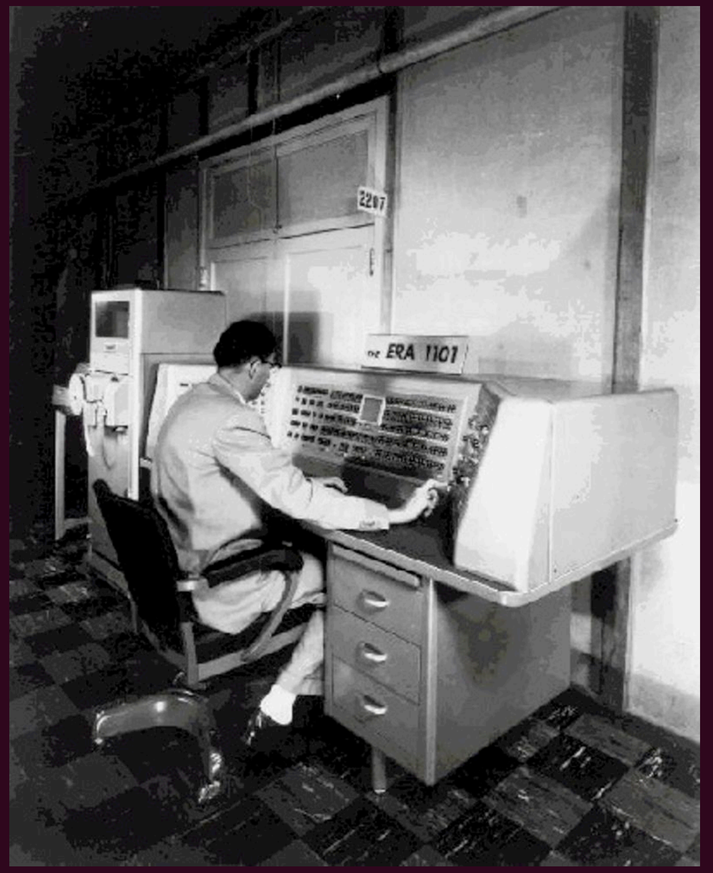
The IBM 7094 at Columbia University

1950s 中断出现

- I/O 设备的速度已经严重低于处理器的速度，中断机制出现



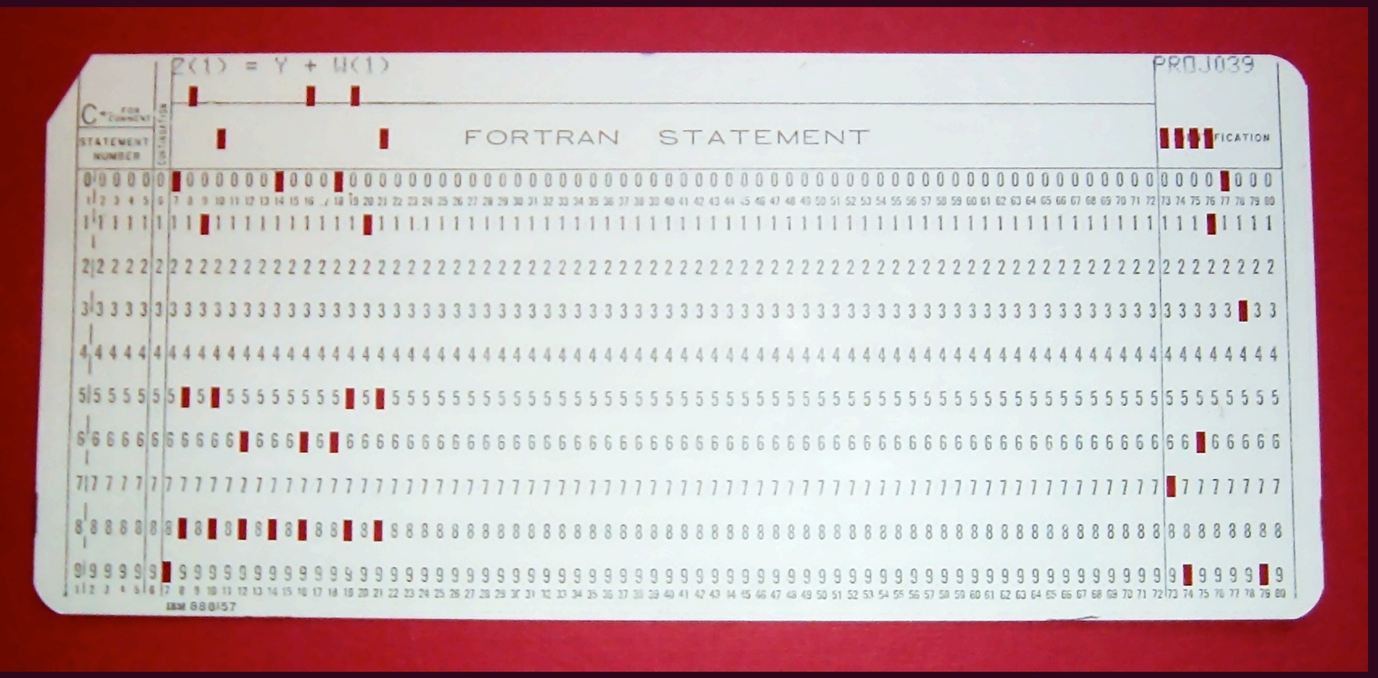
DYSEAC：第一个使用中断的计算机，建在一个卡车里面



中断的出现使得人们不必等待漫长的轮询I/O是否完成，可以转而做其他事情，等待I/O完成之后通过中断机制回来

1950s 的程序

- 更复杂的通用的数值计算
  - 高级语言和 API 诞生 (Fortran, 1957)：一行代码，一张卡片



一个程序需要一堆卡片

1950s 的程序

- 希望使用计算机的人越来越多
  - 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
- 程序和程序之间开始需要协作
  - 程序并不希望直接用指令访问 I/O 设备，而是希望能方便地管理 “文件”

1955s-1965s 的操作系统-批处理系统

- 写程序 (打孔)、跑程序都是非常费事的
  - 计算机非常贵 ($50,000–$1,000,000)，只有大的机构（比如大学）才拥有
  - 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（操作任务系统—Operating System Job）
  - 成批管理程序，一个跑完可以自动的换下一个程序
  - 有了一些通用的库函数
  - Disk Operating Systems (DOS)
    - 操作系统中开始出现 “设备”、“文件”、“任务” 等对象和 API

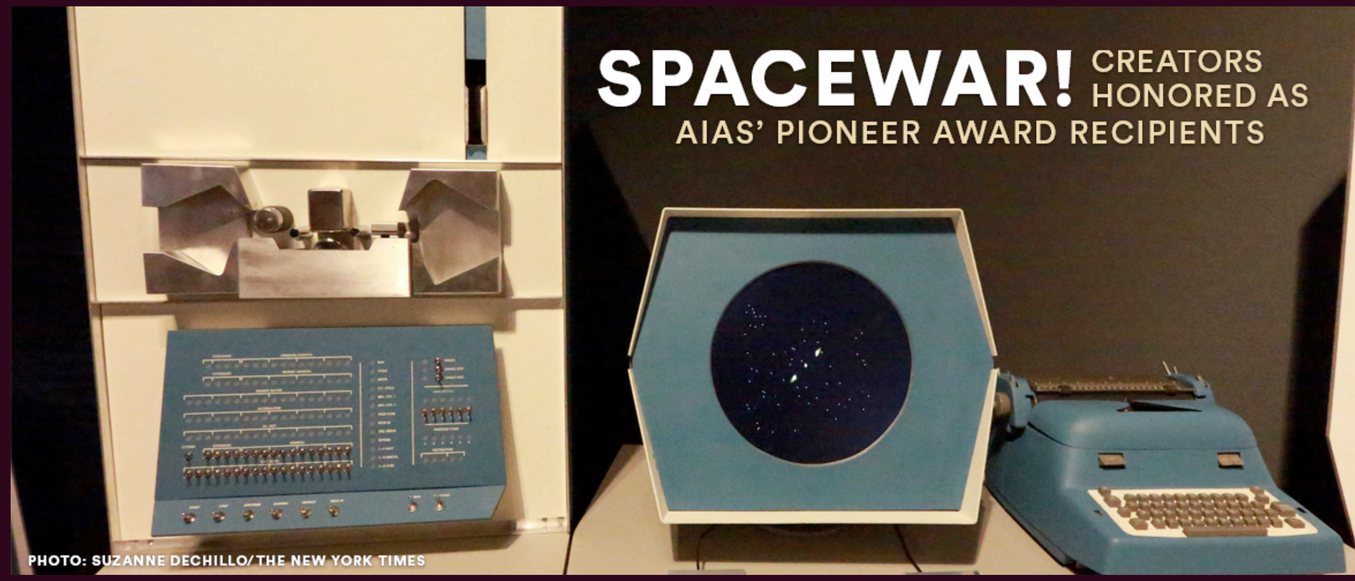
1965s-1980 集成电路时代

- 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

IBM system/360

1965s-1980 的软件

- 更多的高级语言和编译器出现
  - COBOL (1960), APL (1962), BASIC (1965)
  - Bill Gates 和 Paul Allen 在 1975 年实现了 Altair 8800 上的 BASIC 解释器
- 甚至有了游戏



1962年开发在PDP-1上的一款游戏

1965s-1980 的操作系统

- 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
  - 进程在执进程在执行 I/O 时，可以将 CPU 让给另一个进程（多道程序设计 Multiprogramming or multitasking）
    - 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

1965s-1980 的操作系统

- 机器已经不如之前昂贵了
  - 充分利用CPU已经没有那么重要，人的使用体验越来越重要，没人愿意长时间等待机器响应
  - 既然可以在程序之间切换，我们就可以定时切换
    - 虽然更多的切换使得overhead增加了，但有机会让用户得到更快的响应
  - 可以方便的支持多个用户同时使用某个超级计算机（云计算的雏形）
  - Multics (MIT, 1965)：现代分时(Time-Sharing)操作系统诞生

H6180 Multics at Honeywell CISL

1970代 Unix诞生

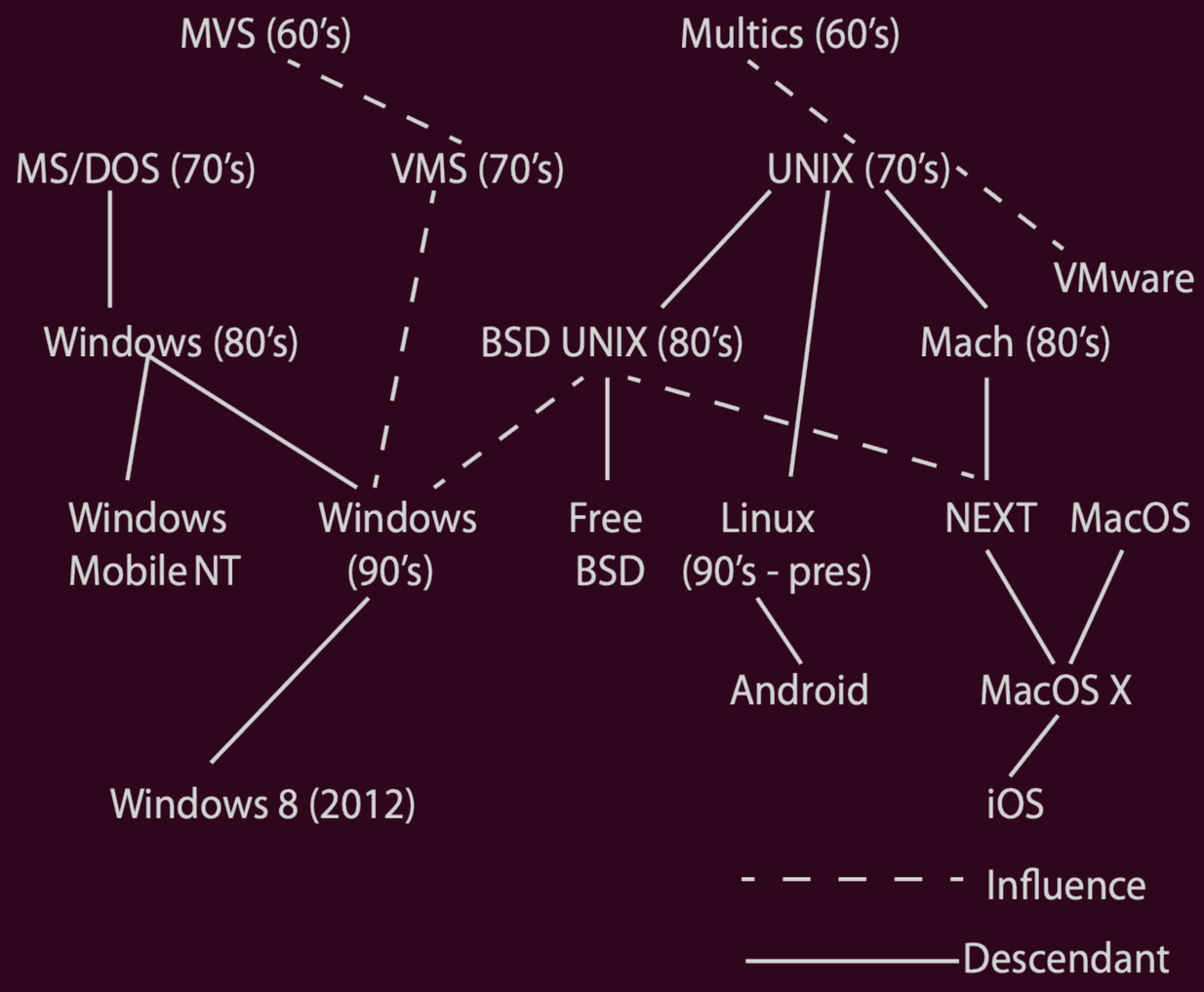
- 基于Multics，Ken Thompson 和Dennis Ritchie’s 在AT&T 写了Unix OS [1970]
- Unix 是第一个可移植的操作系统:
  - 大部分基于C语言
  - 可以将Unix移植到其他平台
- Unix具有高度模块化的设计，其理念影响深远：
  - 操作系统应该提供简单的工具
  - 每个工具都应该具有有限的但是明确的功能

Ken Thompson (sitting) and DeRitchie (standing) running Unix on a PDP-11

Ken Thompson and Dennis Ritchie, principal developers of AT&T Unix [1973]

Unix

- 很多操作系统概念被进一步完善，奠定了现代操作系统的形态。
- 基于Unix，又有很多其他操作系统被开发出
  - System V, BSD (Berkeley Software Distribution), Minix, Linux, Mach, …
- IEEE 基于Unix开发了 POSIX 标准
  - 其规定了Unix类系统需要提供的系统调用
  - 实际上如今大部分操作系统（不只是Unix类系统）都支持这一标准

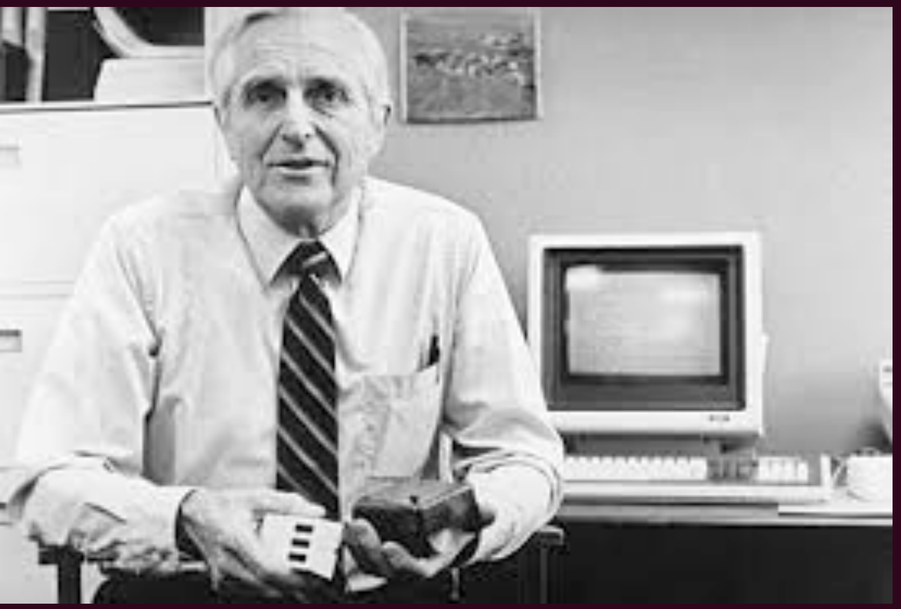


1980s-Present 个人电脑时代

- 大规模集成电路(Large Scale Integration)高度发展
  - 更多的晶体管，更小，性能更强
  - 个人电脑兴起，计算机已经和现代无异
- 1974年Intel 8080 (8-bit cpu)出现，使得CISC指令集更加流行（源于IBM/360）
  - Intel让Kildall写了Disk-based Operating System (DOS) CP/M (Control Program for Microcomputers)
  - 之后Tim Paterson在1980年写了类似的86-DOS，后被微软收购，改名为MS-DOS

1980–至今 个人电脑时代

- 1960 Doug Engelbart 发明了图形用户界面，并把他们用在了Xerox PARC研究中心的机器上

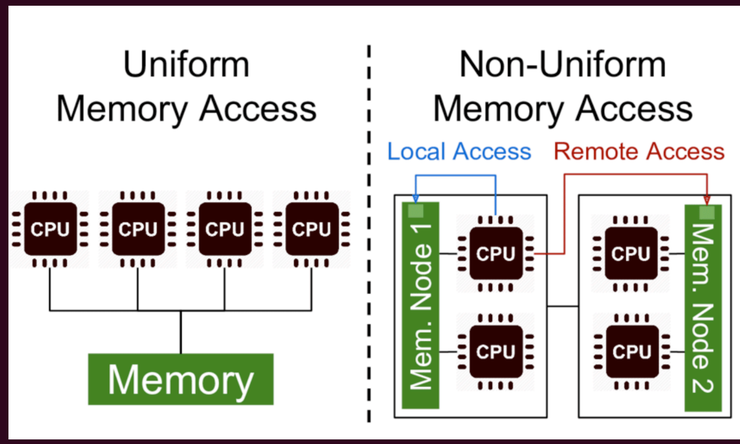


Apple got the idea for the graphical user interface from Xerox. Later, when Steve Jobs accused Bill Gates of stealing from Apple, Gates calmly said, "Well, Steve, I think there's more than one way of looking at it. I think it's more like we both had this rich neighbor named Xerox and I broke into his house to steal the TV set and found out that you had already stolen it."

如今的操作系统

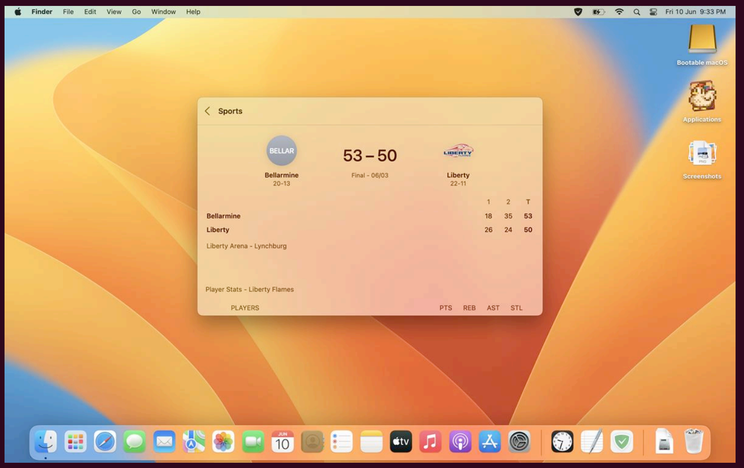
- 更加复杂和多样的硬件环境
  - 面对更复杂的处理器和内存
    - 非对称多处理器 (ARM big.LITTLE; Intel P/E-cores)
    - Non-uniform Memory Access (NUMA)
    - 更多的硬件机制 Intel-VT/AMD-V, TrustZone…
  - 更多的设备和资源
    - SSD、GPU、FPGA...

Cortex A57/A53 MPCore big.LITTLE CPU chip

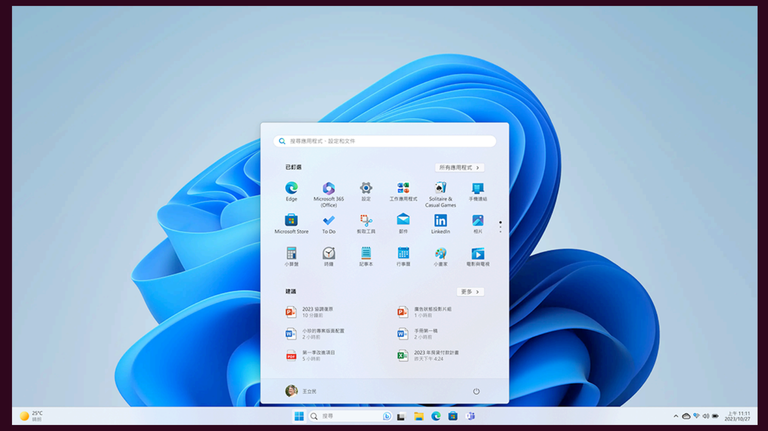


如今的操作系统

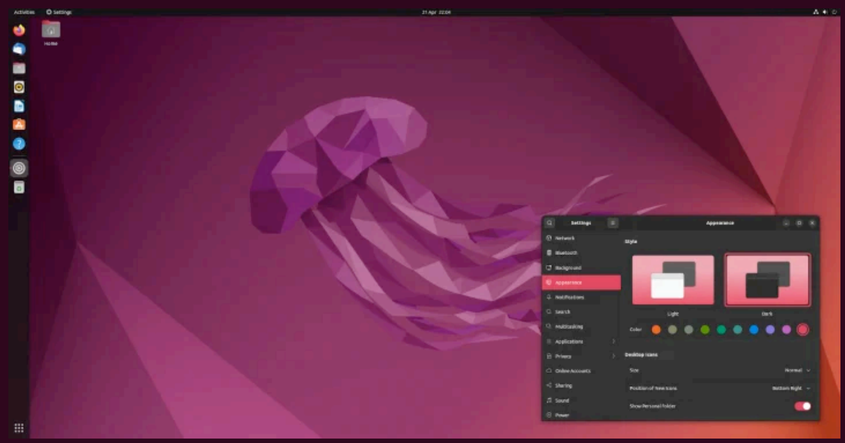
- 主流的PC操作系统



- Apple



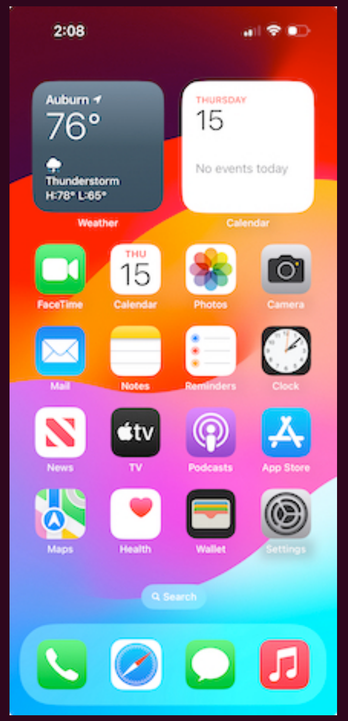
- Microsoft

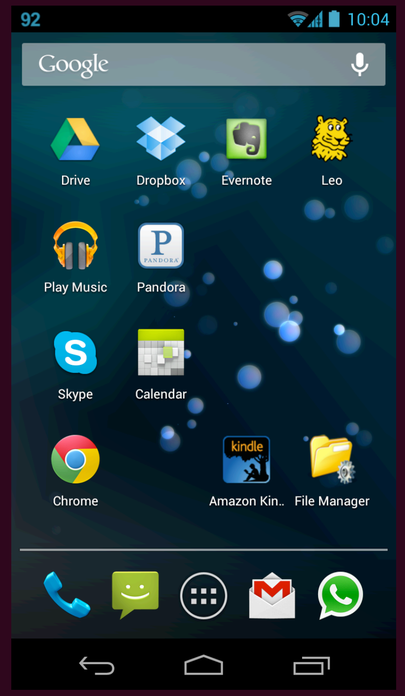


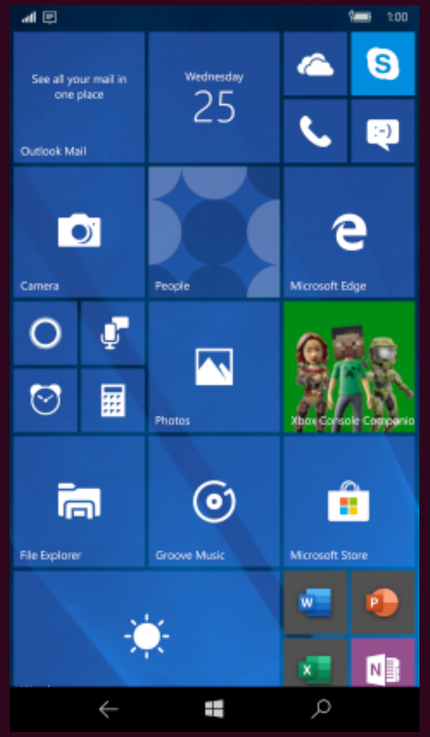
- Linux

如今的操作系统

- 移动计算机: Symbian OS, Blackberry OS , iOS, Google's Android, windows phone, Nintendo Switch system software, 鸿蒙…

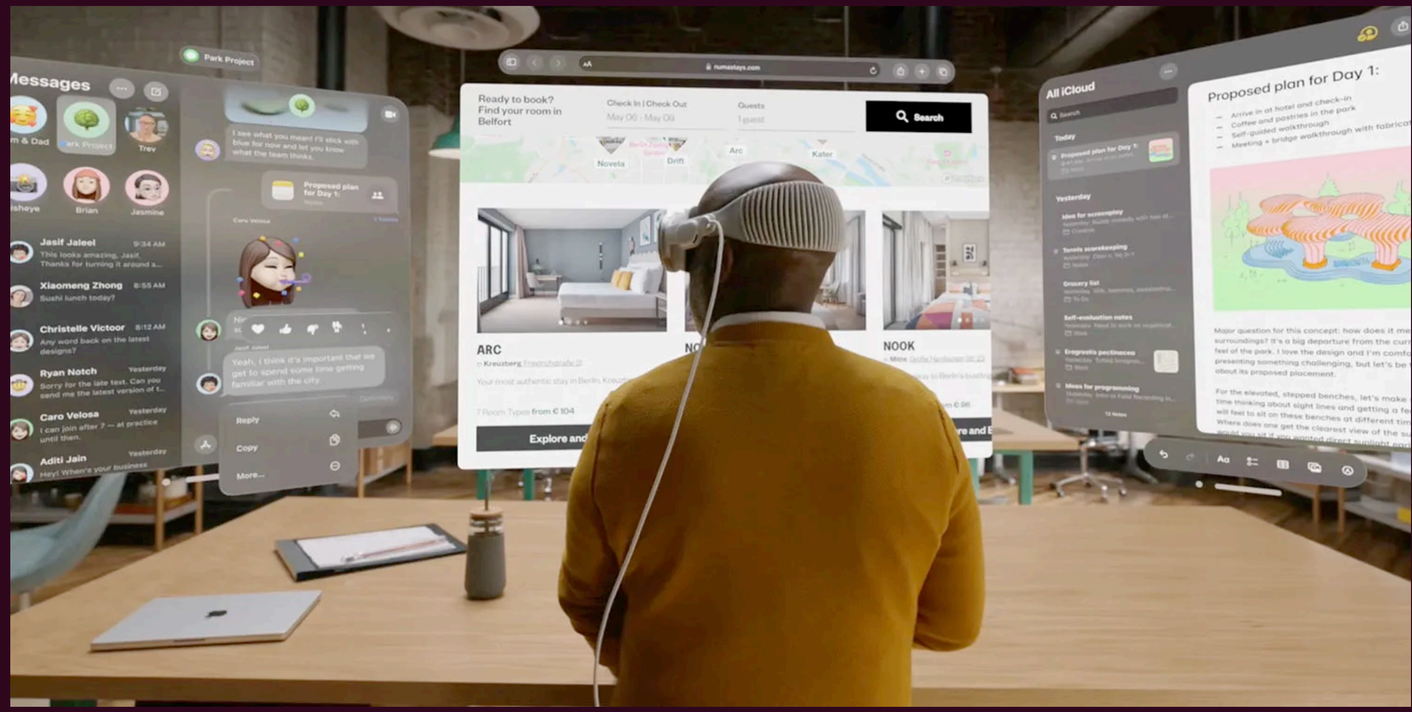






如今的操作系统

- 可穿戴设备操作系统：WatchOS, Glass OS, VisionOS…



操作系统历史中的一些“贝壳”

Maurice Wilkes
1967图灵奖

EDSAC, 1949
Multi-programming
第一台存储程序式电子计算机

Frederick Brooks
1999图灵奖

IBM system/360, 1964

Fernando J. Corbato
1990图灵奖

CTSS, 1961 & Multics, 1969
分时操作系统

Ken Thompson & Dennis Ritchie
1983图灵奖

Unix, 1971
多任务多用户操作系统

Barbara Liskov
2008图灵奖

Venus, 1972
小型低成本交互式分时操作系统

总结

- 操作系统
  - 提供了方便的抽象来处理各种硬件
    - 易用、受保护
  - 协调资源
  - 给应用提供了方便的系统服务，优化性能
  - 历史悠久

阅读材料

- [OSTEP] Ch 1, Ch 2
- 配置好编程环境
  - 准备好一个可用的 Linux 系统(可选以下任意一个方案)
    - 可以直接物理装机
    - Windows subsystem for linux
    - VirtualBox或其他虚拟机安装linux
    - 阿里云等购买linux虚拟主机
  - 熟悉基础命令行工具
    - tldr和man、gcc、gdb、make、git、objdump, vim...