



操作系统原理

南京师范大学计算机与电子信息学院/人工智能学院

夏年

Tel : 15052875526

Email : brastme@qq.com

明理楼一楼108室

回顾

- 处理器结构
- 指令执行
- 中断
- 存储器结构
- I/O通信技术

课程结构



背景

- Chapter 1 计算机系统
- Chapter 2 操作系统

进程

- Chapter 3 进程控制和管理
- Chapter 5 并发：互斥与同步
- Chapter 6 并发：死锁

操作系统原理

内存

- Chapter 7 存储管理
- Chapter 8 虚拟内存

调度

- Chapter 4 处理器调度

输入输出 (I/O)

- Chapter 9 设备管理

文件

- Chapter 10 文件系统

操作系统概论

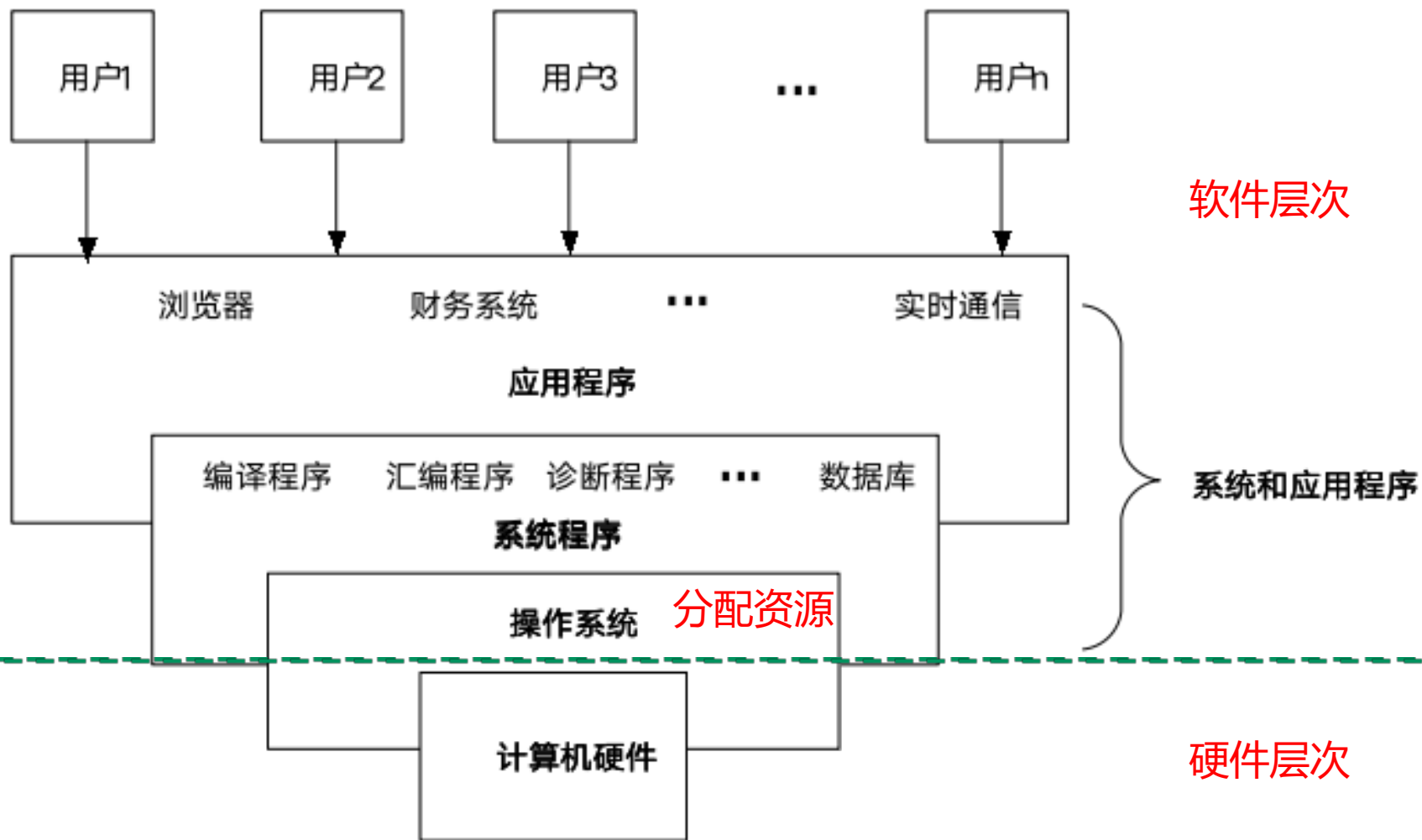
本章教学目标

- 理解操作系统的定义、目标和基本功能
- 理解多道程序设计的概念
- 掌握系统调用的执行方式
- 理解操作系统的结构设计

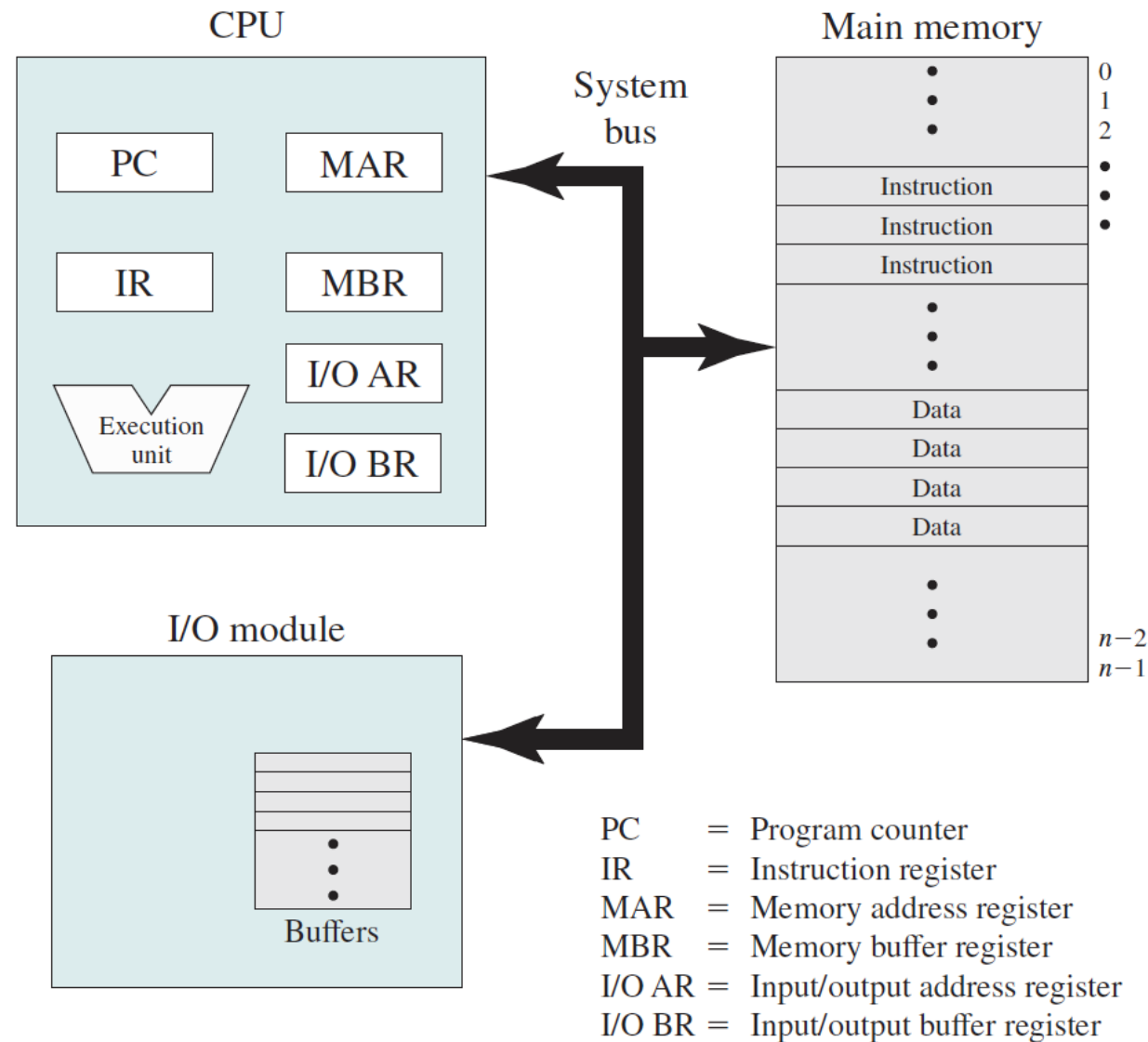
大纲

- **1.1 计算机系统的层次结构**
- 1.2 操作系统的定义和目标
- 1.3 操作系统资源管理技术
- 1.4 操作系统的作用与功能
- 1.5 操作系统的主要特性
- 1.6 多道程序设计
- 1.7 操作系统的形成和发展
- 1.8 操作系统的服务和用户接口
- 1.9 操作系统的结构设计
- 1.10 操作系统的运行模型

1.1 计算机系统的层次结构



1.1 计算机系统的层次结构-硬件



计算机硬件

- 提供基本的可计算性资源，包括处理器、寄存器、存储器，以及各种I/O设备
- 是操作系统和上层软件赖以工作的基础

Figure 1.1 Computer Components: Top-Level View

1.1 计算机系统的层次结构-软件

• 操作系统层

- 最靠近硬件的软件层，对计算机硬件作首次扩充和改造
- 完成资源的调度和分配，信息的存取和保护，并发活动的协调和控制
- 是上层其他软件运行的基础

• 系统程序层

- 各种各样的语言处理程序、数据库管理系统和其他系统程序
- 实用程序，如连接装配程序、库管理程序、诊断排错程序、分类/合并程序等

• 应用程序层

- 解决用户特定的或不同应用需要的问题

大纲

- 1.1 计算机系统的层次结构
- **1.2 操作系统的定义和目标**
- 1.3 操作系统资源管理技术
- 1.4 操作系统的作用与功能
- 1.5 操作系统的主要特性
- 1.6 多道程序设计
- 1.7 操作系统的形成和发展
- 1.8 操作系统的服务和用户接口
- 1.9 操作系统的结构设计
- 1.10 操作系统的运行模型

1.2 操作系统的定义和目标

- 操作系统的定义：

- 没有严格的定义
- 一般认为，是管理系统资源、控制程序执行、改善人机界面、提供各种服务、合理组织计算机工作流程和为用户计算机提供良好运行环境的一种系统软件。



1.2 操作系统的定义和目标

操作系统的**目标**是提供一个能使用户**方便、高效**地执行程序的环境。

具体包括下述目标

- ✓ **方便用户使用**：机械装置→命令行→图形用户界面
- ✓ **扩大机器功能**
- ✓ **管理系统资源**：CPU、存储、文件、...
- ✓ **提高系统效率**：CPU调度、磁盘调度
- ✓ **构筑开放环境**：如POSIX(Portable Operating System Interface)标准

1.2 操作系统的定义和目标

操作系统与上层软件的区别

- **控制与被控制**
 - 操作系统有权分配资源，其它程序只能使用资源
- 操作系统直接作用于硬件之上，隔离其他上层软件，**是软件系统的核心**，是各种软件的基础运行平台
- 操作系统实现资源管理机制，允许应用程序提供资源管理策略

大纲

- 1.1 计算机系统的层次结构
- 1.2 操作系统的定义和目标
- **1.3 操作系统资源管理技术**
- 1.4 操作系统的作用与功能
- 1.5 操作系统的主要特性
- 1.6 多道程序设计
- 1.7 操作系统的形成和发展
- 1.8 操作系统的服务和用户接口
- 1.9 操作系统的结构设计
- 1.10 操作系统的运行模型

1.3 操作系统资源管理技术

资源管理技术的目标

- 解决物理资源有限与竞争使用资源的应用程序众多之间的矛盾，即**资源数量不足**的问题
 - 资源复用
 - 资源虚拟化
- 实现**资源的易用性**
 - 资源抽象

1.3 操作系统资源管理技术

- 资源复用

- 空分复用

- 资源可以进一步分割成更多和更小的单位，供进程使用
 - 资源的不同单位**同时**分配给不同的进程
 - 例如，主存，磁盘

- 时分复用

- 进程可以在一个时间片内以独占方式使用整个物理资源，其它进程则在另外的时间片内使用此资源
 - 例如，磁带机，CPU

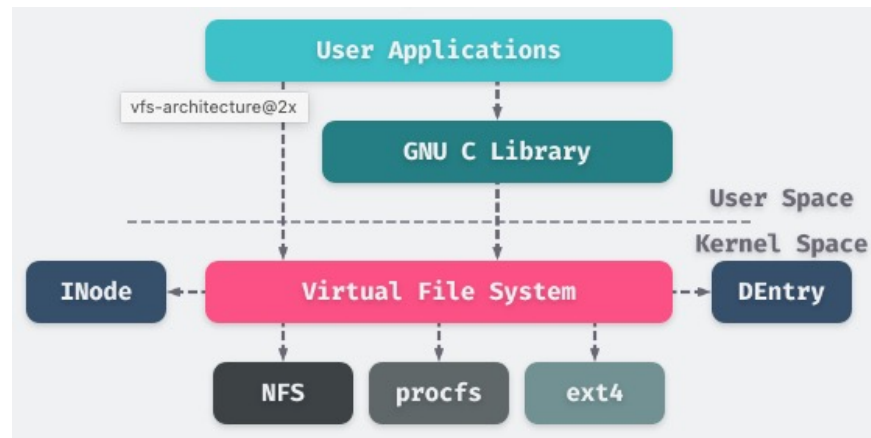
1.3 操作系统资源管理技术

• 资源虚拟化

- 对资源进行转化、模拟或整合，把一个物理资源转变成逻辑上的多个对应物，创建无需共享的多个独占资源的假象，以达到**多用户共享一套计算机物理资源的目的**。

• 例子

- 虚拟内存
- SPOOLing技术(虚拟I/O设备)
- 虚拟文件系统



- **时分复用、空分复用是基本，资源虚化是表现形式，资源虚化最终都要依赖时分复用和空分复用来实现**

1.3 操作系统资源管理技术

- 资源抽象

- 资源抽象软件对内封装实现细节，对外提供应用接口
- 通过分层实现
- 资源抽象的例子：物理设备输出一组字符
 - **Layer-0**：硬件接口，控制寄存器、状态寄存器、数据寄存器，I/O操作命令
 - **Layer-1**：设备驱动程序，屏蔽物理设备处理I/O操作的细节
 - **Layer-2**：系统调用
 - **Layer-3**：库函数

1.3 操作系统资源管理技术

- 机器指令

从内存写入磁盘

```
load(block, length, device)
seek(device, track)
out(device, sector)
```

- 系统调用

```
void write(char *block, int length, int device, int
    track, int sector)
{
    load(block, length, device);
    seek(device, track);
    out(device, sector);
}
```

- 库函数

```
int fprintf(fileID, "%s", datum) {
    ...
    write( );
}
```

1.3 操作系统资源管理技术

操作系统中的基础抽象

- 进程抽象

- 对于进入主存的**当前运行程序**在处理器上操作的状态集的一个抽象
- 是并发和并行操作的基础
- 透明地**时分复用**一个（或多个）处理器
- **处理器虚化**，每个进程都认为独自拥有一个处理器

1.3 操作系统资源管理技术

操作系统中的基础抽象

• 虚存抽象

- 物理主存被抽象成虚拟主存，给每个进程造成一种假象，认为它正在独占和使用整个主存
- 每个进程可以使用**连续的虚拟地址**来引用物理内存单元
- 进程虚拟内存中的内容可以被透明地交换到磁盘，从而为每个用户提供的**虚拟存储空间可以远大于主存**的大小。

1.3 操作系统资源管理技术

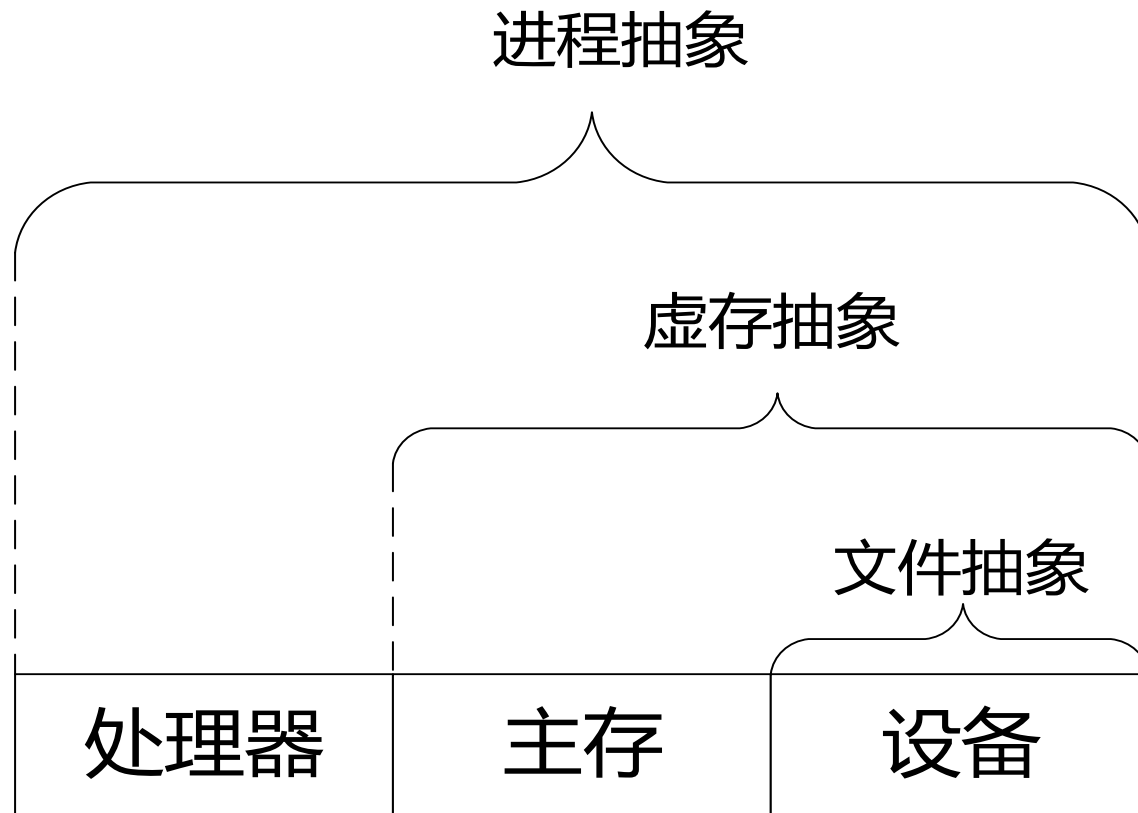
操作系统中的基础抽象

• 文件抽象

- 文件是磁盘、磁带、光盘等设备的抽象，通过将文件中的字节映射到存储设备的物理块中来实现文件抽象。
- 文件抽象的步骤
 - 磁盘→分区：一个物理磁盘可以被划分成多个逻辑上独立的分区
 - 分区→扇区
 - 扇区→簇：解决不同磁盘的扇区大小可能不同的问题
 - 簇→文件系统分区

1.3 操作系统资源管理技术

操作系统中的基础抽象



操作系统的基础抽象

大纲

- 1.1 计算机系统的层次结构
- 1.2 操作系统的定义和目标
- 1.3 操作系统资源管理技术
- **1.4 操作系统的作用与功能**
- 1.5 操作系统的主要特性
- 1.6 多道程序设计
- 1.7 操作系统的形成和发展
- 1.8 操作系统的服务和用户接口
- 1.9 操作系统的结构设计
- 1.10 操作系统的运行模型

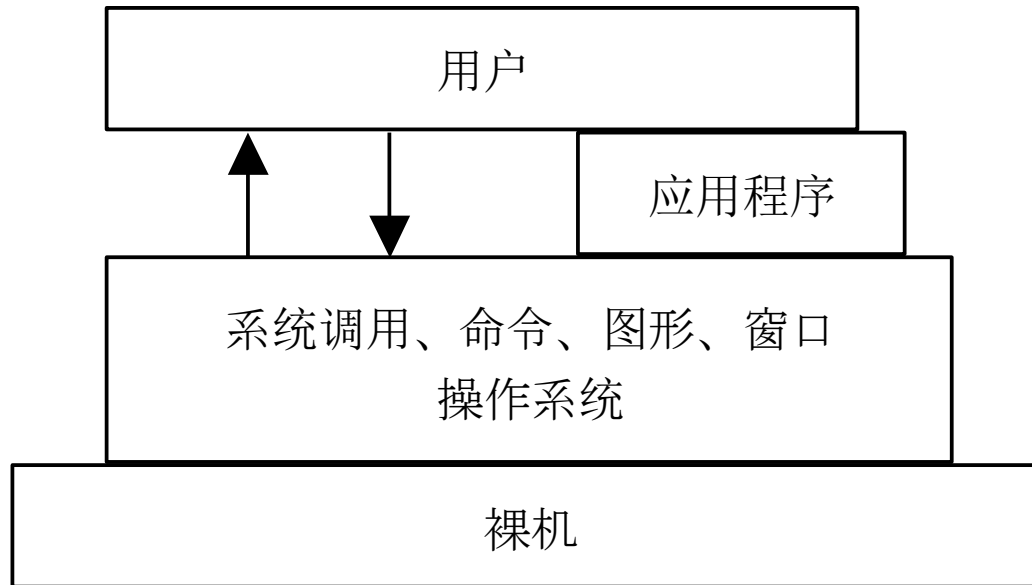
1.4 操作系统的作用与功能

- 操作系统作为**用户接口和服务提供者**
 - 是用户与计算机硬件之间的接口
- 操作系统是计算机系统的**资源管理者和控制者**

1.4 操作系统的作用与功能

- 操作系统是用户与计算机硬件之间的**接口**
 - 操作系统处于用户与计算机硬件系统之间，通过操作来使用计算机系统
 - 操作系统为用户屏蔽硬件细节和复杂性
 - 为用户操作计算机提供多种易用接口
 - 命令方式
 - 系统调用
 - 图形用户界面

操作系统是裸机的第一层扩充。与裸机相比，经过操作系统包装的虚拟机更易于理解和使用。在操作系统的帮助下，用户可以方便、快捷、安全、可靠地操纵计算机硬件并运行自己的程序。



操作系统作为接口的示意图

1.4 操作系统的作用与功能

- 操作系统是计算机系统的**资源管理者**

资源

- 硬件资源：处理器、存储器、I/O设备等
- 信息资源：程序+数据

操作系统的任务：

- 对资源进行抽象，找出各种资源的共性和个性，有序地管理计算机中的硬件、软件资源，跟踪资源使用情况，监视资源的状态，满足用户对资源的需求，协调各程序对资源的使用冲突
- 为用户提供简单、有效的资源使用手段，最大限度实现各类资源的共享，提高资源利用率

1.4 操作系统的作用与功能

操作系统的功能

- 处理机管理/进程管理
- 存储管理
- 设备管理
- 文件管理
- 网络与通信管理

1.4 操作系统的作用与功能

- 处理机管理

- 处理中断事件

- 硬件发现并捕捉中断事件，产生中断信号，但不能处理
 - 操作系统对中断事件进行处理

- 处理器调度

- 单用户单任务系统中，处理器为一个用户的一个任务独占
 - 多道程序或多用户情况下，处理器需要在多个任务或用户之间共享
 - 操作系统引入了**进程和线程**的概念，实现程序的并发执行。处理器的管理和调度最终归结为对进程和线程的管理和调度
 - 将在处理机管理和并发进程两章里详细介绍

1.4 操作系统的作用与功能

处理器调度的功能

- 进程控制和管理
- 进程同步和互斥
- 进程通信
- 进程死锁
- 线程控制和管理
- 处理器调度

1.4 操作系统的作用与功能

- **存储管理**

- 目标：管理存储器资源，为多道程序运行提供有力的支撑，便于用户使用存储资源，提高存储空间的利用率。
- 主要功能包括：
 - 存储分配
 - 地址转换与存储保护
 - 存储共享
 - 存储扩充
- 存储管理的机制与硬件存储器的组织结构和支撑设密切相关。
- 将在存储管理和虚拟内存两章里详细介绍

1.4 操作系统的作用与功能

• 设备管理

- 目标：管理各类外围设备，完成用户提出的I/O请求，加快I/O信息的传送速度，发挥I/O设备的并行性，提高I/O设备的利用率，提供每种设备的设备驱动程序和中断处理程序，为用户隐藏硬件细节，提供方便简单的设备使用方法。
- 主要功能：
 - **提供外围设备的控制与中断处理**
 - **提供缓冲区的管理**
 - **提供设备独立性**
 - **外围设备的分配和去配**
 - **实现共享型外围设备的驱动调度**
 - **实现虚拟设备**
- 将在设备管理一章里详细介绍

1.4 操作系统的作用与功能

- 文件管理

- 目标

- 对用户文件和系统文件进行有效管理，实现**按名存取**；
 - 实现文件的共享、保护和保密，保证**文件的安全性**；
 - 并提供给用户一整套能方便使用文件的**操作和命令**。

- 主要功能：

- 提供文件逻辑组织方法
 - 提供文件物理组织方法
 - 提供文件存取和使用方法
 - 实现文件的目录管理
 - 实现文件的共享和存取控制
 - 实现文件的存储空间管理

- 将在文件管理一章里详细介绍

1.4 操作系统的作用与功能

- **网络与通信管理**
 - 联网操作系统应具备的功能
 - 网上资源管理功能
 - 数据通信管理功能
 - 网络管理功能

大纲

- 1.1 计算机系统的层次结构
- 1.2 操作系统的定义和目标
- 1.3 操作系统资源管理技术
- 1.4 操作系统的作用与功能
- **1.5 操作系统的主要特性**
- 1.6 多道程序设计
- 1.7 操作系统的形成和发展
- 1.8 操作系统的服务和用户接口
- 1.9 操作系统的结构设计
- 1.10 操作系统的运行模型

1.5 操作系统的主要特性

- 并发性
- 共享性
- 异步性

1.5 操作系统的主要特性

• 并发性

- 并发性(**Concurrence**)是指两个或两个以上的事件或活动在**同一时间间隔内**发生。
- 操作系统的并发性指它应该具有处理和调度多个程序同时执行的能力。
 - 多个I/O设备同时在输入输出
 - 设备I/O和CPU计算同时进行
 - 内存中同时有多个系统和用户程序被启动交替、穿插地执行

1.5 操作系统的主要特性

• 并发性

- 并发性使操作系统的设计和实现变得复杂化
- 并发的物理含义
 - 单CPU系统中，实际上是若干道程序之间的空分多路复用
 - **宏观上并发，微观上顺序执行**
 - 在多CPU系统中，程序的执行可以是**并行**的，即两个或两个以上事件或活动在**同一时刻**发生。并行性是并发性的特例。

• **Note: 并发和并行的区别？**

1.5 操作系统的主要特性

并发性引发的问题

- 如何从一个程序切换到另一个程序？
- 以什么样的策略选择下一个运行的程序？
- 如何实现各个运行程序的隔离？
- 如何协调多个运行程序对资源的竞争？

1.5 操作系统的主要特性

- **共享性**

- 共享指操作系统中的资源（包括硬件资源和信息资源）可被多个并发执行的进程共同使用，而不是被一个进程所独占。

1.5 操作系统的主要特性

• 共享性

资源共享的方式

透明资源共享

- 允许同一时间段内多个进程对资源进行访问，好像每个进程都独占资源
- 访问的次序对结果无影响
- 如CPU、主存、磁盘、打印机

显式资源共享

- 同一时间段内只允许一个进程访问资源，这类资源称为临界资源，如磁带机、扫描仪，以及某些数据和表格
- 操作系统必须**提供显式资源共享机制**（申请/释放资源的系统调用，或锁机制），将资源的互斥访问下放给用户决策

1.5 操作系统的主要特性

- 共享性和并发性互为依存

- 资源的共享是因为程序的并发执行而引起的
- 对资源共享实施有效管理是提高程序并发执行效率的前提

1.5 操作系统的主要特性

• 异步性

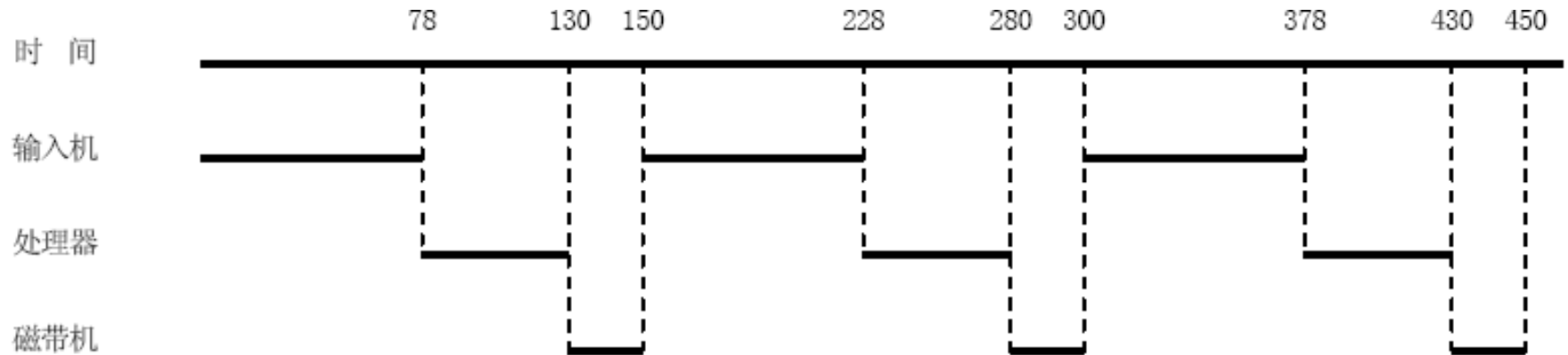
- 在多道程序环境中，允许多个进程并发执行，由于资源有限而进程众多，多数情况，进程的执行不是一贯到底，而是“走走停停”。
- 异步性给系统带来了潜在的危险，有可能导致进程产生与时间有关的错误，但只要运行环境相同，操作系统必须保证多次运行进程，都会获得完全相同的结果。
- 异步性的例子
 - 作业到达系统的时间和类型
 - 操作员发出命令或操作的时间和类型
 - 程序运行发生错误或异常的类型和时刻
 - 中断事件发生的时刻

大纲

- 1.1 计算机系统的层次结构
- 1.2 操作系统的定义和目标
- 1.3 操作系统资源管理技术
- 1.4 操作系统的作用与功能
- 1.5 操作系统的主要特性
- **1.6 多道程序设计**
- 1.7 操作系统的形成和发展
- 1.8 操作系统的服务和用户接口
- 1.9 操作系统的结构设计
- 1.10 操作系统的运行模型

1.6 多道程序设计

- 早期的单道批处理系统中，内存中仅有单个作业在运行，设备利用率低，系统性能差。
- CPU, 外设无法同时工作



单道程序设计运行时CPU效率

$$52/150 \approx 35\%$$

1.6 多道程序设计

- 中断和通道技术的产生使CPU和I/O设备并行工作成为可能，为多道程序设计奠定了基础。
- **多道程序设计是指允许多个程序（作业）同时进入一个计算机系统的内存存储器并启动进行交替计算的方法。**
 - 内存中同时存放了两个以上的程序，它们均处于开始和结束之间
 - 各程序轮流占用CPU，交替执行

1.6 多道程序设计



单道程序设计



多道程序设计

1.6 多道程序设计

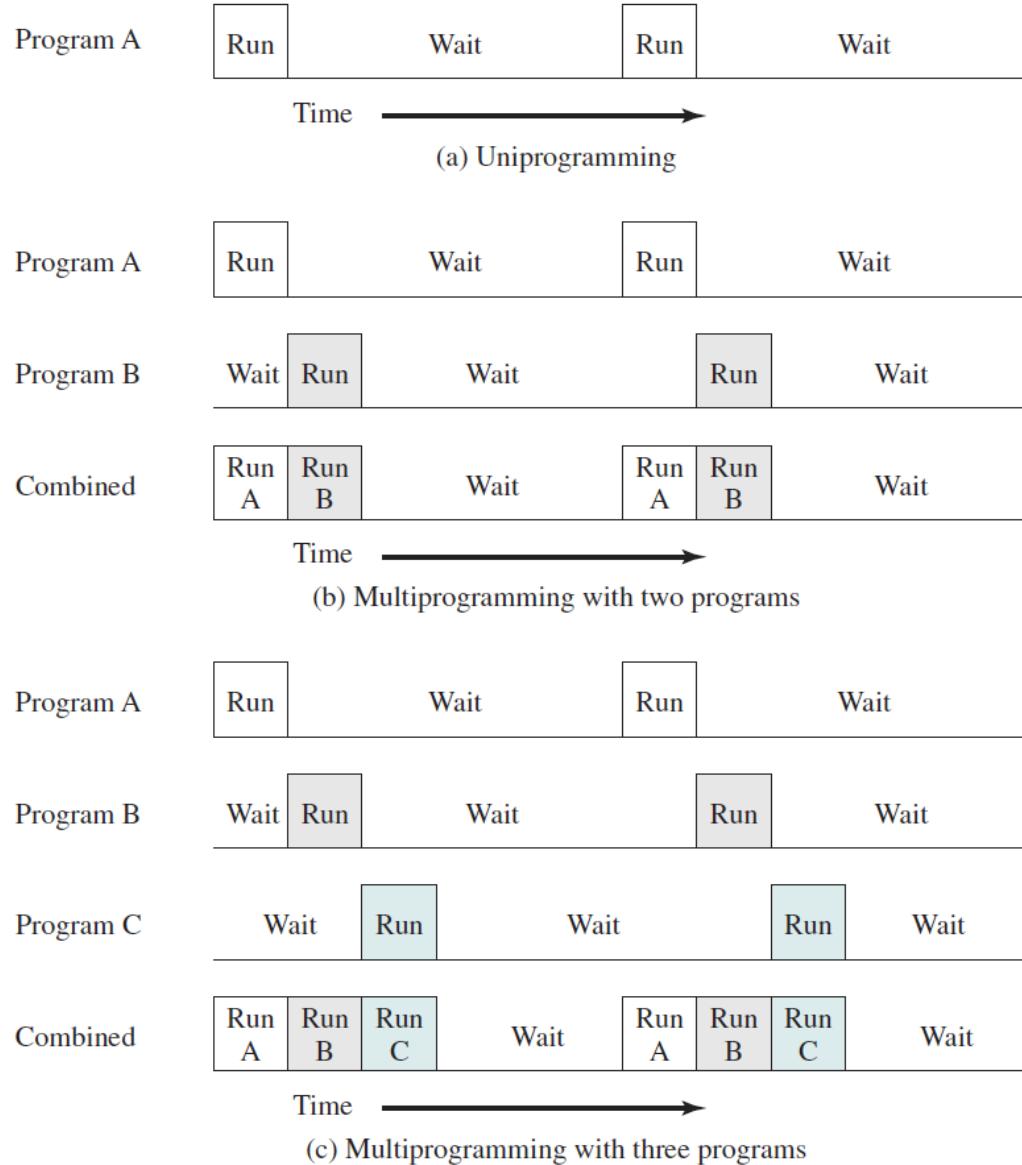


Figure 2.5 Multiprogramming Example

1.6 多道程序设计

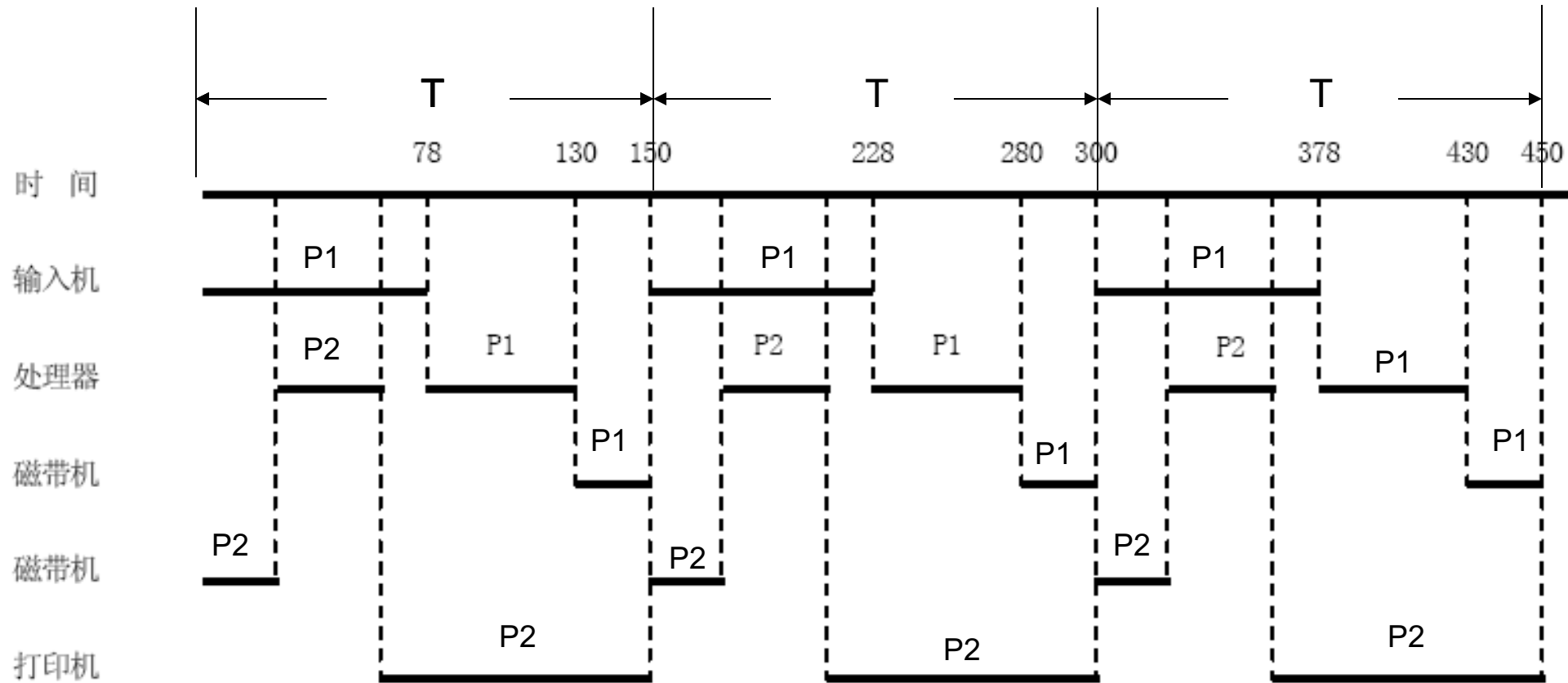
- P1

- a) 输入机输入500个字符(输入机速度6400字符/s), 约78ms
- b) 处理数据, 52ms
- c) 将结果2000个字符存储到磁带机1上(磁带机速度 10^5 字符/s), 20ms

- P2

- a) 从磁带机2上输入2000个字符, 20ms
- b) 处理数据, 42ms
- c) 行式打印机输出两行 (1350行/min) , 约88ms

1.6 多道程序设计



多道程序设计运行时CPU效率

$$(52+42)/150 \approx 63\%$$

1.6 多道程序设计

- 多道程序设计的**道数**指同时**进入内存参与CPU竞争的程序数量**
- 假设内存中有n道程序，程序平均等待I/O操作的时间比例为p,且等待操作相互独立，则所有程序都等待I/O操作的概率是 p^n ，从而，

$$\text{CPU利用率} = 1 - p^n$$

- **Q: 若进程平均花费80%的时间等待I/O操作，则为了使得CPU利用率不低于80%，应至少有多少道程序在主存中运行？**

1.6 多道程序设计

• 优点

- 提高了CPU、内存和I/O设备的利用率
- 改进了系统的吞吐率
- 充分发挥了系统的并行性

• 缺点

- 延长了作业的周转时间

1.6 多道程序设计

多道程序设计需要解决的问题

- **存储保护与程序浮动(地址重定向)**

- 多道程序之间避免相互干扰，一道程序的错误不影响其他程序
- 程序或程序的部分应能从某个内存区域移动到另一区域

- **处理器的管理和分配**

- 一个程序可以处于运行、等待或就绪三个状态；程序可以在三个状态之间切换
- 合理搭配具有**不同特性**的多道程序同时运行，如I/O密集型和CPU密集型任务的合理搭配

- **系统资源的管理和调度**

- 资源为多道程序所共享，需要解决好资源的竞争与协作、共享与安全

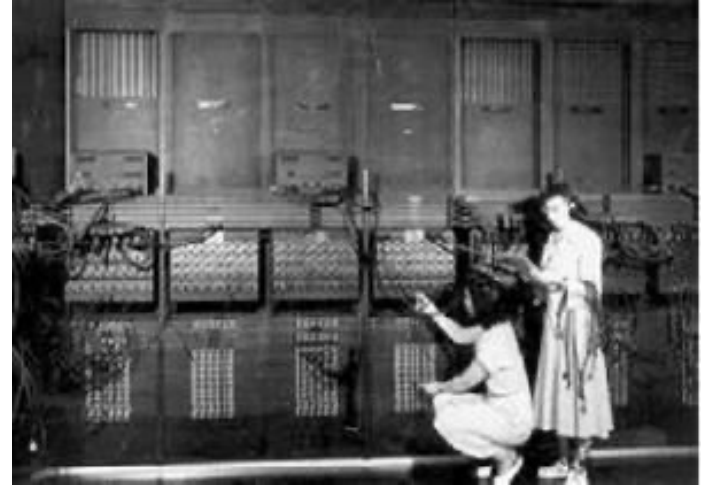
大纲

- 1.1 计算机系统的层次结构
- 1.2 操作系统的定义和目标
- 1.3 操作系统资源管理技术
- 1.4 操作系统的作用与功能
- 1.5 操作系统的主要特性
- 1.6 多道程序设计
- **1.7 操作系统的形成和发展**
- 1.8 操作系统的服务和用户接口
- 1.9 操作系统的结构设计
- 1.10 操作系统的运行模型

1.7 操作系统的形成和发展

• 人工操作阶段

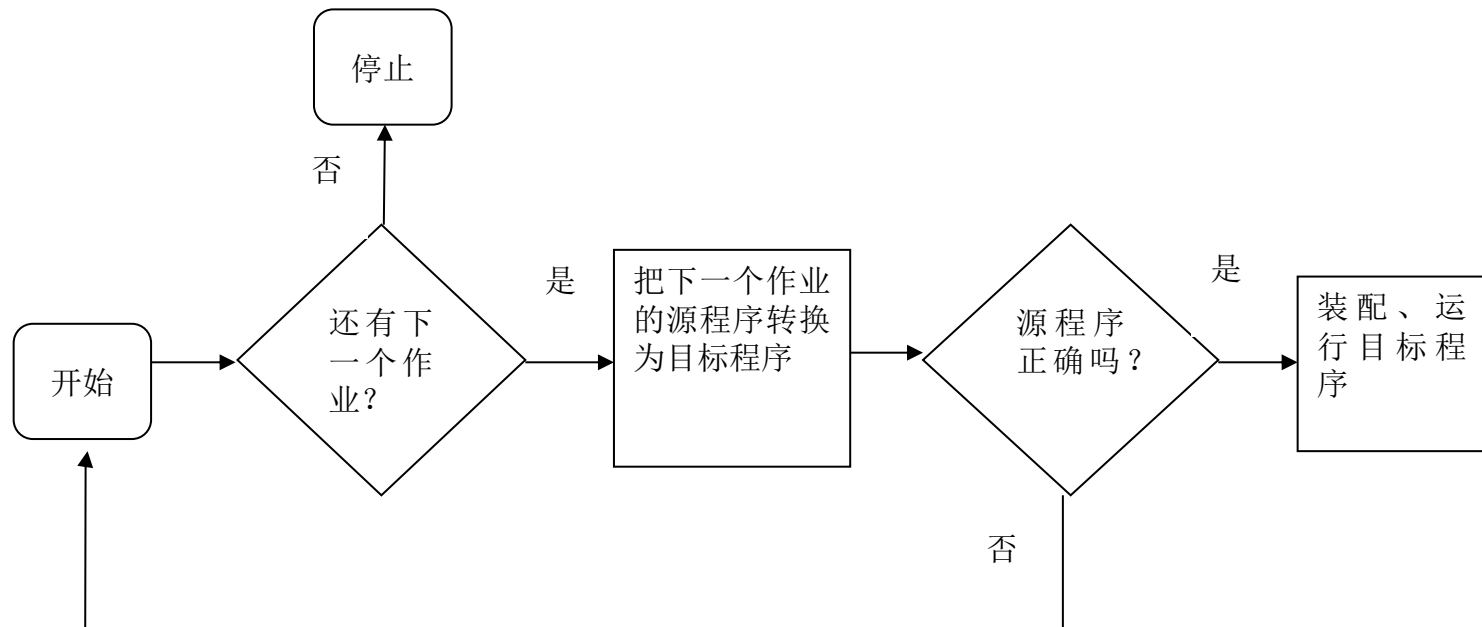
- 手工方式直接控制、使用计算机硬件
 - 使用机器语言编程
 - 将准备好的程序和数据穿孔在纸带或卡片上
 - 从纸带或卡片输入机将程序和数据输入计算机
 - 通过控制台的按钮、开关和氛灯来操纵和控制程序
- ## • 缺点
- 用户独占资源
 - 人工干预多
 - 计算时间拉长



1.7 操作系统的形成和发展

• 管理程序阶段/简单批处理

- 自动控制和处理作业流
- 当一个作业完成时，自动返回到管理程序，选择下一个作业载入运行



单道批处理系统处理流程图

1.7 操作系统的形成和发展

- 管理程序阶段/简单批处理

- 管理程序包括

- 一组控制命令和解释器
 - 设备驱动和I/O控制功能
 - 库程序和程序装配功能
 - 简单的文件管理功能

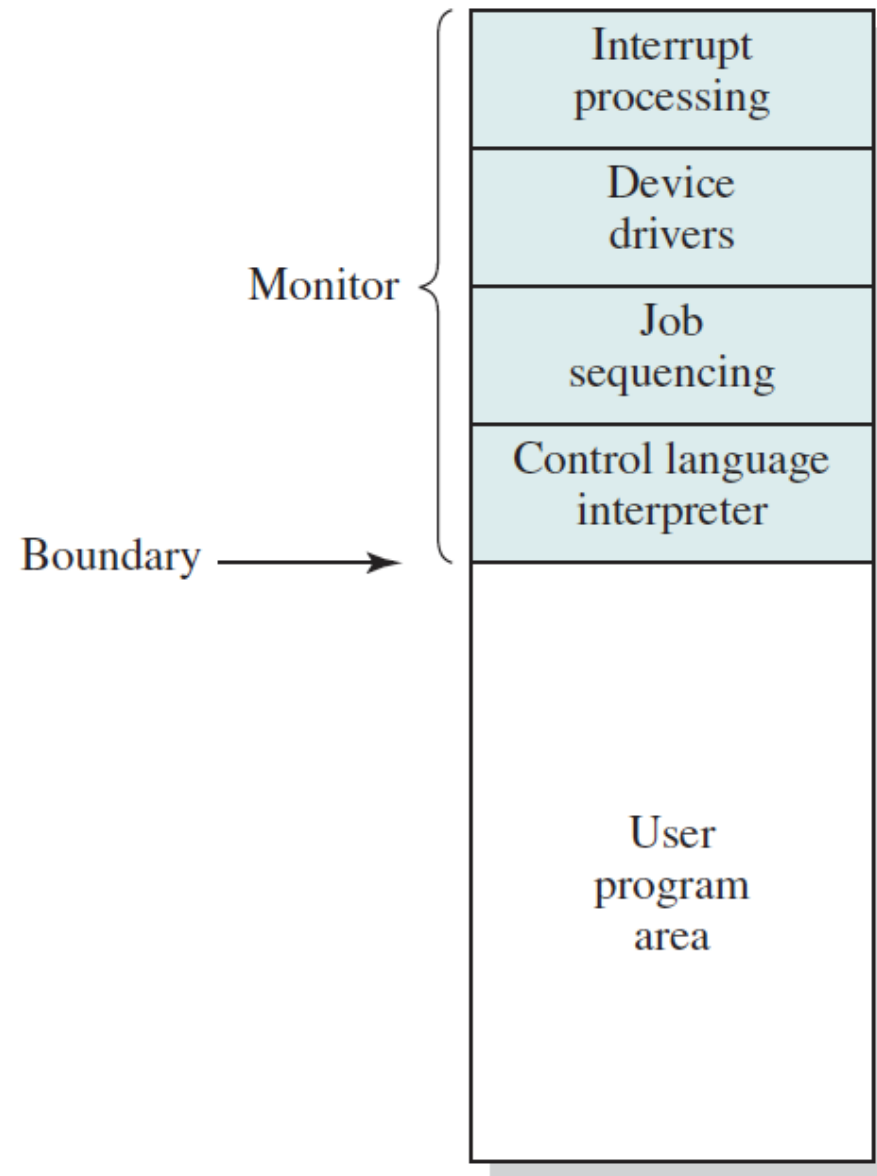
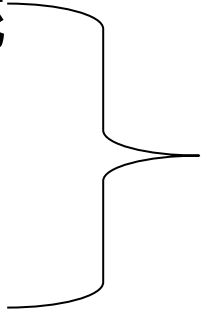


Figure 2.3 Memory Layout for a Resident Monitor

1.7 操作系统的形成和发展

操作系统的类型

- 批处理操作系统
 - 分时操作系统
 - 实时操作系统
 - 并行操作系统
 - 网络操作系统
 - 分布式操作系统
 - 嵌入式操作系统
- 三种基本类型
- 

1.7 操作系统的形成和发展

批处理操作系统

- 批处理系统的特征
 - 用户脱机工作
 - 作业控制语言JCL(Job Control Language)
 - 程序+数据+作业说明书
 - 用户无法在程序执行过程中对其进行控制
 - 成批处理作业
 - 操作员集中一批作业并对作业进行预输入
 - 操作系统调度和控制用户作业的执行
 - 单/多道程序运行
 - 单道批处理系统
 - 多道批处理系统
- 优缺点
 - 优点：系统资源利用率高，作业吞吐量大
 - 缺点：作业周转时间长，不具备交互式能力，不利于开发和调试

1.7 操作系统的形成和发展

分时操作系统

- 多个联机用户通过终端（键盘/显示器）同时直接使用一个计算机系统进行交互式计算
 - 用户在各自终端上进行会话
 - 程序、数据和命令在会话过程中提供，以问答方式控制程序的运行
 - 时间片轮转将处理器分配给各个终端
- 分时操作系统的特性
 - 同时性
 - 独立性
 - 及时性
 - 交互性

1.7 操作系统的形成和发展

实时操作系统

- 当外部事件或数据产生时，能对其予以接收并以足够快的速度进行处理，所得结果能够在规定的时间内控制生产过程或对控制对象作出快速响应，并控制所有实时任务协调运行
 - 事件驱动
 - 及时响应
 - 高可靠性
- 实时操作系统的分类
 - 过程控制系统
 - 信息查询系统
 - 事务处理系统
- 过程控制系统的处理步骤
 - 数据采集
 - 加工处理
 - 操作控制

- **硬实时任务**

- 对截止时间的要求极其严格

- **软实时任务**

- 要求有一个截止时间，但要求并不十分严格

大纲

- 1.1 计算机系统的层次结构
- 1.2 操作系统的定义和目标
- 1.3 操作系统资源管理技术
- 1.4 操作系统的作用与功能
- 1.5 操作系统的主要特性
- 1.6 多道程序设计
- 1.7 操作系统的形成和发展
- **1.8 操作系统的服务和用户接口**
- 1.9 操作系统的结构设计
- 1.10 操作系统的运行模型

1.8 操作系统的服务和用户接口

操作系统的服务

- **提供给程序和用户的共性服务包括：**
 - 创建/执行程序
 - I/O设备访问
 - 信息存取
 - 通信服务
 - 错误检测和处理
- **为保证自身效率和正确性提供的功能**
 - 资源分配
 - 统计
 - 保护

1.8 操作系统的服务和用户接口

操作系统提供的用户接口

- 程序接口
 - 可被应用程序直接使用
 - 由一组系统调用组成
- 操作接口
 - 通过实用程序提供
 - 用户借助命令管理或shell来请求执行

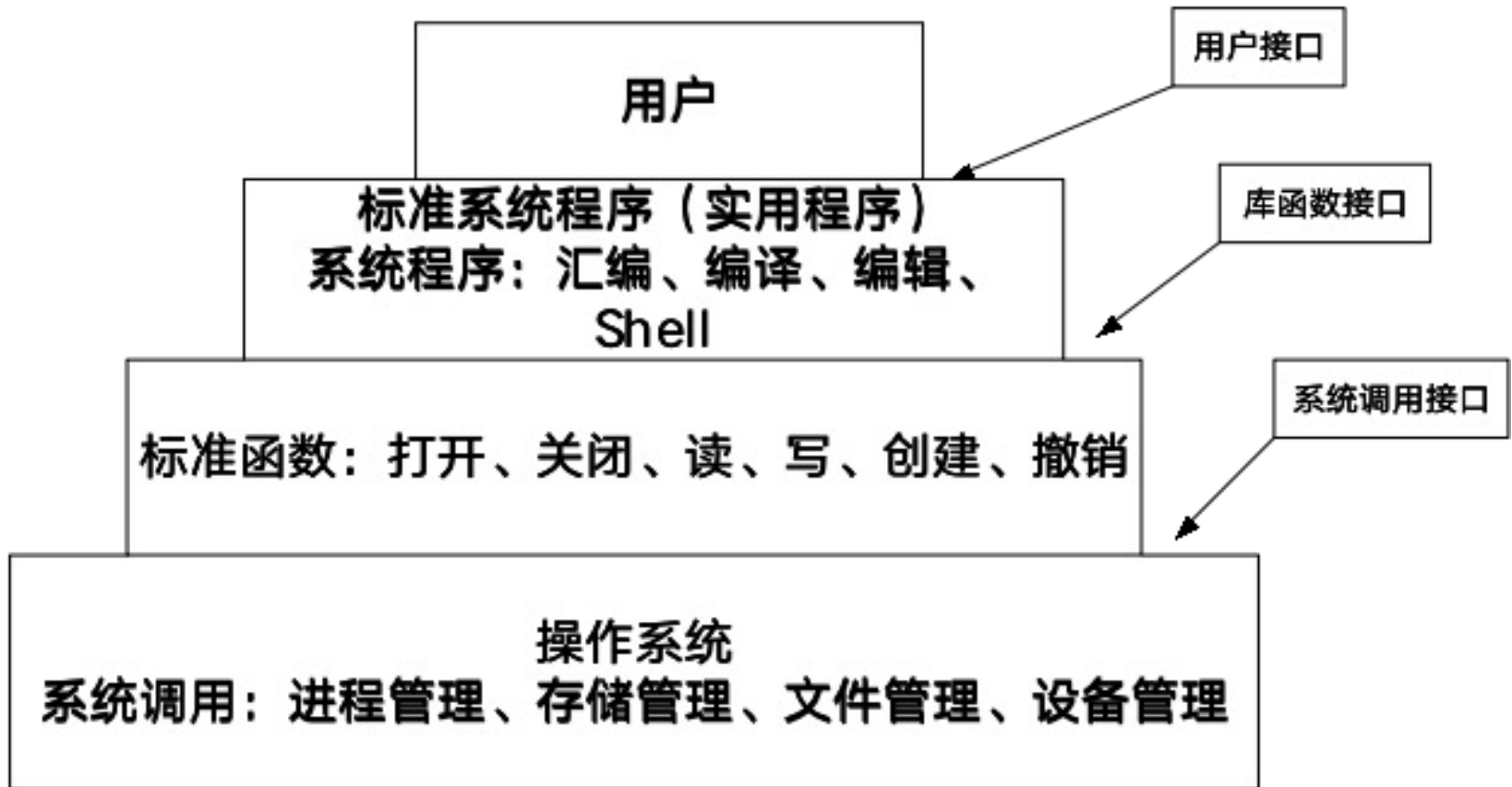
1.8 操作系统的服务和用户接口

系统调用

- 系统调用是为了扩充机器功能、增强系统能力、方便用户使用而建立的。
- 系统调用是用户程序或其他系统程序获得操作系统服务的唯一途径。
- 操作系统内核的主体是系统调用的集合，可以将**内核看成特殊的公共子程序**
- 应用程序和系统调用分别在**用户空间和内核空间**运行，在逻辑上相互隔离，起到了保护系统的作用

1.8 操作系统的服务和用户接口

- 系统调用



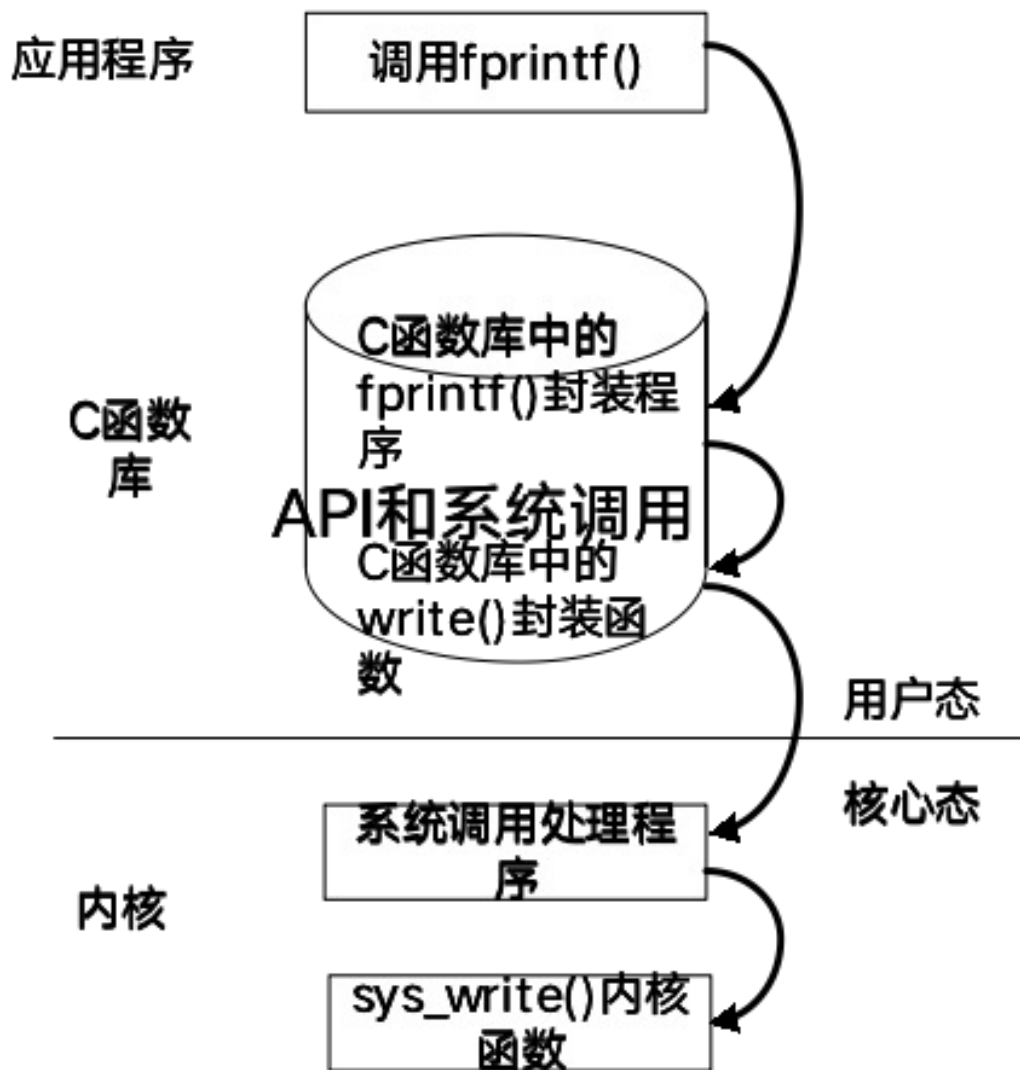
UNIX/Linux操作系统上系统程序、库函数和系统调用的关系

1.8 操作系统的服务和用户接口

API和系统调用

- 不同操作系统提供的系统调用功能类似，但实现细节不尽相同
- 系统调用属低层接口，接口复杂，跨平台移植困难
- **API对系统调用及其它复杂功能进行封装，提供用户所需的服务，标准化，易于使用，易于移植**
 - POSIX API标准
 - Win32 API
 - Java API
- API为函数定义，**仅说明获得给定服务的方式，不规定过程实现**
- 一个API的实现可能会用到一个或多个系统调用，也可能完全不使用系统调用；多个API可能封装同一个系统调用。
- 调用API在用户态，如果API使用了系统调用，则在执行过程中切换到核心态

API和系统调用



UNIX/Linux系统中应用程序执行API `fprintf()`的过程

1.8 操作系统的服务和用户接口

系统调用的分类

- **进程管理**：创建和撤销进程、终止或异常终止进程、阻塞和唤醒进程、挂起和激活进程等
- **文件操作**：建立文件、删除文件、打开文件、关闭文件、读写文件等
- **设备管理**：申请设备、释放设备、设备I/O操作和重定向、获得和设置设备属性等
- **内存管理**：申请和释放内存
- **信息维护**：获取和设置日期及时间、获取和设置系统数据、生成诊断和统计数。
- **进程通信**：建立和断开通信连接、发送和接受消息、链接和断开共享内存等。

1.8 操作系统的服务和用户接口

系统调用的实现

- 实现系统调用功能的机制称为**陷入(trap)机制**。
- 系统调用将引起**处理器中断**，相应的机器指令称**访管指令** (如INT)。
- **每个系统调用都事先规定了编号**，称功能号，在陷入指令中必须指明对应系统调用的功能号。

1.8 操作系统的服务和用户接口

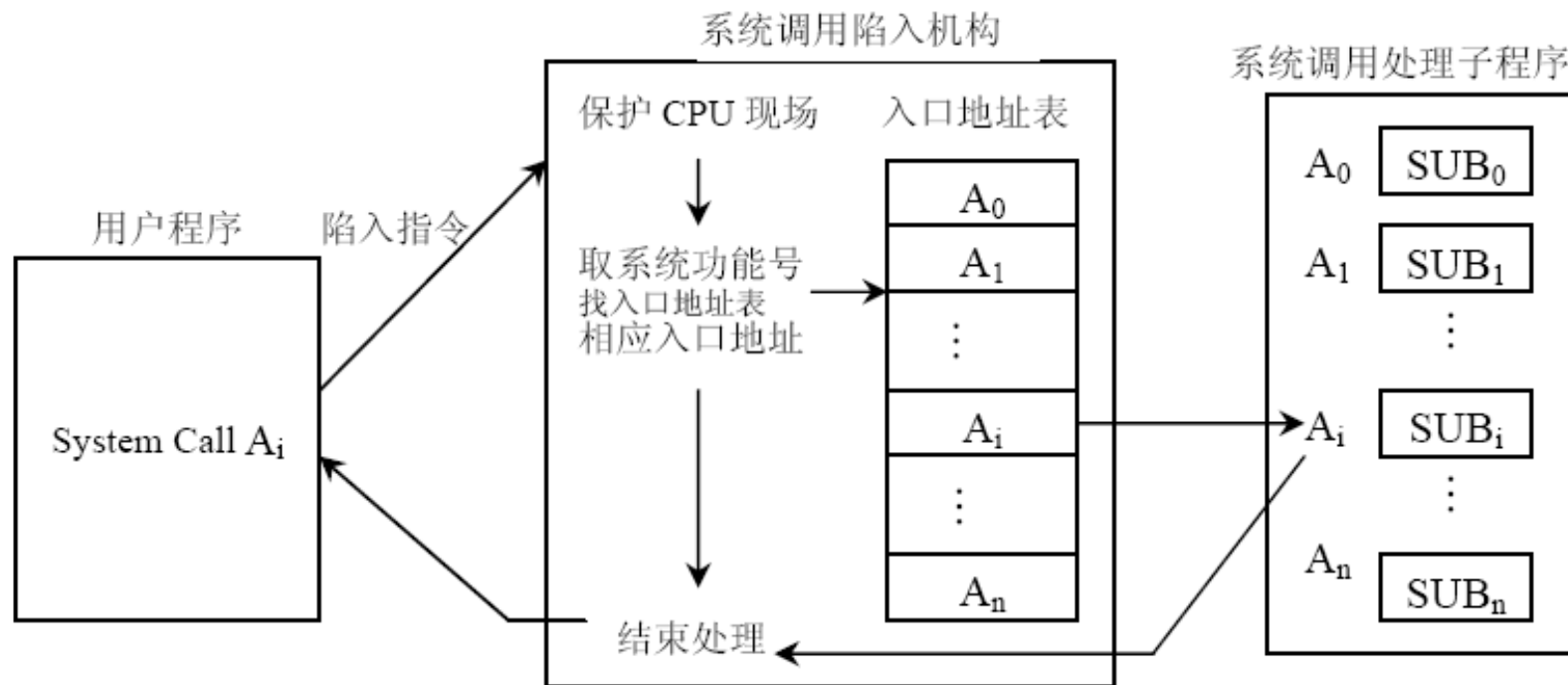
系统调用的实现

• 实现系统调用的步骤

- 编写系统调用处理程序
- 设计一张系统调用入口地址表，每个入口地址都指向一个系统调用的处理程序，有些还包含系统调用自带参数的个数
- 陷入处理机制，需开辟现场保护区，保存发生系统调用时的处理器现场

1.8 操作系统的服务和用户接口

系统调用的实现



陷入机构和系统调用处理过程

1.8 操作系统的服务和用户接口

系统调用的参数传递方法

- **陷入指令自带参数**

- 直接参数：规定指令之后若干单元存放参数
- 间接参数：指令之后紧靠单元存放参数地址，指出参数的存放区

- **CPU通用寄存器传递**

- 将参数值直接放入通用寄存器，仅适用少量参数传递
- 在内存中的一个区或表中存放参数，将其首地址送入寄存器

- **堆栈传递**

1.8 操作系统的服务和用户接口

系统调用与普通函数调用的区别

- 调用形式不同
 - 普通函数调用使用一般调用指令，其转向地址是固定不变的，包含在跳转语句中
 - 系统调用中不包含处理程序入口，仅提供功能号，**按功能号调用**。
- 被调用代码位置不同
 - 普通函数调用是一种静态调用，调用者和被调用者在同一程序内，经过编译连接后作为目标代码的一部分，当函数被修改后，需要重新编译连接。
 - 系统调用是一种动态调用，系统调用的**处理代码在调用程序之外**（在操作系统中）

1.8 操作系统的服务和用户接口

系统调用与普通函数调用的区别

- 提供方式不同
 - 普通函数往往由编译系统提供，不同编译系统提供的函数可以不同
 - 系统调用由操作系统提供，**给定的操作系统上系统调用的功能、种类和数量固定不变**
- 调用的实现不同
 - 普通函数调用一般使用跳转机器指令(JUMP)来实现调用，在用户态运行
 - 系统调用是通过陷入机制实现，需要从**用户态转变为核心态**

1.8 操作系统的服务和用户接口

中断与陷入关系

- 中断由**与现行指令无关的中断信号触发**，通常在**两条机器指令之间才可以响应中断**，一般来说，中断处理程序提供的服务不是为当前进程所需的。
- 陷入则是由**处理器正在执行现行指令而引起的**，通常，陷入处理程序提供的服务是为当前进程所用的。
- 中断与陷入采用的实现技术基本一样，都需要进行处理器模式切换，保护CPU运行现场，然后转向中断/系统服务例程，待服务结束后恢复CPU现场返回被中断程序。

1.8 操作系统的服务和用户接口

操作系统提供的用户接口

- 作业控制方式
 - 联机用户接口
 - 命令行
 - Command arg1 arg2 ... argn
 - 由命令解释器执行
 - 批命令
 - BAT, Shell文件
 - 图形化用户界面
 - 脱机用户接口
 - 专为批处理作业用户提供的
 - 作业控制语言JCL

1.8 操作系统的服务和用户接口

操作系统提供的用户接口

- 命令解释器的种类
 - Shell: Bourne shell, C shell, Bourne-Again shell, Korn shell.
 - MS-DOS
- 命令解释器的两种实现方式
 - 解释器自身包含执行命令的代码
 - 能支持的命令数决定了命令解释器的大小，可扩展性差
 - 命令解释器并不理解命令，而只是利用命令定位实现该命令的系统程序，将其载入内存并加以执行
 - 程序员很容易添加新命令
 - 命令解释器可以很小，并无需改变

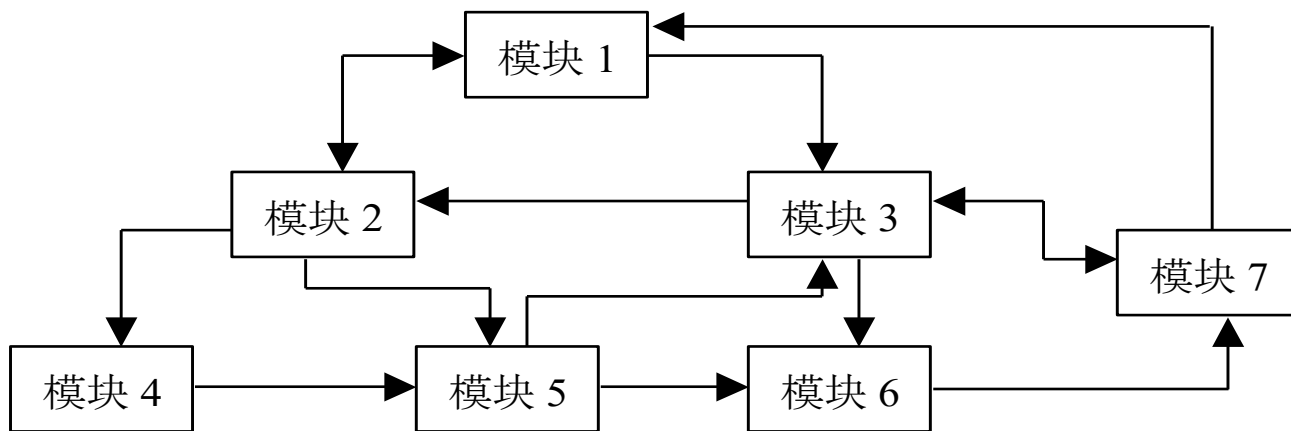
大纲

- 1.1 计算机系统的层次结构
- 1.2 操作系统的定义和目标
- 1.3 操作系统资源管理技术
- 1.4 操作系统的作用与功能
- 1.5 操作系统的主要特性
- 1.6 多道程序设计
- 1.7 操作系统的形成和发展
- 1.8 操作系统的服务和用户接口
- **1.9 操作系统的结构设计**
- 1.10 操作系统的运行模型

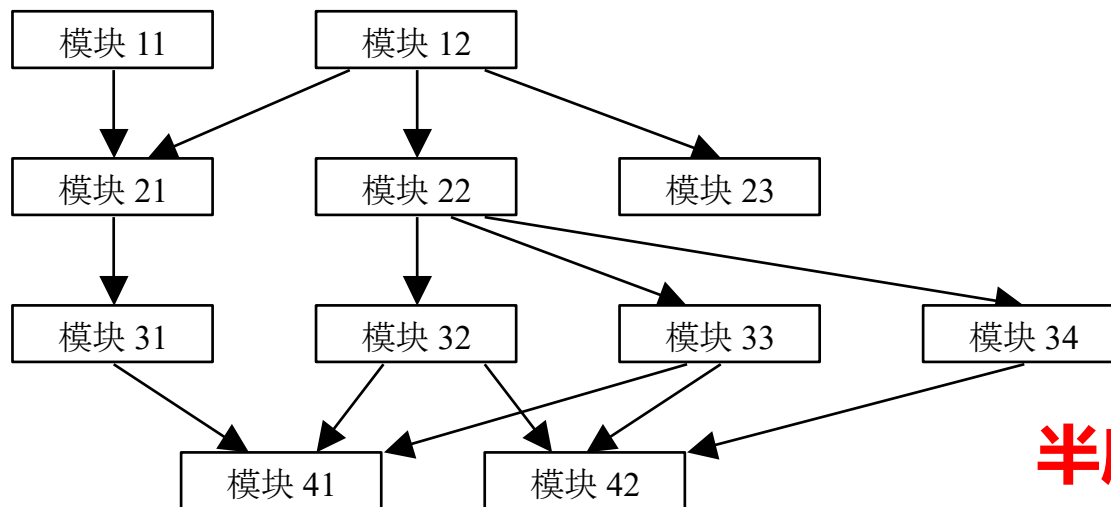
1.9 操作系统的结构设计

- 操作系统的设计呈现下述特征
 - 复杂程度高
 - 生成周期长
 - 正确性难保证
- 操作系统结构设计的内涵
 - 整体结构，包括功能分块、模块交互
 - 局部结构，包括数据结构和控制结构
 - 运行时组织，如采用进程还是线程，在系统空间还是用户空间运行

1.9 操作系统的结构设计



整体式结构模型示意图



半序层次式？

全序层次式操作系统结构模型

1.9 操作系统的结构设计

操作系统的构件

- **内核**：内核是提供支持系统运行的基本功能和基本操作的一组程序模块
- **进程**：进程操作系统资源分配的基本单位
- **线程**：进程之间的通信和切换开销相当大，限制了系统中并发执行的进程数目，于是引入了更细粒度的线程，系统调度的基本单位不再是进程而是线程
- **管程**：管程管理共享资源的程序（一种进程同步机制），对管程的调用表示对共享资源的请求与释放
- **类程**：类程管理私有资源，对类程的调用表示对私有资源的操作。它仅能被进程及起源于同一进程的其他类程或管程嵌套调用链所调用。

1.9 操作系统的结构设计

内核

- 什么是内核？
 - 内核是操作系统对裸机的第一次改造，为进程的并发执行提供了良好的运行环境
 - 内核是一组可信的程序模块
 - 内核运行于核心态
 - 内核具有访问硬件设备和所有主存空间的权限
 - 内核是仅有的能够执行特权指令的程序

1.9 操作系统的结构设计

内核的分类

- 单内核
 - 整体式结构
 - 层次结构
- 微内核
- 虚拟机

1.9 操作系统的结构设计

单内核

- 运行时是一个大二进制映像
- 模块之间的交互通过直接调用其他模块中的函数来实现
- Unix和Linux都是单内核结构，但Linux引入了动态加载模块和卸载模块机制

1.9 操作系统的结构设计

整体式结构

- 整体式结构的特性
 - 模块是操作系统的基本单元
 - 按照功能将系统分解为若干具有一定独立功能的模块
 - **模块间可不加控制地自由调用**
 - 各模块分别设计、编码、调试，最后将所有模块连接成一个整体
- 缺点
 - 模块独立性差，调用关系复杂
 - 系统结构不清晰，正确性难以保证
- 优点
 - 结构紧密，组合方便
 - 系统效率较高

1.9 操作系统的结构设计

层次式结构

- 层次式结构的特点
 - 模块按照功能的调用次序排列成若干层次
 - 各层次之间是**单向调用关系**
- 优点
 - 接口少而简单
 - 下层模块的正确性为上层模块的正确性提供了基础，使系统的正确性大大提高
 - 增加、修改或替换一个层次不会影响其它层次
- 缺点
 - 合理地定义不同层次的功能较为困难
 - 系统的效率会降低。每一层都可能需要修改参数，传递数据。每一层都可能增加系统调用的开销。

1.9 操作系统的结构设计

微内核

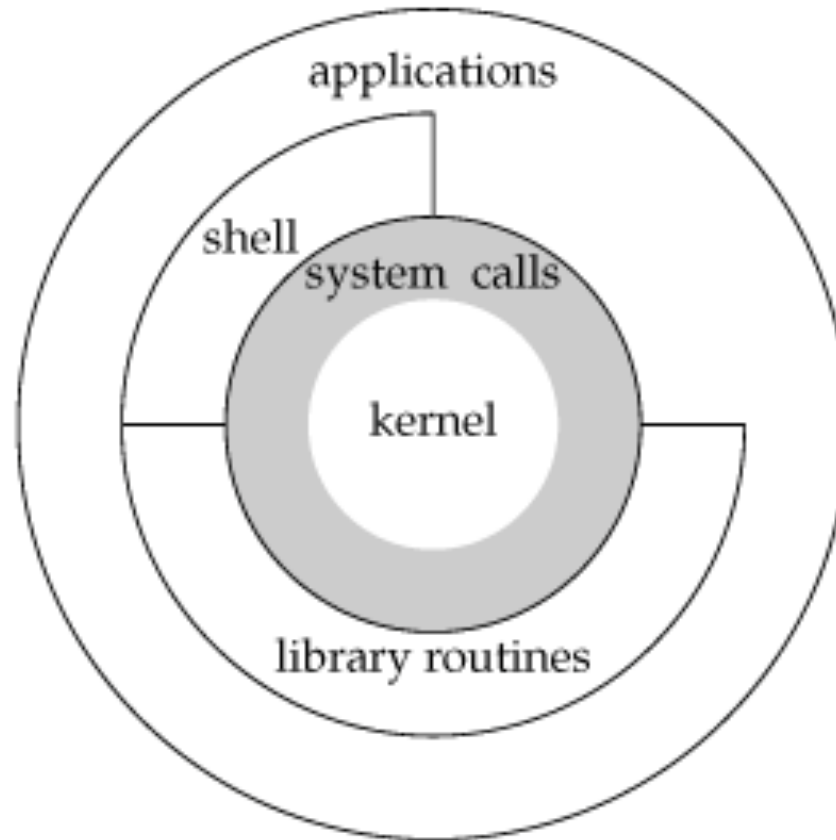
- 操作系统分为两部分
 - **核心态的内核**
 - 微内核应该包括哪些服务并没有确切的共识，通常，至少包含进程管理及调度，消息传递和设备驱动，I/O和中断管理，存储管理。
 - **用户态进程**
 - 将所有非必要的组件从内核移出，作为系统或用户程序出现，如文件管理服务，统计服务等。
- 不同用户空间的进程之间的通信由消息传递实现。但是，这些进程并不直接通信，而是通过微内核间接通信。

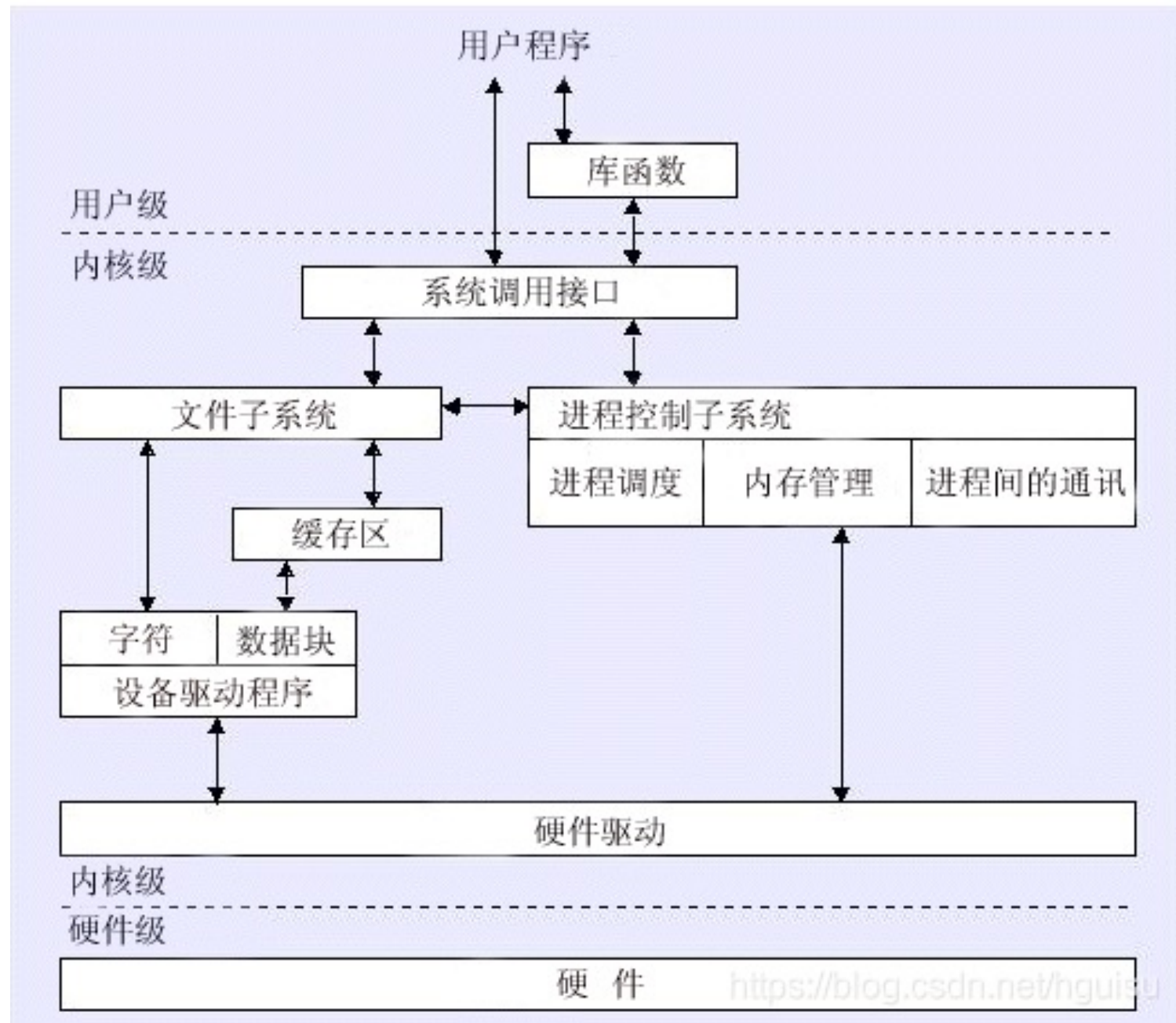
1.9 操作系统的结构设计

微内核

- 优点：
 - 提供一致性接口
 - 可扩充性和易修改性强
 - 可移植性好
 - 对分布式系统提供有力的支撑
- 缺点
 - 效率低，进程之间必须通过内核的通信机制才能进行通信

Figure 1.1. Architecture of the UNIX operating system





<https://blog.csdn.net/hguisu>

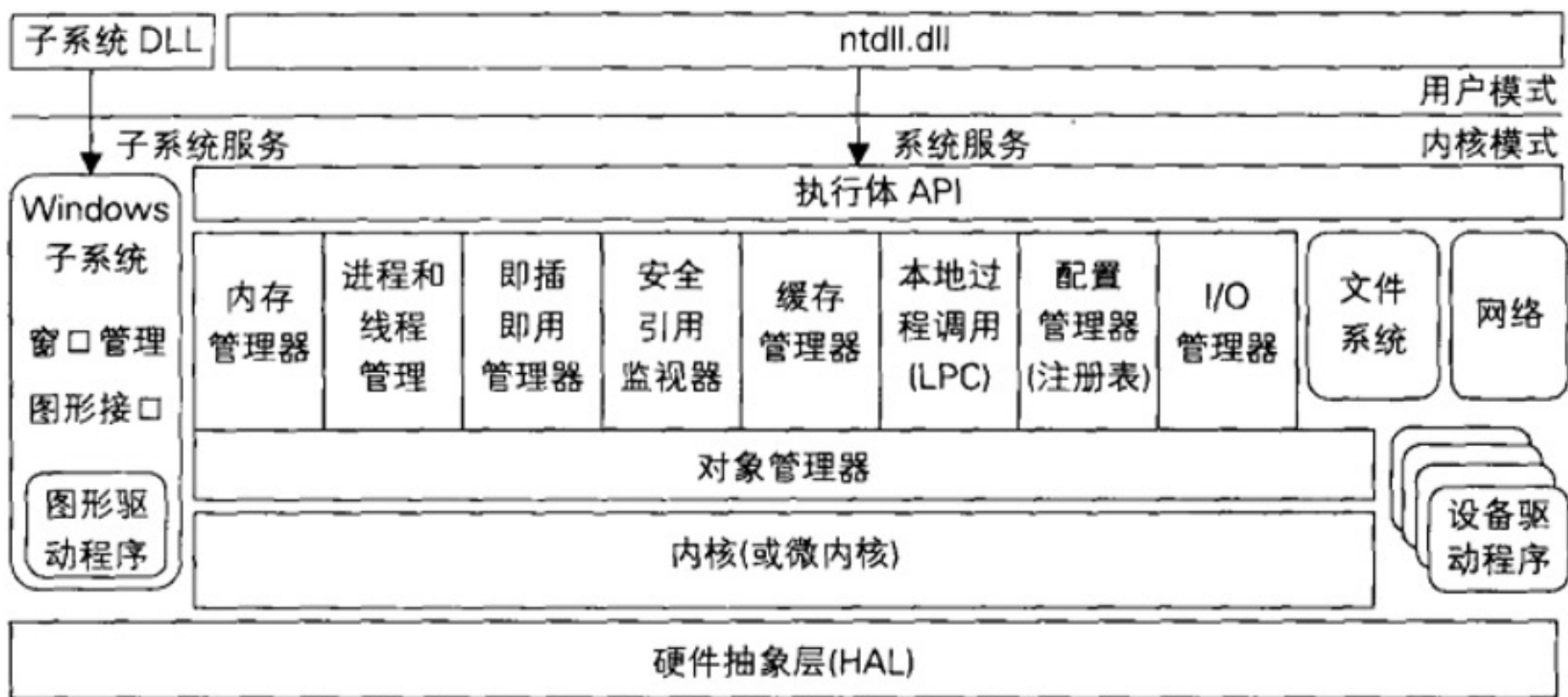
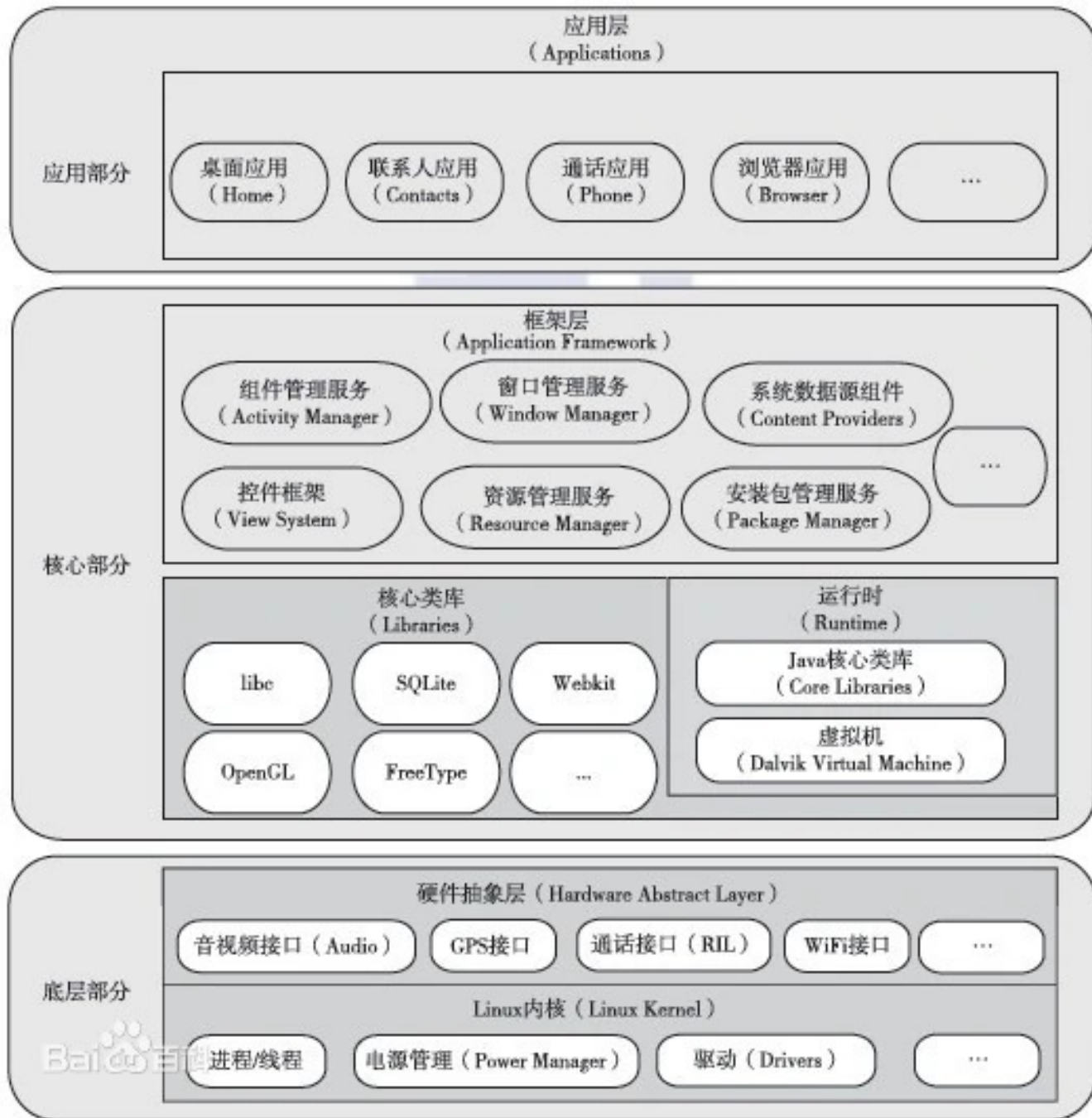
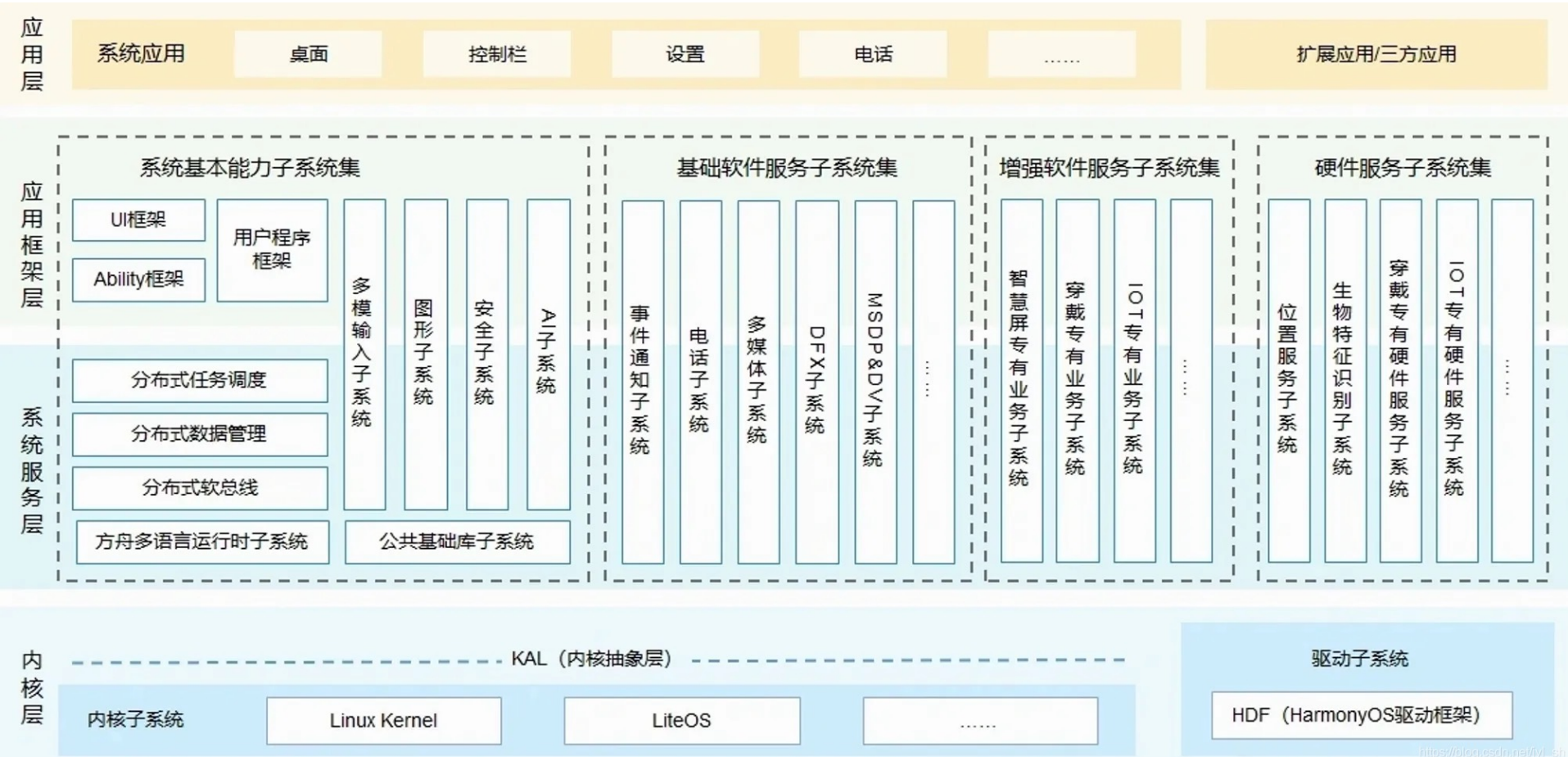


图 2.3 Windows 内核的组成结构

<https://blog.csdn.net/youyou519>





<https://blog.csdn.net/jyksh>

1.9 操作系统的结构设计

内核的主要功能

- 中断处理
- 时钟管理
- 短程调度
- 原语管理

1.9 操作系统的结构设计

内核执行的属性

- 中断驱动
- 不可抢占
- 在屏蔽中断状态下执行
- 可使用特权指令

1.9 操作系统的结构设计

虚拟机

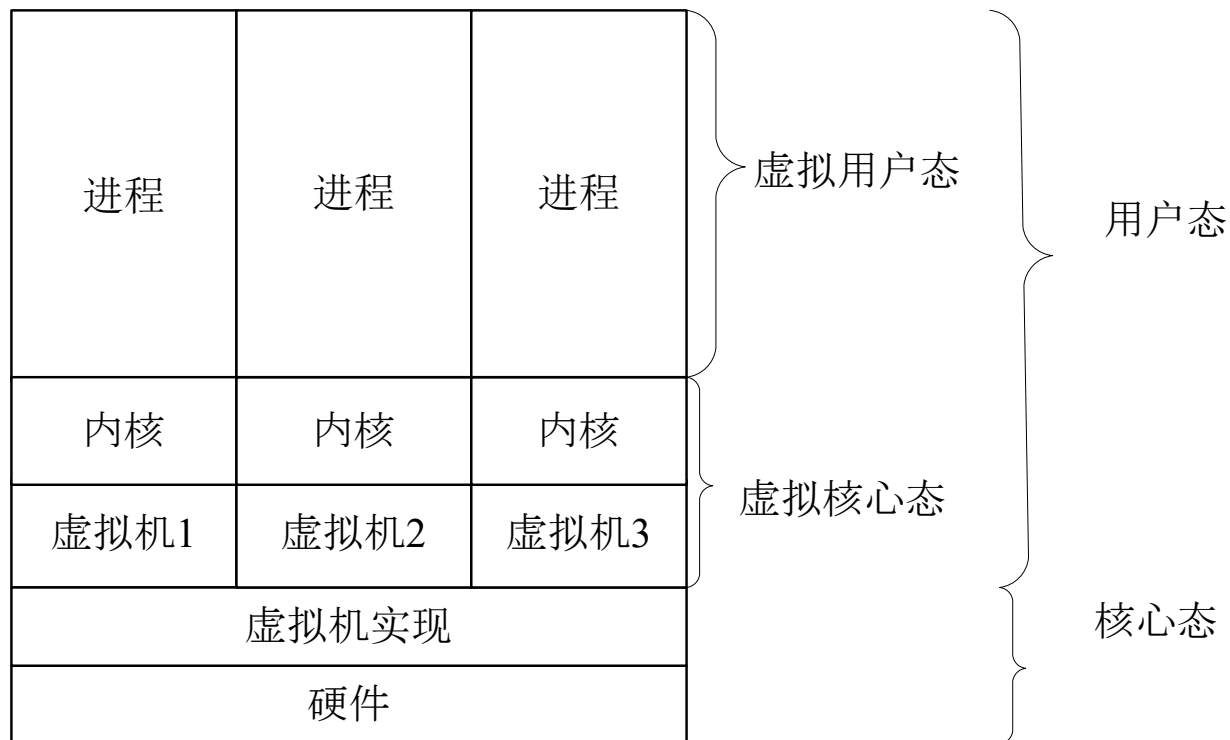
- 将单台计算机的硬件（CPU, 内存，磁盘驱动，网卡等）抽象成多个不同的执行环境，从而创造每个不同的执行环境都运行在自己的私有计算机上的幻觉。
- 虚拟机仅提供与底层裸硬件等同的接口
- 虚拟机软件可以运行在核心态，由于它提供操作系统的功能。
- 而虚拟机自身则只能在用户态运行。
- 虚拟机的运行引入了虚拟用户态和虚拟核心态

1.9 操作系统的结构设计

虚拟机



(a) 非虚拟机结构



(b) 虚拟机结构

大纲

- 1.1 计算机系统的层次结构
- 1.2 操作系统的定义和目标
- 1.3 操作系统资源管理技术
- 1.4 操作系统的作用与功能
- 1.5 操作系统的主要特性
- 1.6 多道程序设计
- 1.7 操作系统的形成和发展
- 1.8 操作系统的服务和用户接口
- 1.9 操作系统的结构设计
- **1.10 操作系统的运行模型**

1.10 操作系统的运行模型

• 独立运行的内核模型

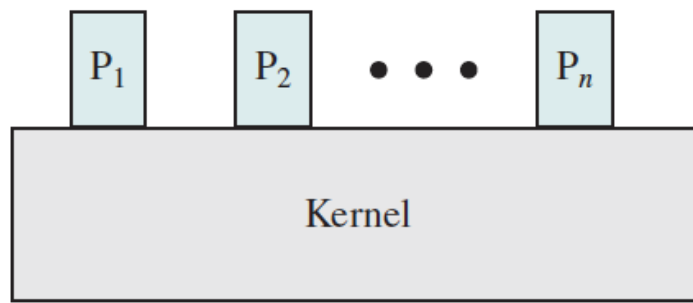
- 进程仅针对应用程序而言
- 操作系统作为独立实体运行在内核模式
- 内核有独立的核心栈，控制函数的调用和返回
- 内核函数难以并发执行

• 操作系统功能在用户进程内执行的模型

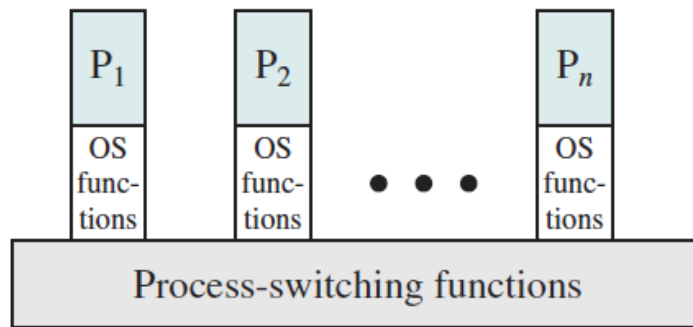
- 创建应用进程时，为其分配一个核心栈，用于运行操作系统的内核函数
- 应用程序调用操作系统的功能时进行模式切换

• 操作系统功能作为独立进程执行的模型

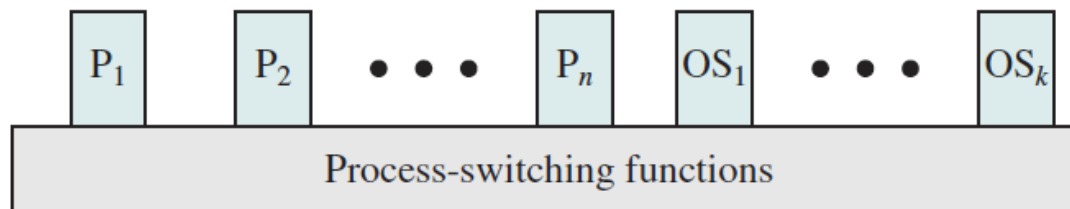
- 操作系统组织成一组系统进程
- 操作系统的小部分功能在核心态运行，大部分功能以独立进程的方式在用户态运行
- 有利于系统性能的提升



(a) Separate kernel



(b) OS functions execute within user processes



(c) OS functions execute as separate processes

Figure 3.15 Relationship between Operating System and User Processes

流行操作系统简介

- DOS操作系统
- Windows操作系统
- UNIX操作系统
- Linux操作系统
- IBM系列操作系统