

操作系统的前世今生

陈海波 / 夏虞斌

上海交通大学并行与分布式系统研究所

<https://ipads.se.sjtu.edu.cn>

版权声明

- 本内容版权归**上海交通大学并行与分布式系统研究所**所有
- 使用者可以将全部或部分本内容免费用于非商业用途
- 使用者在使用全部或部分本内容时请注明来源
 - 内容来自：上海交通大学并行与分布式系统研究所+材料名字
- 对于不遵守此声明或者其他违法使用本内容者，将依法保留追究权
- 本内容的发布采用 Creative Commons Attribution 4.0 License
 - 完整文本：<https://creativecommons.org/licenses/by/4.0/legalcode>

大纲

- 什么是操作系统
- 操作系统的历史
- 操作系统当前的挑战
- 为什么要学习操作系统

► 操作系统的定义

什么是操作系统？

- 你觉得以下哪些属于操作系统？

- A. Windows 10所包含的所有软件
- B. Linux内核以及所有设备的驱动
- C. 在Macbook上下载安装的第三方NTFS文件系统
- D. 华为Mate 30出厂时所有的软件
- E. 大疆无人机出厂时所有的软件
- F. 火星车上运行的软件

- 你觉得应当如何定义操作系统？

操作系统是在硬件和应用之间的软件层

应用

操作系统

硬件

操作系统和应用的关系：

- 应用功能越来越多
- 操作系统沉淀越来越多功能、内涵与外延不断扩大

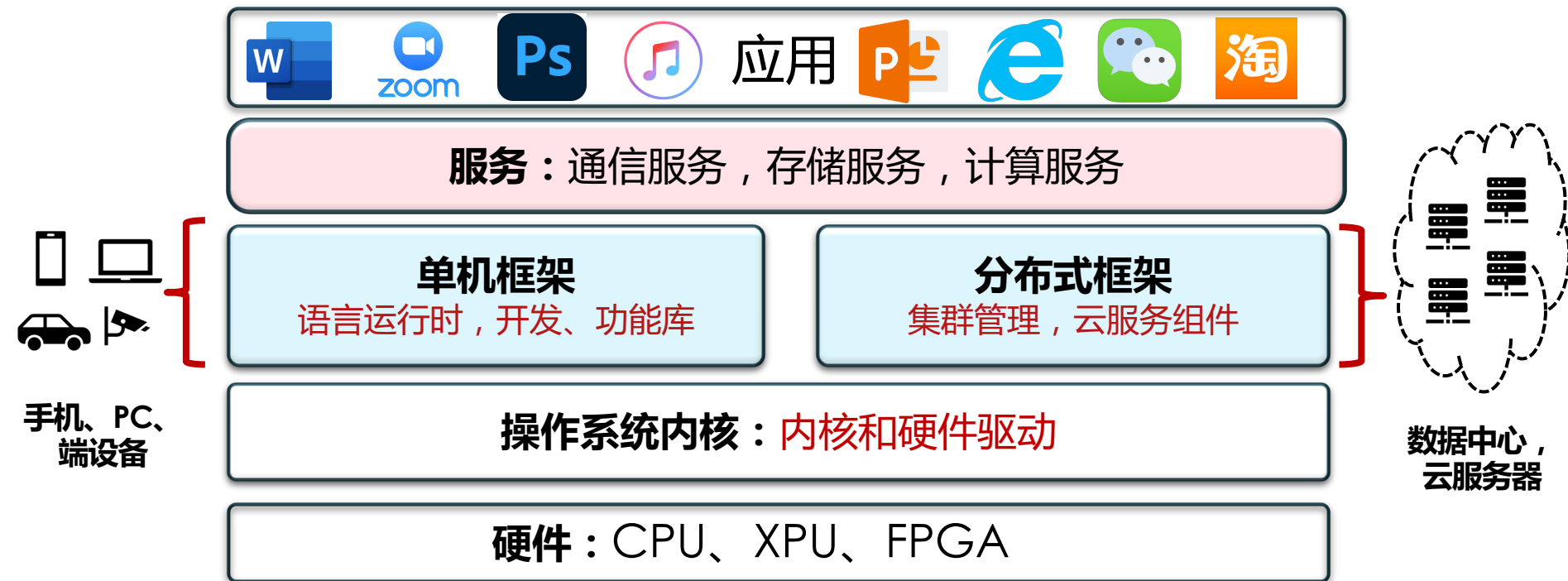
操作系统和硬件的关系：

- 意识 VS. 身体

"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(第2版)]

操作系统：对下管理硬件，对上构筑应用生态



从 Hello World 说起

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

运行hello时，操作系统的作用？

```
bash$ gcc hello.c -o hello
```

```
# 运行一个hello world程序
```

```
bash$ ./hello
```

```
Hello World!
```

```
# 同时启动两个hello world程序
```

```
bash$ ./hello & ./hello
```

```
[1] 144
```

```
Hello World!
```

```
Hello World!
```

```
[1]+ Done      ./hello
```

操作系统考虑的一些问题

- hello 这个可执行文件存储在什么位置？是如何存储的？
- hello 这个可执行文件是如何加载到 CPU 中运行？
- hello 这个可执行文件是如何将"Hello World!"这行字输出到屏幕？
- 两个hello 程序同时运行的过程中如何在一个 CPU 中运行？
- ...

操作系统需要：1、服务应用；2、管理应用

操作系统为应用提供的一些服务

- **为应用提供计算资源的抽象**

- CPU：进程/线程，数量不受物理CPU的限制
- 内存：虚拟内存，大小不受物理内存的限制
- I/O设备：将各种设备统一抽象为文件，提供统一接口

- **为应用提供线程间的同步**

- 应用可以实现自己的同步原语（如spinlock）
- 操作系统提供了更高效的同步原语（与线程切换配合，如pthread_mutex）

- **为应用提供进程间的通信**

- 应用可以利用网络进行进程间通信（如loopback设备）
- 操作系统提供了更高效的本地通信机制（具有更丰富的语义，如pipe）
 - 例：Shell Pipe

操作系统对应用的管理

- **生命周期的管理**
 - 应用的加载、迁移、销毁等操作
- **计算资源的分配**
 - CPU：线程的调度机制
 - 内存：物理内存的分配
 - I/O设备：设备的复用与分配
- **安全与隔离**
 - 应用程序内部：访问控制机制
 - 应用程序之间：隔离机制，包括错误隔离和性能隔离

问题

- **问题一**：如果一台机器有且只有一个应用程序，开机后自动运行且不会退出，是否还需要操作系统？
- **问题二**：如果一个应用希望自己完全控制硬件而不是使用操作系统提供的抽象，是否还需要操作系统？

操作系统 = 管理 + 服务

- **管理和服务的目标有可能存在冲突**
 - 服务的目标：单个应用的运行效率最大化
 - 管理的目标：系统的资源整体利用率最大化
 - 例：单纯强调公平性的调度策略往往资源利用率低
 - 如细粒度的round-robin导致大量的上下文切换

操作系统的定义

- **操作系统的核心功能：**
 - 将有限的、离散的资源，高效地抽象为无限的、连续的资源
- **从软件角度的定义：**
 - 硬件资源虚拟化+管理功能可编程
- **从结构角度的定义：**
 - 操作系统内核+系统框架

操作系统 VS. "The Matrix/黑客帝国"



The Matrix	操作系统
民众	进程
Oracle/先知	进程调度器
列车员	进程间通信
Key Maker	密钥管理程序
特工/Smith	杀毒程序（后来演变为病毒）
电话	系统调用
建筑师	内核监控程序

应用与操作系统的交互：系统调用

- 什么是系统调用？
 - 应用调用操作系统的机制，实现应用不能实现的功能
- 例如：`printf()` -> `write()`->`sys_write()`
 - `write(1, "Hello World!\n", 13)`
- 使应用调用操作系统的功能就像普通函数调用一样

Hello运行中的系统调用（strace）

```
/* 运行hello程序 */
```

```
execve("./hello", ["./hello"], 0x7ffed5a79e80 /* 64 vars */) = 0
```

```
...
```

```
/*将`Hello World!\n`写到标准输出中，在这里，1代表标准输出，  
13代表一共写了13个字符。*/
```

```
write(1, "Hello World!\n", 13Hello World!) = 13
```

```
/* 执行结束后，hello程序退出*/
```

```
exit_group(0)
```

系统调用过程

应用
程序

```
#include <stdio.h>
int main() {
    printf("Hello World!\n");
    return 0;
}
```

用户
地址
空间

libc

```
write(1, "Hello World!\n" ,13)
{
    ...
    /*参数处理 */
    push $__NR_write /* 第一个参数: 系统调用号 */
    push $1 /* 第二个参数: file descriptor */
    push "Hello World!\n"
    push $13
    syscall (不同平台对于的指令不同, 例如ARM中的svc, x86中的sysenter, RISC-V中的ecall)
    ...
}
```

内核
地址
空间

下陷
处理

```
sys_syscall:
...
call syscall_table[__NR_write]
...
```

系统
调用
处理

```
sys_write {
...
return error_no;
}
```

操作系统的功能：管理

- 避免一个流氓应用独占所有资源
- 方法-1：每10ms发生一个时钟中断（时间片）
 - 调度器决定下一个要运行的任务
- 方法-2：可通过信号等打断当前任务执行
 - 如：kill -9 1951

rogue.c

```
int main () {  
    while (1);  
}
```

操作系统的功能：管理

- 如何卡死一个OS？

- 例：rogue-1.c 可以fork出无数的进程

rogue-1.c

```
int main () {  
    while (1)  
        fork();  
}
```

- 如何解决这个问题？

- 资源配额：cgroup/Linux
- 虚拟化：虚拟机
- 万能方法：重启机器
- 制度约束：AppStore的程序预准入机制

这个是很现实的...

刚写完代码，就被开除了

只看楼主

收藏

回复



西伯利亚蓝眼睛



淼淼



```
/**
 * 获取下一天的日期
 * @return
 */
public static Date getNextDay() {
    try {
        Thread.sleep(24*60*60*1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return new Date();
}
```

是不是因为我没写注释

图片来自：<https://www.bilibili.com/read/cv1844639/>

▶ 操作系统的历史

批处理操作系统：GM-NAA I/O



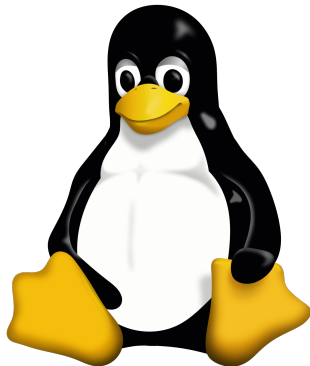
- Robert L. Patrick和Owen Mock于1956年建设
 - 运行在IBM 704上
 - 主要功能：批处理运行任务

通用操作系统：OS/360



- **IBM System/360 OS , 1964**
 - 首个通用操作系统，首次将操作系统与计算机分离
 - 架构师：Gene Amdahl (Amdahl's Law)
 - 项目经理：Fred Brooks (《人月神话》，1999年图灵奖得主)

分时与多任务：Multics/Unix/Linux



Multics: Fernando Corbató
(1990年图灵奖) MIT/GE, 1964
: 分时，文件系统，动态链接等



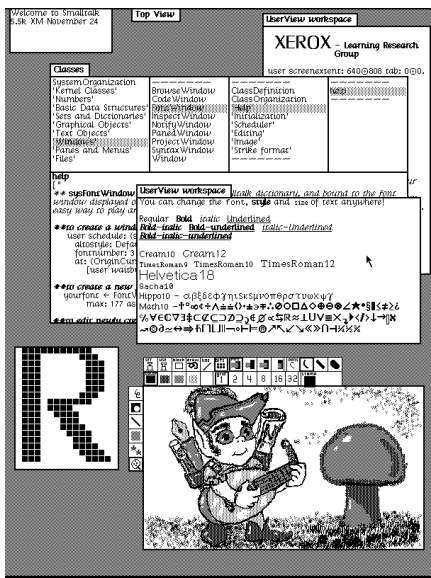
Unix: Ken Thompson, Dennis
Ritchie (1983年图灵奖), 1969
Shell，层次化文件系统



Linux: Linus Torvalds, 1991
最流行的开源操作系统



图形界面：Xerox Alto/MacOS/Windows



Xerox Alto (1973) : 第一个图形化操作系统，首次使用鼠标 (Chuck Thacker , 2009年图灵奖)



Mac OS (Apple LISA, 1983): 1979年乔布斯访问Xerox PARC，意识到GUI的重要性，买下了GUI进行研究



Windows 1.0 (1985) : 基于图形界面的操作系统

► 操作系统当前的挑战

AIoT: AI + IoT + 5G + Cloud + ...

5G



10-100X ↑吞吐量
10-100X ↓时延

AI 硬件



100X 计算能力

IoT



100X 设备数

构筑万物互联的智能世界！

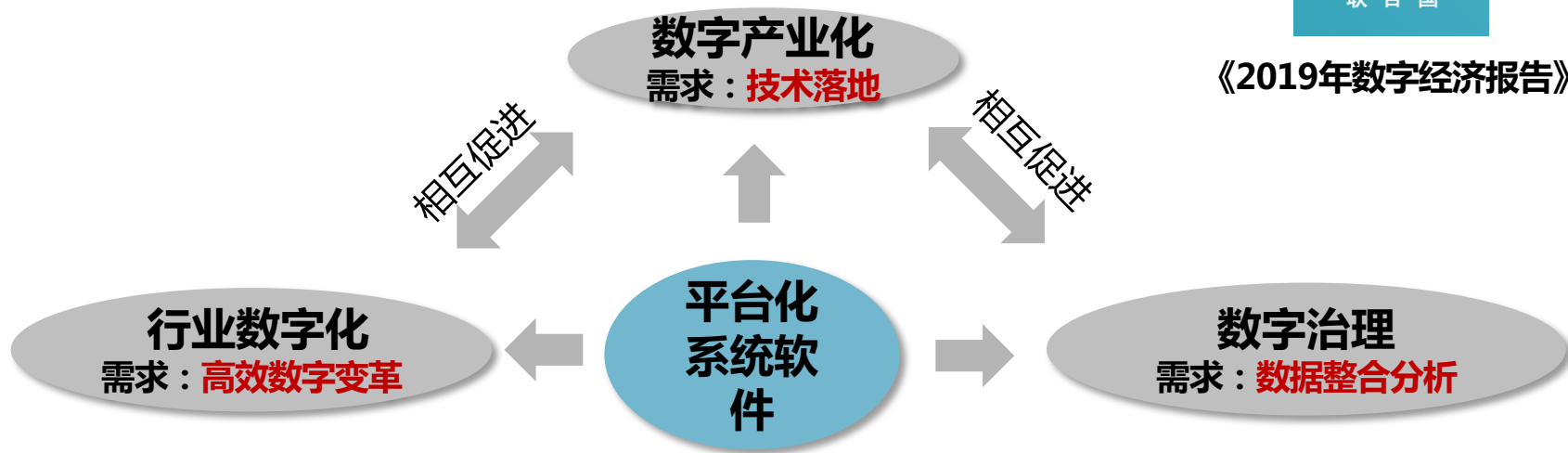
“数字化转型，以操作系统为核心的平台化系统软件是关键”

数字经济转型已成共识，产业数字化转型正加速。

操作系统作为创新平台为代码和内容制作者开发应用程序和软件创造环境。



《2019年数字经济报告》



系统软件是支持数字经济平台化的基础和关键

趋势-1：从封闭到开放，再到封闭

- 例子1：2018年10月起 Google正式对欧盟区域的Android进行收费，初步高达40美元每设备

CAN'T WAIT TO HAVE BING INSTALLED ON MY PHONE —

Google to charge Android OEMs as much as \$40 per phone in EU

After the EU ruling, OEMs can unbundle Google's Android apps, but it will cost them.

- 例子2：2018年10月 IBM 340亿美元收购 RedHat，构筑其云计算竞争力

IT'S OFFICIAL: IBM is acquiring software company Red Hat for \$34 billion

- 例子3：2016年起谷歌投入600+人力，数十亿美元，研发面向智能端设备的自研OS Fuchsia

**操作系统并不是免费的午餐，
而是构筑与控制生态的黑土地**

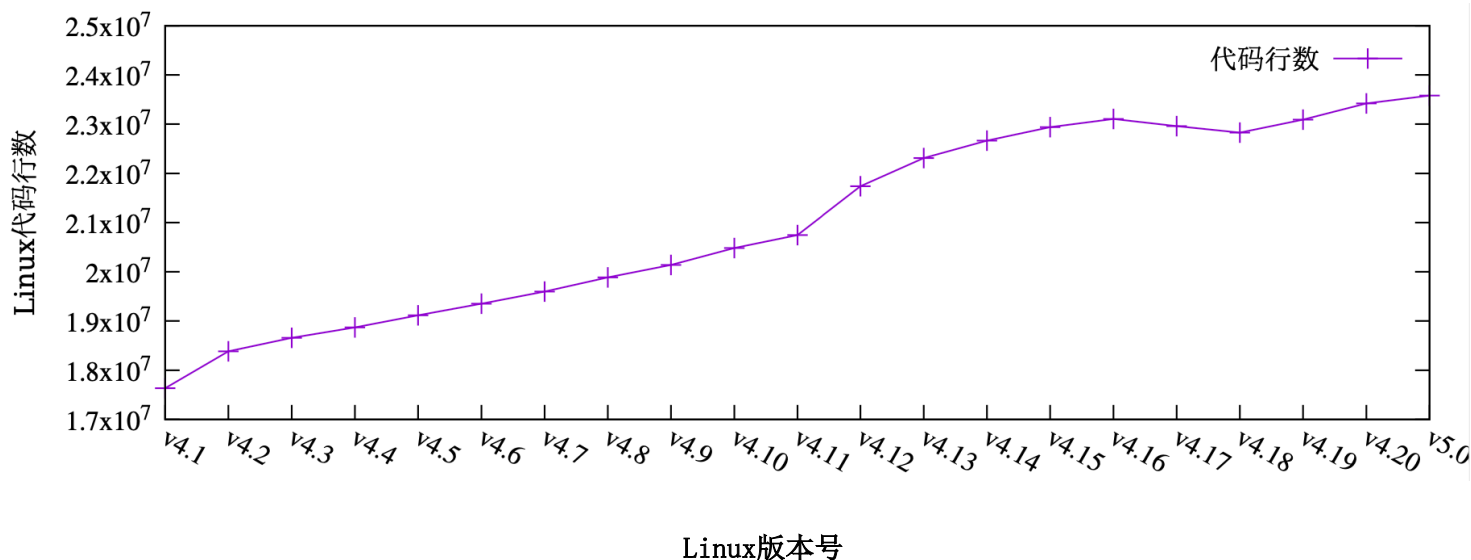
趋势-2：从专用到通用，再到专用

- 专用 → 通用 → 专用（领域化）
 - "I think there is a world market for maybe five computers."
 - Thomas Watson, president of IBM, 1943
 - 计算机系统正在从传统的分层解耦走向云+端的垂直整合

操作系统的类型和数量均会大幅增加

趋势-3：从简单到复杂，到更复杂

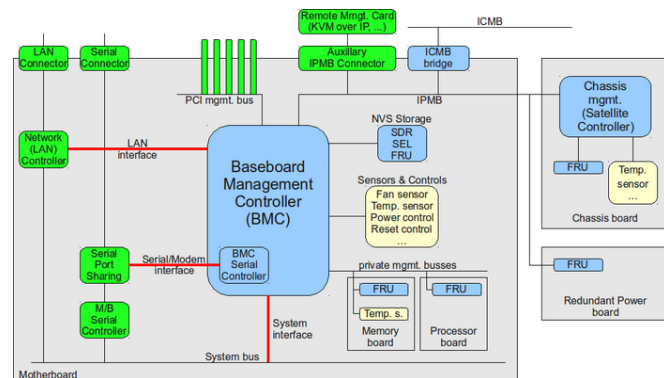
- Linux代码规模已超过3000万行
- 仍然以每年200万行的数量在增加/更新



一个芯片上的OS不是单一OS，而是一组OS

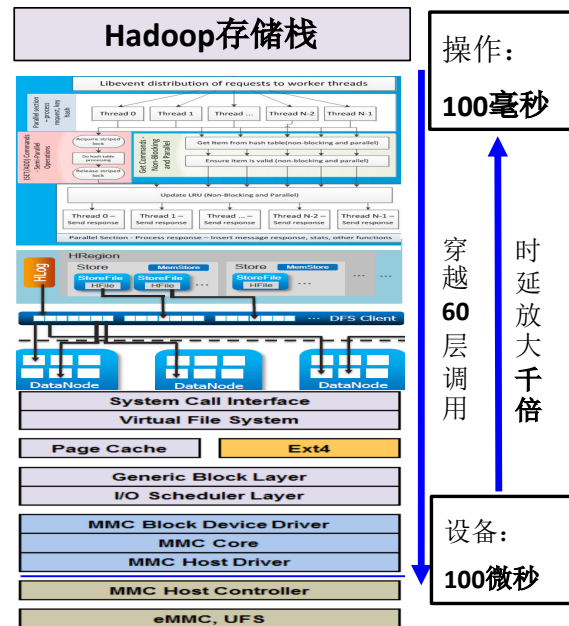
- 复杂硬件：芯片上的数据中心
 - 异构计算
 - ARM: 大小核(big.medium.little)设计, 加速器
 - 新型可编程设备 (智能网卡, 智能存储 (SSD), AI加速器)
 - 趋势: 分布式的, 可编程的异构设备

- 含义: 一组OS运行在一个计算机/芯片上



计算机硬件在新应用需求下迅猛发展

- **计算**：从通用计算走向领域计算，各种xPU不断繁荣
 - GPU、TPU、NPU、IPU等支撑人工智能算力需求
- **存储**：智能存储，存算一体，非易失内存（SCM），内存与持久存储走向融合
- **数据中心网络**：Infiniband等网络走向纳秒级时延
- **广域网络**：5G大连接、**低时延**、高可靠使能新型高吞吐、低时延广域计算



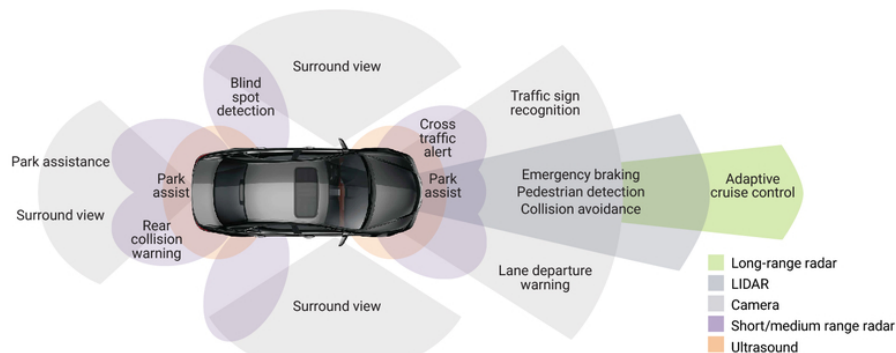
新型硬件发展需要新的
操作系统抽象与设计来充分释放算力

操作系统：传统开源模式在特定场景开始走向窘境

- **例：Linux代码已超过3000万行，仍以每年200万行在增加/更新**
 - 适用于云、IT等通用系统
 - Linux的主要贡献者来自IT大厂
- **观点：Linux历史包袱过重，对IT外的领域场景协同演进困难**
 - 例1：Android涉及Linux代码不超过300万行
 - 例2：苹果自研iOS内核: 200万行代码左右，常年较为稳定
 - 结果：很多场景追随Linux的演进代价太大，生命周期管理困难，且不易构成竞争力
- **业界：开源License不断变得受限，更难自主可控**

智能驾驶、智能家庭等新场景需要新的操作系统

- 智能驾驶汽车包含完整的传感器、网络互联、计算单元，是新一代移动数据中心的主要载体



单一功能
(MCU、3~5K
LoC)

富功能
(AI芯片、通用处
理器)

融合功能
(融合计算、网
络、存储等MDC)

家居环境越来越智能化

- 无缝连接/协作支撑良好用户体验



为什么学习操作系统

为什么学习操作系统？

- **操作系统是一个成熟的领域**
 - Windows曾经一统天下数十年
 - 让用户改变操作系统是很困难的
 - 我们是否需要新的操作系统？
- **"新的"操作系统不断出现**
 - Linux、Mac OS、Android、iOS、ROS...
 - 大疆用什么操作系统？谷歌数据中心用什么操作系统？
火星车用什么操作系统？...

为什么学习操作系统？

- 操作系统是系统领域的基石
- 系统领域有大量的公司
 - 微软、谷歌、IBM、EMC、VMware、华为、阿里...
 - 谷歌的核心技术
 - 集群、GFS、MapReduce、BigTable
 - 都是系统领域的优秀工作
- 学好操作系统，for fun and profit
 - 优秀的公司，优秀的大学，更多的机会

图灵奖与操作系统的演变



Maurice Wilkes
1967年图灵奖

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



Frederick Brooks
1999年图灵奖

IBM System/360, 1964



Fernando J. Corbató
1990年图灵奖

CTSS, 1961 & Multics, 1969
分时操作系统



Ken Thompson & Dennis Ritchie
1983年图灵奖

Unix, 1971
多任务多用户操作系统



Barbara Liskov
2008年图灵奖

Venus, 1972
小型低成本交互式分时操作系统

2019
分布式操作系统



这门课程希望给大家的能力

- **成为一个更高效的程序员**
 - 更快找到并消灭bug的能力
 - 理解并调试程序性能的能力
- **理解复杂系统设计与实现的能力**
- **构建一个操作系统的能力**

课程的特点：抽象与具体

- 许多课程强调抽象
 - 如：抽象数据类型、渐进分析等
- 这些抽象通常不可避免的带来限制
 - 尤其是当bug存在的时候
 - 需要理解底层的实现细节，才能突破限制

“What I cannot create, I do not understand”

—— Richard Fynman (CIT), referred to by 陆奇

为什么操作系统比较难/有意思？

- **深入事情本质：直接管理硬件细节**

- 好处：实现资源的高效利用，从根本上解决问题，做一个高效程序员（降维）
- 挑战：需要理解与处理硬件细节，硬件甚至可能出错
- “把复杂留给自己、把简单留给用户”
 - 对比：用户态写一个Hello World和在操作系统内核中输出一个Hello World (Lab1)

- **锻炼系统架构能力**

- 将复杂问题进行抽象与化简
- 最好的体现了M.A.L.H原则的使用
- 计算机科学中**30%**的原则是从操作系统中来的 (13/41, greatprinciples.org)

- **各种问题的交互与相互影响**

- `fd = open (); ... ; fork();`

操作系统的机遇和挑战

- **新的硬件层出不穷**

- 硬件种类越来越多：如自动驾驶、无人机、各种IoT设备等
- 硬件互联越来越复杂：如AirDrop、摄像头和驾驶系统等

- **应用需求日新月异**

- 对时延的要求越来越苛刻，如自动驾驶系统
- 对资源利用率要求越来越高，如数据中心混部

- **新的硬件和应用呼唤新的抽象和管理——新的操作系统**

课程信息

教师

陈海波 (haibochen@sjtu.edu.cn)

夏虞斌 (xiayubin@sjtu.edu.cn)

糜泽羽 (yzmizeyu@sjtu.edu.cn)

助教

杨健邦 (jianbangyang@sjtu.edu.cn)

黄政 (huangzheng@sjtu.edu.cn)

徐天强 (2286455782@qq.com)

胡雨奇 (yuki.h@sjtu.edu.cn)



<http://ipads.se.sjtu.edu.cn/courses/os>

课程评分

- **40% : 期末考试**
- **45% : Lab 1-4 (在arm64平台上实现一个小的微内核架构OS)**
 - Lab1 : PC Booting
 - Lab2 : Virtual Memory
 - Lab3 : User Process
 - Lab4 : Preemptive Multitasking
 - Lab5(可选) : 文件系统
- **10% : 平时作业**
- **5% : 课程网站讨论与课程表现**

欢迎进入操作系统的世界！

