

操作系统

Operating Systems

华东师范大学
计算机科学与技术学院
石亮

课程目录

- 第1章 引论
- 第2章 进程与线程
- 第3章 存储管理
- 第4章 文件系统
- 第5章 输入/输出
- 第6章 死锁

第1章 引论

- 课程相关信息
- 何为操作系统？它的作用是什么？
- 操作系统的发展历史
- 操作系统大观园
- 计算机硬件概述
- Unix的简单剖析
- 评价操作系统优劣的标准
- 操作系统组成部件

课程的必要性

- 本科培养计划的要求
- 其他专业课的先导课程
 - ↳ 计算机网络
 - ↳ 计算机安全
 - ↳ 分布式系统
 - ↳ 实时系统
 - ↳ 多媒体系统
- 未来规划的需要
 - ↳ 计算机专业研究生入学必考科目
 - ↳ 毕业后工作需要

关于这门课程…

原理

- 操作系统概念
- 操作系统设计
- 一些相关的理论
- 基本原理
- 编程与实践

目标

- 理解操作系统的原理与设计
- 为今后学习与研究打下基础
- 亲自参与实践环节

课程考核

- 随堂作业: 10%
- 期中考试: 10%
- 实验部分: 20%
- 期末考试: 60%



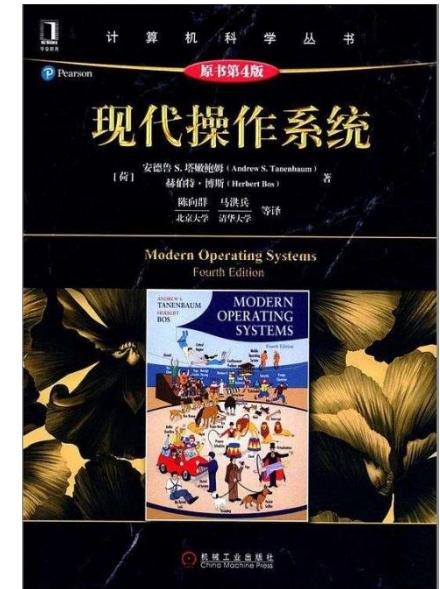
群聊: 2024操作系统-石亮-拔尖



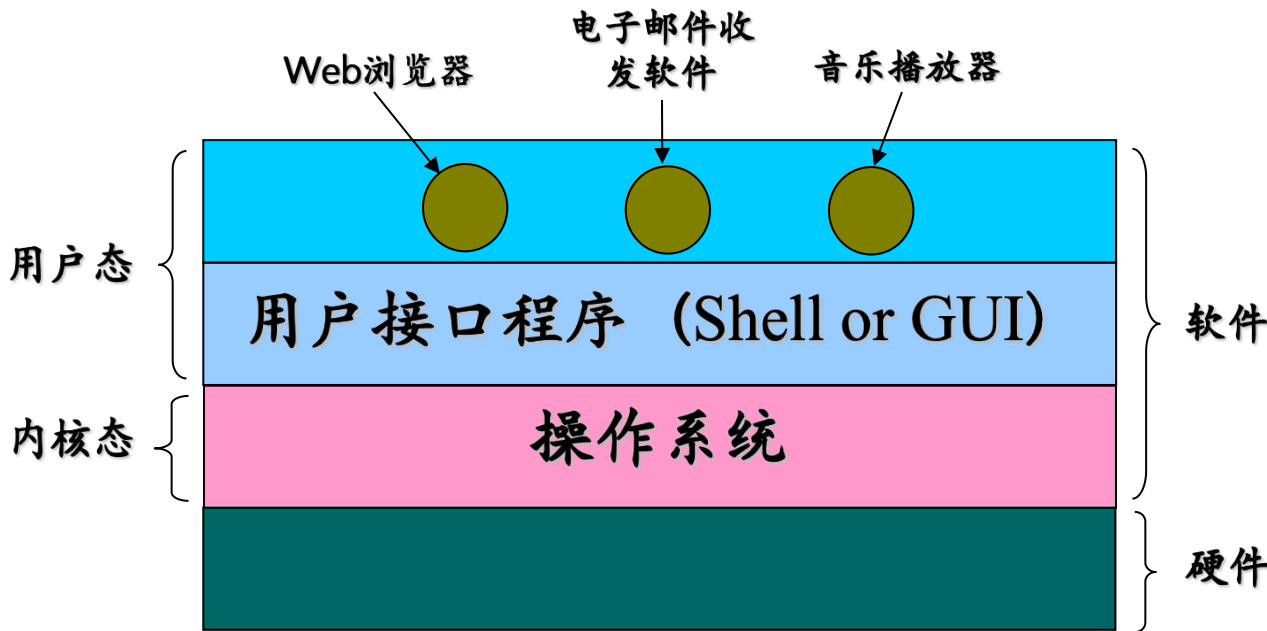
该二维码7天内(9月12日前)有效，重新进入将更新

参考教材

- 现代操作系统（原书第4版）作者：[荷] Andrew S. Tanenbaum / [荷] Herbert Bos 出版社：机械工业出版社 原作名：Modern Operating Systems (4th Edition) 译者：陈向群 / 马洪兵 等
- 其他参考教材：
- Operating Systems: Three Easy Pieces 操作系统：三大简易元素
- Operating System Concepts 操作系统概念
- 操作系统：原理与实现，陈海波 夏虞斌，机械工业出版社



何为操作系统?



操作系统所处的位置

何为操作系统?

是代码，位于：

- 程序与硬件之间
- 不同的程序之间
- 不同的用户之间

操作系统 (OS) 的作用：

在相互竞争资源的程序之间，提供对处理器、存储器、IO设备等资源有序的控制、管理与分配。

现实生活中的：

↳ 政府？

何为操作系统?

资源 (资源管理者)

- 资源分配
- 资源保护
- 资源回收
- 虚拟

服务 (裸机的扩展)

- 抽象
- 简化
- 方便
- 标准化

OS使计算机使用更加简单，功能更强！

何为操作系统?

政府

资源 (资源管理者)

- 资源分配
- 资源保护
- 资源回收
- 虚拟

有限的资源
竞争的需求

例子：

- CPU
- Memory
- Disk
- Network

有限的预算,
土地,
石油,
天然气,

Linux or Windows? ← → 民主党 or 共和党?

何为操作系统?

资源 (资源管理者)

- 资源分配
- 资源保护
- 资源回收
- 虚拟

你不能干扰 (伤害) 我,
我不能干扰 (伤害) 你。

在某种程度上意味着安全

政府

法律与次序

何为操作系统?

资源 (资源管理者)

- 资源分配
- 资源保护
- 资源回收
- 虚拟

一切资源：
由OS分配
由OS回收

自愿回收：不需要资源时
抢占式回收：基于合作

政府

收入税

何为操作系统?

资源 (资源管理者)

- 资源分配
- 资源保护
- 资源回收
- 虚拟

OS构造了一个无限多的、私有资源的“假象”。

内存 versus 磁盘
分时使用CPU

更多更优的可能性。

政府

社会保障

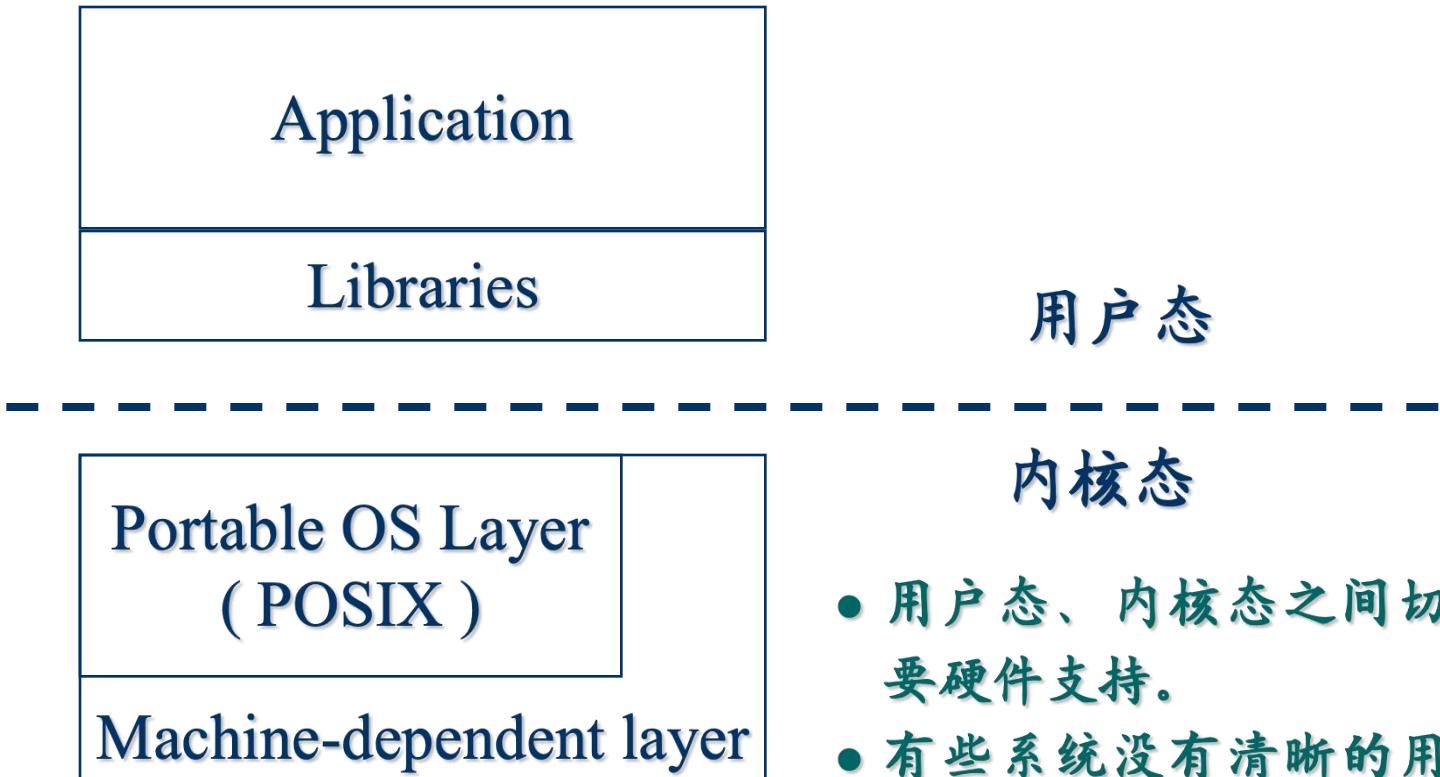
评价OS优劣的标准

- 可移植性(Portability)
- 安全性(Security)
- 公平性(Fairness)
- 稳定性(Robustness)
- 资源利用效率(Efficiency)
- 接口友好性(Interfaces)

政府

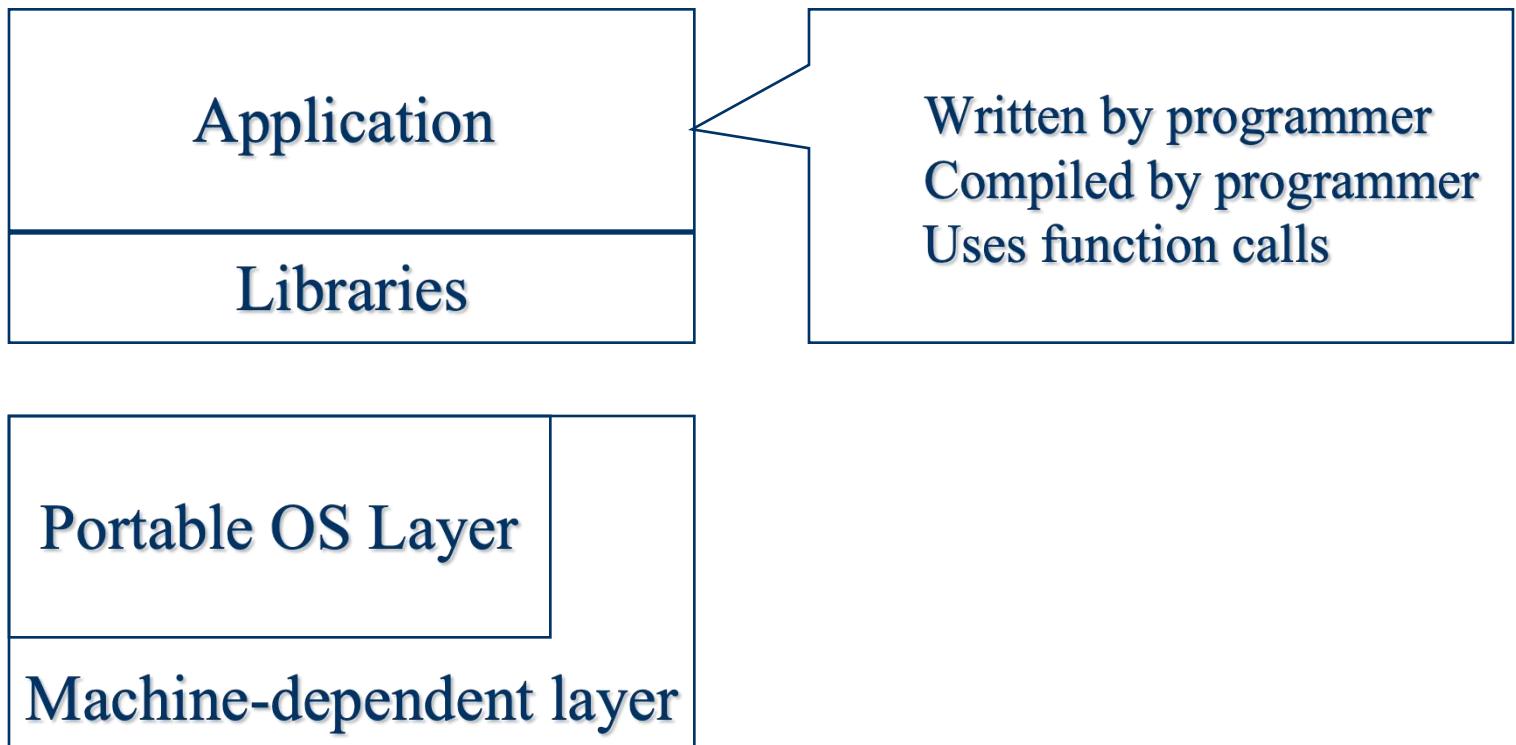
- 不受地域限制
- 国土安全
- 公平、公正、正义
- 反战争和恐怖主义
- 消除官僚、腐败现象!
- 提供优良的服务

Unix 简单剖析

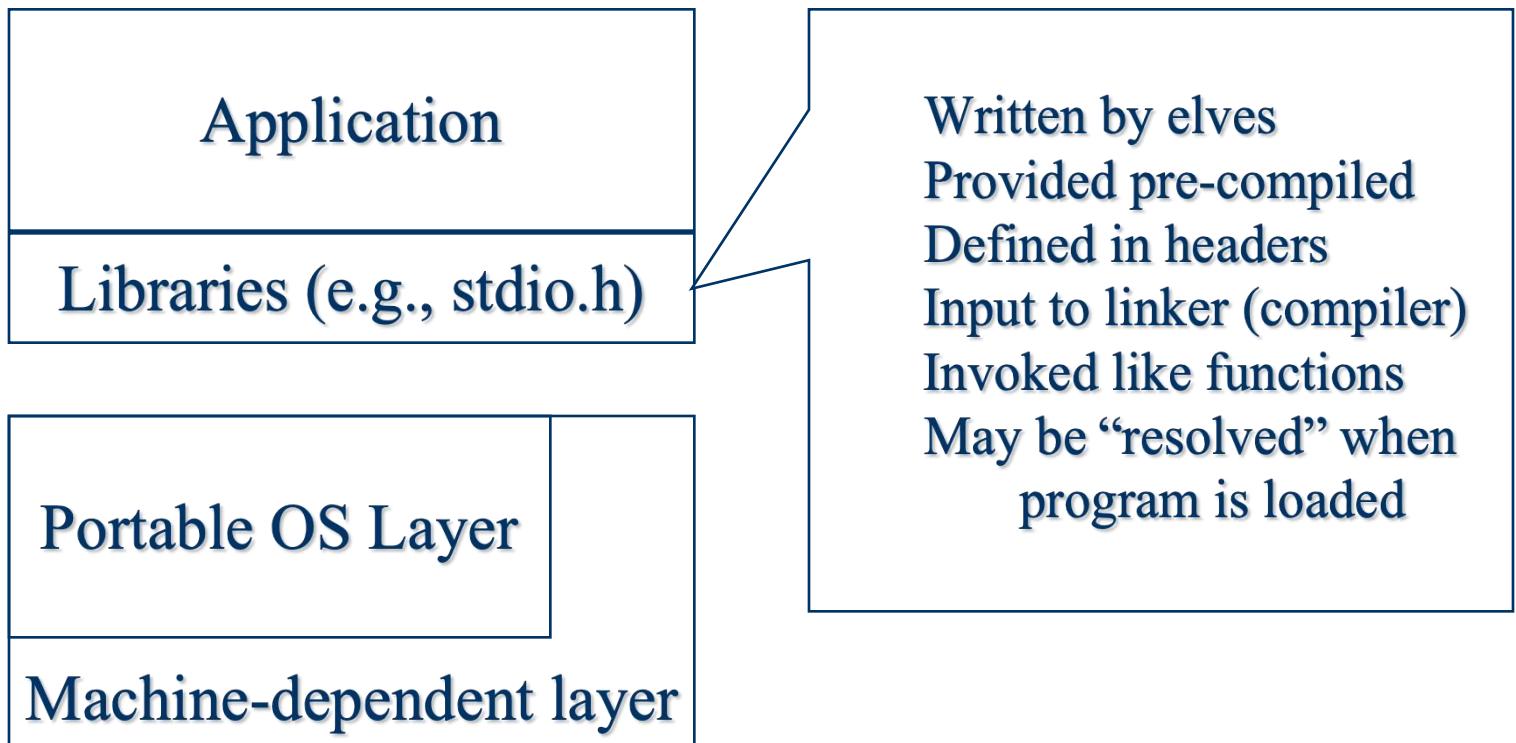


- 用户态、内核态之间切换需要硬件支持。
- 有些系统没有清晰的用户态与内核态之间的边界。

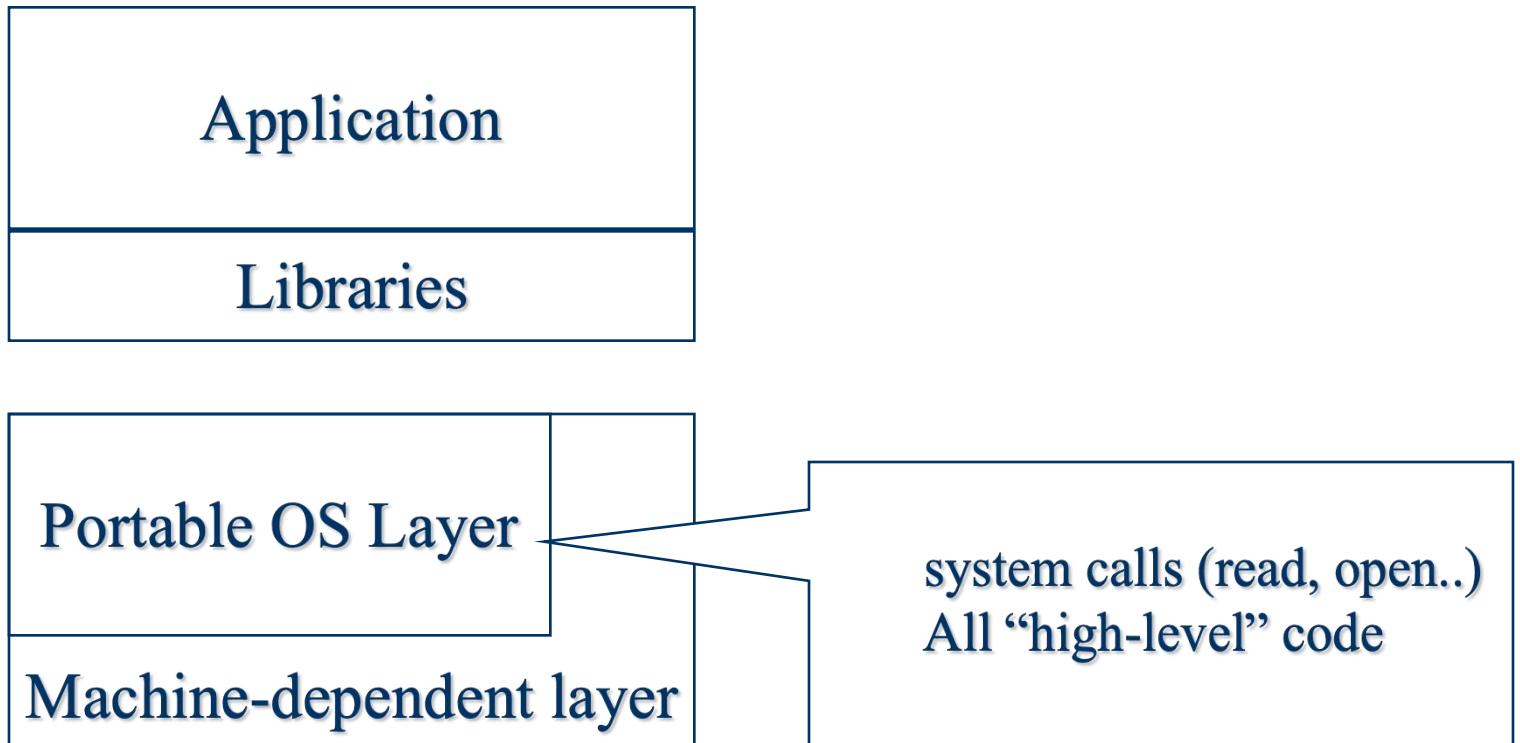
Unix: Application



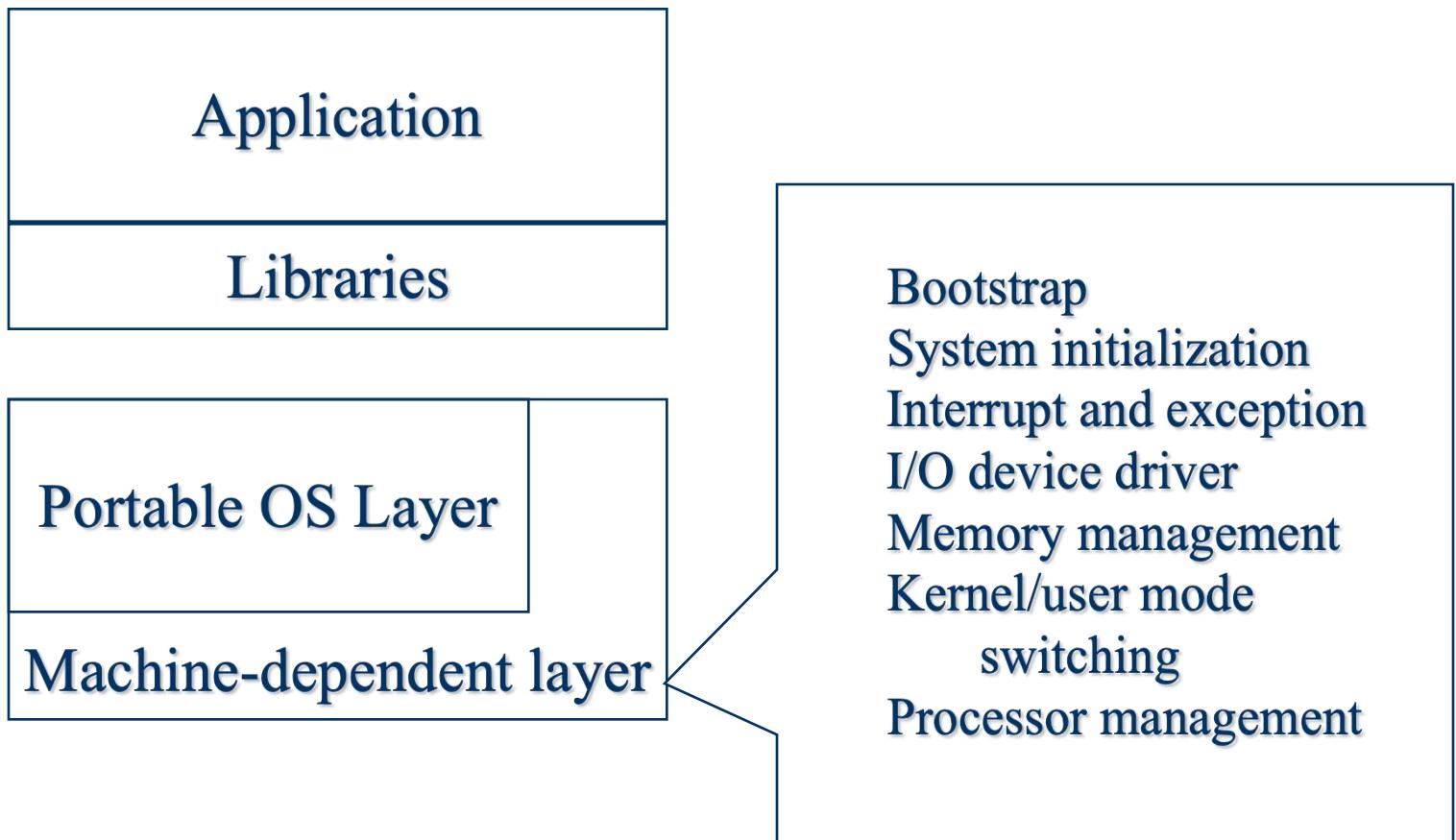
Unix: Libraries



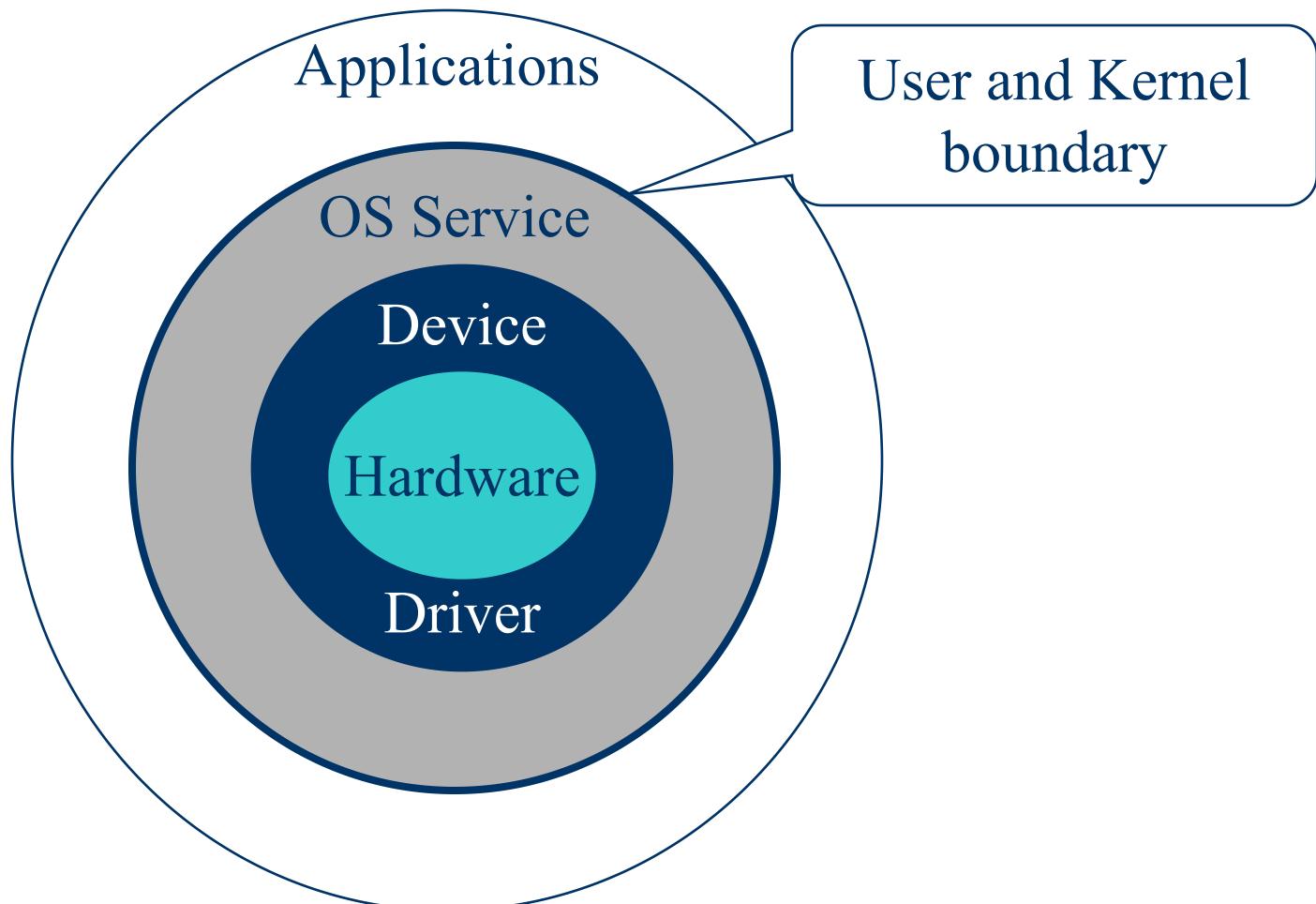
典型Unix OS结构



典型Unix OS结构



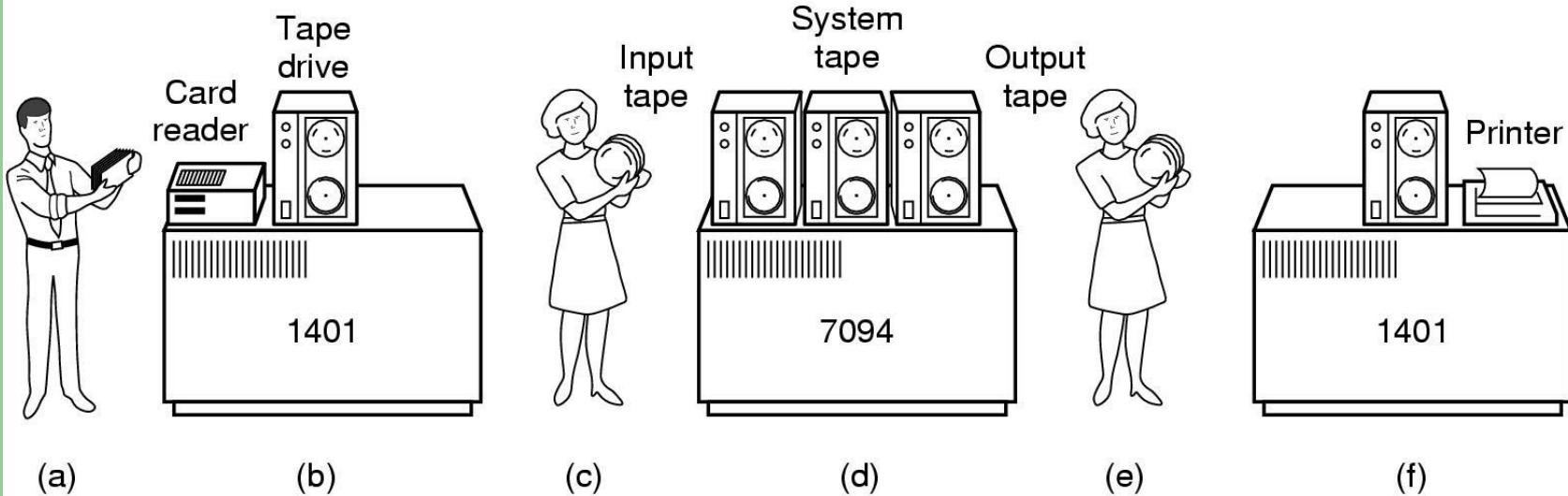
另一个视角：Unix 的“洋葱”结构



操作系统发展历史

- 第一代 1945 – 1955
 - └ 真空管, 穿孔卡片, 插件板
- 第二代 1955 – 1965
 - └ 晶体管, 批处理系统(Batch Systems)
- 第三代 1965 – 1980
 - └ 集成电路芯片, 中断、通道机制
 - └ 多道程序设计 (Multiprogramming) 环境
 - └ 多道批处理系统 (Multiprogramming Batch System)
 - └ 多道分时系统 (Multiprogramming Time-sharing Systems)
- 第四代 1980 – 现在
 - └ 个人计算机
 - └ Windows/Unix/Linux

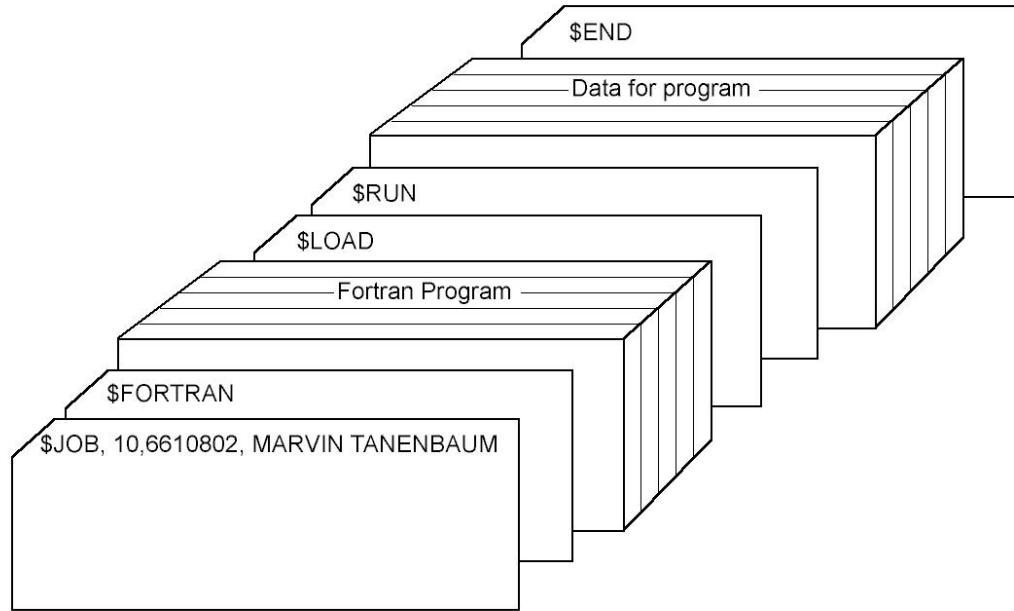
操作系统发展历史



早期的批处理系统(脱机输入输出技术):

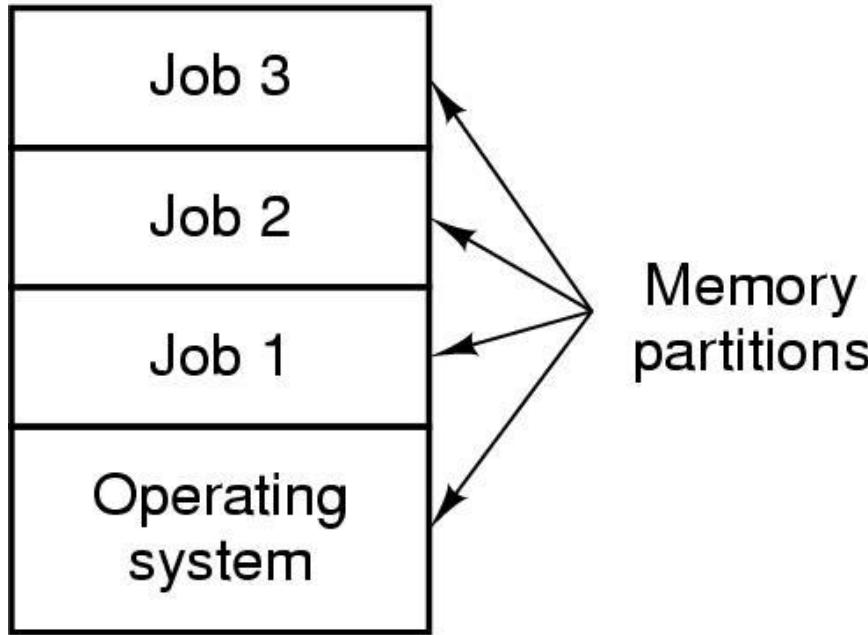
- └ 程序员将卡片拿到1401机处
- └ 1401机将批处理作业读到磁带上
- └ 操作员将输入带送至7094机
- └ 7094机进行计算，并将结果输出到输出带中
- └ 操作员将输出带送到1401机
- └ 1401机打印输出

操作系统发展历史



典型的FMS工作结构 - 第二代(JCL)

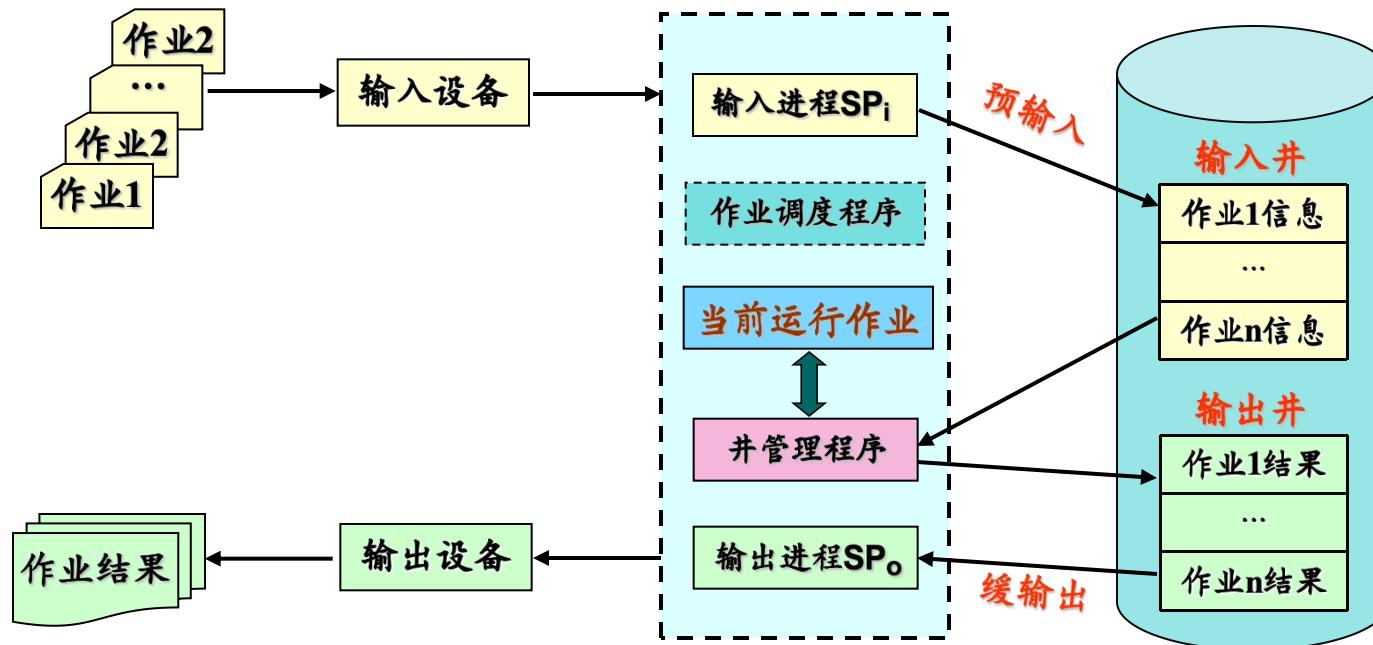
操作系统发展历史



多道程序系统

- └ 3个作业在内存中 - 第三代
- └ Spooling技术 (假脱机技术)

Spooling技术



SPOOLing系统的组成

Spooling技术

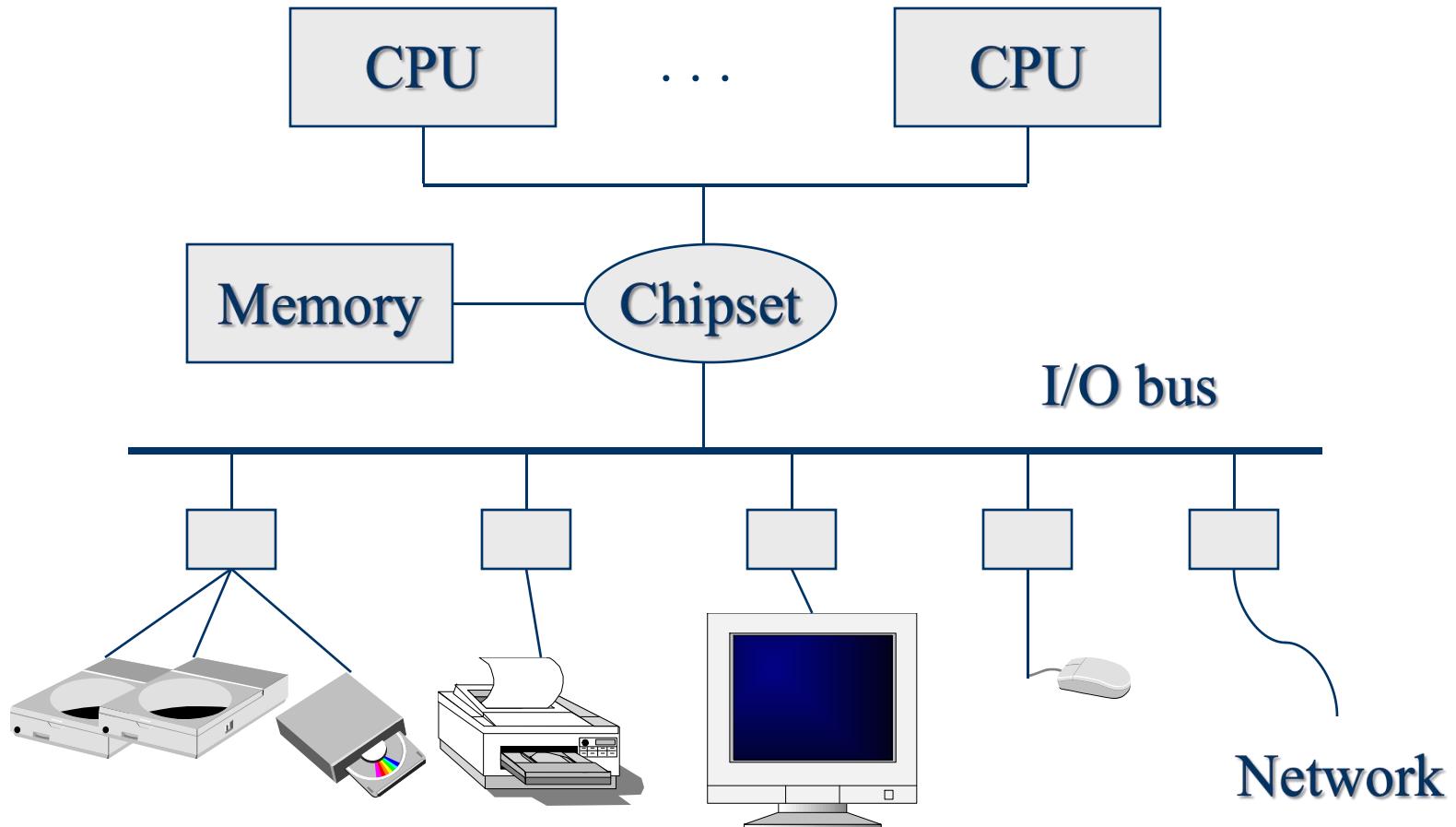
SPOOLing技术 (Simultaneous Peripheral Operation On-Line) ,

即为联机同时外围设备操作技术，也可称为假脱机技术。

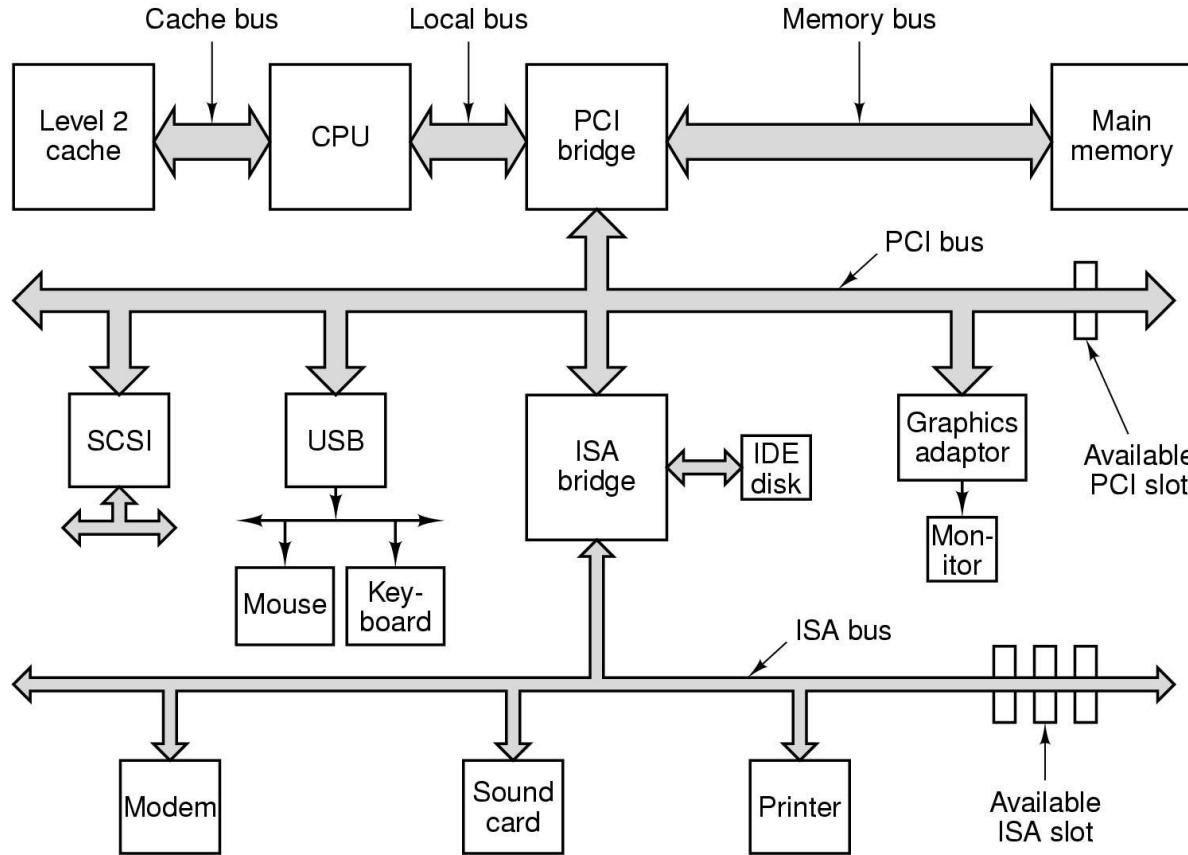
它的实现思想是：利用中央处理器和通道并行工作的能力，在多任务系统中分别以一个**输入进程**和一个**输出进程**代替脱机技术中的输入外围机和输出外围机，这样用一台机器完成脱机外围设备操作技术中三台机器的工作。SPOOLing系统在磁盘中划分出专门称为“井”的区域，它分为**输入井**和**输出井**。输入进程把作业流中作业信息**预输入**到输入井保存，作业在执行时只要通过井管理程序从输入井中获取数据，而不去启动低速的外围设备。作业执行时产生的结果也不直接输出到低速外设上，而是先通过井管理程序先输出到输出井，由输出进程将输出井中的数据**缓输出**到低速设备上。

应用SPOOLing技术能实现**虚拟设备**，即将需要互斥访问的临界资源虚拟成可以同时访问的共享资源，提高了资源的利用率，改善了用户体验。基于**打印队列的网络打印技术**就是利用了SPOOLing技术。

从硬件角度看计算机系统

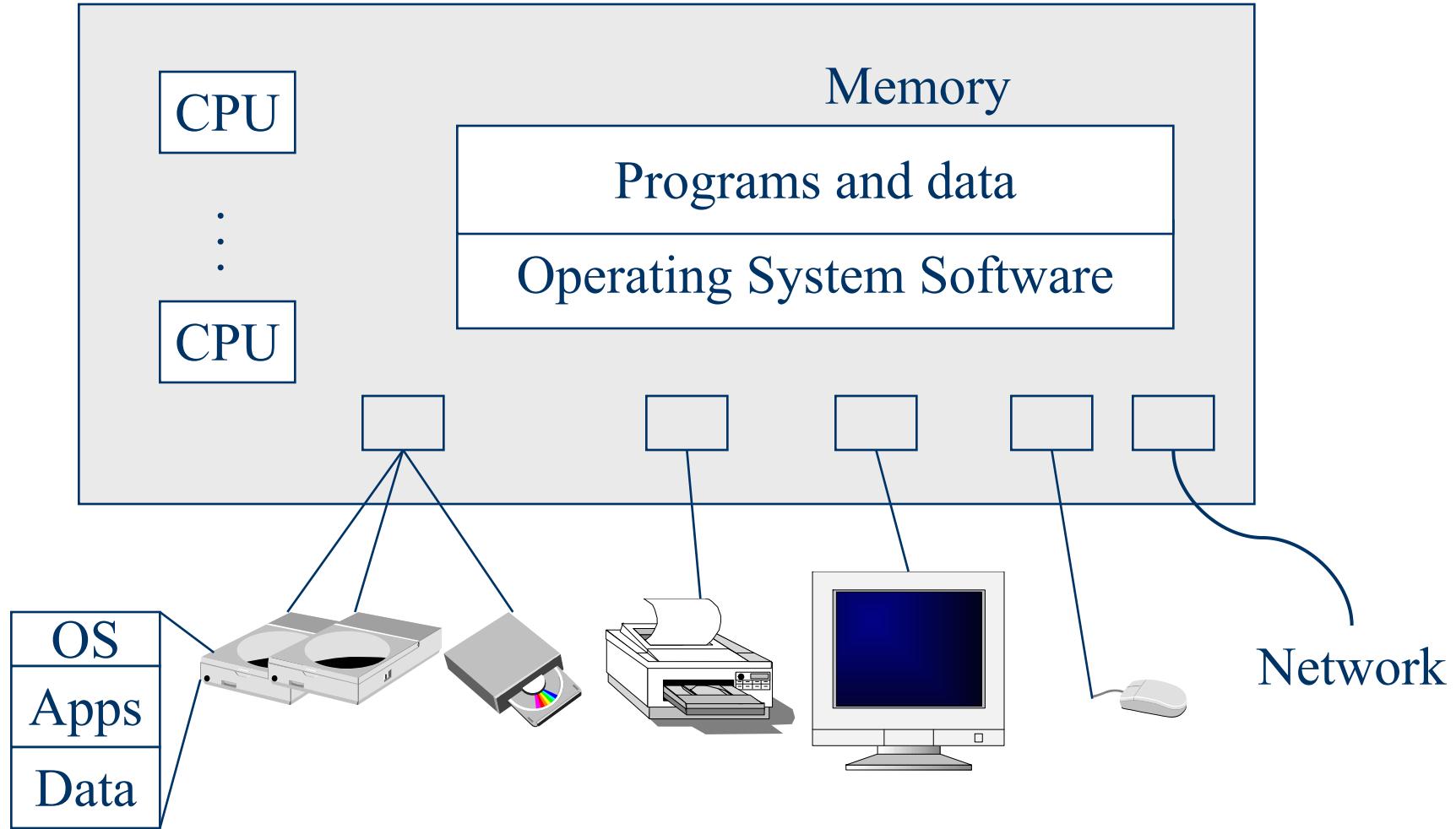


Pentium计算机系统



大型Pentium系统的结构

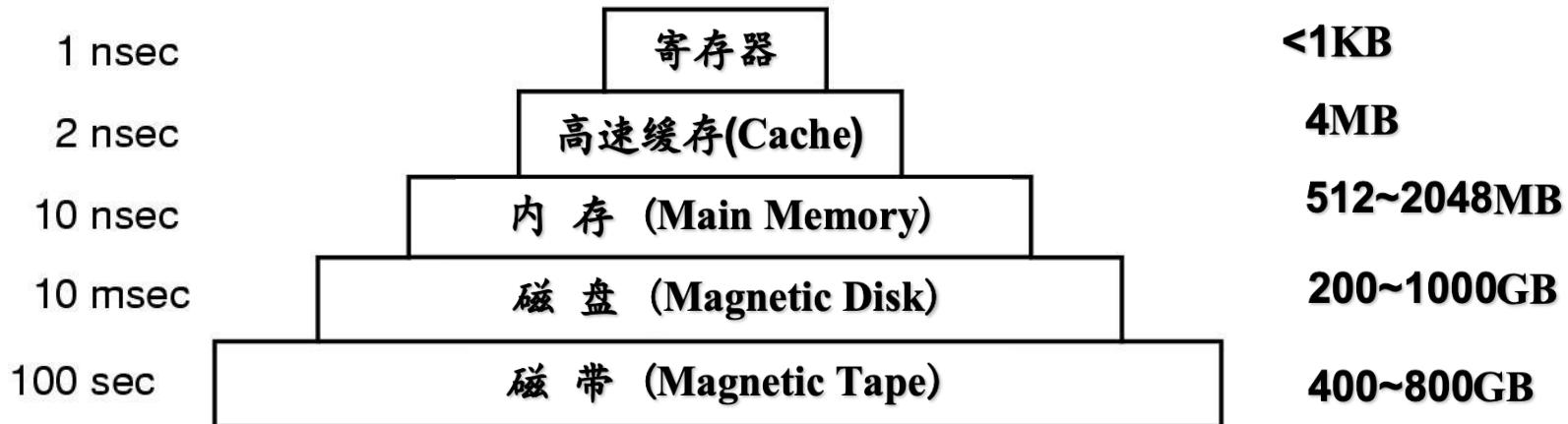
一个典型的计算机系统(黑盒)



存储层次结构(Memory-Storage Hierarchy)

典型的访问时间

典型的容量



操作系统大观园(1)

- 大型机操作系统(Mainframe Operating Systems, OS/390、Unix、Linux)
- 服务器操作系统(Server Operating Systems, Solaris、Unix、Linux、Windows Server)
- 多处理器操作系统(Multiprocessor Operating Systems, Windows、Linux)
- 个人计算机操作系统(Personal Computer Operating Systems)
- 实时操作系统(Real-time Operating Systems , VxWorks 、 e-Cos、 RTLinux)
- 掌上计算机操作系统(PDA OS, Android、iOS、Windows Phone、BlackBerry OS和Symbian)
- 嵌入式操作系统(Embedded Operating Systems, VxWorks 、 QNX、 RTLinux、 Windows CE)
- 传感器节点操作系统(Tiny OS)
- 智能卡操作系统(Smart Card Operating Systems)

操作系统大观园(2)

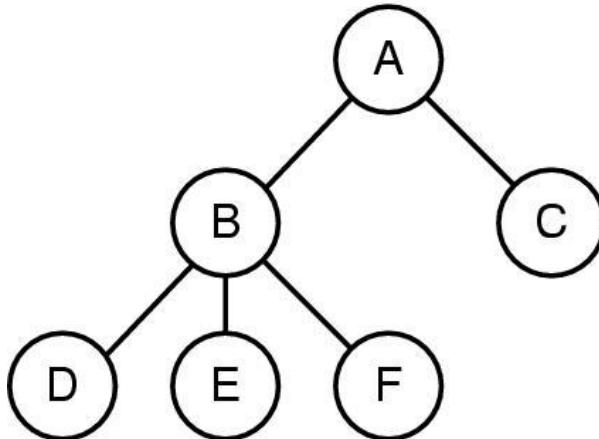
- 多道批处理系统(Multiprogramming Batch Operating Systems)
- 分时操作系统(Time-sharing Operating Systems)
- 实时操作系统(Real-time Operating Systems)
- 通用操作系统(General Operating Systems)
- 网络操作系统(Network Operating Systems)

OS的主要组成

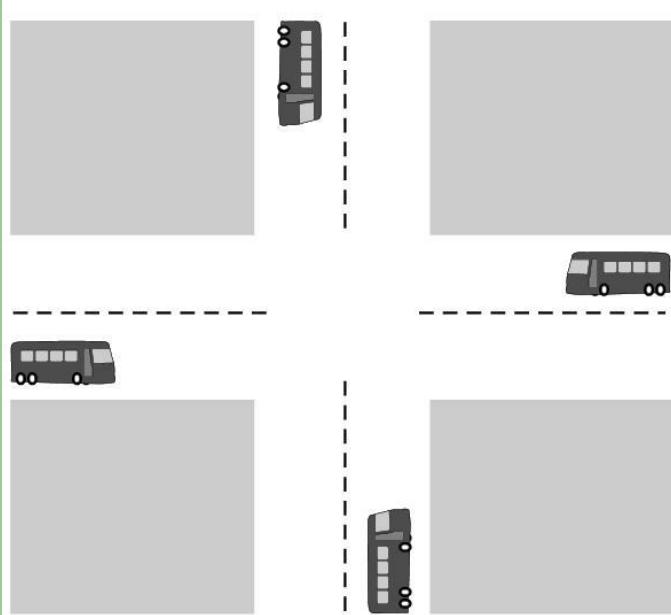
- 进程和线程(Process and Thread Management)
- 资源管理(Resource Management)
 - └ 处理器(CPU)
 - └ 存储器(Memory)
 - └ 设备(Device)
- 文件系统(File system)
- 系统自举(Bootstrapping)

进程: 一个运行中的程序

- 一个进程(Process)包括:
 - └ 内存地址空间(Address Space)
 - └ 进程表表项(PTE, Process Table Entry)
- 进程树(Process Tree)
 - └ 进程A创建了两个子进程, 进程B和进程C。
 - └ 进程B创建了三个子进程, 进程D、进程E和进程F。

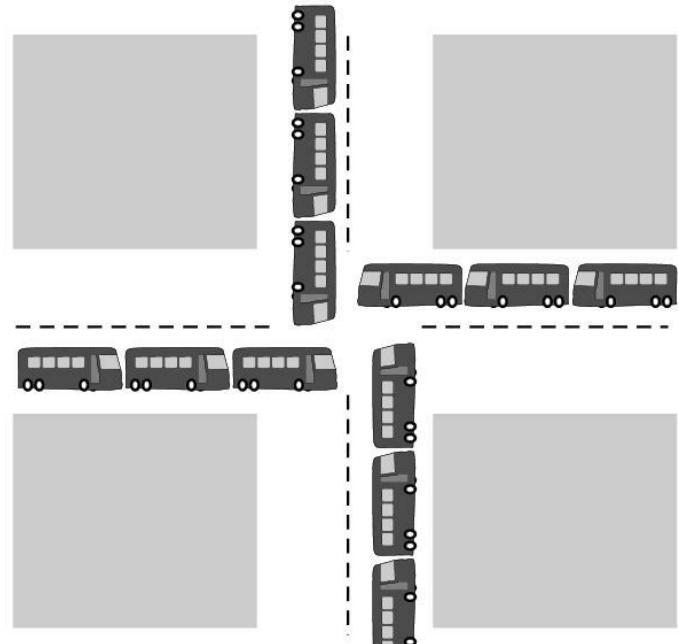


死锁(Deadlock)



(a)

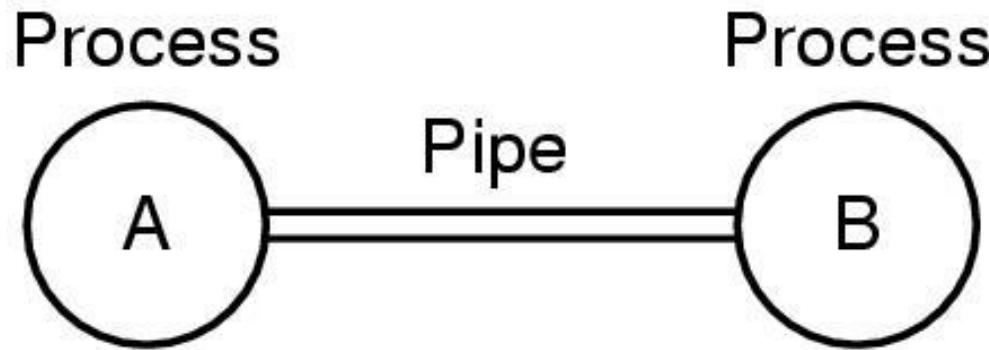
(a)一个潜在死锁



(b)

(b)一个实际死锁

进程通信(Process Communication)



通过管道 (Pipe) 进行通信的两个进程

处理器管理

- 目标

- └ 分时共享(Time Sharing)
- └ 多处理器分配(Multiple CPU Allocations)

- 需解决的问题

- └ 提高CPU使用效率
- └ 进程互斥与同步(Synchronization and Mutual Exclusion)
- └ 公平性(Fairness)
- └ 避免死锁(Deadlock Free)



Analogy: Video Games

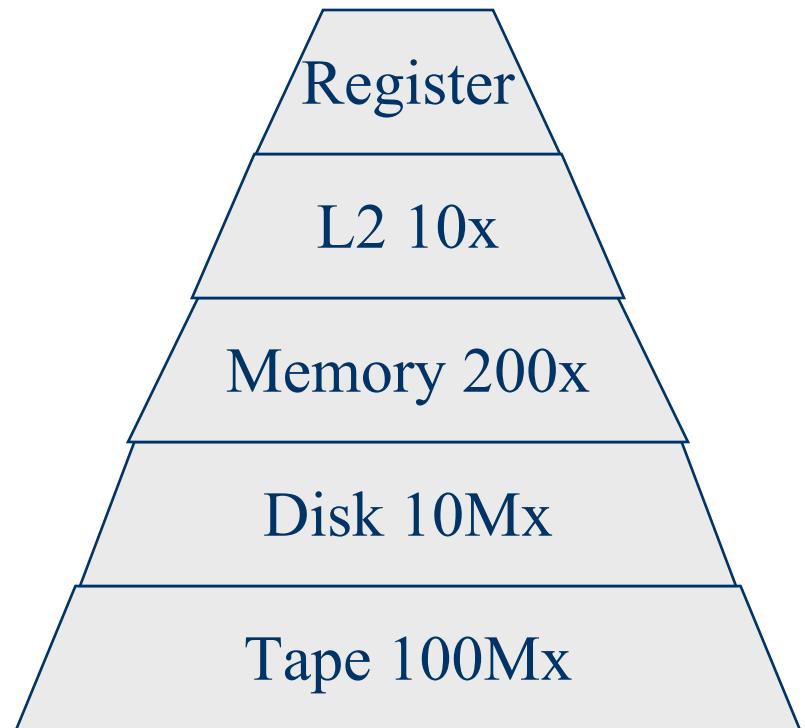
存储器管理

- **目标**

- └ 支持多道程序同时运行
- └ 存储空间的分配与管理
- └ 支持虚拟存储技术

- **需解决的问题**

- └ 提高内存使用效率，方便用户使用。(可能是一对矛盾)
- └ 公平性
- └ 运行环境的保护



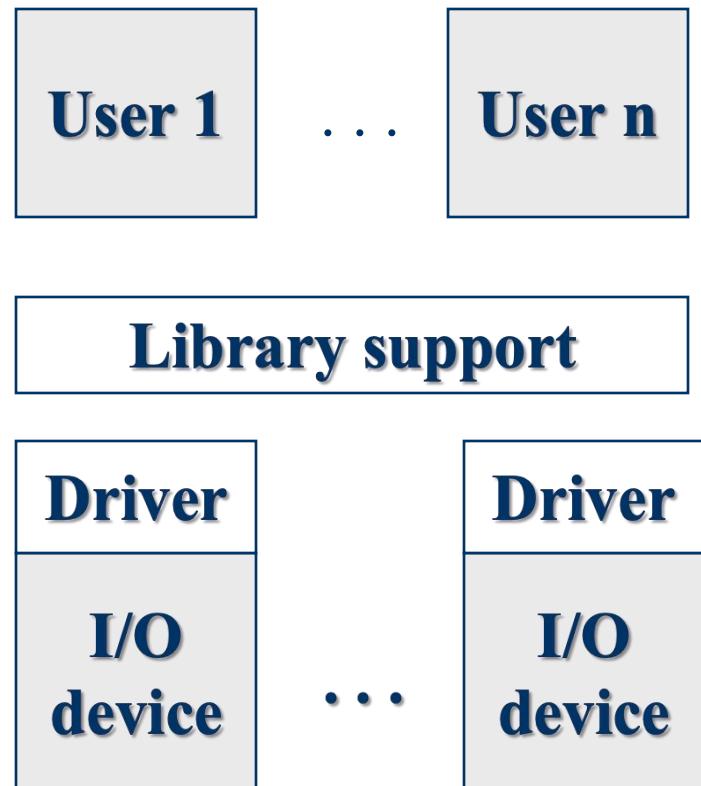
I/O设备管理

- **目标**

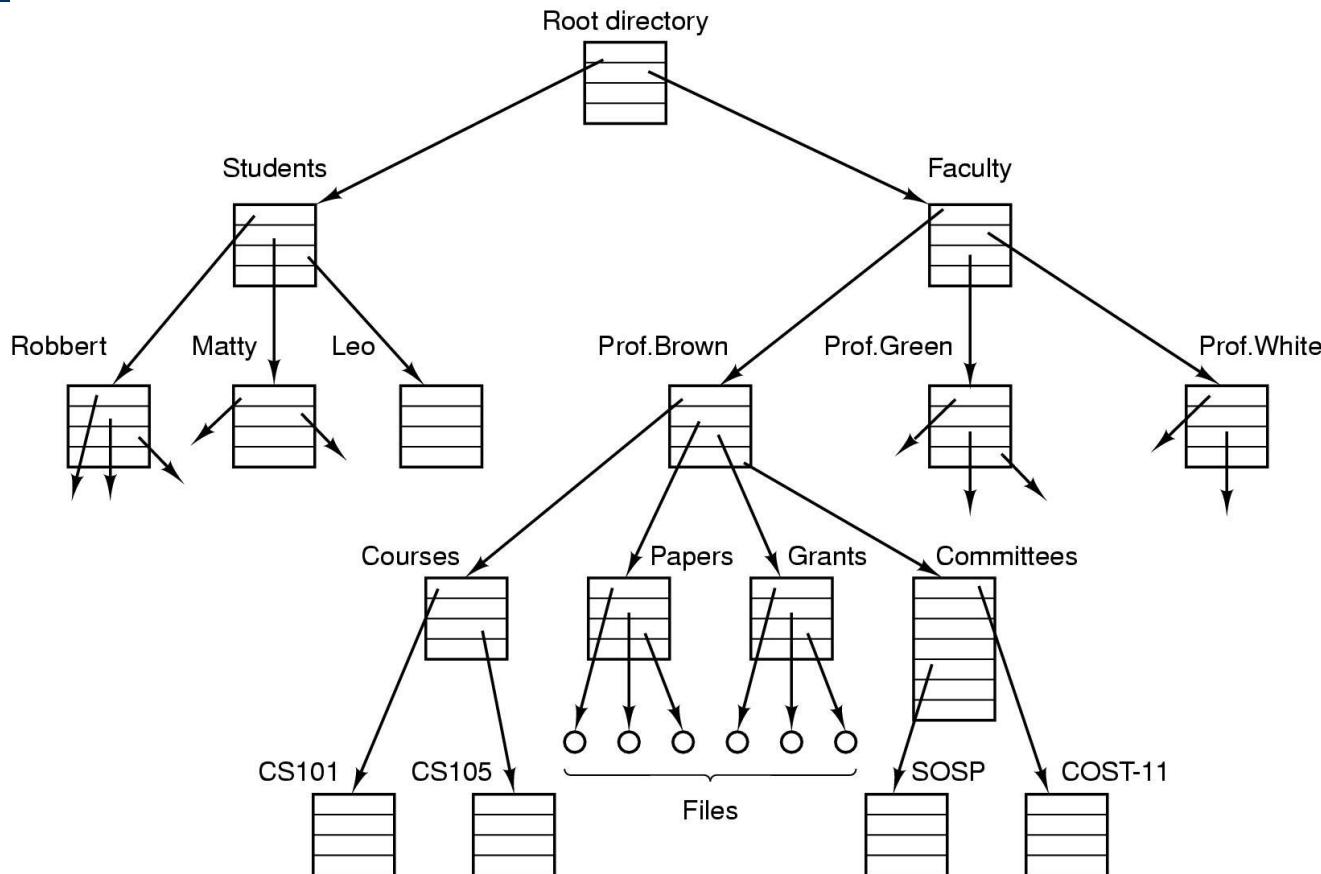
- 『 提供应用程序与设备之间
的访问接口
- 『 接纳新设备的能力

- **需解决的问题**

- 『 提高设备使用效率
- 『 公平性
- 『 共享与保护



文件系统



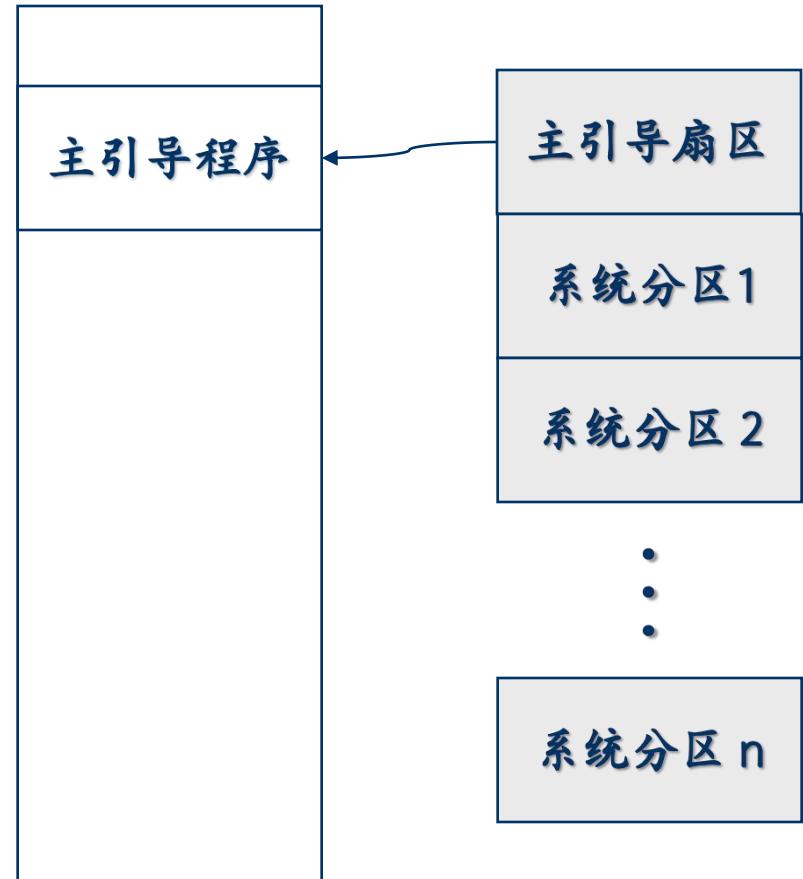
文件系统

- 基本功能
 - ↳ 按名存取文件
 - ↳ 访问权限的控制
 - ↳ 读写文件
- 该服务能否在用户态下实现?



系统自举

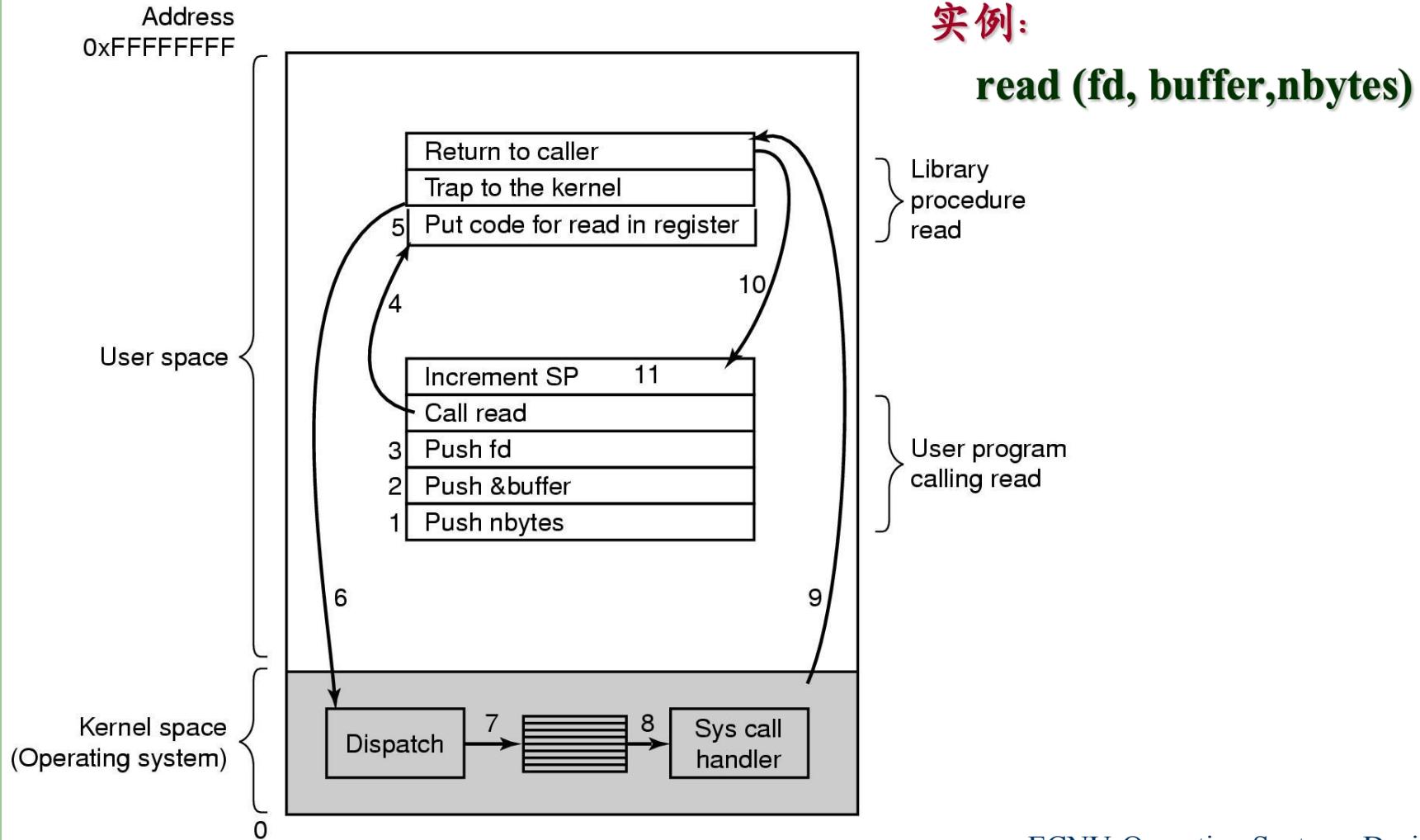
- 计算机系统加电
- 加载BIOS
- 加载执行主引导程序，
读分区表
- 执行活动分区的操作系
统引导程序
- 加载操作系统
- 系统初始化



概念的循环重用(Recycling of Concepts)

- 文件 – 连续分配方式(Contiguous Allocation)
 - ↳ CD-ROM文件系统
 - ↳ 虚拟存储系统(Virtual Memory) – 外存对换区(Swap Space)
- 动态链接(Dynamic linking)
 - ↳ MULTICS: 为了实现在系统运行时替换一些库过程，采用了动态链接技术。
 - ↳ 现代OS: 允许多个程序去共享同一个库过程，不需要产生多个私有副本，也可以修改一些库过程而不影响系统。
- An idea that is obsolete today may be the star of the party tomorrow!

系统功能调用(System Call)的过程



进程管理的系统调用

Process management

| Call | Description |
|---------------------------------------|------------------------------------------------|
| pid = fork() | Create a child process identical to the parent |
| pid = waitpid(pid, &statloc, options) | Wait for a child to terminate |
| s = execve(name, argv, environp) | Replace a process' core image |
| exit(status) | Terminate process execution and return status |

一个shell程序框架：

```
while (TRUE) {
    type_prompt();
    read_command (command, parameters)

    if (fork() != 0) {
        /* Parent code */
        waitpid( -1, &status, 0);}
    else {
        /* Child code */
        execve (command, parameters, 0);}
}
```

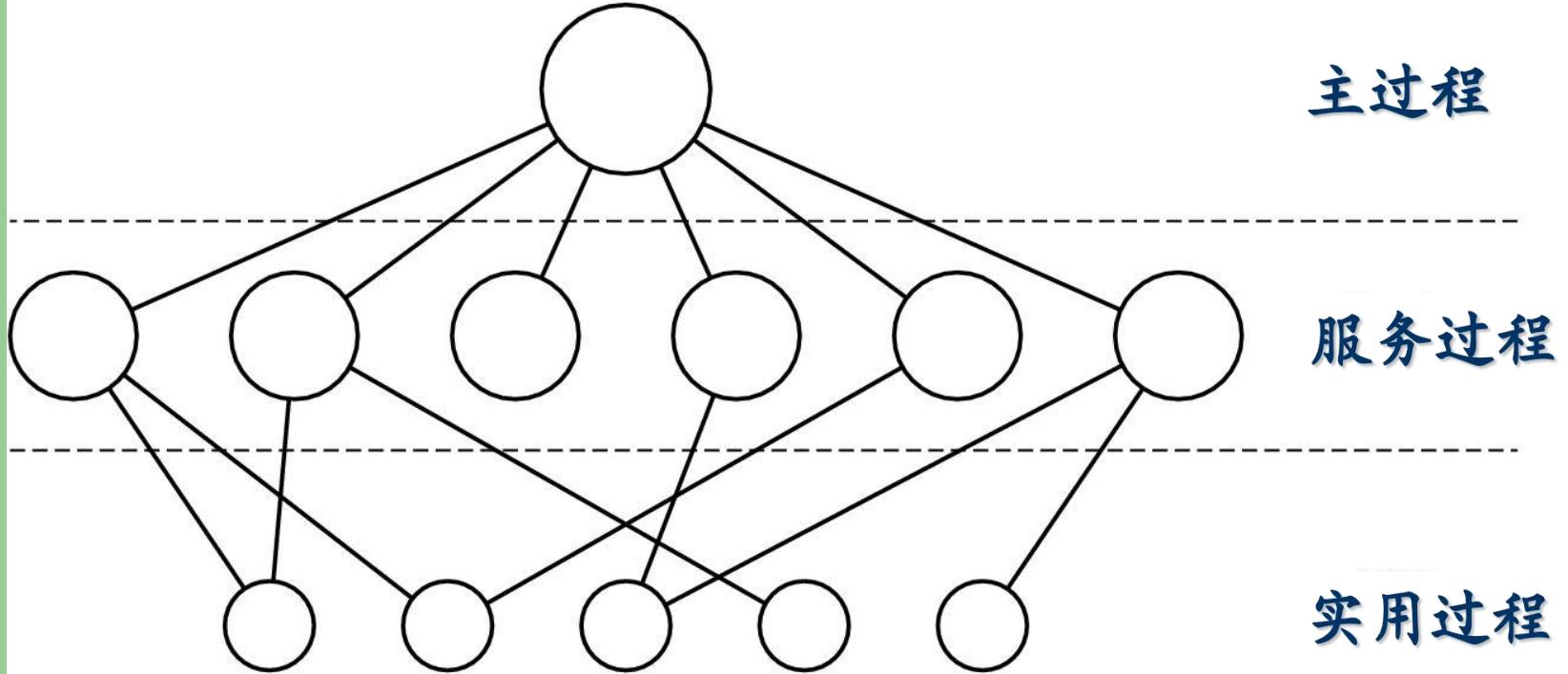
```
/* repeat forever */
/* display prompt */
/* input from terminal */

/* fork off child process */
/* wait for child to exit */
/* execute command */
```

中断与异常

- 早期的计算机，各程序只能串行执行，系统资源利用率低，为了解决这个问题，就发明了操作系统（作为计算机的管理者），引入了中断机制，实现了多道程序并发执行
- 概念
 - ① 当中断发生之后，CPU立即进入核心态
 - ② 当前运行的进程暂停运行，并由操作系统内核对中断进行处理
 - ③ 对于不同的中断信号，会进行不同的处理
- 发生了中断，就意味着需要操作系统介入，开展管理工作，由于操作系统的管理工作（比如进程切换，分配I/O设备等）需要使用特权指令，因此CPU需要从用户态切换到核心态。中断可以使CPU从用户态切换为核心态，使操作系统获得计算机的权限。有了中断，才能实现多道程序并发执行

操作系统结构



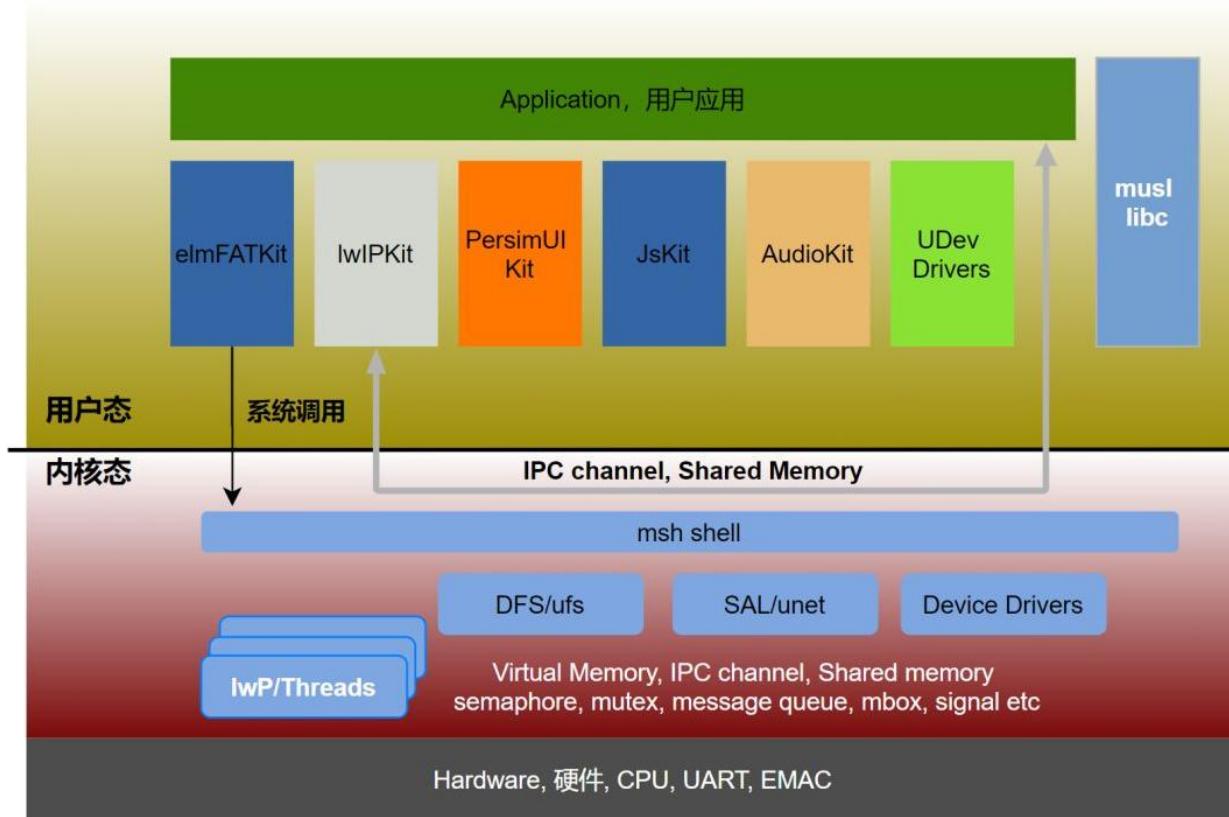
简单的单体结构模型

操作系统结构

| 层号 | 功能 |
|----|--------------|
| 5 | 操作员 |
| 4 | 用户程序 |
| 3 | 输入/输出管理 |
| 2 | 操作员-进程通信 |
| 1 | 存储器和磁鼓管理 |
| 0 | 处理器分配和多道程序环境 |

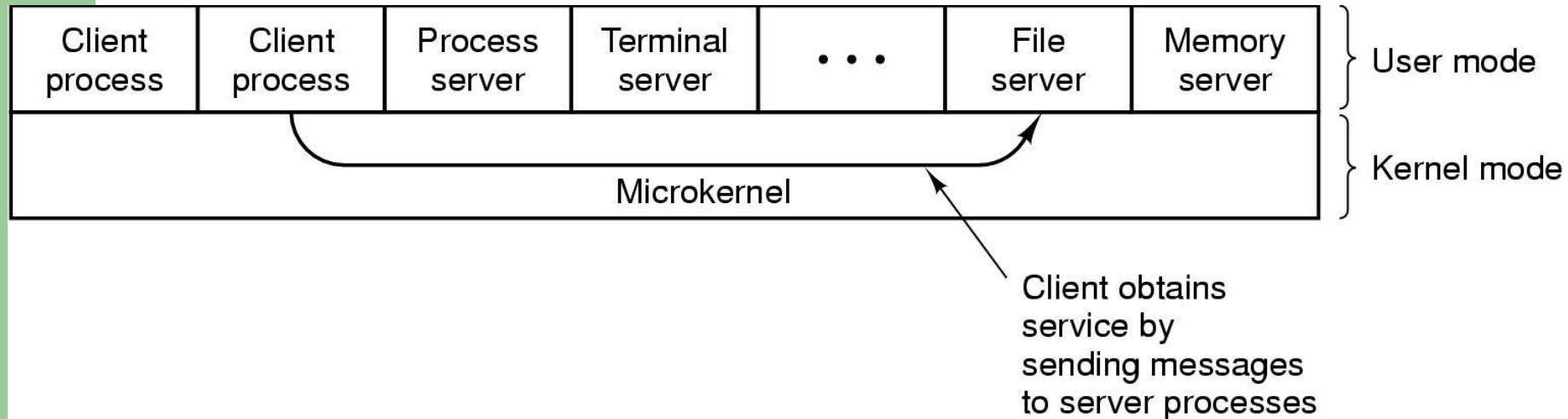
THE操作系统的结构(层次式系统)

操作系统结构



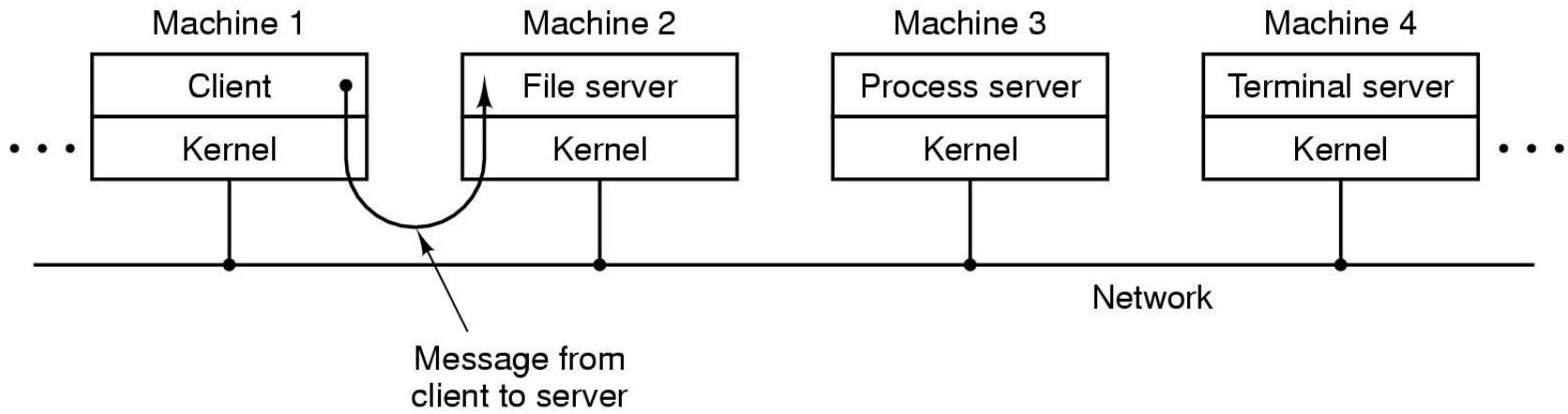
微内核设计

操作系统结构



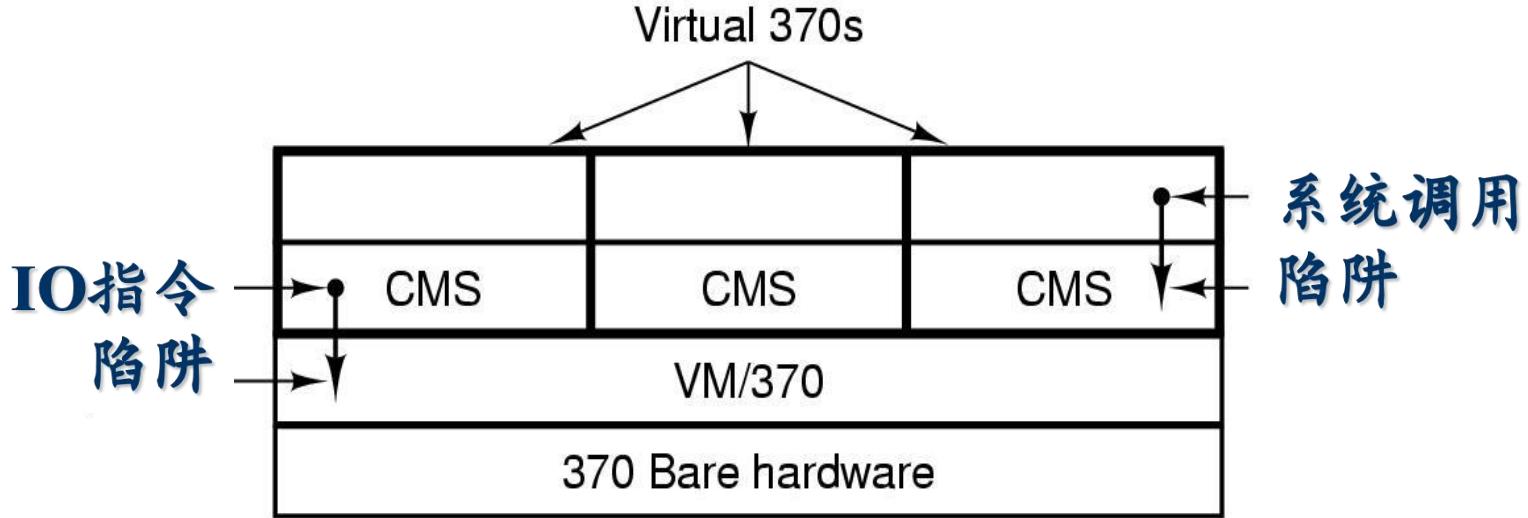
客户-服务器模式

操作系统结构



在分布式系统上的客户-服务器模型

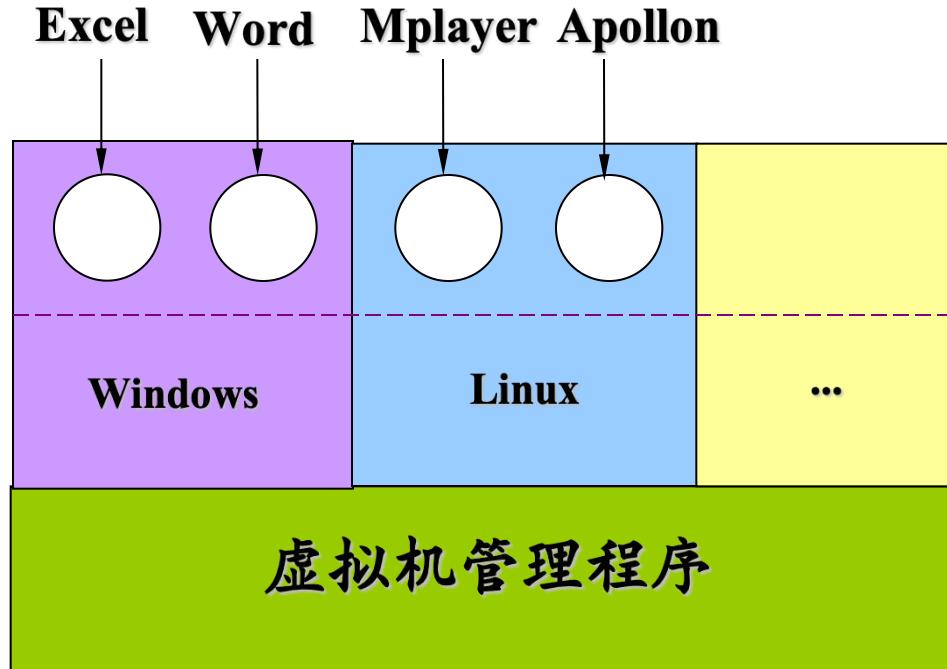
操作系统结构



配有CMS的VM/370结构(虚拟机)

- 可向上层提供若干个虚拟机
- 每个虚拟机是裸机硬件的精确复制
- 可同时运行多个批处理操作系统和分时操作系统

操作系统结构



现代硬件虚拟化技术