# 1. 操作系统演化

计算机诞生时的操作员

批处理操作系统：GM-NAA I/O

通用操作系统：OS/360

- 首个通用操作系统

- 《人月神话》Fred Brooks

当时的硬件不允许：没有ISA，没有形成软硬件分离的机制。为了实现软硬件分离：必须要有ISA概念。划一条线：ISA，下面硬件遵守ISA。

why 360：360度全覆盖。

不用管硬件了，可以做很多事情：

Multics：1964分时，文件系统，动态链接

（简化版Multics）Unix：层次化文件系统 1969shell（一帮人在构思一个巨复杂的操作系统：把它做减法才产生的，multics做不出来。。。）bell实验室因此开始收license fee（莱昂斯源码分析）

Linux（unix不让用）：1991最流行的开源操作系统

图形界面：Xerox（第一个图形化操作系统，首次使用鼠标）、macOS、Windows

软件导致硬件的问题被发现。

MacOS 1979年乔布斯访问Xerox：意识到GUI重要性

Windows 1.0：1985

modern OS

鸿蒙，Euler，矿鸿，Apple：RealityOS

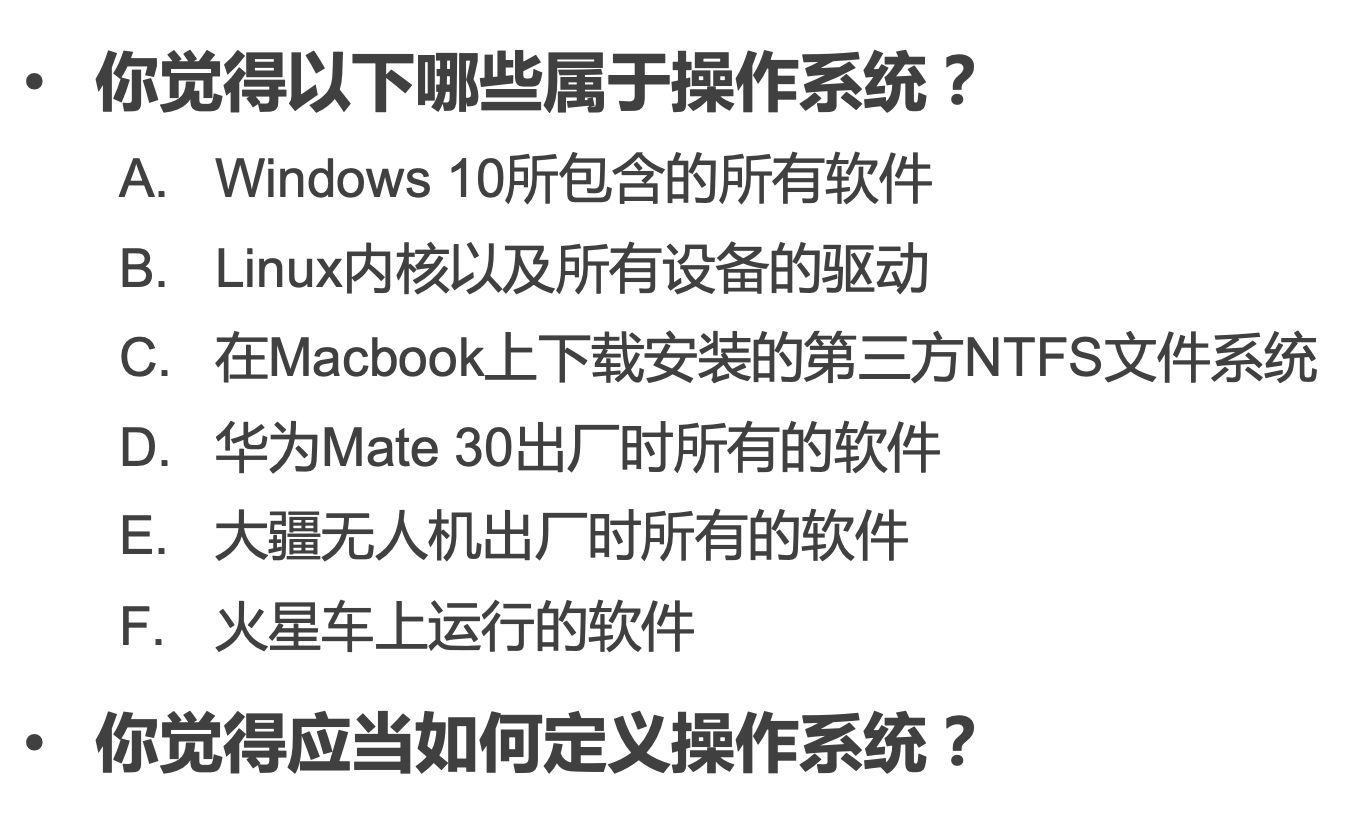
Q：CPU里有OS吗？

A：有。Intel内嵌OS：management （微内核的minix）做非常critical的工作（安全相关）。！！minix有bug。

Q：SSD里有OS吗？

A：有。泛化的OS。固件firmware。做地址转换（把相同的地址翻译到不同的物理地址，防止写一块区域写的太多被写穿）某一块地方数据做矩阵运算：加载到内存里太慢，直接在SSD上做——NDP（near data computing）

什么是OS：

A 一些应用软件不算。

B

C.文件系统属于OS。

D 应用软件不算。

E

F

G 运行在用户态的IO框架是否算？

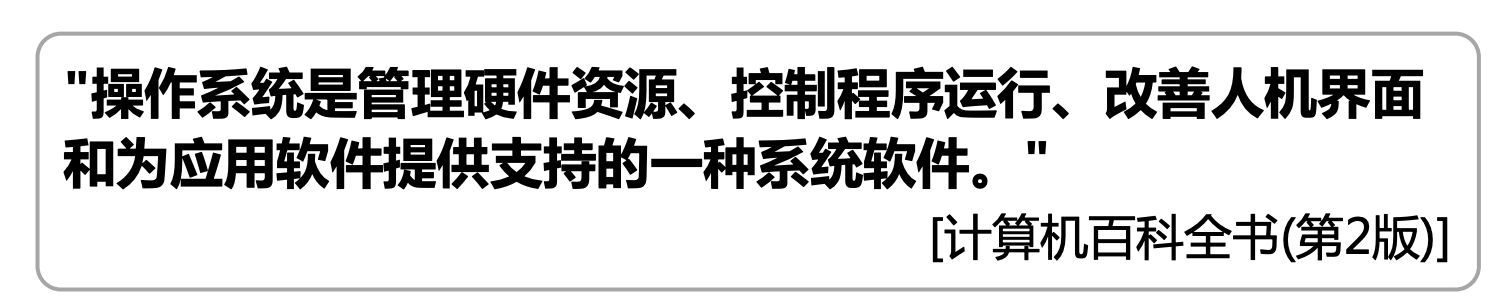


操作系统和应用的关系：服务应用，管理应用。

管理：不能让一个应用占太多资源，kill。governer。

服务：server

管理硬件 AND 抽象硬件



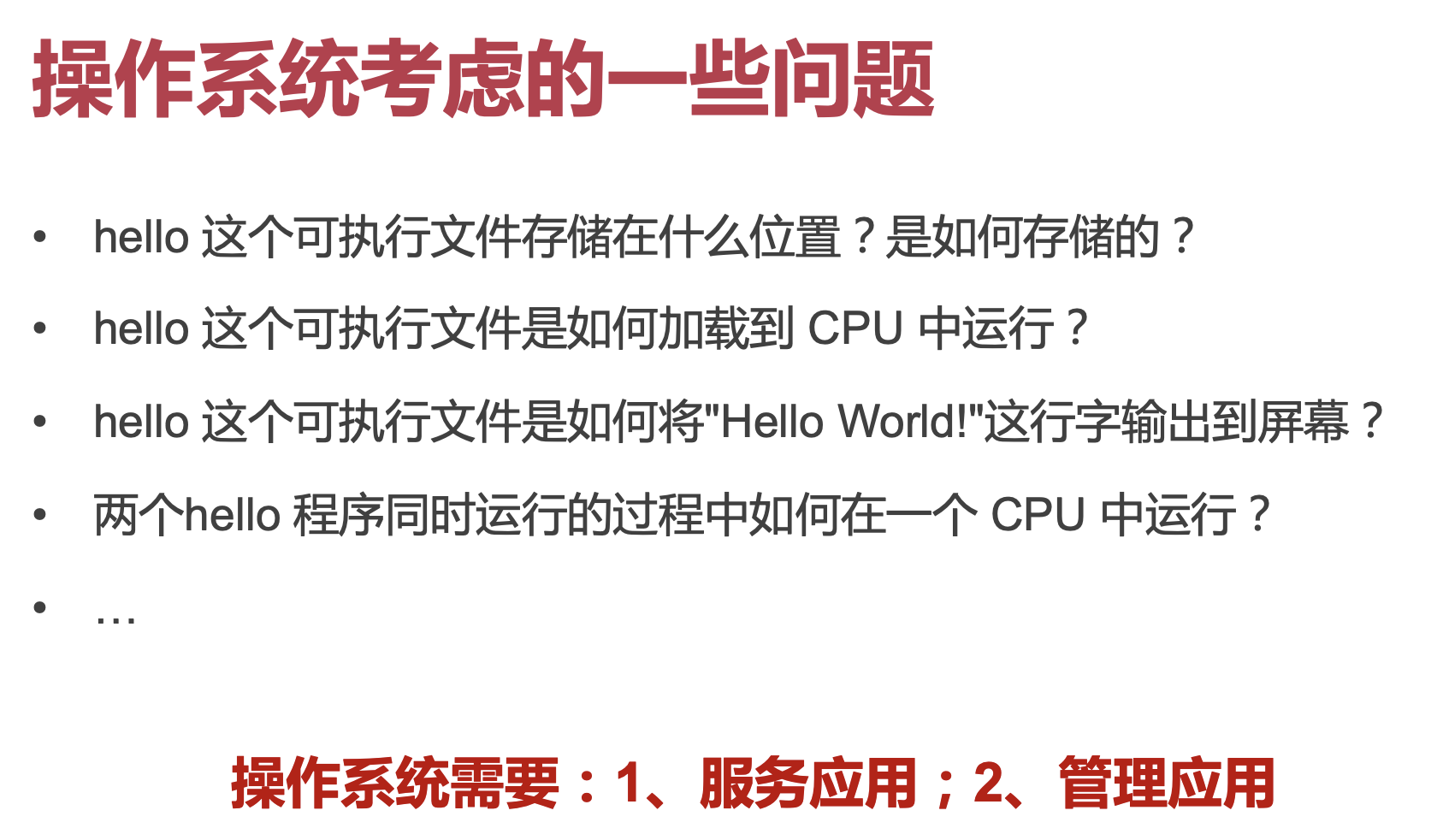
人机界面：有些过时：有可能是机与机之间的界面。

操作系统并没有严格的唯一定义。

操作系统可以包含运行在用户态的框架。

内核态用户态为了适应软件分层才产生的概念。

安卓应用通信：IPC：与图形界面交互：android提供service运行在用户态。



1.文件系统相关。

2.内存布局in ICS。

3.屏幕作为一个设备是如何抽象的。

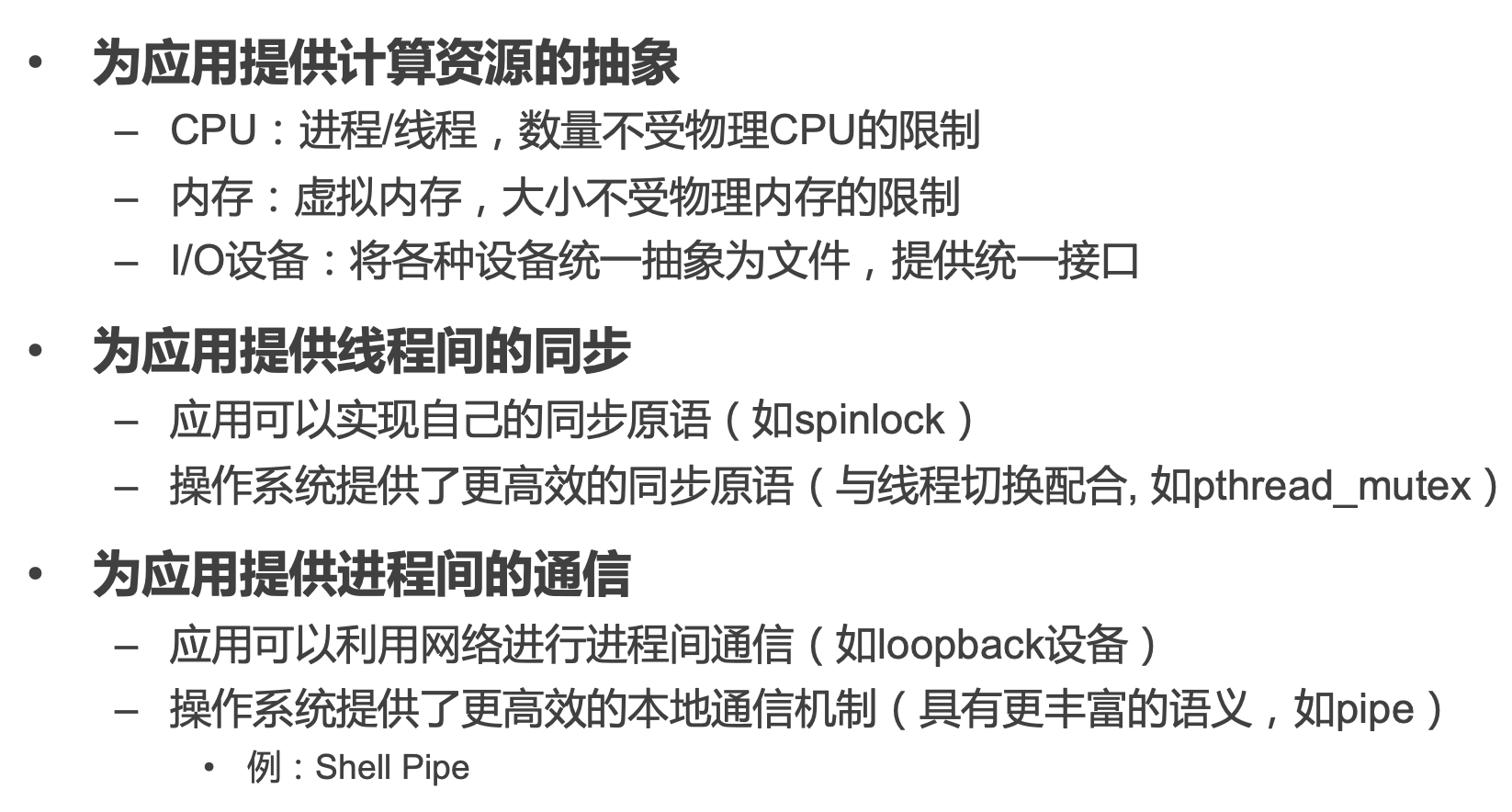
4.时分复用。——管理

5.两个hello如何通信？——服务。

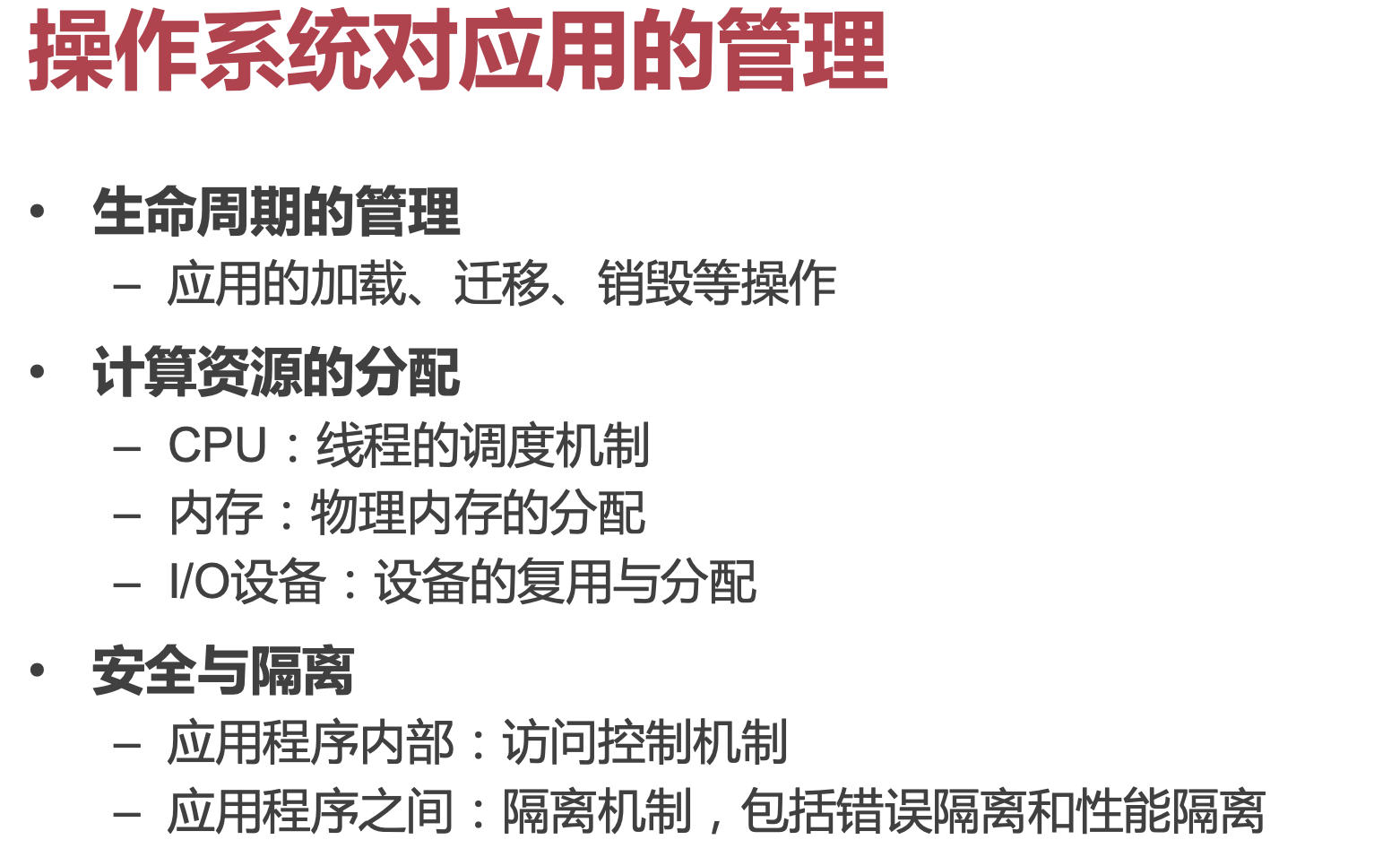
6.如果其中一个bug了，如何保证另一个不受影响正常运行？

7.如何同时运行1000个，10000个。。。

操作系统对应用的服务。



OS中对锁的支持is important。



Q：如何避免流氓应用独占CPU资源？

A：1.切换（10ms发生时钟中断，调度器决定下一个任务）2.信号中断。

Q：fork bomb？

int main() {

while(1)

fork();

}

A： 1.重启。

2.代码运行在虚拟机里。（虚拟机外面只能看到一个进程。给虚拟机配#个虚拟CPU就有几个线程）如果运行fork bomb：会把四个虚拟核占满。

3.Linux cgroup：预先设置某个应用允许占用的资源。（容器中的方法）

之所以我们要讨论什么是OS：今天有越来越多的人声称我们在做OS。（城市OS，对城市的控制）云上有云OS。在我们看来：都是用户态的应用程序：上下文语义发生了变化。安卓UI界面属于OS的一部分。

## 为什么要有OS？

最早：应用OS没有解耦：OS就是应用，应用就是OS。

> 计算器上有OS吗？不care。

什么时候care：想在计算机上装软件（装个俄罗斯方块）。

功能越来越多：开始出现分工和分化：底层功能被频繁使用（对存储设备的操作）：常用工能形成特定的模块（存储模块）——为什么不能就作为一个库。

特权级的必要性：

1.多道应用。

2.保证不同应用间的隔离。功能上有了划分：让应用降权，升权。

3.至少两种权限。（推理出来的，Intel早期有四种权限0到3最后真正用的就是0和3）

OS取最大公约数：所有的处理器至少两种权限。为了兼容性。

## 对CPU要求：

1.CPU对软件提供的接口称为ISA。

2.分化出两个模式特权非特权。

trap

非特权部分与特权部分的交互：Syscall

Q：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，do we still need OS？

A：OS可以演变为库的形式，libos：不负责管理应用，只负责对硬件抽象。

Q：应用想要fully control硬件（database），是否需要OS？

A：除非只有一个应用，OS在这的作用就是管理。一台计算机有10块硬盘，每个应用管自己的硬盘：OS管理应用不能写其他人的磁盘——XO kernel

向上提供抽象 AND 提供管理可以去掉一个留一个。

## OS演化：

### 外部演化：接口演化：

- POSIX接口

- Linux：不完全遵循POSIX（实现POSIX），但有自己提供的新接口。

- 鸿蒙：分布式软总线（写一个应用程序：既可以跑在手表上，也可以跑在大屏幕上，类似卡片，不用开发多套软件——提高生产力）

### 内部演化：

架构演化：宏内核——微内核（也可以实现POSIX API）——外核结构——多内核架构

为了：更好的扩展，容错，安全，兼容等等。

Future: CPU，GPU，DPU，SSD上也有PU OS装CPU上，每个人都有OS，如何协调——多内核，一个应用程序在Kernel中切换，想跑CPU跑CPU，想跑硬盘跑硬盘。

## 图形界面用户态OR内核态：

### Windows GUI：

设计上是微内核

- 早期放在内核态，代码量过于庞大导致内核臃肿

- 后来放到用户态，windows NT初期

- 后来有部分回到内核态：从而获得更高的性能

架构上微内核，实际上跑在内核态

### Linux

X window完全跑在用户态，交互通过TCP/IP，即使在单机上。

任意应用的界面可以在任意显示器显示（类似远程桌面）

性能差，时延过高

linux社区想把越来越多与显示相关的东西放到内核态。（即使是驱动的形式）

Nowadays：EBPF

## 当今的挑战

1.设备发生了巨大的变化

- 5G，6G互联时延越来越低

- AI设备导致装机的发生了变化：GPU is expensive。。。

- IOT

封闭到开放

windows从收费到免费

安卓全开源

——再到封闭

\*\*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\*\*

专用到通用

linux没有通过汽车领域的标准：实时性不行

矿鸿。。。

计算机系统从传统的分层解耦走向云+端的垂直整合

——再到专用

操作系统的类型和数量均会大幅增加

从简单到复杂

——到更复杂

芯片上的OS不是单一OS，而是一组OS

摩尔定律终结时，计算机下一步发展是什么？

算法用python在通用CPU上写 VS 为这个算法特定的硬件

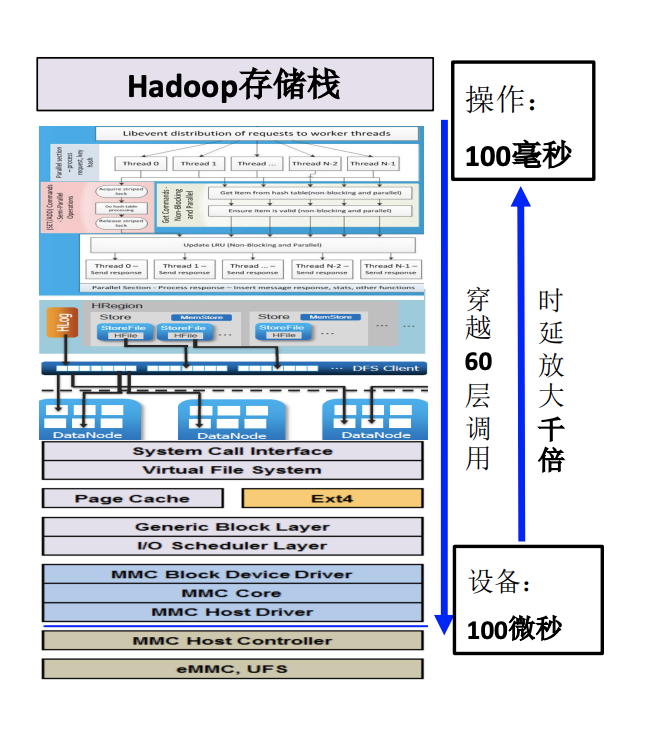
性能相差6w倍

现有算法一个个拆解变成一个个专用算法

Future：DSA domain specidic architecture——每个硬件只做一件事

》矿机算hash快

上层OS发生巨大变化：传统程序无法发挥硬件巨大能力



层层叠叠抽象把硬件的好处都遮蔽了。

现在OS过于追求通用。

## 操作系统历史包袱太重

linux代码more than 3000w行，200w行increase per year

开源license不断变得受限

## why we need to learn OS？