



L1 操作系统概述

宋卓然

上海交通大学计算机系

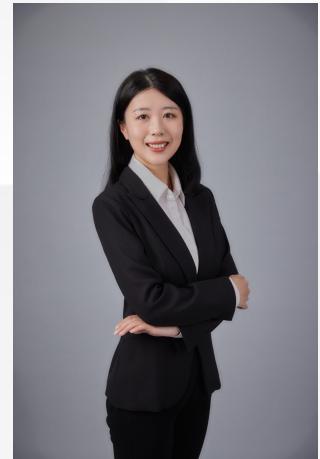
songzhuoran@sjtu.edu.cn



饮水思源 · 爱国荣校



我是谁?



上海交通大学计算机系助理教授

电信群楼3-125

主页:

<https://songzhuoran.github.io>

Email: songzhuoran@sjtu.edu.cn





课程信息

■ 助教：齐春宇；邮箱：qichunyu@sjtu.edu.cn

■ 课程ppt

■ <https://songzhuoran.github.io>

■ Canvas

■ 课程书本

■ Abraham Silberschatz等编著，操作系统概念（第7版、第8版、第9版、第10版，中英文版本均可）。

■ Abraham Silberschatz et al. Operating System Concepts (7th, 8th, 9th, 10th Edition in English or Chinese).





课程考核



■ 分数比例

- 课堂作业 15%
- 课后作业 10%
- 课程设计 15%
- 期末考试 60%

■ 课程设计

- 3人一组
- 每组20分钟演讲
- 主题为面向新算法（AI算法）、新硬件（GPU A100、V100、tensor core）的操作系统设计
- 参考会议包括： ASPLOS、OSDI、NSDI、EuroSys、USENIX ATC、HPCA
- 年份包括2021-2024





学术诚信

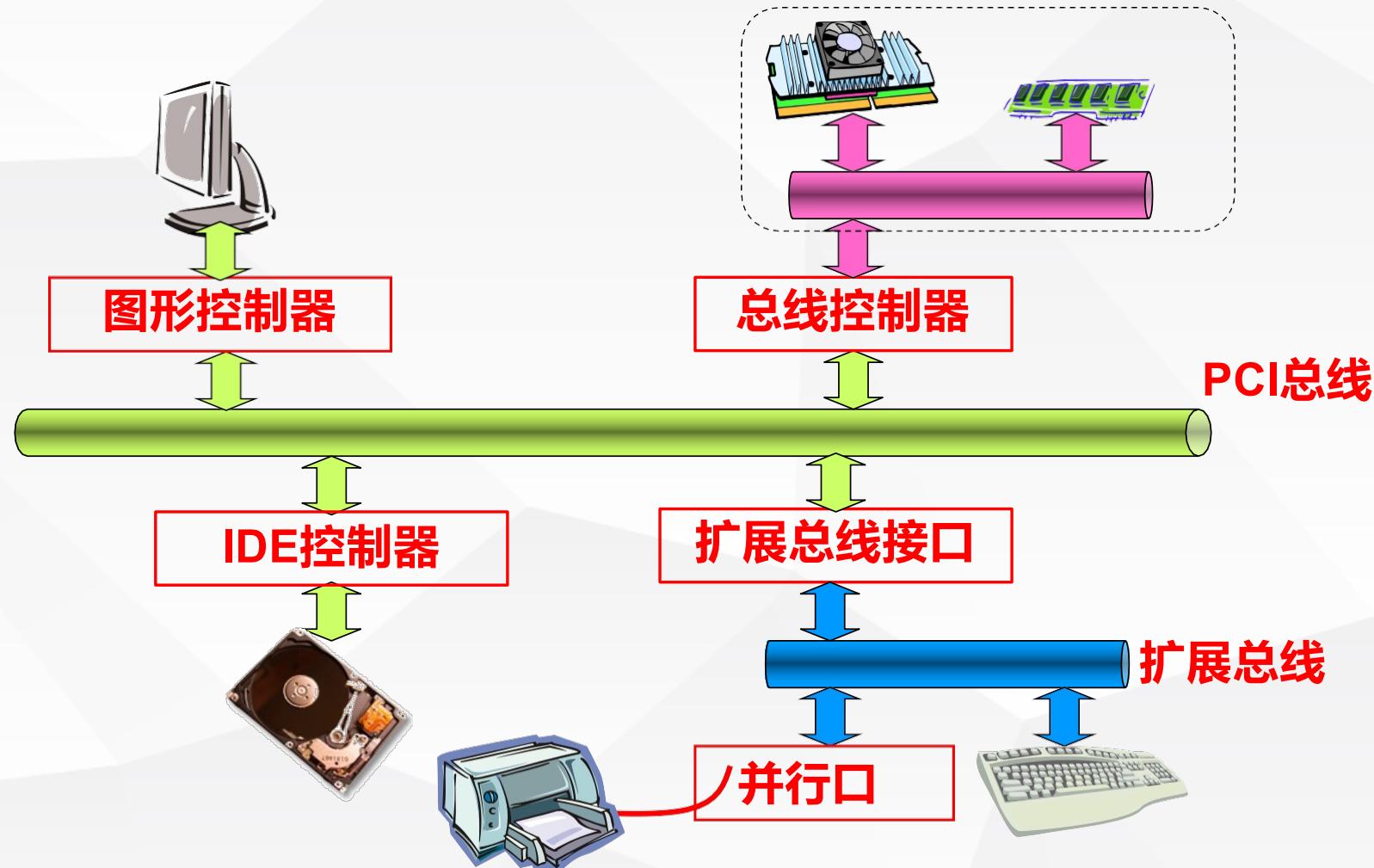
- 不要与其他同学共享你的作业/程序
- 如果需要用到别人发表的工作中的内容，请适当引用

NO PLAGIARISM!





计算机系统组件

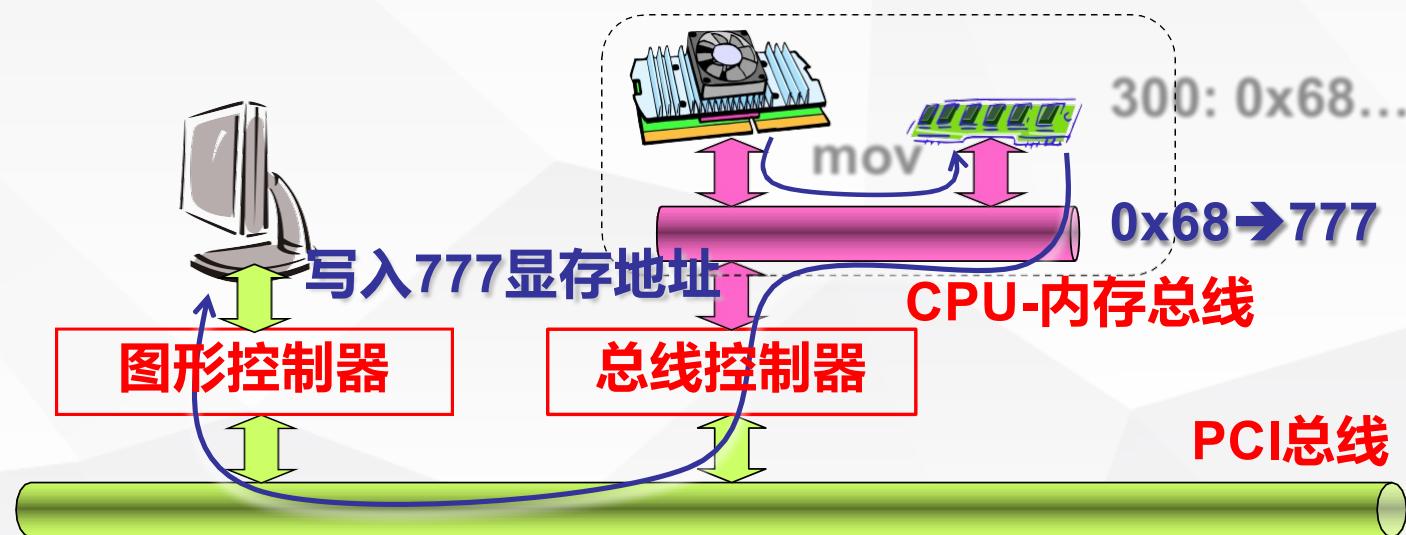




计算机系统组件



- 计算机的作用是?
 - 帮助人们解决实际问题
- 如何在屏幕中显示一串字符“hello world”?

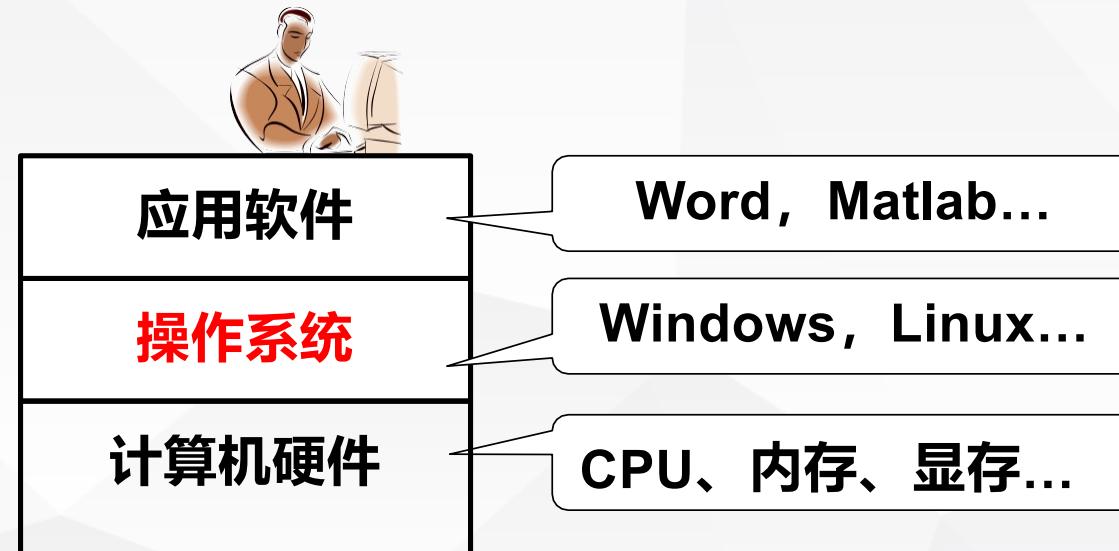




操作系统是在硬件和应用之间的软件层



- 计算机硬件，有人戏称为“裸机”，要使用计算机就要控制计算机硬件
 - 但上层程序员可以通过“printf”在屏幕上显示，如何做到？
 - 需要给计算机硬件穿上衣服----操作系统！



在穿上了衣服的计算机上再次：屏幕上输出
“hello!”





什么是操作系统



- 是计算机硬件和应用之间的一层软件
 - 方便我们使用和管理硬件 (CPU、内存、显示器)
 - 管理哪些硬件

CPU管理

内存管理

终端管理

磁盘管理

文件管理

网络管理

电源管理

多核管理





从 Hello World 说起



```
#include <stdio.h>

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

运行Hello时，操作系统的角色是？

```
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、管理硬件； 2、服务应用





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)





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





操作系统的[设计目标](#)有可能存在冲突

- 服务应用的目标:单个应用的运行效率最大化
- 管理硬件的目标: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





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);  
}
```





从用户视角看操作系统



- 单一用户使用计算机
- 操作系统目的：
 - 用户使用方便
 - 性能（次要）
 - ~~资源开销~~





从用户视角看操作系统



- 多个用户共享使用大型计算机、服务器
- 操作系统目的：
 - 优化资源利用率
 - 确保CPU、内存、I/O利用率高
 - 确保没有用户使用超额的硬件资源

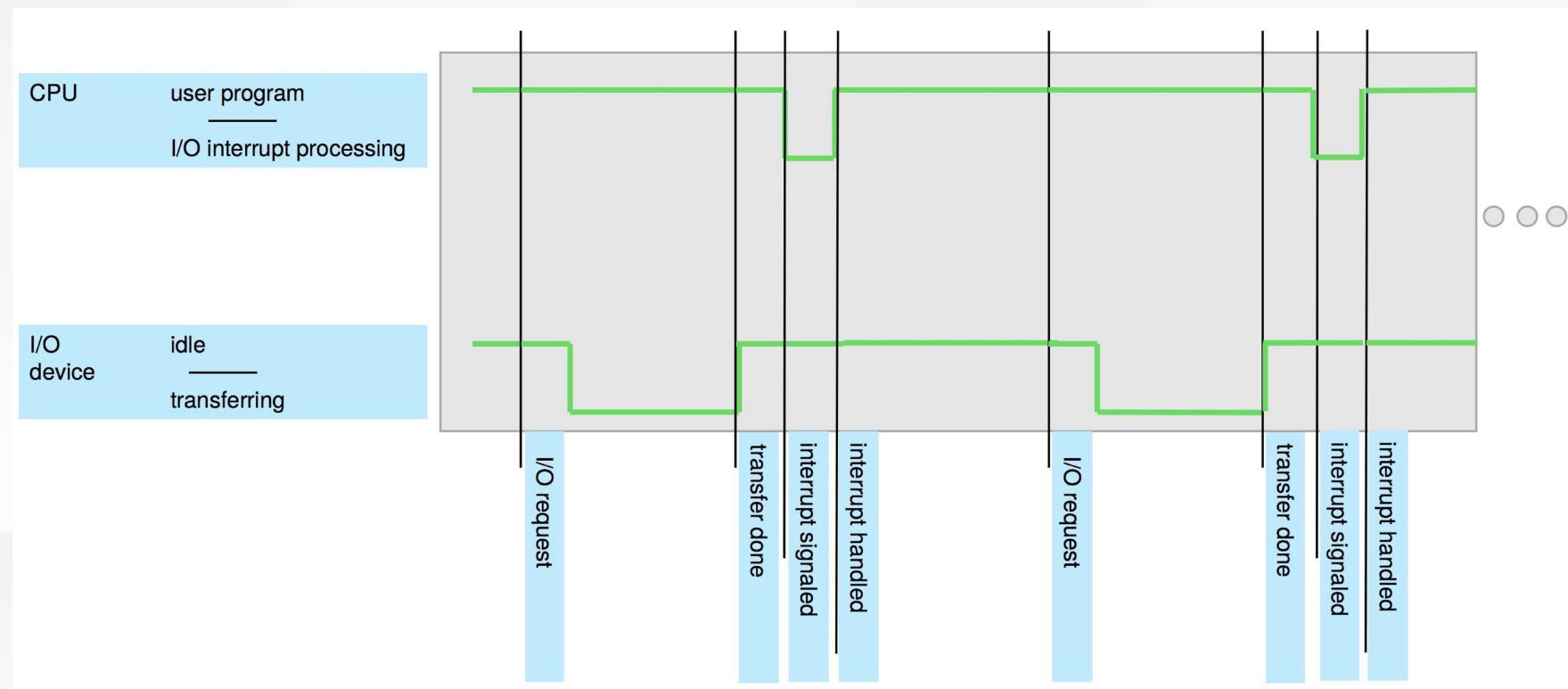




中断



- 外部中断：硬中断
 - 可屏蔽中断，通过INTR信号线进入CPU，正常的响应请求
 - 不可屏蔽中断，表示出现致命错误，需立即响应





中断

- 可屏蔽中断，分为上下部分
 - 上部分：需要立即响应，限时执行的中断处理程序（如中断应答、硬件复位）
 - 下部分：在下半部分执行时，如有新的中断发生，此时旧中断的下半部分会被换下CPU，先执行新中断处理程序的上半部，然后在等待线程调度机制将其调度到CPU上完成下半部分的执行





中断

- 内部中断
 - 软中断：由软件主动发起的中断，并不是某种内部错误
 - 异常：CPU在执行指令期间发生错误导致的中断成为异常，比如分母为0
 - 故障：CPU恢复到异常前的状态，CPU将返回地址依然指向导致fault异常的那条指令，给机会让其重新执行
 - 陷阱：程序在运行时掉进了CPU设置的陷阱从而停止运行，通常用于调试中，比如int3指令便可引发该异常，CPU将中断处理程序的返回地址指向导致异常指令的下一个指令地址
 - 终止：错误无法修复，将此程序从进程表中去掉





计算机系统体系结构

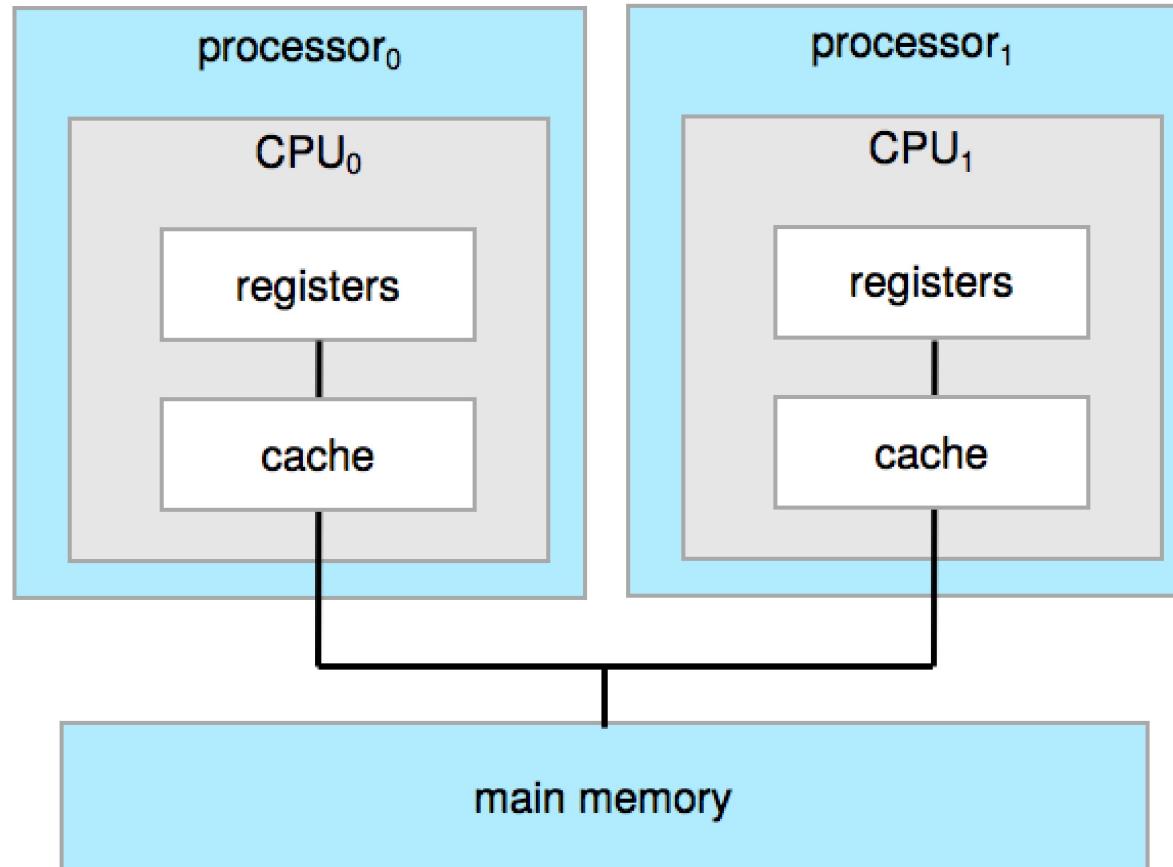


- 单处理器系统
 - 一个主CPU
- 多处理器系统（并行系统）
 - 增加吞吐量
 - 规模经济
 - 增加可靠性
- 多核处理器

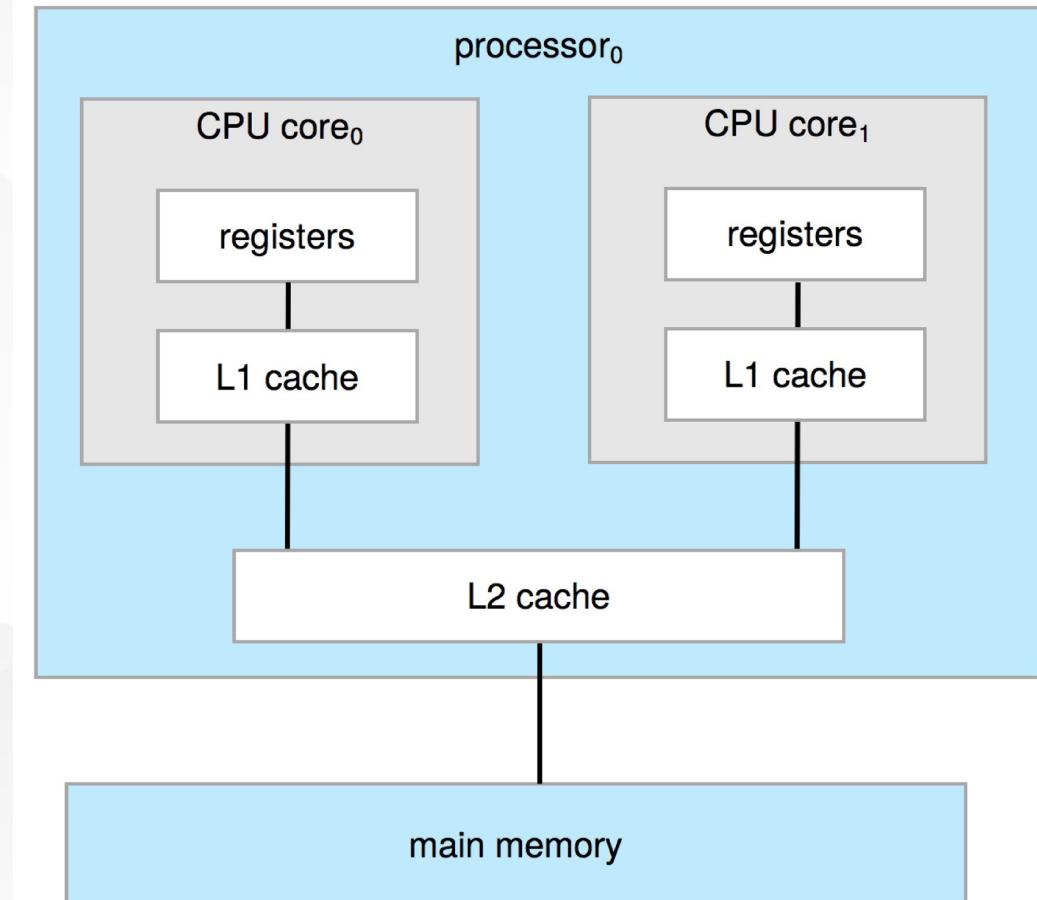




计算机系统体系结构



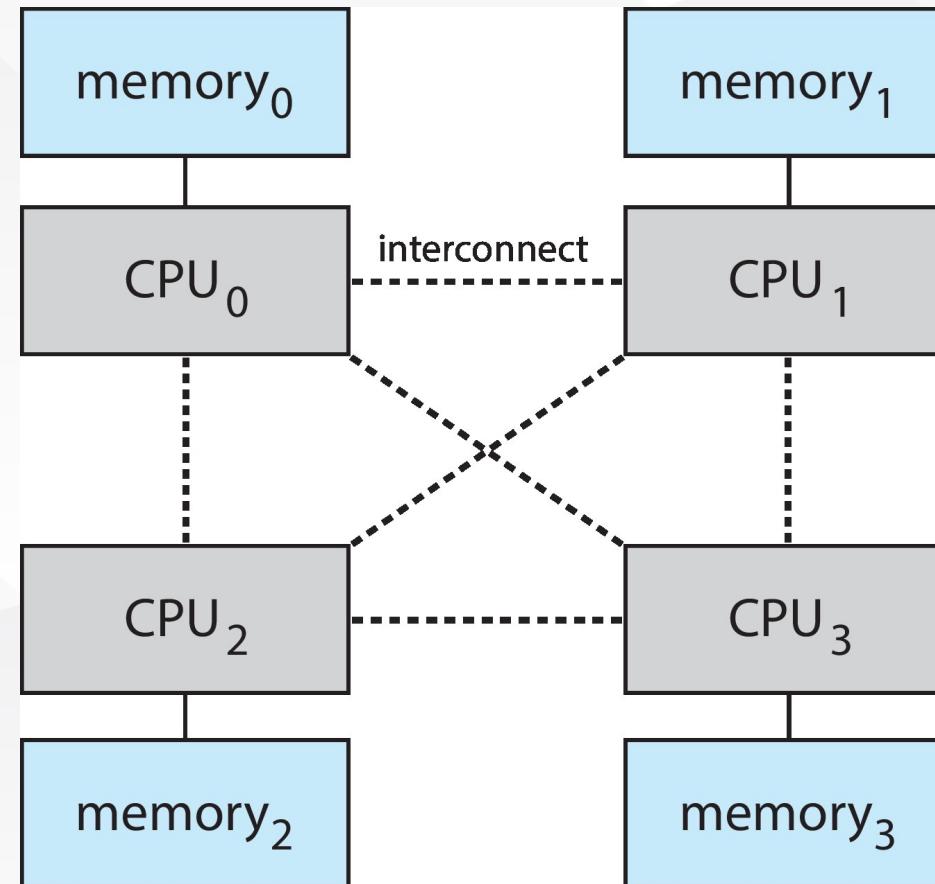
多处理器系统



多核处理器



- 均匀内存访问 (Uniform memory access)
- 非均匀内存访问 (Non-Uniform memory access) : CPU访问内存时间不一致

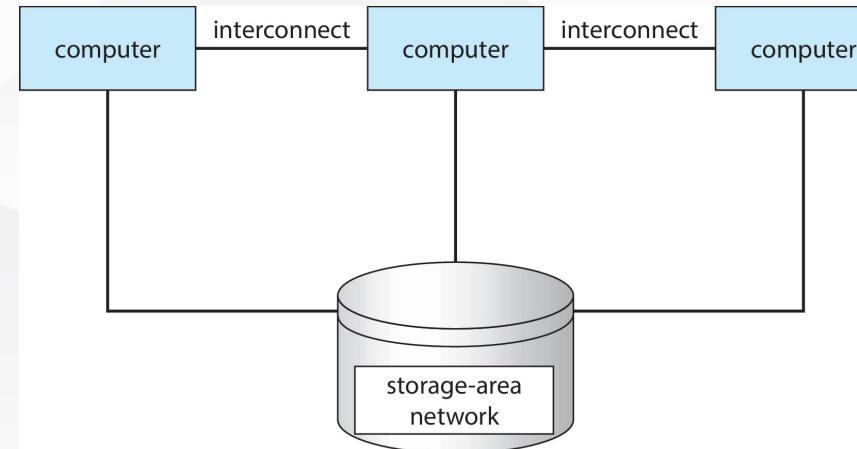




集群系统



- 两个或多个独立系统（节点）组成
- 采用LAN(Local Area Network, 局域网)连接或更快的内部连接，如 InfiniBand
- 对称集群：每个处理器都参与完成操作系统的所有任务，处理器间没有从属关系
- 非对称集群：包含热备份处理器

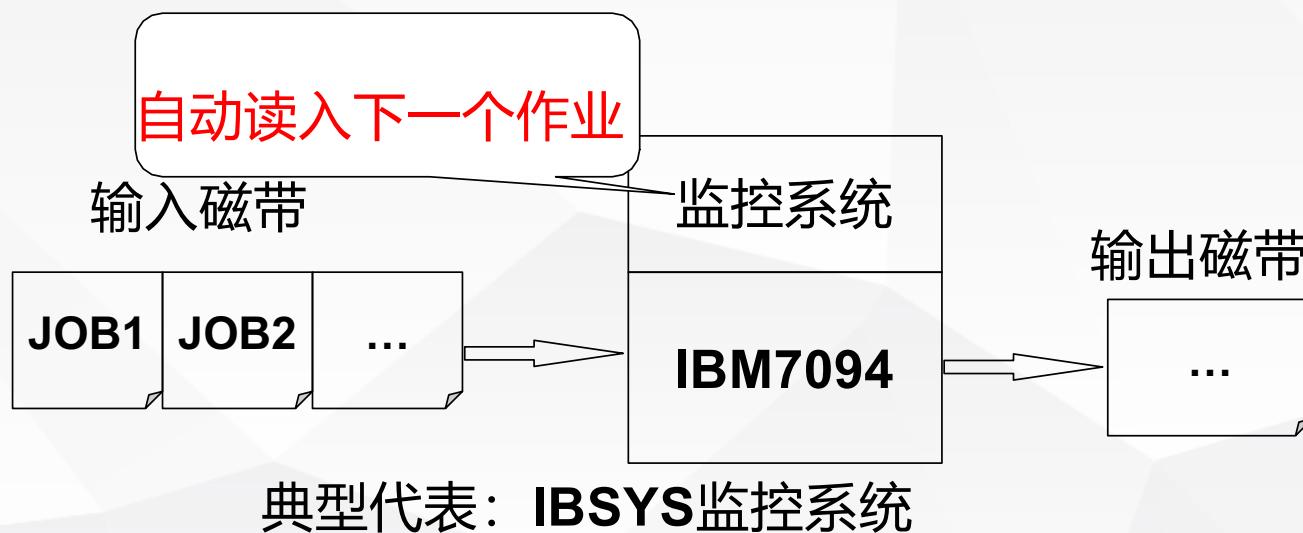




操作系统简史



- (1955-1965)计算机非常昂贵，上古神机**IBM7094**，造价在**250万美元以上**
 - 计算机硬件的使用原则：只专注于计算
 - 批处理**操作系统**(Batch system)





从IBSYS到OS/360(1965-1980)



计算密集型

访存密集型

- 计算机开始进入多个行业：科学计算(IBM 7094机器)，银行(IBM 1401机器，性能越来越高，价格越来越便宜，摩尔定律)
 - 需要让一台计算机干多种事，批处理操作系统已经不适合
 - 多道程序(multiprogramming)
 - 作业之间的**切换和调度**成为核心：因为既有I/O任务，又有计算任务，需要让CPU忙碌
 - 典型代表：IBM OS/360(360表示全方位服务)，开发周期**5000**个人年

多进程结构和进程管理概念萌芽！

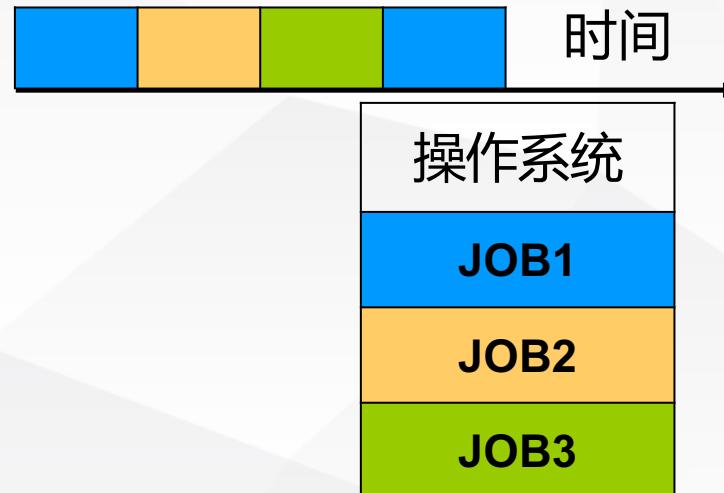




从OS/360到MULTICS(1965-1980)



- 计算机进入多个行业, **使用人数增加**
 - 如果每个人启动一个作业, 作业之间快速切换
 - 分时系统(timesharing), 为了响应时间, **分时/定期地**切换作业, 而不是**等到阻塞**
 - 代表: MIT MULTICS (MULTIplexed Information and Computer Service)
 - 核心仍然是任务切换, 但是资源复用的思想对操作系统影响很大, 虚拟内存就是一种复用

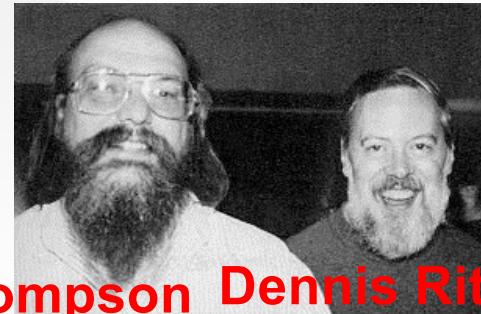




从MULTICS到UNIX(1980-1990)



- 小型化计算机出现，PDP-1每台售价120,000美元，售价不足IBM 7094的5%
 - 越来越多的个人可以使用计算机
 - 1969年：贝尔实验室的Ken Thompson、Dennis Ritchi（图灵奖获得者）等在一台没人使用的PDP-7上开发一个简化 MULTICS，就是后来的UNIX
 - UNIX是一个简化的MULTICS，核心概念差不多，但更灵活和成功



Ken Thompson Dennis Ritchi





从UNIX到Linux(1990-2000)



- 1981, IBM推出IBM PC; 个人计算机开始普及
 - 很多人可以用计算机并接触UNIX
 - 1987年Andrew Tanenbaum发布了 MINIX(非常类似UNIX)用于教学
 - Linus Torvalds在386sx兼容微机上学习minix, 作出小Linux于1991年**开源发布**
 - 1994年, Linux 1.0发布并采用GPL协议, 1998年以后互联网世界里展开了一场历史性的Linux产业化运动





核心思想、技术

- 历史使人明智
 - 作为管理者，操作系统要让**多个程序合理推进**，就是**进程管理**
 - 用户通过执行程序来使用计算机(吻合 冯诺依曼的思想)
 - 多进程(用户)推进时需要内存复用等等

软件实现

- 对于操作系统，实现很重要OS/360->UNIX
- 需要真正的群体智慧 UNIX->Linux





历史是多线条的：PC与DOS



- PC机的诞生一定会导致百花齐放；IBM推出PC，自然需要为这个机器配备一个操作系统
 - 1975年Digital Research为Altair 8800开发了操作系统CP/M
 - CP/M：写命令让用户用，执行命令对应的程序
 - 1980年出现了8086 16位芯片，从CP/M基础上开发了QDOS系统

```
Loading CPM.SYS...
CP/M-86 for the IBM PC/XT/AT, Vers. 1.1 (Patched)
Copyright (C) 1983, Digital Research

Hardware Supported :

Diskette Drive(s) : 3
Hard Disk Drive(s) : 1
Parallel Printer(s) : 1
Serial Port(s) : 1
Memory (Kb) : 640
CP/M

D>a:
A>dir
A: PIP      CMD : STAT      CMD : SUBMIT    CMD : ASM86     CMD
A: GENCMD   CMD : DDT86    CMD : TOD       CMD : ED        CMD
A: HELP     CMD : HELP     HLP : SYS       CMD : ASSIGN    CMD
A: FORMAT   CMD : CLDIR    CMD : WRTLDR    CMD : BOOTPCDS SYS
A: BOOTWIN  SYS : CPM      HB6 : WINSTALL  SUB : PD        CMD
A: WCPM     SYS : DISKUTIL CMD
A>-
User 0          0:00:11           Jan. 1, 2000
```





从QDOS到MS-DOS



- Bill Gates进入历史舞台
 - 1975年，22岁的Paul Allen和20岁的Bill Gates为Altair 8800开发了BASIC解释器，据此开创了微软
 - 1977年Bill Gates开发FAT管理磁盘
 - QDOS的成功在于以CP/M为基础将BASIC和FAT包含进来
 - 文件管理与编程环境都是用户关心的
 - 1980年IBM想和Digital Research协议授权使用CP/M，但没有达成，转向和微软合作
 - 1981年微软买下QDOS，改名为MS-DOS，和IBM PC打包一起出售



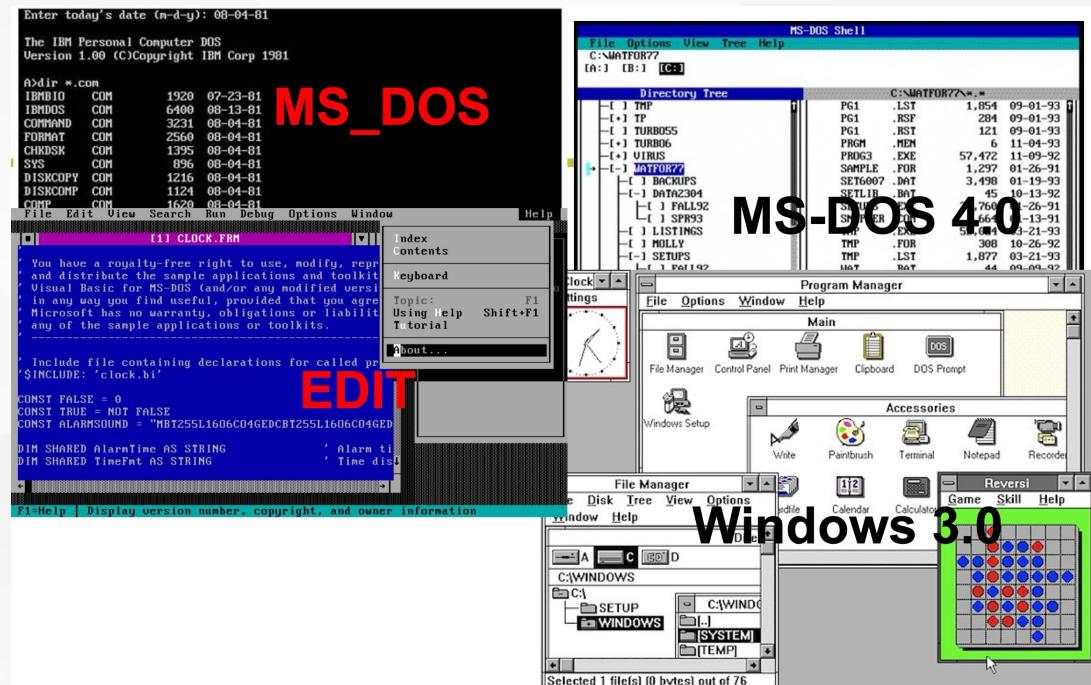


从MS-DOS到Windows



历史背景

- 为了使用户使用更加便利
 - 1989年，MS-DOS 4.0出现，支持了鼠标和键盘，此时微软已经决定放弃MS-DOS
 - 不久后Windows 3.0大获成功
 - 后来一发不可收拾，95, XP, Win7, Win8
- 文件、开发环境、图形界面对操作系统来说至关重要**





Mac OS与iOS



- 1984年，苹果推出PC（麦金塔机，Macintosh），简称Mac机，其处理器使用IBM、Intel或AMD等，核心在于屏幕、能耗等
 - 与Mac机一起发布System X系统，一上来就是GUI
 - 在System7以后改名为Mac OS 8
 - 2007年发布iOS，核心依然是Mac OS，引入手势
 - Mac OS专注于**界面、文件、媒体**等和用户有关的内容





核心思想、技术

- 历史使人明智
 - 仍然是程序执行、多进程、程序执行带动其他设备使用的基本结构
 - 但用户体验被更加重视：文件系统、编程环境、图形界面

软件实现

- 如何通过文件存储代码、执行代码、操作屏幕
- 如何让文件和操作变成图标、点击或触碰

操作系统的任务： (1) 管理底层硬件，实现多进程调度 (CPU、内存)； (2) 服务应用与用户，实现文件操作视图 (I/O设备、磁盘)





操作系统课程总览





进程管理

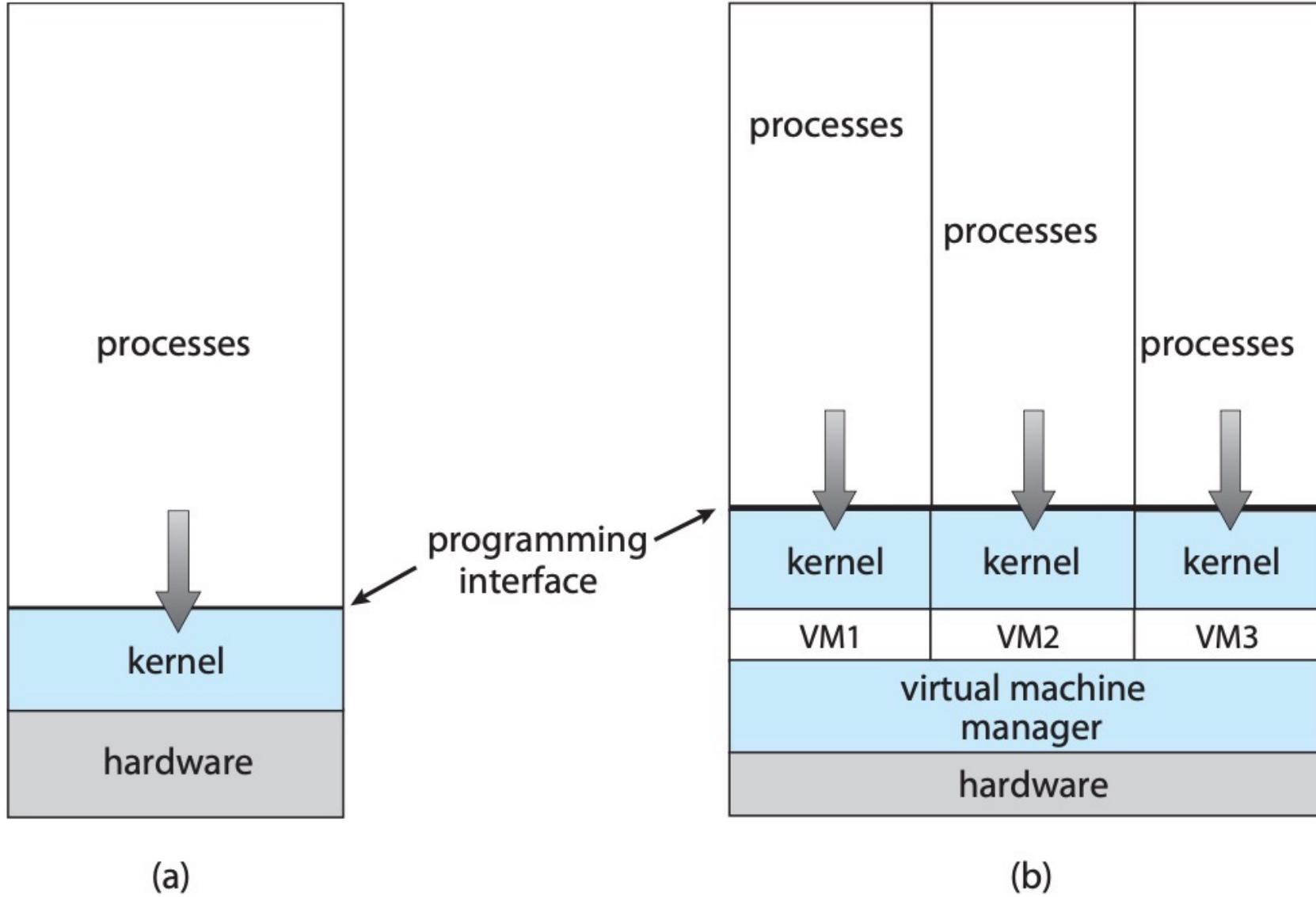


- 执行的程序称为进程
 - 微信、网页
- 程序本身不是进程，是被动实体，如同存储在磁盘的文件
- 进程是主动实体
- 进程需要资源来完成工作
 - CPU、内存、I/O设备
- 单线程进程有一个程序计数器（PC），指定了下一条要执行的指令
- 多线程进程有多个PC，每个指向下一个给定线程所需执行的指令
- 系统中有多个进程





单进程 vs 多进程





进程管理的活动

- 在CPU上调度进程和线程
- 创建和删除用户进程和系统进程
- 挂起和重启进程
- 提供进程同步机制
- 提供进程通信机制





CPU调度



- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-First (SJF) Scheduling
- Priority Scheduling
- Round-Robin Scheduling (时间片)
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling





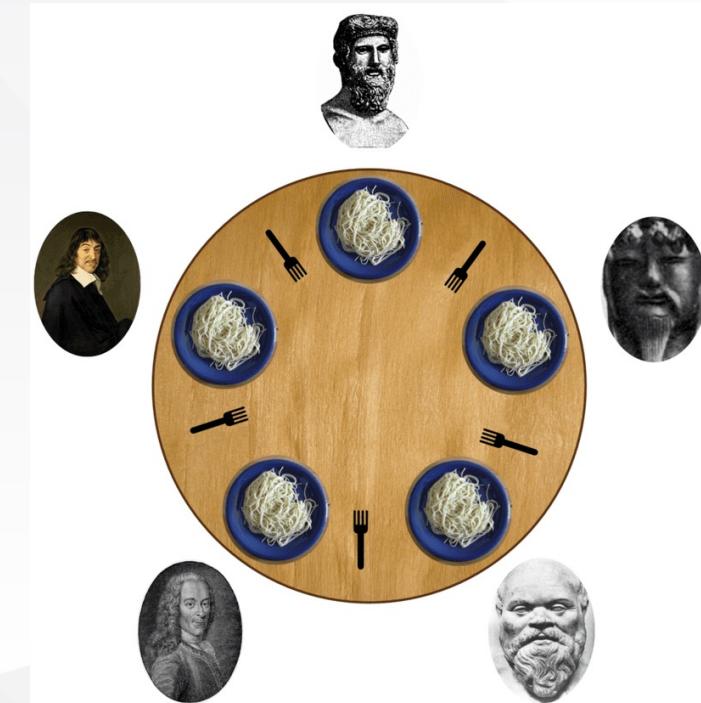
进程同步与死锁



- 哲学家就餐问题

假设有五位哲学家围坐在一张圆形餐桌旁，做以下两件事情之一：吃饭，或者思考。吃东西的时候，他们就停止思考，思考的时候也停止吃东西。假设哲学家必须用两只餐叉吃东西。他们只能使用自己左右手边的那两只餐叉。

哲学家从来不交谈，这就很危险，可能产生**死锁**，每个哲学家都拿着左手的餐叉，永远都在等右边的餐叉（或者相反）。





避免死锁



- 银行家算法

3种资源：A (10)、B (5)、C (7)

Snapshot at time T_0 :

	Max	Allocation	Need	Available
	A B C	A B C	A B C	A B C
P_0	7 5 3	0 1 0	7 4 3	3 3 2
P_1	3 2 2	2 0 0	1 2 2	
P_2	9 0 2	3 0 2	6 0 0	
P_3	2 2 2	2 1 1	0 1 1	
P_4	4 3 3	0 0 2	4 3 1	

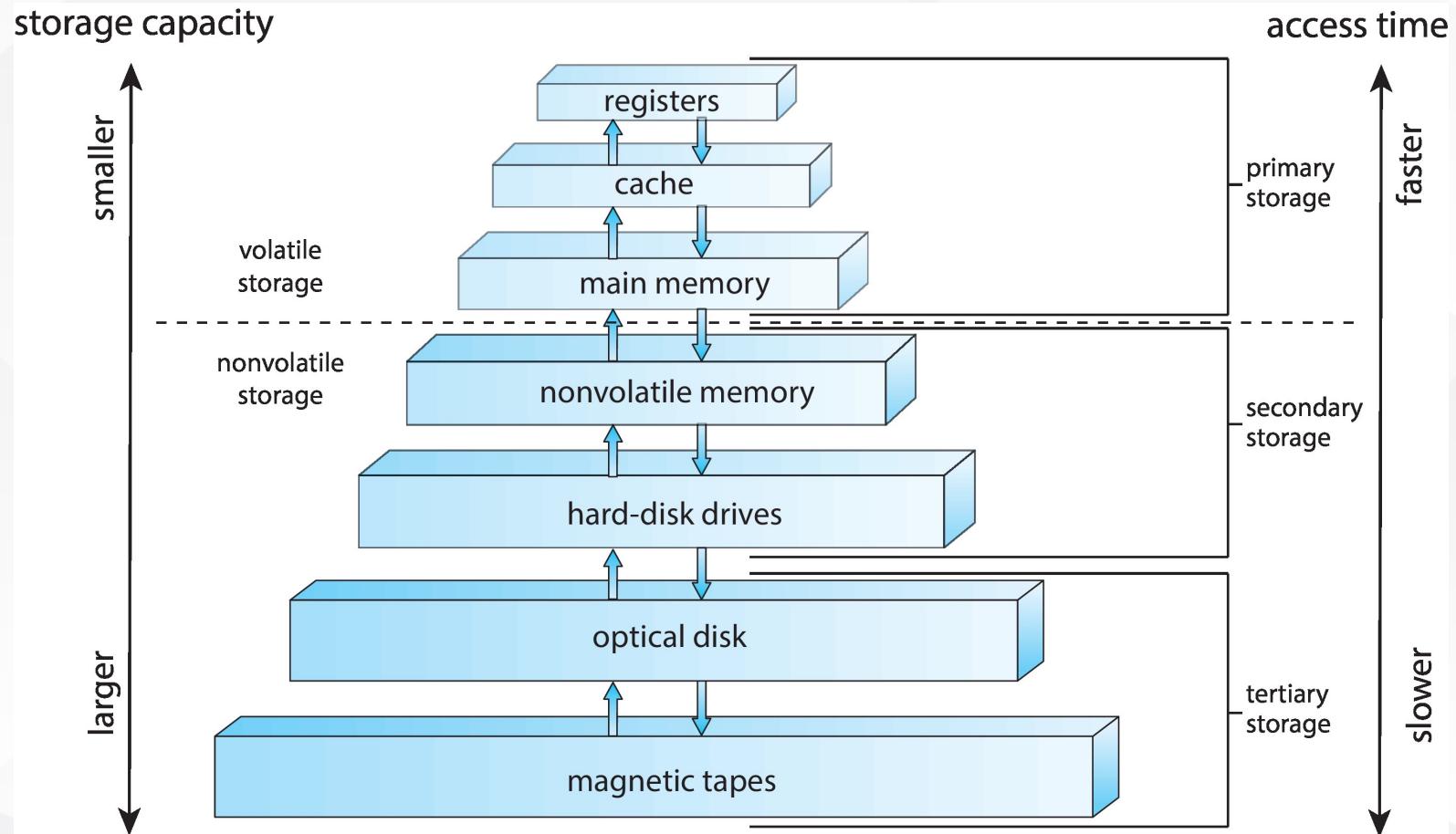
$\langle P_1, P_3, P_0, P_2, P_4 \rangle$ 是安全的调度方案





计算机存储结构

- 易失性存储结构
 - 寄存器
 - 高速缓存
- 非易失性存储结构
 - 磁盘
 - 硬盘





计算机存储结构



Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape





存储定义与符号



- 基本单位：位或比特 (bit) ---0/1
- 字节 (byte) : 8bit
- 字 (word) : 一个或多个字节
- 千字节 (kilobyte, KB) : 1024个字节
- 兆字节 (megabyte, MB) : 1024^2 个字节





内存管理

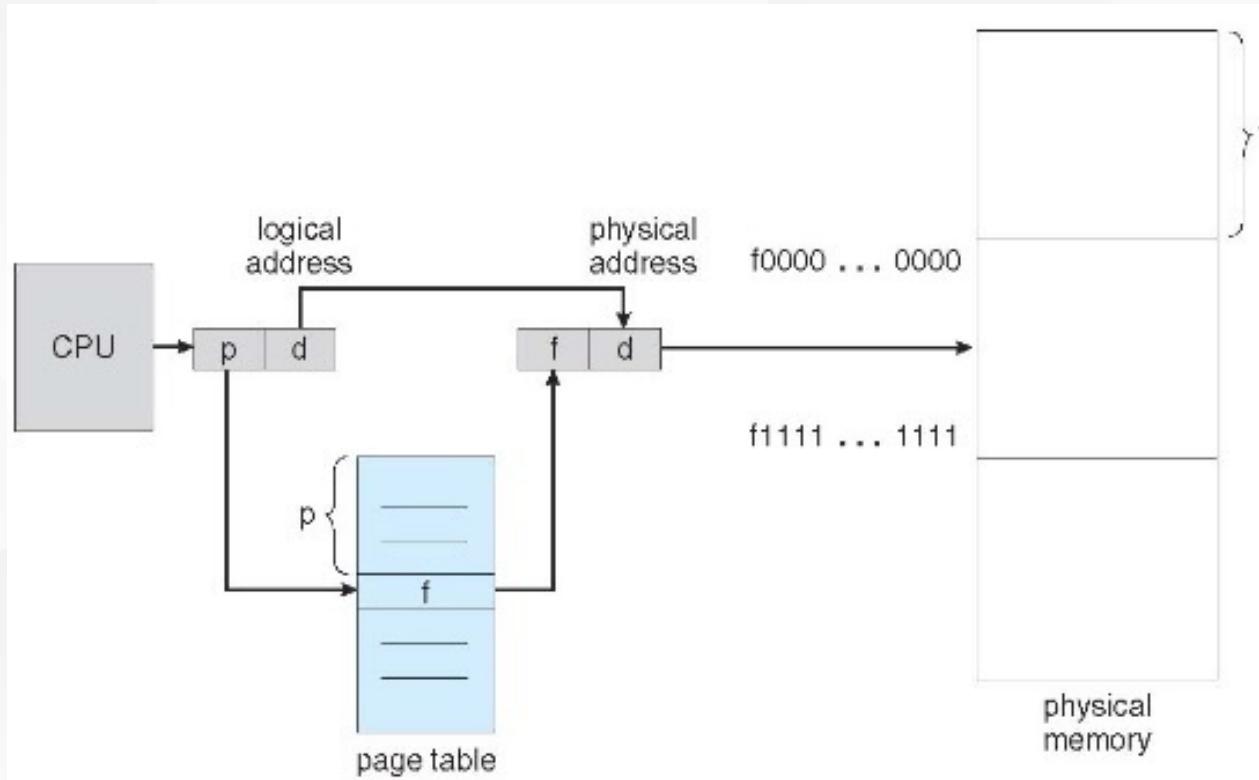
- 内存 (main memory) 是现代计算机系统执行的中心
- 大的字节数组
- 地址寻址
- CPU从内存中获取指令、数据
- 操作系统负责内存管理的以下活动：
 - 记录内存的哪部分被使用及被谁使用
 - 决定哪些进程会调入/调出内存
 - 根据需要分配和释放内存空间





内存管理

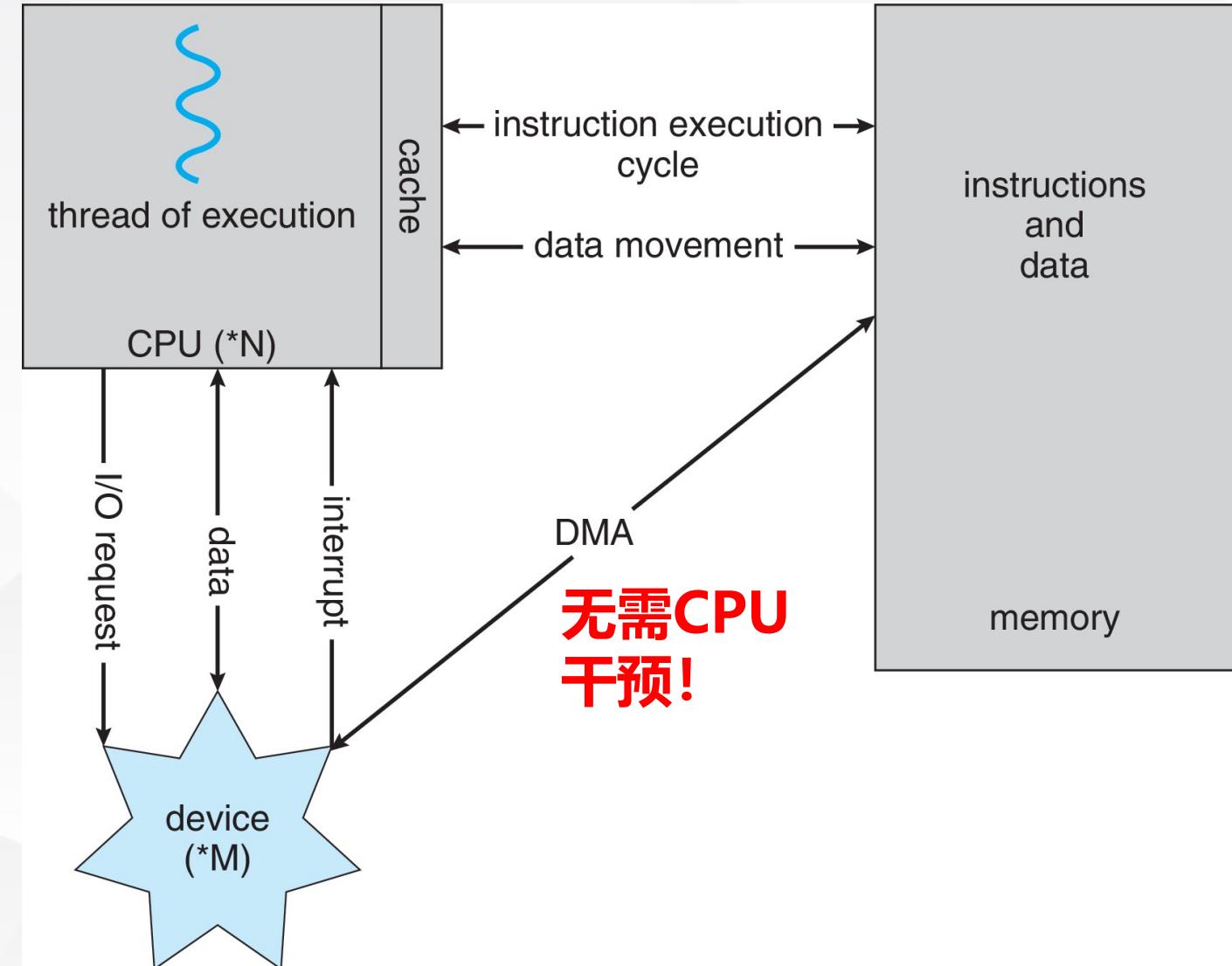
- 内存分配：为每道程序分配内存空间，保证进程之间地址独立，内存保护机制
- 地址映射：虚实地址转换，提高内存利用率





存储管理-I/O设备

- 存储器
- 键盘
- 显示器
- 设备控制器
- 设备驱动程序加载设备控制器的适当寄存器，决定采取什么操作
- 完成数据传输，通过中断通知设备驱动程序





存储管理-文件系统管理



- 文件是创建者定义的相关信息组合
- 文件内容为程序和数据
- 操作系统管理大容量存储介质并控制它们，以实现文件这一抽象概念
 - 创建和删除文件
 - 创建和删除目录，以便组织文件
 - 提供文件和目录的操作原语
 - 映射文件到外存
 - 备份文件到稳定的存储介质

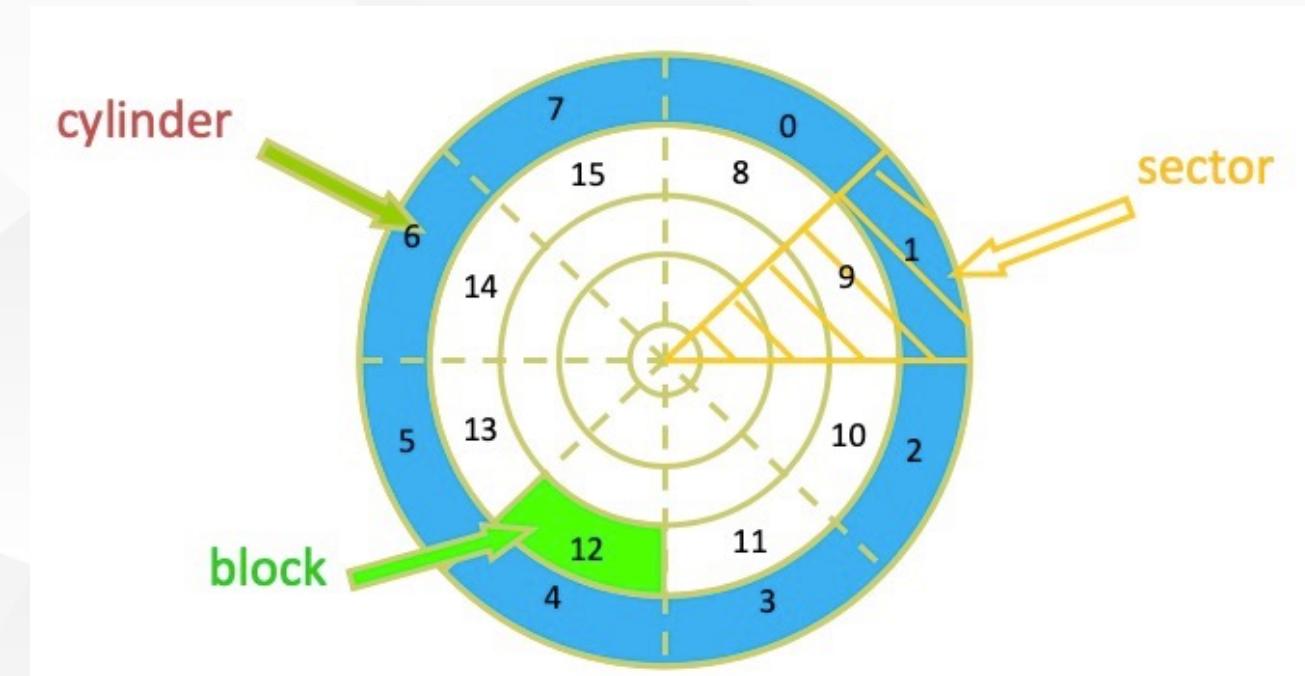




存储管理-大容量存储器管理



- 非易失性存储器
- 备份内存数据
- 操作系统负责有关硬盘管理的以下活动：
 - 空闲空间管理
 - 进程的存储空间分配
 - 磁盘调度





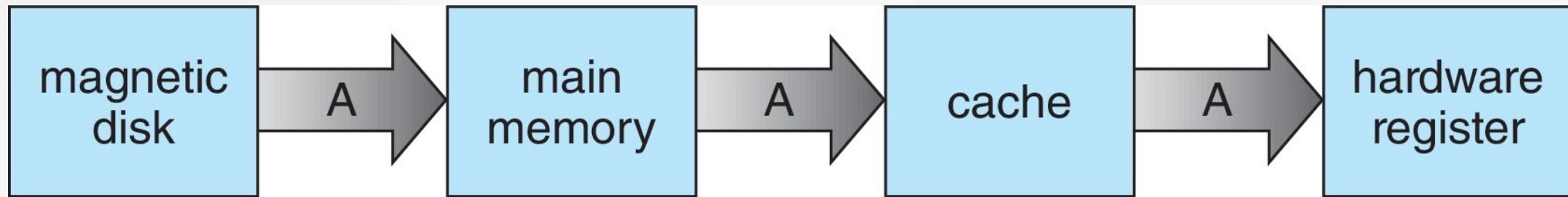
存储管理-高速缓存



- 信息通常保存在一个存储系统中(如内存)，使用时，它会被临时复制到更快存储系统，即高速缓存 (caching)
 - Hit (命中)
 - Miss (缺失)
- 高速缓存大小有限，高速缓存管理至关重要
 - 高速缓存大小
 - 替换策略



- 多任务调度环境需要尤其关注多进程访问同一数据A
 - 确保每个进程得到最新更新的A



- 对于多处理器环境，A可能保存在多个高速缓存上
 - 高速缓存一致性的问题
 - 通常由底层硬件负责数据的正确性



保护与安全



- **保护**: 多用户、多进程的情况下，数据访问需要加以控制，经过操作系统授权，进程才能使用相应资源
 - 文件
 - 内存（进程有自己的地址空间）
 - CPU
- **安全**: 防止系统受到外界或内部的攻击
 - 病毒
 - 蠕虫
 - 身份偷窃

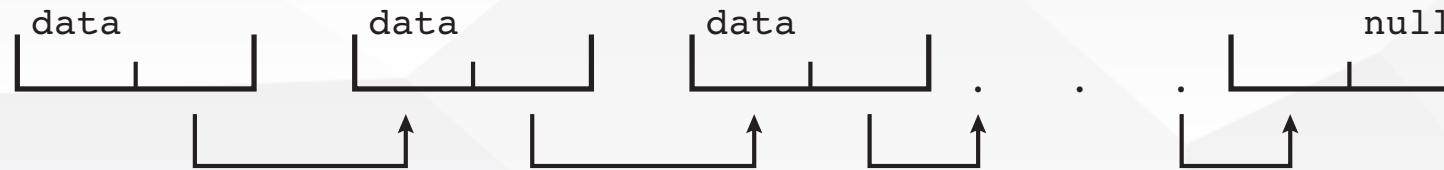




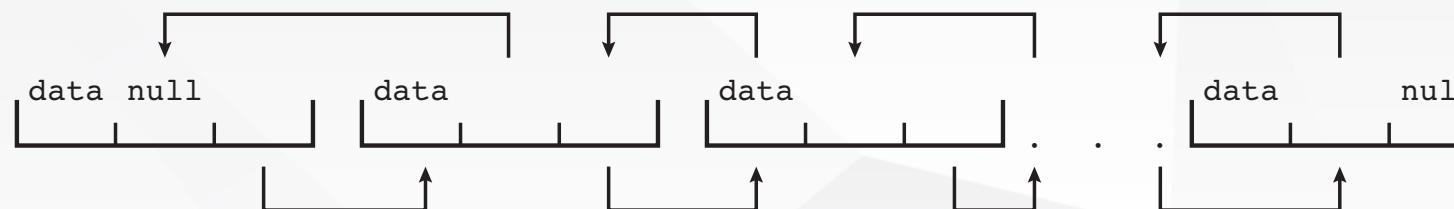
内核数据结构



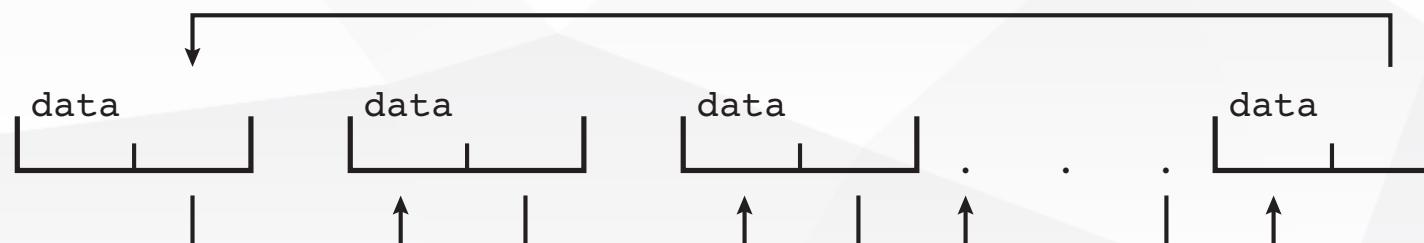
- 单向链表：每项指向它的后继



- 双向链表：每项指向它的前驱和后继



- 循环链表：最后一项指向第一项

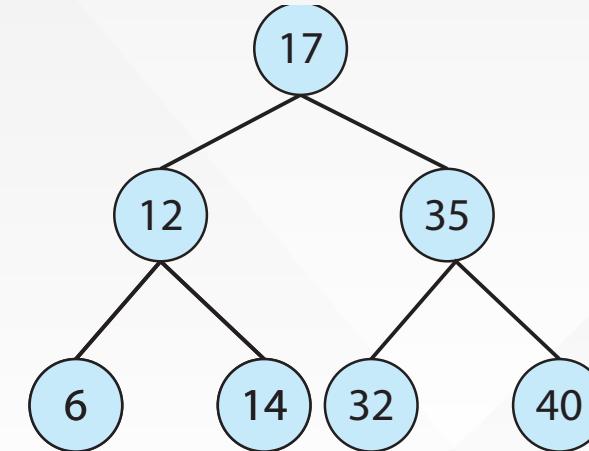




内核数据结构



- 二叉查找树
 - 左子节点 \leq 右子节点
 - 最坏的搜索复杂度为 $O(n)$
- 平衡二叉查找树
 - 左子节点 \leq 右子节点
 - 左右均为平衡二叉树
 - 左右高度相差不超过1
 - 最坏的搜索复杂度为 $O(\lg n)$

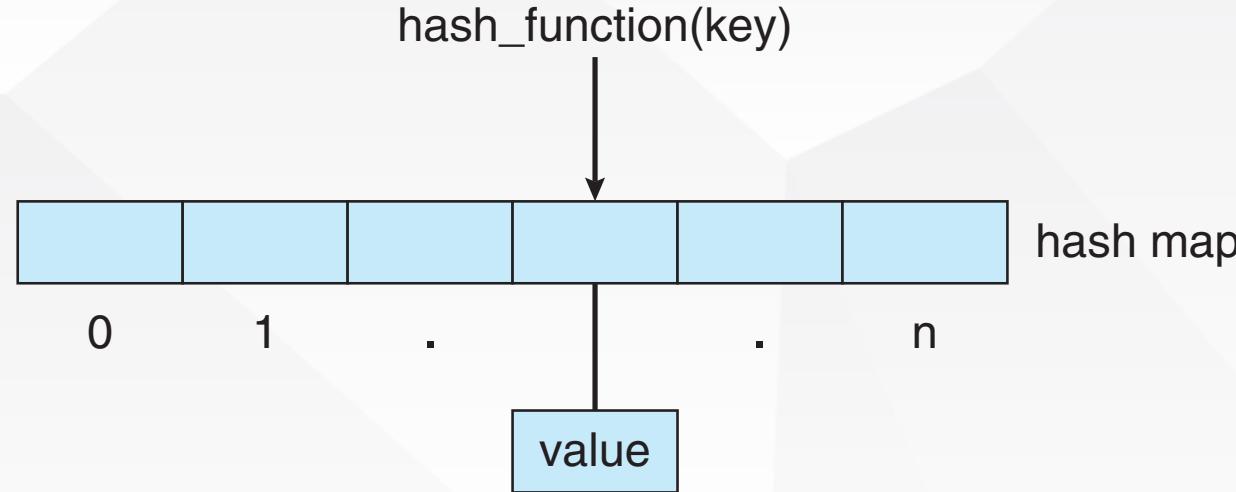




内核数据结构



- 哈希函数，以一个数据作为输入，进行运算，返回另一个数据



- 搜索复杂度可能只用 $O(1)$
- 哈希碰撞：两个输入可能产生同样的输出





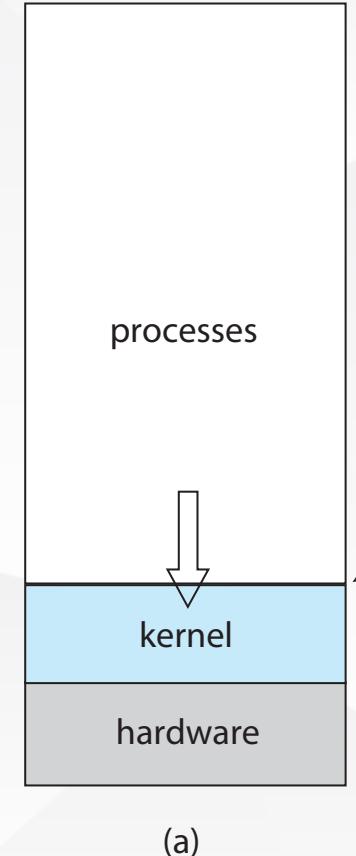
内核数据结构

- 位图：n个二进制位的串，表示n项的状态
 - 0表示资源可用
 - 1表示资源不可用
- 001011101：第2、4、5、6、8资源不可用，其他可用
- 应用范围：管理大量的磁盘块

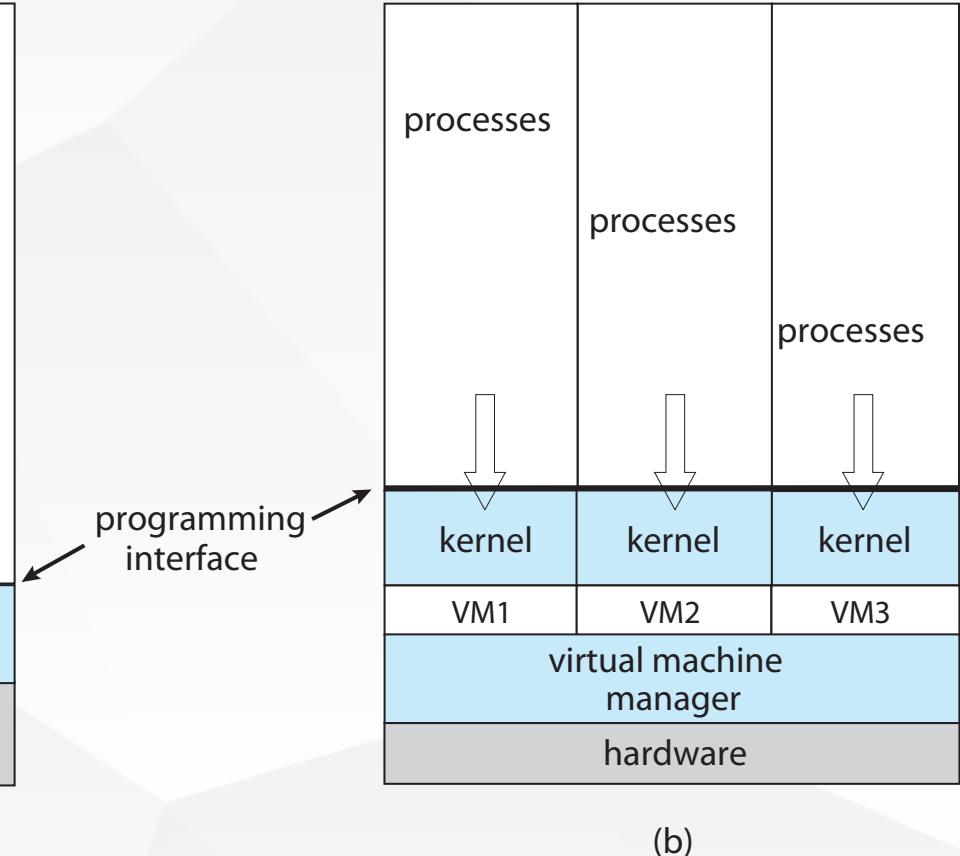




- 虚拟化技术：使操作系统可以在其他操作系统上运行应用程序
 - 以支持不同类型的CPU
 - 如苹果电脑换成Intel x86 CPU，利用Rosetta软件，允许为IBM CPU编译的应用程序运行在Intel CPU上
 - Vmware：运行一个或多个Window系统，其中一个Windows为主机操作系统



(a)



(b)



课堂习题

1. 回顾前面的Hello world程序，操作系统的作用有哪些？
2. 请描述操作系统中多进程的作用是什么？

