

# 第 1 讲：操作系统概述

## 第一节：课程概述

向勇、陈渝

清华大学计算机系

*xyong,yuchen@tsinghua.edu.cn*

2020 年 5 月 5 日

# 课程信息

- 主讲教师：向勇、陈渝
- 助教：
  - 王润基、贾越凯、戴臻旸、刘丰源
  - 吴一凡、潘庆霖、张译仁、陈嘉杰

# 预备知识

- 程序设计语言（汇编、C 和 Rust）
  - :( 不是开发应用程序
  - :) 而是开发系统程序
- 数据结构
  - :) 理解基本数据结构即可
- 计算机组成原理
  - :( 康总的 x86/mips 原理
  - :) Patterson 的 RISC-V 原理
- 编译原理
  - :) 没学过影响不大
  - :( 但还是要了解高级语言 <->RISC-V 汇编语言

# 课程信息

- 课程 WIKI
  - 所有课程信息的入口，也是最新课程信息的官方发布网站
- 课程视频
  - 学堂在线：操作系统 (RISC-V)
- 作业
  - 网络学堂 – 向老师
  - 网络学堂 – 陈老师
- 实验楼的在线实验环境
  - ucore os 在线实验环境
  - rcore os 在线实验环境
- 讨论区
  - Piazza 交流论坛
  - 微信：OS2020

# 第 1 讲：操作系统概述

## 第二节：教学安排

向勇、陈渝

清华大学计算机系

*xyong,yuchen@tsinghua.edu.cn*

2020 年 5 月 5 日

# 参考教材

## 参考教材

- Operating Systems: Three Easy Pieces  
操作系统：三大简易元素
- Operating System Concepts  
操作系统概念
- Operating Systems:Internals and Design Principles  
操作系统：精髓与设计原理

## 上课时间地点

- 星期一 (1-13 周) 上午第 2 大节 09:50-11:25 五教 5105、五教 5305
- 星期四 (1-13 周) 上午第 1 大节 08:00-09:35 五教 5105、五教 5305

# 教学内容



## 操作系统原理与实现

- 操作系统结构
- 中断及系统调用
- 内存管理
- 进程管理
- 处理机调度
- 同步互斥
- 文件系统
- I/O 子系统

# 作业与实验

- 平时作业
  - 课上练习与交流
  - 课后练习
- 基础实验
  - uCore 实验：基于 X86 用 C 写教学操作系统
  - rCore 实验：基于 RISC-V 用 rust 写教学操作系统
- 课程设计（大实验）

# 基础实验

uCore 实验：基于 X86 用 C 写教学操作系统

- 实验零：操作系统实验准备
- 实验一：系统软件启动过程
- 实验二：物理内存管理
- 实验三：虚拟内存管理
- 实验四：内核线程管理
- 实验五：用户进程管理
- 实验六：调度器
- 实验七：同步互斥
- 实验八：文件系统

# 基础实验

rCore 实验：基于 RISC-V 用 rust 写教学操作系统

- 第一章：独立式可执行程序
- 第二章：最小化内核
- 第三章：中断
- 第四章：内存管理
- 第五章：内存虚拟化
- 第六章：内核线程
- 第七章：线程调度
- 第八章：进程
- 第九章：文件系统
- 第十章：同步互斥

# 课程设计

- 各种 CPU 平台上的操作系统移植
  - RISC-V、x86-64、x86-32、MIPS、ARM
- 操作系统内核功能实现和扩展
  - GUI、驱动、内核可加载模块、微内核
- 操作系统分析工具
  - 错误分析、行为分析、模拟器
- 操作系统教学实验设计
  - uCore、rCore、zCore
- 操作系统新方向探索
  - rust、内核语言

# 成绩评定

- 作业：5 分
- 实验：15 分
  - 独立完成 uCore 或 rCore 中的操作系统功能实现，并提交实验报告
- 考试或课程设计：80 分
  - 期中考试：35 分
  - 期末考试：45 分
  - 有余力和兴趣的同学，可用课程设计替代考试

总成绩加权方法：上述各项成绩的总和会做一次调整，基本原则是，各分数段保持一定的比例，可能的

参考比例为 A+/A/A- 占 25%、B+/B/B- 占 45%、C+/C/C- 占 20% 和 D+/D/F 占 10%。

详见：[操作系统课程的成绩评定标准说明](#)

# 调查问卷

- 为什么要学这门课?
- 你打算如何来学这门课?
- 对自己的课程学习要求是什么?
- 你愿意如实报告是否独立完成实验任务?
- 你希望在操作系统课上学到什么知识和什么能力?
- 以前的学习情况?
- 对计算机专业的看法是什么?
- 采集仅限于操作系统课内注册的同学信息

# 第 1 讲：操作系统概述

## 第三节：什么是操作系统

向勇、陈渝

清华大学计算机系

*xyong,yuchen@tsinghua.edu.cn*

2020 年 5 月 5 日

# 操作系统定义

没有公认的精确定义

## 操作系统定义

操作系统是管理硬件资源、控制程序运行、改善人机界面和为应用软件提供支持的一种系统软件。[计算机百科全书]

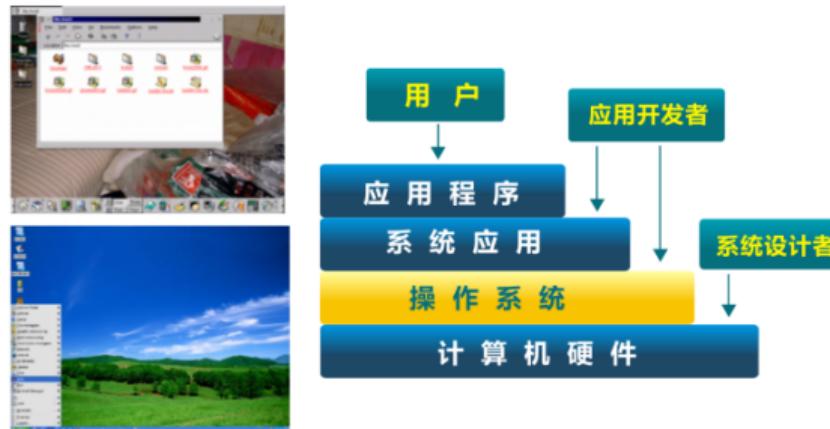
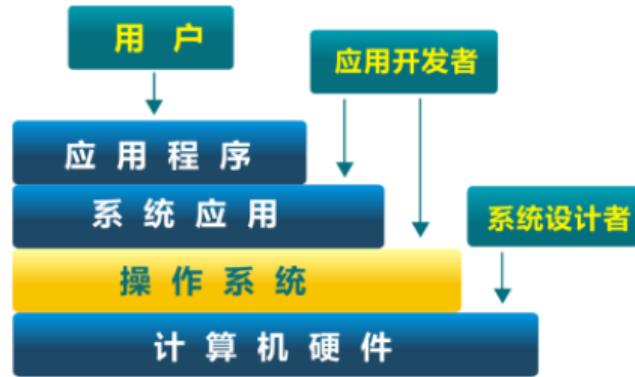
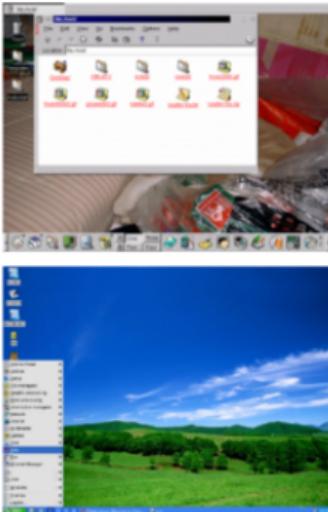


图: 承上启下的操作系统

# 操作系统解释



- 没有公认的精确定义
- 操作系统是一个控制程序
  - 一个系统软件
  - 控制程序执行过程, 防止错误
  - 执行用户程序, 给程序提供服务
  - 方便用户使用计算机系统
- 操作系统是一个资源管理程序
  - 应用程序与硬件之间的中间层
  - 管理各种软硬件资源
  - 提供访问软硬件资源的高效手段
  - 解决访问冲突, 确保公平使用

# 操作系统软件的分类

- Shell – 命令行接口
- GUI – 图形用户接口
- Kernel – 操作系统的内部

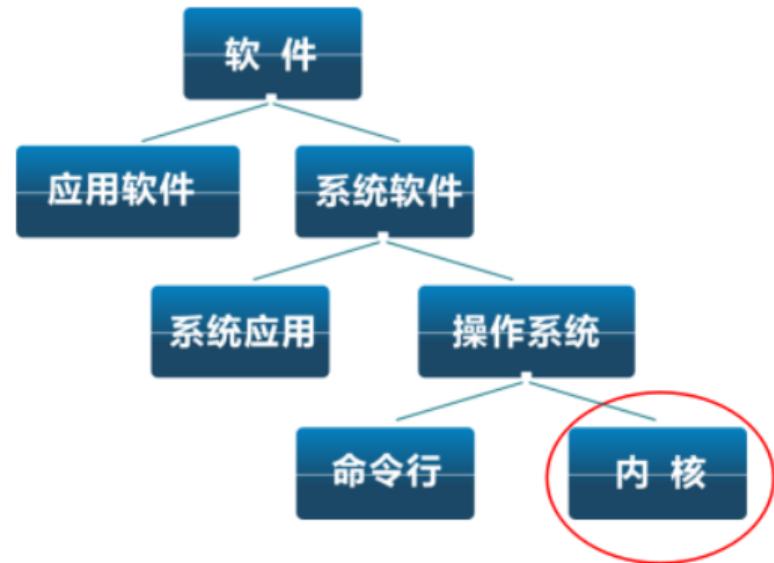


图: 操作系统的软件分类

# uCore/rCore 教学操作系统内核



图: uCore/rCore 操作系统内核

# 操作系统内核的抽象与特征

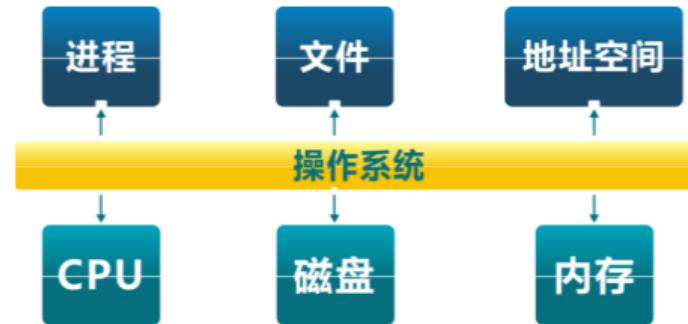


图: 操作系统抽象

## 操作系统内核的特征

- 并发：计算机系统中同时存在多个运行程序
- 共享：程序间“同时”访问互斥共享各种资源
- 虚拟：每个程序“独占”一个完整的计算机
- 异步：服务的完成时间不确定，也可能失败

# 第 1 讲：操作系统概述

## 第四节：为什么要学习和如何学习操作系统

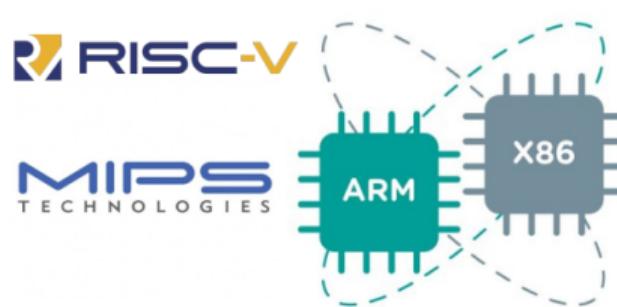
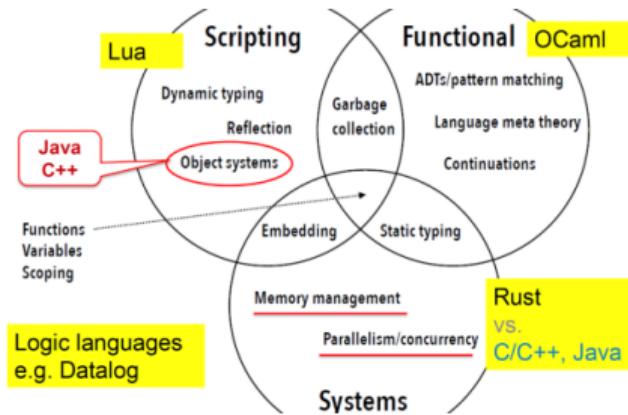
向勇、陈渝

清华大学计算机系

*xyong,yuchen@tsinghua.edu.cn*

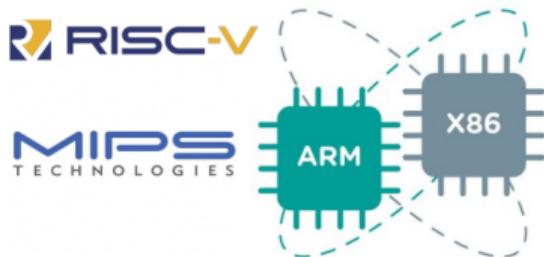
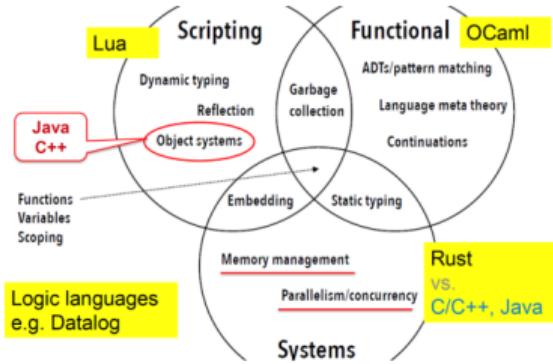
2020 年 5 月 5 日

# 操作系统课是多门课程的综合



- 综合课程-结合许多不同的课程
  - 程序设计语言
  - 数据结构
  - 计算机组成原理/体系结构
  - 编译技术
- OS 基本理论基础
  - 操作系统概念和原理
- OS 设计和实现
  - 操作系统源代码 & 实践技能

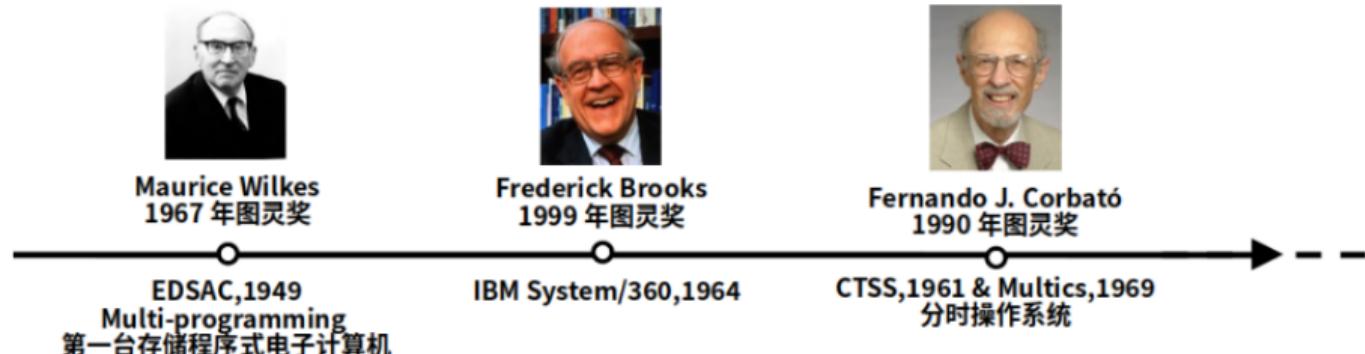
# 操作系统软件的地位



操作系统：计算机科学研究的基石之一

- 计算机系统的基本组成部分和核心支撑软件
- 贯穿程序语言、运行时系统、应用、体系结构
- 联系计算机科学和计算机系统的典范
- 操作系统的知识影响到专业人员的素质
- 大量专业工作与操作系统技术相关

# 操作系统研究相关的图灵奖



# 操作系统研究相关的产业

为了应对欧盟的反垄断裁决，Google 要向手机厂商收费了

公司

2018-10-17 14:07

5  
评论



三个月前 Google 收到了欧盟开出的 43.4 亿欧元（约合 339.3 亿元人民币）天价反垄断罚单，并被要求在三个月内终止非法行为，三个月后的今天 Google 正式作出了回应：要开始向手机厂商收取许可费了。



IBM 以340 亿美元收购 RedHat (红帽) ，史上最贵！

多易 发布于 2018-10-29

分类：云计算

来源：云头条



# 哪里在做操作系统研究？

- 顶尖大学的计算机科学部门
  - MIT, Stanford, Berkeley, ...
- 计算机产业
  - 旧时: Xerox (PARC), IBM, DEC (SRC), Bell Labs
  - 现代: Microsoft, Google, Yahoo, IBM, HP, Sun, Intel, VMware, Amazon, ...
  - 国内: 华为、阿里巴巴、腾讯...
- 学术研究协会
  - SOSP OSDI HotOS
  - ACM SIGOPS Hall-of-Fame Awards
  - USENIX USENIX-ATC

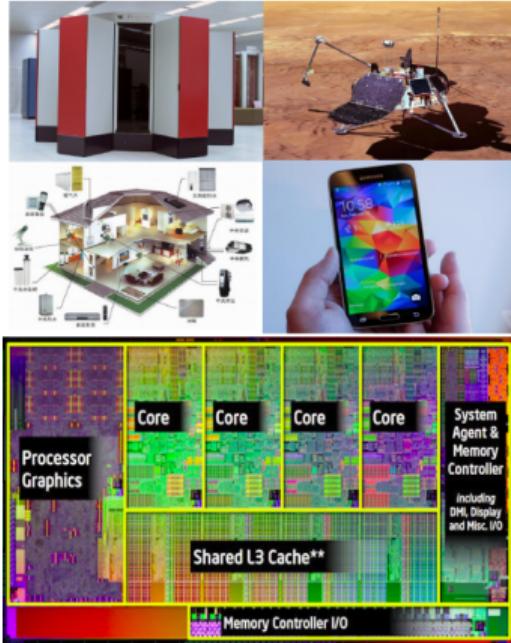
# 学习操作系统的目的



## ● 动机

- 已有操作系统很好，我将来的工作不会写操作系统
  - Windows,Linux,android,ios
- 已有操作系统是否解决了所有的事?
  - multi-core/parallel
  - Non-Volatile Storage
  - security
  - Energy
  - mobility/AIoT
  - correct/verification
- 为什么我要学习它?

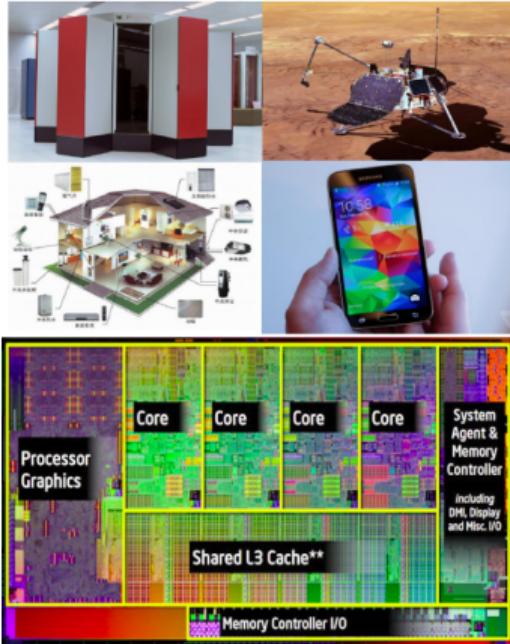
# 掌握操作系统具有挑战性 (1)



## 抓住操作系统的关键问题

- 操作系统很庞大
  - Windows XP 有 4500 万行
- 操作系统管理并发
  - 并发导致有趣的编程挑战
- 操作系统代码管理原始硬件
  - CPU、内存、磁盘
  - 时间依赖行为, 非法行为, 硬件故障
- 操作系统代码必须是高效的, 低耗能, 安全可靠
  - 操作系统要及时地给应用提供合理资源
  - 操作系统出错, 就意味着机器出错
  - 操作系统必须比用户程序拥有更高的稳定性

# 掌握操作系统具有挑战性 (2)



## 学习操作系统需要具有系统思维

- 操作系统并不仅仅是琐碎的调度算法
  - 磁盘调度算法大多已被硬件实现
  - 进程调度是个比较小话题
- 并发性是操作系统的一小部分内容
  - 内核里不存在管程和哲学家问题
  - 内核中锁机制需要考虑应用和硬件
- 权衡资源
  - 时间与空间 – 性能的可预测性与公平性
- 软硬协同
  - 如何让中断、异常、上下文切换真正有效?
  - TLB 是如何工作的? 这对页表又意味着什么?

# 如何上好操作系统这门课？

《儒效篇》 -荀子

不闻不若闻之，闻之不若见之，见之不若知之，知之不若行之；学至于行之而止矣。

Thomas Edison

天才是 1% 的灵感加上 99% 的汗水

往届同学

“最有趣的三年级课程！” “最无聊的三年级课程！” “难过的三年级课程！”

# 第 1 讲：操作系统概述

## 第五节：操作系统实例

向勇、陈渝

清华大学计算机系

*xyong,yuchen@tsinghua.edu.cn*

2020 年 5 月 5 日

# Multics OS

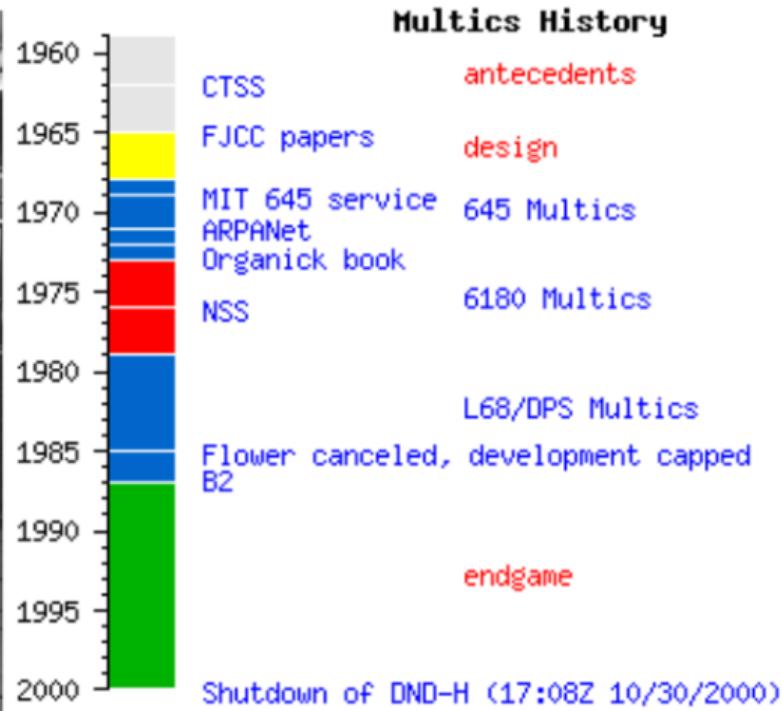
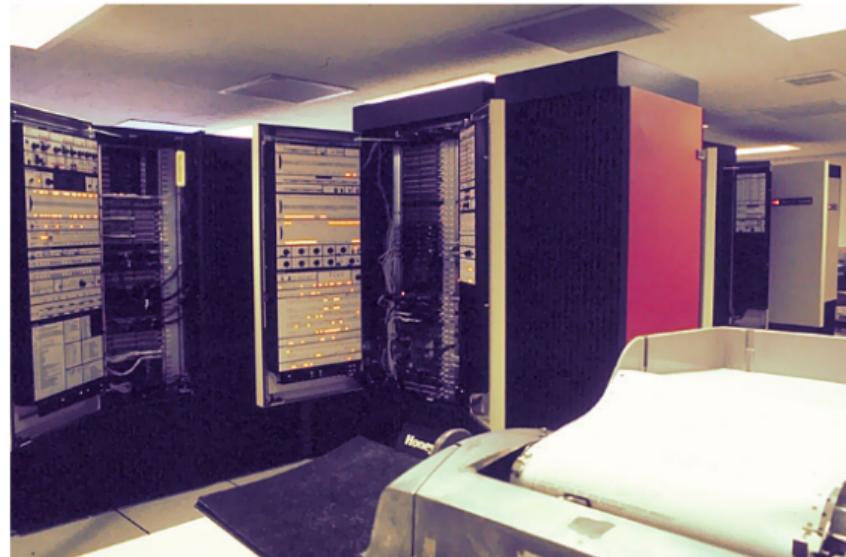


图: Multics OS—当今操作系统的鼻祖

# Multics OS

MULTICS (MULTplexed Information and Computing System)，是 1964 年由 MIT, 贝尔实验室及美国通用电气公司所共同参与研发的，是一套安装在大型主机上多人多任务的操作系统

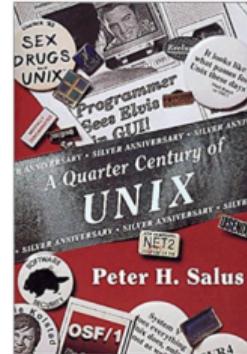
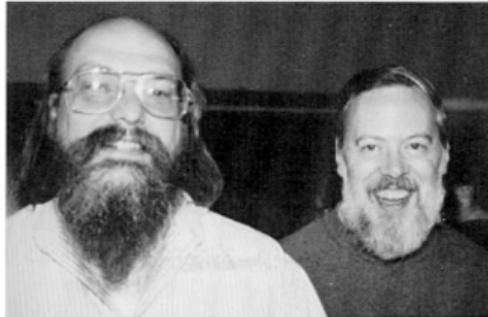
MULTICS 以 Compatible Time-Sharing System (CTSS) 做基础，建置在美国通用电力公司的大型机 GE-645。目的是连接 1000 部终端机，支持 300 位用户同时上线



PL/I    分层文件目录    分时调度    二维虚拟内存

图: Multics OS 简介

# UNIX 家族



iOS 6

图: UNIX 家族

# UNIX 历史

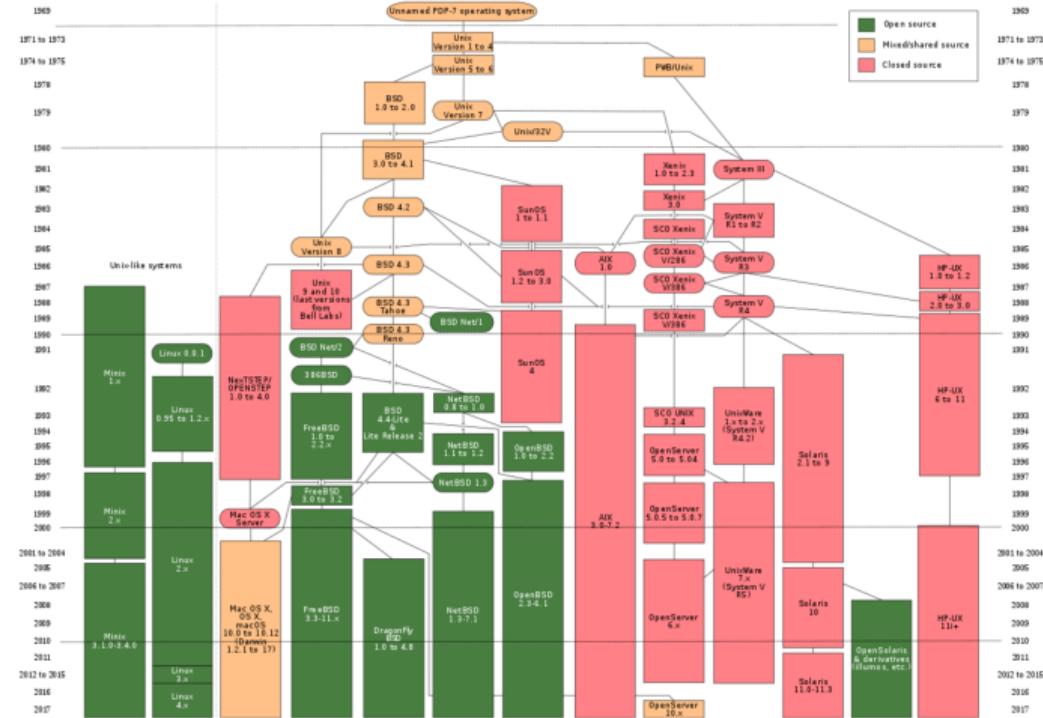


图: UNIX 历史

# Linux 家族



图: Linux 家族

# MacOS 家族

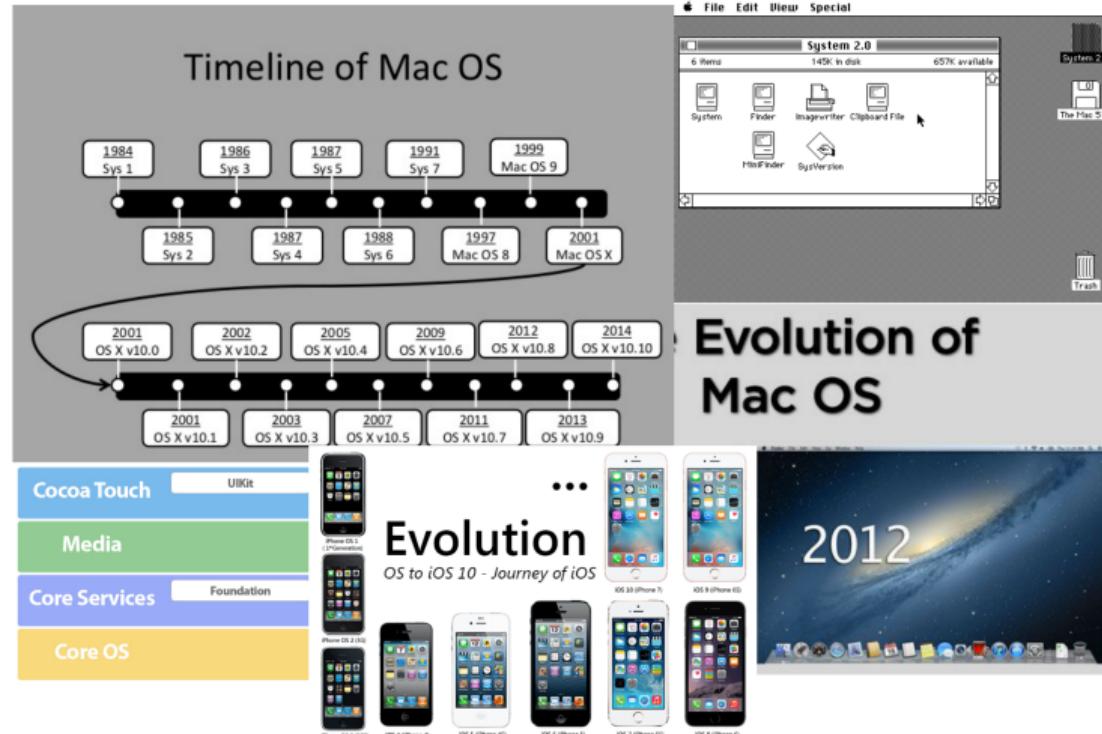


图: MacOS 家族

# MacOS 家族

RIG  
~ Mach 0.8, mid 1970s  
Accent  
~ Mach 0.9, circa 1979

Mach 1.0  
Project started in 1984  
USENIX paper in 1986

Mach 2.0

4.3BSD

NEXTSTEP 0.8  
October 1988

Mach 2.5 NeXT additions to Mach

NEXTSTEP 1.0  
September 1989

2.0  
September 1990

2.1  
March 1991

3.0  
September 1992

3.1  
May 1993

3.2  
October 1993

3.3  
February 1995

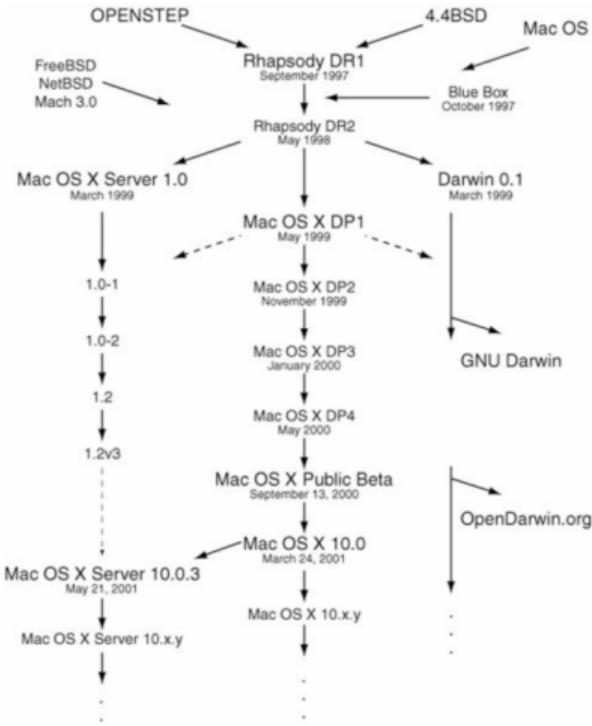
4.1  
December 1996

4.2  
January 1997

OpenStep Specification  
1994

OPENSTEP 4.0  
July 1996

Apple buys NeXT  
February 4, 1997



# Windows 家族

微软从DEC聘请 Dave Cutler 做Windows NT主要设计师

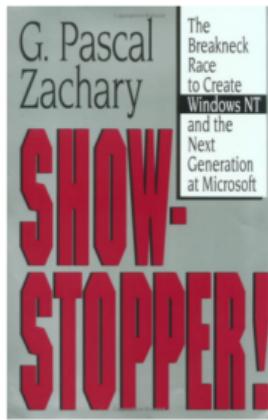


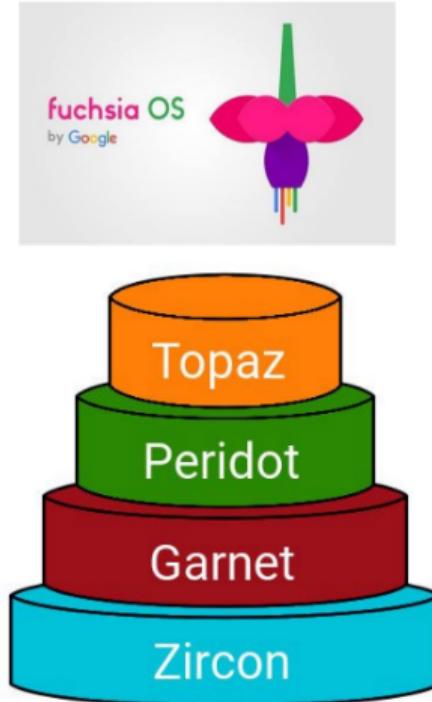
图: Windows 家族

# Android 操作系统



图: Android 操作系统-手机操作系统的代表之一

# Fuchsia 操作系统



**Topaz** : Runtime/ 前端框架和系统 UI/ 系统程序，提供 Flutter 支持，及其应用程序

**Peridot** : 系统框架和相关，处理 Fuchsia 的模块化应用程序设计，跨设备保存信息

**Garnet** : 平台相关基础库和服务，包括硬件的驱动程序（网络，图形等）和软件安装

**Zircon** 内核：处理硬件访问和软件之间的通信

图: Fuchsia 操作系统–AIoT 操作系统的代表之一

# 第 1 讲：操作系统概述

## 第六节：操作系统历史

向勇、陈渝

清华大学计算机系

*xyong,yuchen@tsinghua.edu.cn*

2020 年 5 月 5 日

1

## 第六节：操作系统历史

- 单用户系统
- 批处理系统
- 多道程序系统
- 分时系统
- 个人计算机
- 分布式计算

# 单用户系统

## 单用户系统 (1945-1955)

- 手动连线/纸带传输进行程序输入
- 机器成本远大于人力成本
- 操作系统 = 装载器 (loader) + 程序库 (libraries)
- 问题：昂贵组件的低利用率

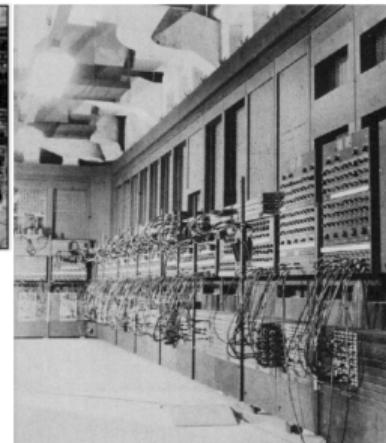
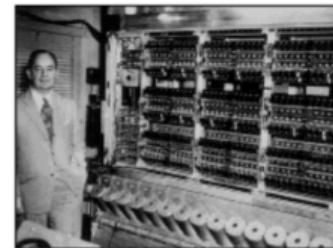
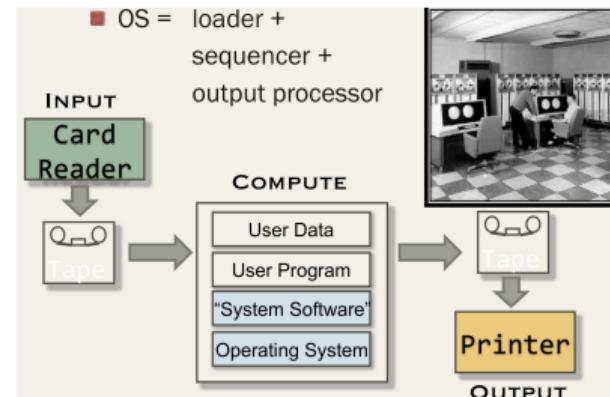


图: 单用户系统

# 批处理系统

## 批处理系统 (1955-1965)

- 磁带/磁盘传输进行程序输入
- 机器成本大于人力成本
- 操作系统 = 装载器 (loader) + 程序控制器 (sequencer) + 输出处理器 (output processor)
- 问题：相比以前利用率提高



图：批处理系统

# 批处理系统

## 批处理系统 (1955-1965)

- 磁带/磁盘传输进行程序输入
- 机器成本大于人力成本
- 操作系统 = 装载器 (loader)+ 程序控制器 (sequencer)+ 输出处理器 (output processor)
- 演变：相比以前利用率提高

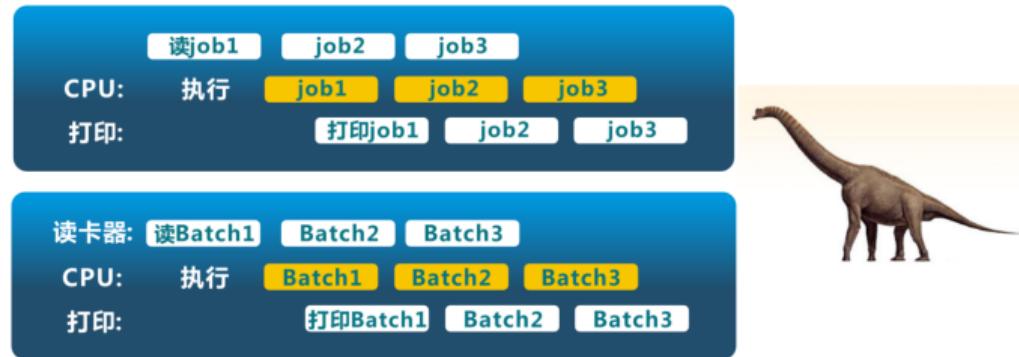


图: 批处理流程

# 多道程序系统

## 多道程序系统 (1955-1980)

- 多个程序驻留内存中
- 多个程序轮流使用 CPU
- 操作系统 = 装载器 + 程序调度 + 内存管理 + 输出管理
- 演变：相比以前利用率提高

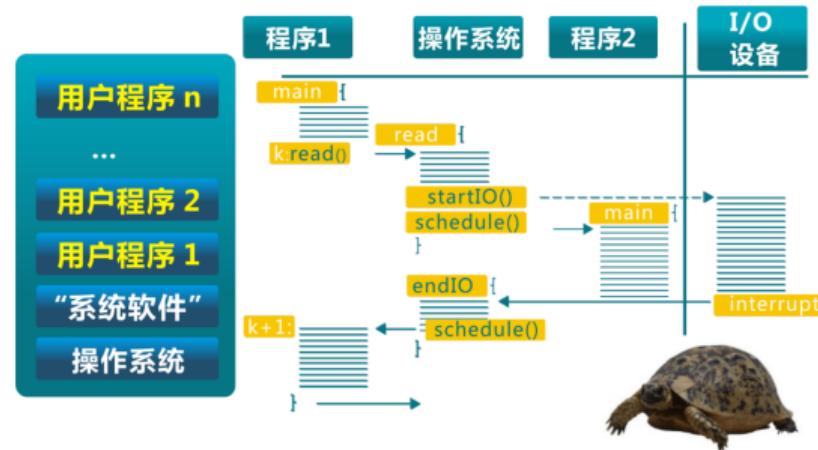
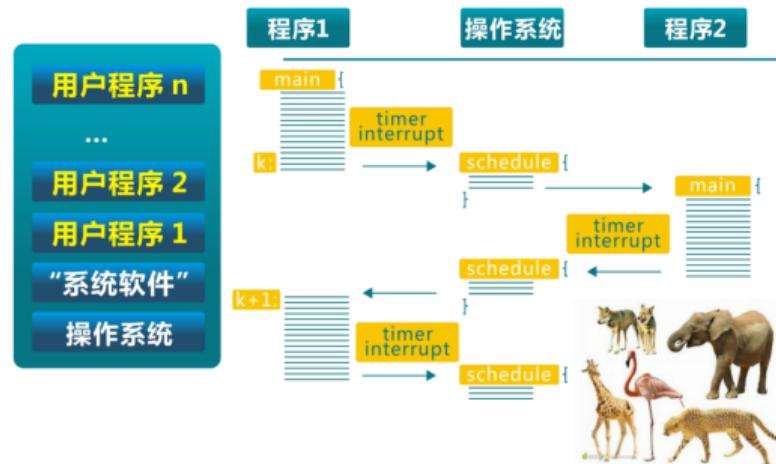


图: 多道程序处理流程

# 分时系统

## 分时系统 (1970- )

- 多个程序驻留内存中
- 多个程序分时使用 CPU
- 操作系统 = 装载器 + 程序调度 + 内存管理 + 中断处理 + ...
- 演变：相比以前利用率提高



图：分时处理流程

# 个人电脑

## 个人电脑 (1981- )

- 单用户
- CPU 利用率已不再是最重要的关注点
- 重点是用户界面和多媒体功能
- 操作系统 = 装载器 + 程序调度 + 内存管理 + 中断处理 + ...
- 演变：走向大众，老的服务和功能不存在，越来越多的安全问题



图: 个人电脑

# 分布式系统

## 分布式系统 (1990- )

- 分布式多用户
- 分布式系统利用率是关注点
- 重点是网络/存储/计算的效率
- 操作系统 = 分布式 (装载器 + 程序/OS 调度 + 内存管理)
- 演变：走向大众，走向网络，新的挑战 (不可靠/不确定)

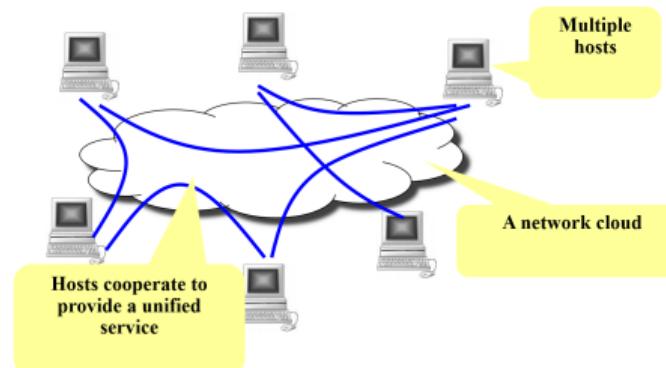


图: 分布式系统

# AIoT 系统

## AIoT 系统 (2000- )

- 分布式多设备
- 分布式系统利用率/可用性是关注点
- 重点是网络/存储/计算的效率
- 操作系统 = 分布式 (程序/OS 调度 + 内存管理 + 安全/更新)
- 演变：走向设备，走向网络，新的挑战 (不可靠/大数据)



图: AIoT 系统

# 第 1 讲：操作系统概述

## 第七节：操作系统结构

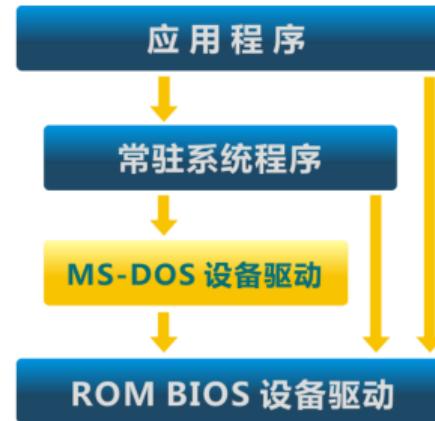
向勇、陈渝

清华大学计算机系

*xyong,yuchen@tsinghua.edu.cn*

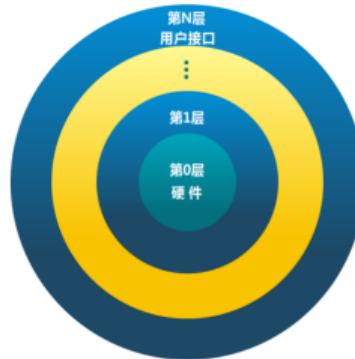
2020 年 5 月 5 日

# 简单结构



- MS-DOS：在最小的空间，设计用于提供大部分功能（1981–1994）
  - 没有拆分为模块
  - 主要用汇编编写
  - 没有安全保护

# 单体分层结构



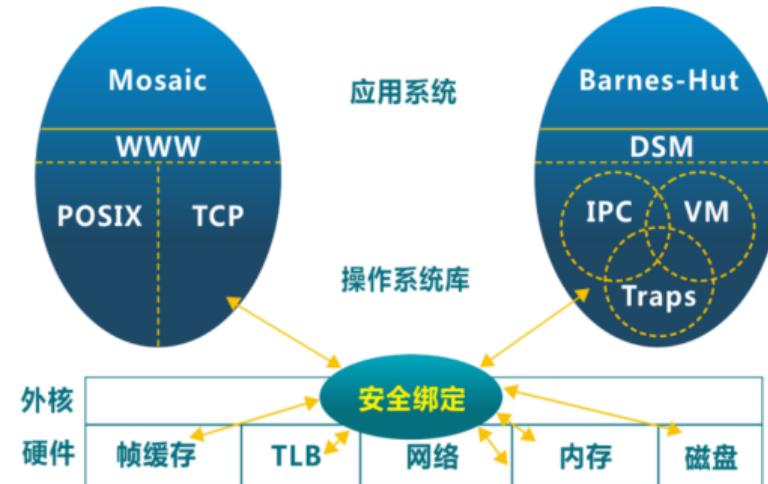
- 将单体操作系统 (Monolithic OS) 划分为多层 (levels)
  - 每层建立在低层之上
  - 最底层 (layer 0), 是硬件驱动
  - 最高层 (layer N) 是用户界面
- 每一层仅使用更低一层的功能和服务。

# 微内核结构



- 尽可能把内核功能移到用户空间
- 用户模块间的通信使用消息传递
- 好处: 灵活/安全...
- 缺点: 性能

# 外核结构 Exokernel



- 让内核分配机器的物理资源给多个应用程序，并让每个程序决定如何处理这些资源
- 程序能链接到操作系统库 (libOS) 实现了操作系统抽象
- 保护与控制分离

# 虚拟机结构 VMM



- 虚拟机管理器将单独的机器接口转换成很多的虚拟机, 每个虚拟机都是一个原始计算机系统的有效副本, 并能完成所有的处理器指令。

# 第 1 讲：操作系统概述

## 第八节：OS 实验概述

向勇、陈渝

清华大学计算机系

*xyong,yuchen@tsinghua.edu.cn*

2020 年 5 月 5 日

# OS 实验概述

## ● 设计思路

- 设计 ucore/rcore，覆盖操作系统的关键点，内容如下：
  - 外设：I/O 管理/中断管理
  - 内存：虚存管理/页表/缺页处理/页替换算法
  - CPU：进程管理/调度器算法
  - 并发：信号量实现和同步互斥应用
  - 存储：文件系统 + 磁盘驱动
- 完整代码量控制在 10000 行以内
- 提供实验讲义和源码分析文档

# OS 实验内容

## OS 实验内容

- ① OS 启动/中断/异常
- ② 物理内存管理
- ③ 虚拟内存管理
- ④ 内核模式线程管理
- ⑤ 用户模式进程管理
- ⑥ 处理器调度
- ⑦ 多处理与同步互斥
- ⑧ 文件系统



图: OS 实验框架

### Lab1: Bootloader/Interrupt/Debug

启动操作系统的 bootloader，了解操作系统启动前的状态和要做的事，了解运行操作系统的硬件支持，操作系统如何加载到内存中，理解两类中断-“外设中断”，“异常”等。

- 编译运行直接与硬件交互的系统程序
- 启动 bootloader 的过程
- 实现中断处理机制
- 输出字符的方法
- 调试系统程序的方法

### Lab2: 物理内存管理

理解分页模式，了解操作系统如何管理连续空间的物理内存。

- 理解内存地址的转换和保护
- 实现页表的建立和使用方法
- 实现物理内存的管理方法
- 了解常用的减少碎片的方法

## Lab3：虚拟内存管理

了解页表机制和换出（swap）机制，以及中断-“故障中断”、缺页故障处理等，基于页的内存替换算法。

- 理解换页的软硬件协同机制
- 实现虚拟内存的 Page Fault 异常处理
- 实现页替换算法

## Lab4: 内核模式线程管理

了解如果利用 CPU 来高效地完成各种工作的设计与实现基础，如何创建相对与用户进程更加简单的内核态线程，如何对内核线程进行动态管理等。

- 建立内核线程的关键信息
- 实现内核线程的管理方法

### Lab5: 用户模式进程管理

了解用户态进程创建、执行、切换和结束的动态管理过程，了解在用户态通过系统调用得到内核态的内核服务的过程。

- 建立用户进程的关键信息
- 实现用户进程管理
- 分析进程和内存管理的关系
- 实现系统调用的处理过程

## Lab6：调度

理解操作系统的调度过程和调度算法。

- 熟悉系统调度器框架，以及内置的 Round-Robin 调度算法
- 基于调度器框架实现一个调度器算法

### Lab7：同步互斥

了解进程间如何进行信息交换和共享，并了解同步互斥的具体实现以及对系统性能的影响，研究死锁产生的原因，以及如何避免死锁。

- 熟悉同步互斥的实现机制
- 理解基本的 spinlock、semaphore、condition variable 的实现
- 实现基于各种同步机制的同步问题。

### Lab8：文件系统

了解文件系统的具体实现，与进程管理等的关系，了解缓存对操作系统 IO 访问的性能改进，了解虚拟文件系统（VFS）、buffer cache 和 disk driver 之间的关系。

- 掌握基本的文件系统系统调用的实现方法
- 了解一个基于 inode 方式的 SFS 文件系统的设计与实现
- 了解文件系统抽象层-VFS 的设计与实现

# OS 实验内容

labX

## LabX：大实验

前提：已经完成基本实验

尝试完成一些有一定挑战性且有趣的 OS 实验。

- 重新设计 zircon 操作系统
- 在一个 OS(如 Linux) 实现一个 Hypervisor
- OS 直接支持运行被隔离的应用程序
- 支持动态更新的 OS
- 驱动程序运行在用户态的 OS
- 支持标签化 CPU 的 OS
- 一个可验证正确性的 OS
- 运行在抽象计算机上可动态调试的 OS

# OS 实验内容

labX

选题方向	大实验题目
RISC-V	ucore on RISC-V
RISC-V	简易版 rcore 开发与教学文档编写 && rcore plus 开发
RISC-V	FPGA 上运行 RISC-V rCore 构建路由器
x86_64	对标 Biscuit OS 真实应用真实网卡及性能测试
x86_64	rCore 内核可加载模块和动态链接库
MIPS	第三届全国大学生系统能力培养大赛
Arm	Python (and more) on rCore on RPi
GUI	GUI
GUI	适配 mini GUI

# OS 实验内容

labX

选题方向	大实验题目
驱动	IO 复用
rust	Audio support for rCore
内核语言	编译原理/操作系统综合实验
错误分析	在 ucore 获得稳定触发竞争条件的漏洞样本
行为分析	Program Analysis via Memory Access Patterns
微内核	调研 Fuchsia 的微内核，尝试 rcore 微内核的修改
内核可加载模块	rethink 用户/内核态
模拟器	操作系统中常用算法的性能分析及优化
教学实验设计	对简易版 rcore 的进一步维护和更新