

# 操作系统

韩明峰

QQ 2022446359



# 为什么要学习《操作系统》？

- (1) 操作系统的功能在软件中广泛应用。
- (2) 操作系统的方法，比如虚拟技术、缓冲技术、缓存技术、策略与机制相分离等，也在软件中广泛应用。
- (3) 我们非常渴望知道计算机在软件层面是如何运转的。



# 《操作系统》 经典教材

- [1] (荷) Andrew S. Tanenbaum 等著. 《现代操作系统》 (第4版).
- [2] (美) Abraham Silberschatz等著. 《操作系统概念》 (第9版).
- [3] (美) William Stallings著. 《操作系统：精髓与设计原理》 (第9版).
- [4] (美) Remzi H. Arpaci-Dusseau等著.  
《Operation Systems: Three Easy Pieces》.



# 第一章 操作系统引论

1.1 操作系统的作用

1.2 操作系统的发展过程

1.3 操作系统的基本特性

1.4 操作系统的主要功能

1.5 OS结构设计

# 1.1 操作系统的作用

## 1.1.1 操作系统的作用

### 1. OS作为用户与计算机硬件系统之间的接口

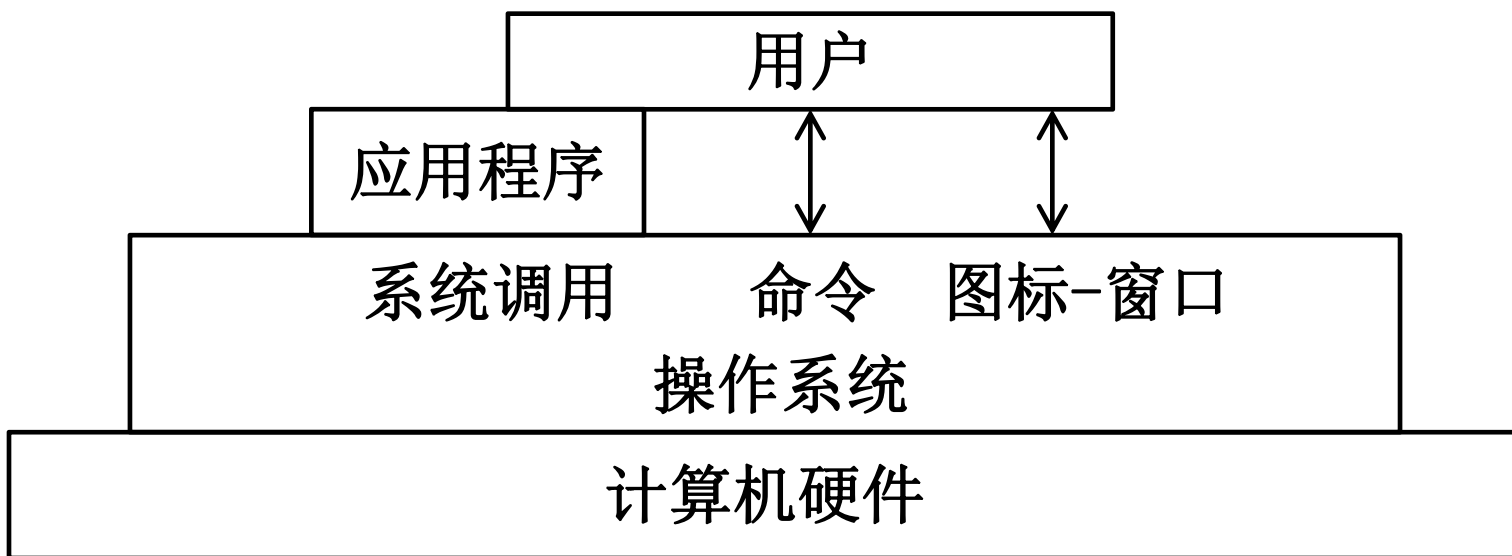


图1-1 OS作为接口的示意图



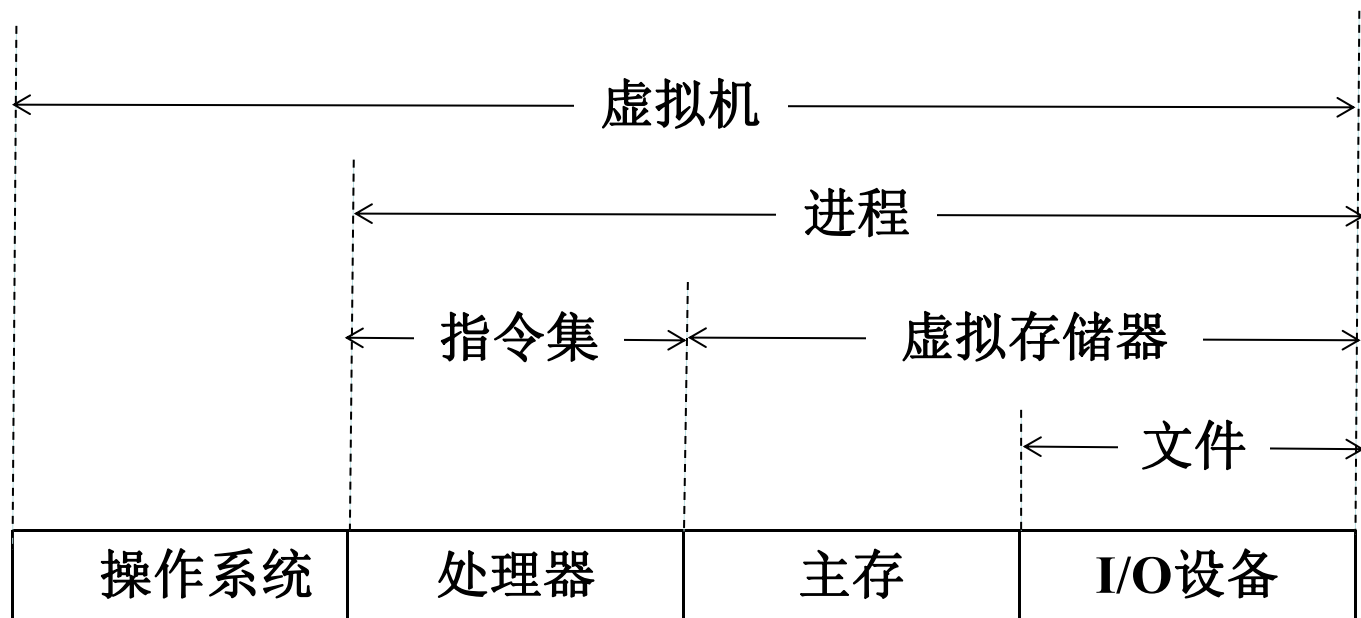
## 2. OS作为计算机系统资源的管理者

OS对处理机、存储器、I/O设备以及文件(数据和程序)进行管理。

## 3. OS实现了对计算机资源的抽象

计算机系统中的一个重大主题就是提供不同层次的抽象表示，来隐藏实现的复杂性。





计算机系统提供的一些抽象



## 1.1.2 推动操作系统发展的主要动力

1. 不断提高计算机资源利用率
2. 方便用户
3. 器件的不断更新换代
4. 计算机体系结构的不断发展：单CPU，多CPU，多计算机，分布式系统。
5. 不断提出新的应用需求





## 1.2 操作系统的发展过程

### 1.2.1 未配置操作系统的计算机系统

#### 1. 人工操作方式

由程序员将事先已穿孔的纸带(或卡片), 装入纸带输入机(或卡片输入机), 再启动他们将纸带(或卡片)上的程序输入计算机, 然后启动计算机运行。

仅当程序运行完毕并取走计算结果后, 才允许下一个用户上机。

## 2. 脱机输入/输出 (Off-Line I/O) 方式

为了解决人机矛盾及CPU和I/O设备之间速度不匹配的矛盾，20世纪50年代末出现了脱机I/O技术。

该技术利用外围机将低速输入设备的数据输入到高速磁盘，或者将高速磁盘上的数据输出到低速输出设备上。

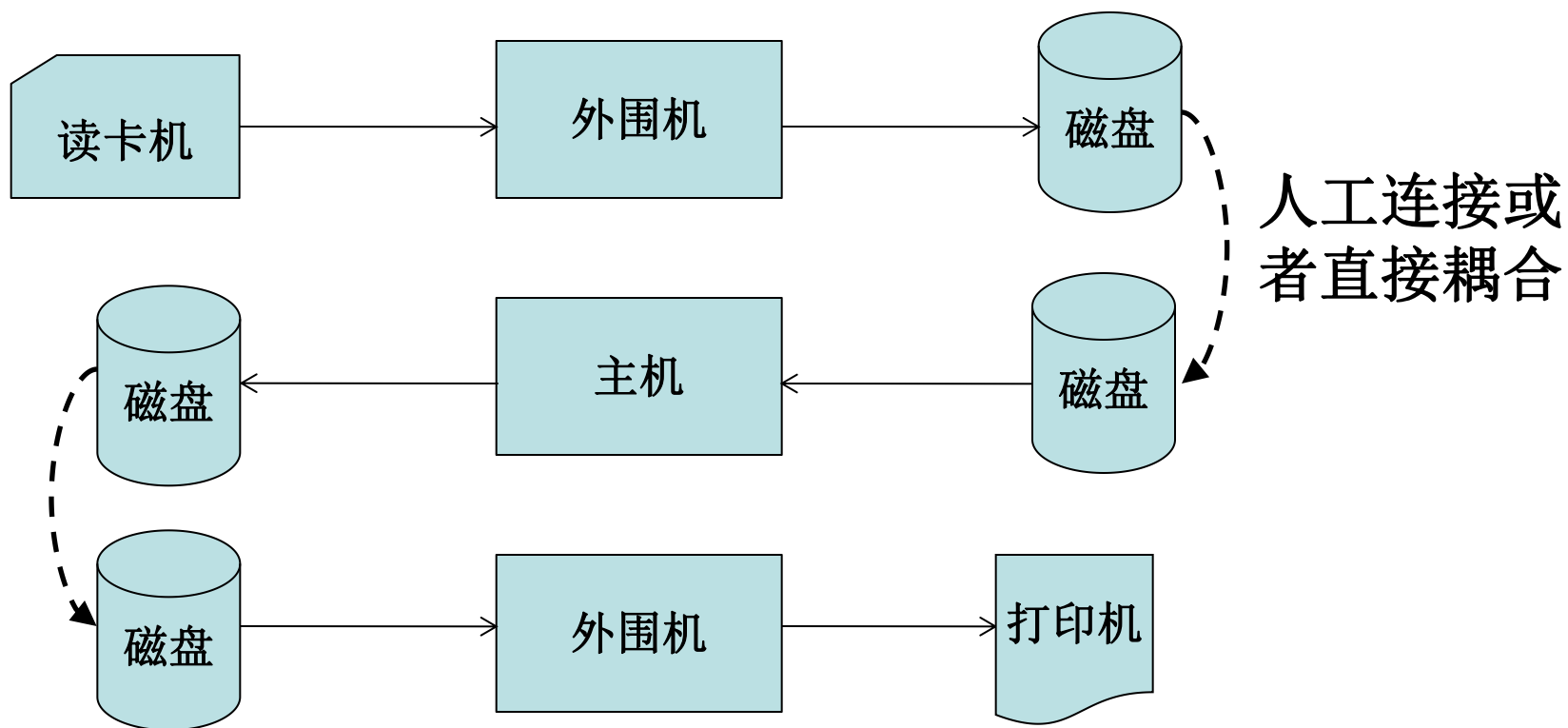


图1-3 脱机I/O示意图

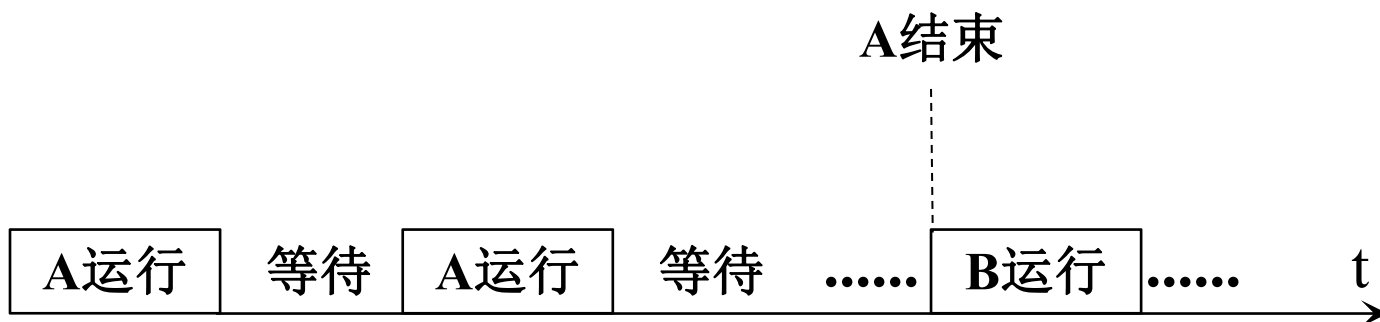
## 1.2.2 单道批处理系统

### 1. 单道批处理系统的处理过程

为实现对作业的连续处理，需要先把一批作业以脱机方式输入到磁带上，并在系统中配上监督(Monitor)程序，在它的控制下，使这批作业能一个接一个地连续处理。

### 2. 单道批处理系统的缺点

系统中的资源得不到充分的利用。



## 1.2.3 多道批处理系统

### 1. 多道程序设计的基本概念

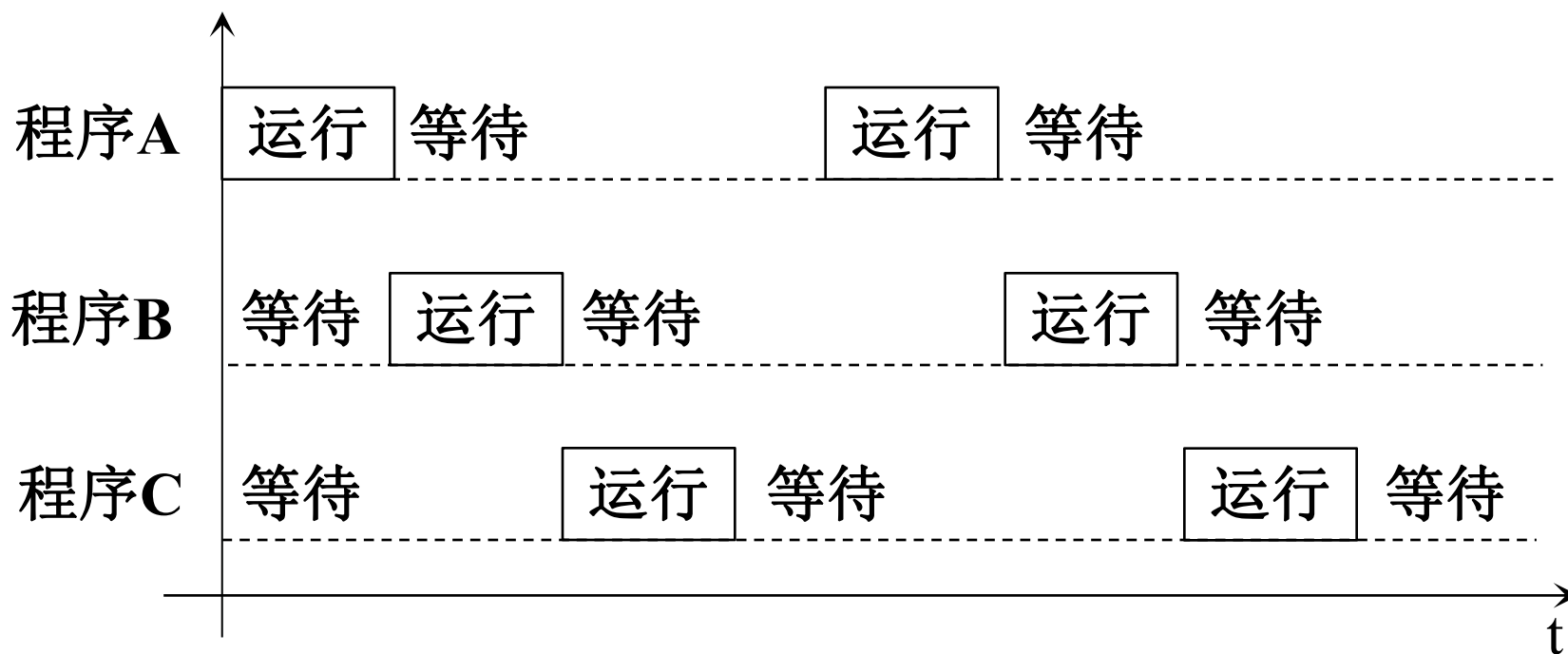


图1-6 多道程序的运行情况



## 2. 多道批处理系统的优缺点

(1) 资源利用率高。

(2) 系统吞吐量大。

(3) 平均周转时间长。由于作业要排队依次进行处理，因而作业的周转时间较长，通常需几个小时，甚至几天。

(4) 无交互能力。用户一旦把作业提交给系统后，直至作业完成，用户都不能与自己的作业进行交互，修改和调试程序极不方便。



## 1.2.4 分时系统(Time Sharing System)

### 1. 分时系统的引入

推动多道批处理系统形成和发展的主要动力是提高资源利用率和系统吞吐量；

推动分时系统形成和发展的主要动力，则是为了满足用户对人机交互的需求。



## 2. 分时系统实现中的关键问题

(1) 及时接收：当用户在自己的终端上键入命令时，系统应能及时接收。

(2) 及时处理：作业直接进入内存；采用轮转运行方式。






## 1.2.5 实时系统 (Real Time System)

系统的正确性，不仅由计算的**逻辑结果**来确定，而且还取决于**产生结果的时间**。

### 1. 实时性的概念

**开始截止时间**：某任务在某时间以前必须开始执行。

**完成截止时间**：某任务在某时间以前必须完成。

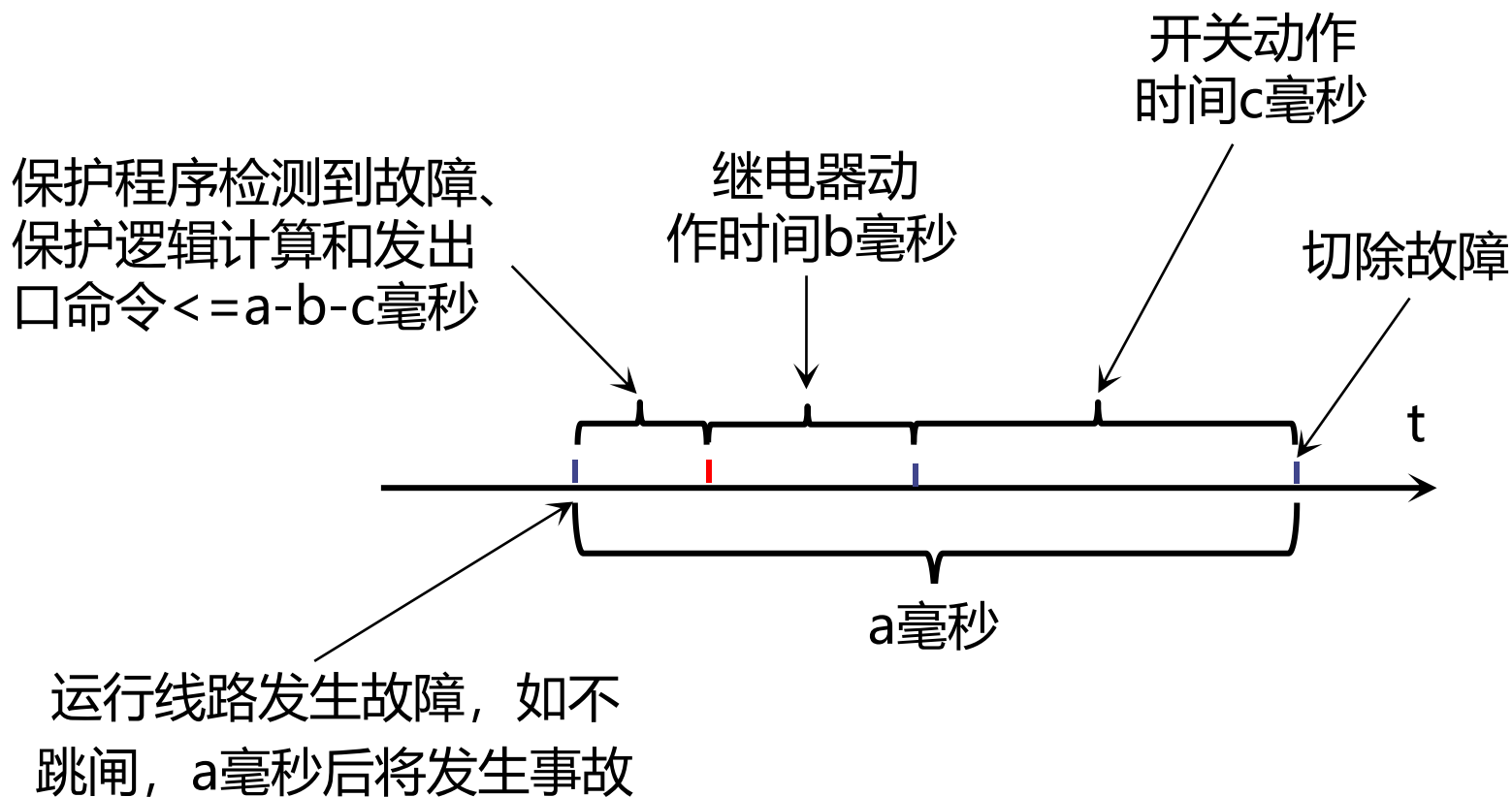


## 2. 实时任务的类型

**硬实时任务**：系统必须满足任务对截止时间的要求，否则可能出现难以预测的后果。

**软实时任务**：虽然也联系着一个截止时间，但并不严格，若偶尔错过了任务的截止时间，对系统产生的影响也不会太大。

### 3. 实时系统举例



电力系统继电保护跳闸实时性图示



## 1.2.6 微机操作系统的发展

### 1. 单用户单任务操作系统

如早期的DOS操作系统。

### 2. 单用户多任务操作系统

单用户操作系统是指一台计算机在同一时间只能由一个用户使用。Windows的一些客户端操作系统是单用户多任务操作系统。

### 3. 多用户多任务操作系统

如UNIX操作系统。



## 1.2.7 嵌入式操作系统

- (1) 嵌入式操作系统在用来控制设备的计算机中运行。
- (2) 嵌入式操作系统和实时操作系统往往是合在一起的。



## 1.2.8 网络操作系统

- (1) 网络操作系统的用户知道多台计算机的存在。
- (2) 网络操作系统与单处理机的操作系统没有本质区别。



## 1.2.9 分布式操作系统

(1) 分布式系统由全世界范围内完整的计算机通过网络连接在一起。

(2) 分布式操作系统是以一种传统单处理机操作系统的形式出现在用户面前。

(3) 分布式系统允许一个应用在在多个计算机上同时运行。



## 1.3 操作系统的基本特性

### 1.3.1 并发(Concurrence)

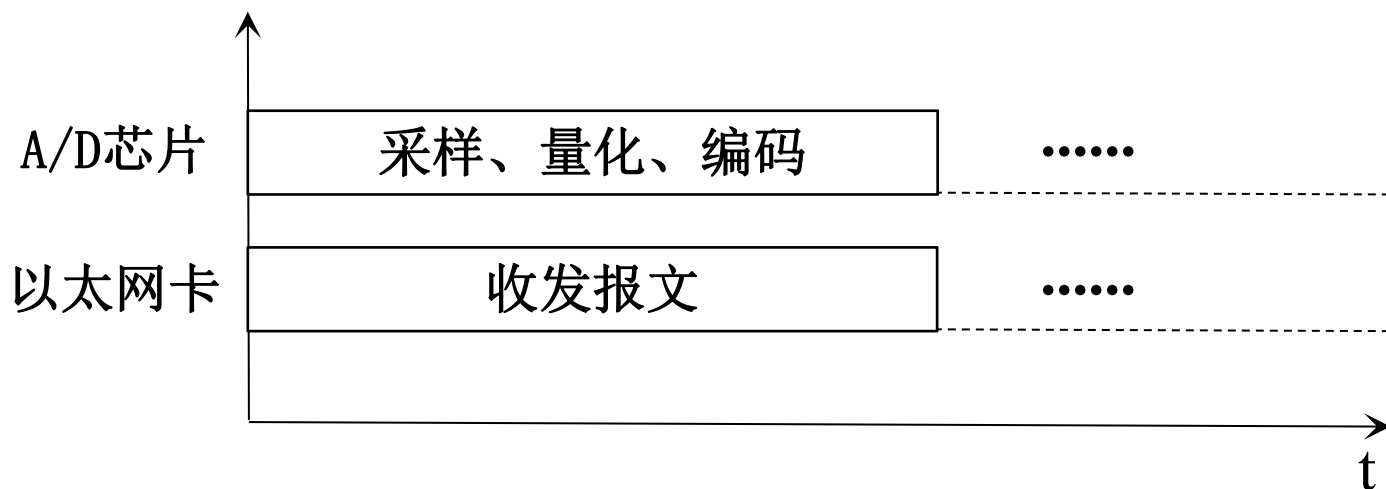
#### 1. 并行、并发与串行

- (1) **并行**：是指两个或多个事件在同一**时刻**发生。
- (2) **并发**：是指两个或多个事件在同一时间**间隔**内发生。
- (3) **串行**：是指两个或多个事件**依次**发生。



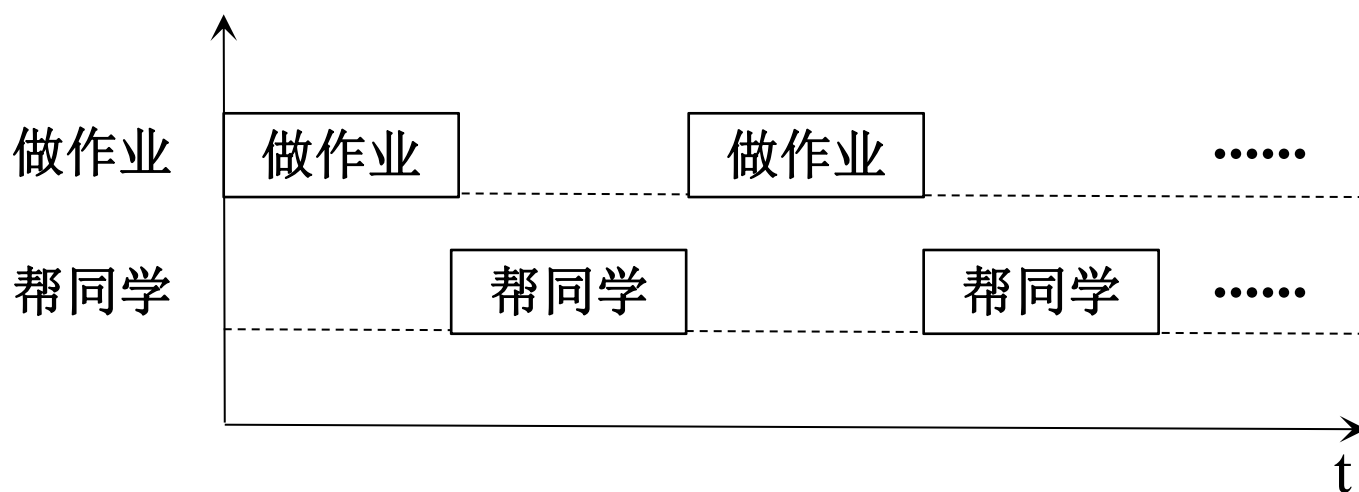
举例1：一个嵌入式系统的A/D芯片正在进行采样、量化和编码，而以太网卡正在进行通信报文的收发。

此例是并行、并发还是串行？



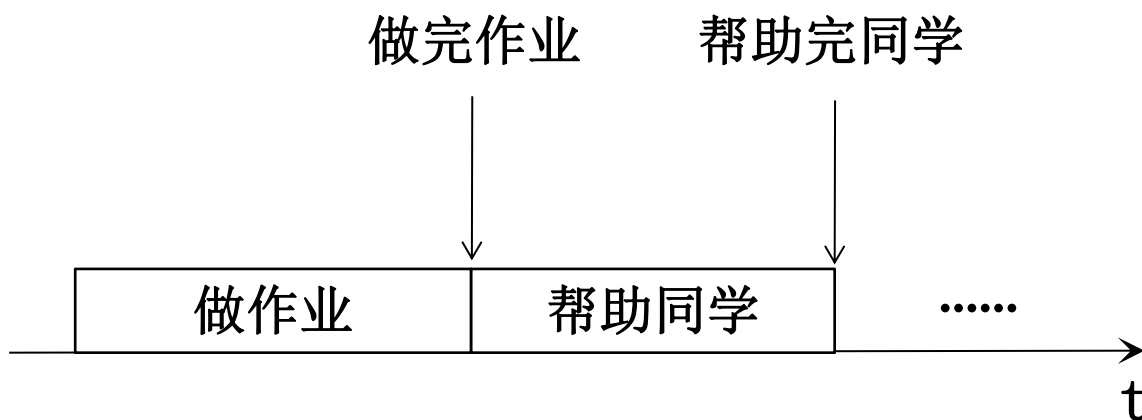
举例2：她一会儿做自己的作业，一会儿帮助同学解答问题，反复了好几次。

此例是并行、并发还是串行？



举例3：她先做完自己的作业，再帮助同学，然后两项活动都结束了。

此例是并行、并发还是串行？





## 2. 引入进程

若对内存中的多个程序都分别建立一个进程，它们就可以并发执行，这样便能极大地提高系统资源的利用率，增加系统的吞吐量。



### 1.3.2 共享(Sharing)

假设一个I/O操作是原子的。

#### 1. 互斥共享方式

如果一个逻辑上完整的数据通过多次I/O操作才能完成，那么就需要独占使用设备，其方式类似于申请互斥信号量。

如打印机等是互斥共享方式。

#### 2. 同时访问方式

一个逻辑上完整的数据通过一次I/O操作即可完成。此类设备总是能申请成功，但是将I/O请求排入队列，由驱动程序来处理（可能涉及I/O调度）。

典型的可供多个进程“同时”访问的资源是磁盘设备。



### 1.3.3 虚拟(Virtual)

将一个物理实体变为若干个逻辑上的对应物的功能称为虚拟。

#### 1. 时分复用技术

- (1) 虚拟处理机技术：多进程共享处理机。
- (2) 虚拟设备技术：SPOOLing系统。

#### 2. 空分复用技术

- (1) 通信领域的频分复用。
- (2) 计算机中对存储空间的空分复用。



### 1.3.4 异步 (Asynchronism)

由于资源等因素的限制，使进程的执行通常都不可能“一气呵成”，而是以“停停走走”的方式运行，即以人们不可预知的速度向前推进。



## 1.4 操作系统的主要功能

### 1.4.1 处理机管理功能

1. 进程控制
2. 进程同步
3. 进程通信
4. 调度

(1) 作业调度：根据算法将外存的作业调入内存，为它们创建进程、分配资源并放入就绪队列。

(2) 进程调度：根据算法，决定就绪队列中的哪个进程应获得处理机，恢复选中进程的处理机现场信息并运行。





## 1.4.2 存储器管理功能

### 1. 内存分配

内存分配的主要任务是：

- (1) 为每道程序分配内存空间，使它们“各得其所”。
- (2) 提高存储器的利用率，尽量减少不可用的内存空间（碎片）。
- (3) 允许正在运行的程序申请附加的内存空间，以适应程序和数据动态增长的需要。



## 2. 内存保护

(1) 绝不允许用户程序访问操作系统的程序和数据。

(2) 也不允许用户程序转移到非共享的其它用户程序中去执行。

## 3. 地址映射

利用内存管理单元MMU，将地址空间中的逻辑地址转换为内存空间中与之对应的物理地址。

## 4. 内存扩充

即虚拟存储技术，要求：

(1) 请求调入功能；

(2) 置换功能。





### 1.4.3 设备管理功能

设备管理的主要任务如下：

(1) 完成用户进程提出的I/O请求，为用户进程分配所需的I/O设备，并完成指定的I/O操作。

(2) 提高CPU和I/O设备的利用率，提高I/O速度，方便用户使用I/O设备。

为实现上述任务，设备管理应具有：

- (1) 缓冲管理；
- (2) 设备分配；
- (3) 设备驱动；
- (4) 虚拟设备等功能。



## 1.4.4 文件管理功能

1. 文件存储空间的管理
2. 目录管理
3. 文件的读/写管理和保护



## 1.4.5 操作系统与用户之间的接口

### 1. 操作系统与用户之间的接口

(1) 命令行接口

(2) 图形接口

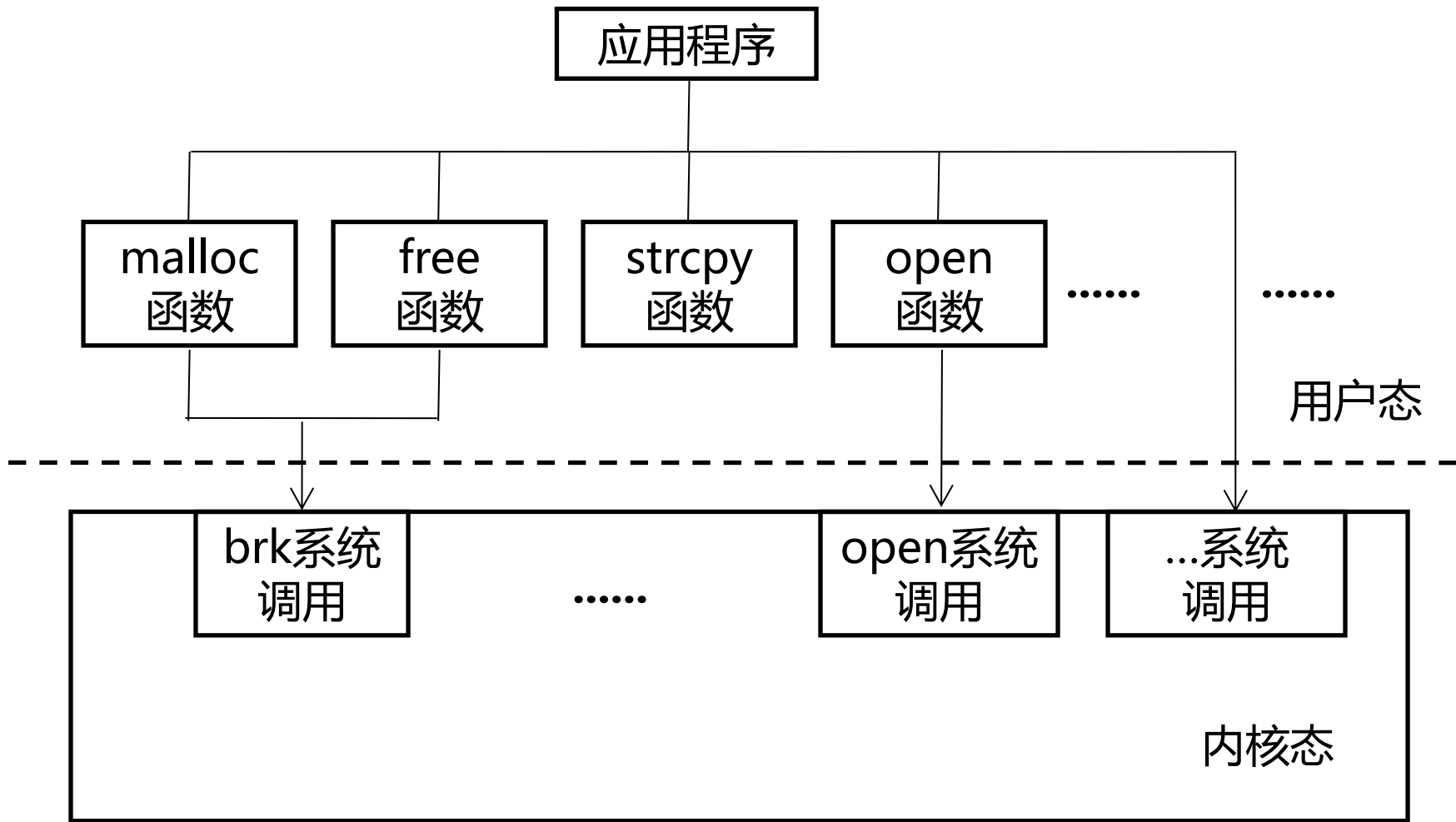
(3) 应用编程接口(API)。



## 2. 应用编程接口(API)

程序接口是用户程序取得操作系统服务的唯一途径。它是由一组系统调用组成的，每一个系统调用都是一个能完成特定功能的子程序。

**注意：**引发系统调用的实现机制是非常依赖于机器的，而且必须用**汇编代码**表达，所以通过提供**过程库**可使高级语言中能够使用系统调用。



库函数与系统调用的关系图



## 系统调用和库函数的关系：

(1) 系统调用通常只提供最基本的、最小的接口集合，而库函数常常提供更完全且更强大的功能。（体现了什么设计思想？）

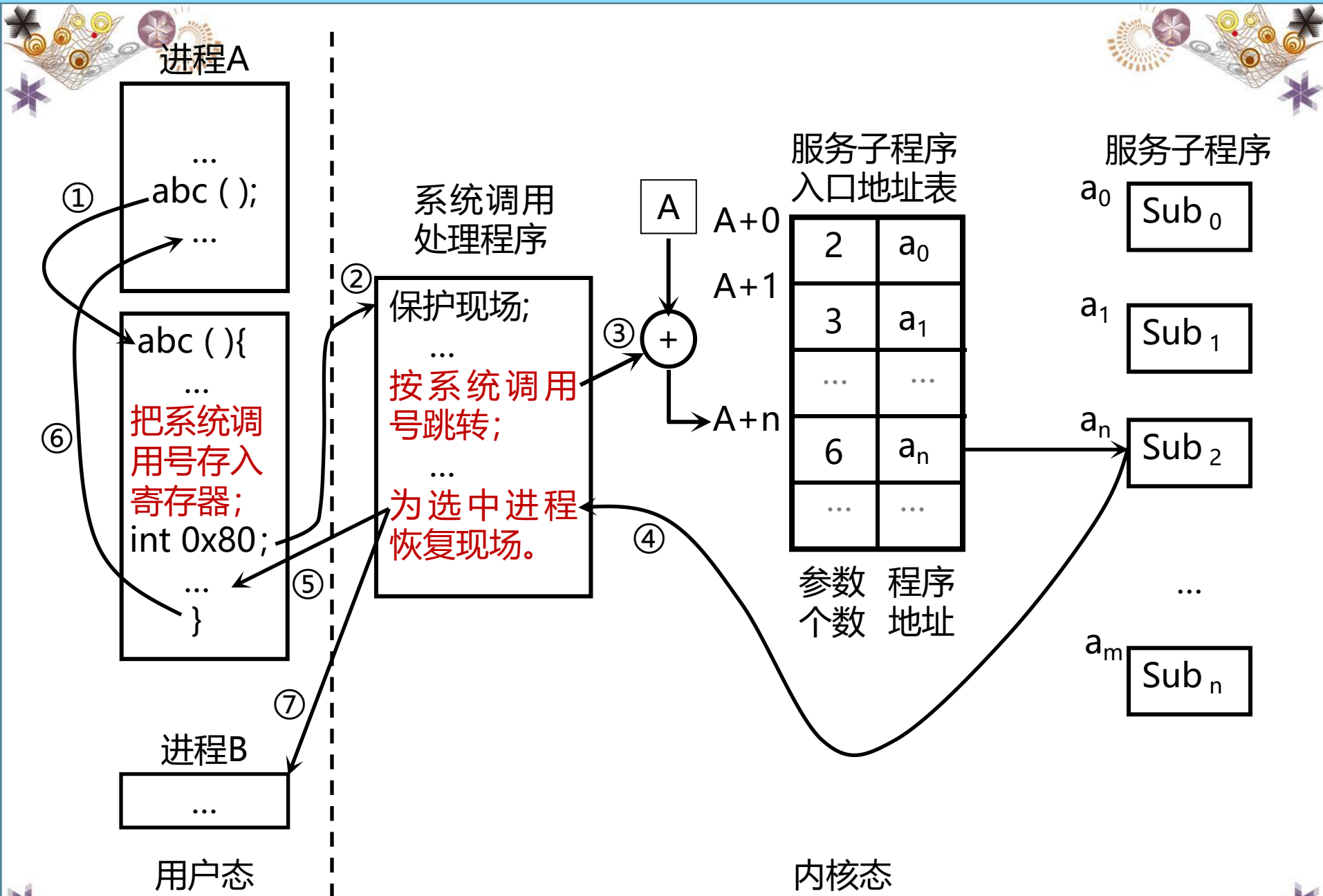
(2) 不同的库函数可以调用同一个系统调用。

(3) 一个库函数也可能调用多个系统调用。



(4) 更有些库函数并不依赖于任何系统调用。

(5) 在UNIX中，系统调用和系统调用所使用的库过程几乎是一一对应的；而在Windows中，库调用和实际的系统调用几乎是不对应的。





系统调用的执行过程示意图



**思考：**根据系统调用的执行过程示意图，说明系统调用和函数调用的区别。



## 1.5 OS结构设计

### 1.5.1 传统操作系统结构

#### 1. 无结构操作系统

OS是为数众多的一组过程的集合，每个过程可以任意地相互调用其它过程，致使操作系统内部既复杂又混乱，因此，这种OS是无结构的，也有人把它称为整体系统结构。





## 2. 模块化结构OS

### (1) 模块化程序设计技术的基本概念

模块化程序设计技术基于“分解”和“模块化”的原则来控制大型软件的复杂度。为使OS具有较清晰的结构，OS不再是由众多的过程直接构成的，而是按其功能精心地划分为若干个具有一定独立性和大小的模块。

### (2) 模块独立性

高内聚低耦合原则。



### (3) 模块化结构设计仍存在下述问题：

1) 在OS设计时，对各模块间的接口规定很难满足在模块设计完成后对接口的实际需求。

2) 模块化结构设计中，各模块的设计齐头并进，无法寻找一个可靠的决定顺序，造成各种决定的“无序性”，因此模块-接口法又被称为“无序模块法”。

### 3. 分层式结构OS

#### (1) 分层式结构的基本概念

分层法的设计任务是，在目标系统 $A_n$ 和裸机系统 $A_0$ 之间，铺设若干个层次的软件 $A_1$ 、 $A_2$ 、 $A_3$ 、...、 $A_{n-1}$ ，使 $A_n$ 通过 $A_{n-1}$ 、 $A_{n-2}$ 、...、 $A_2$ 、 $A_1$ 层，最终能在 $A_0$ 上运行。

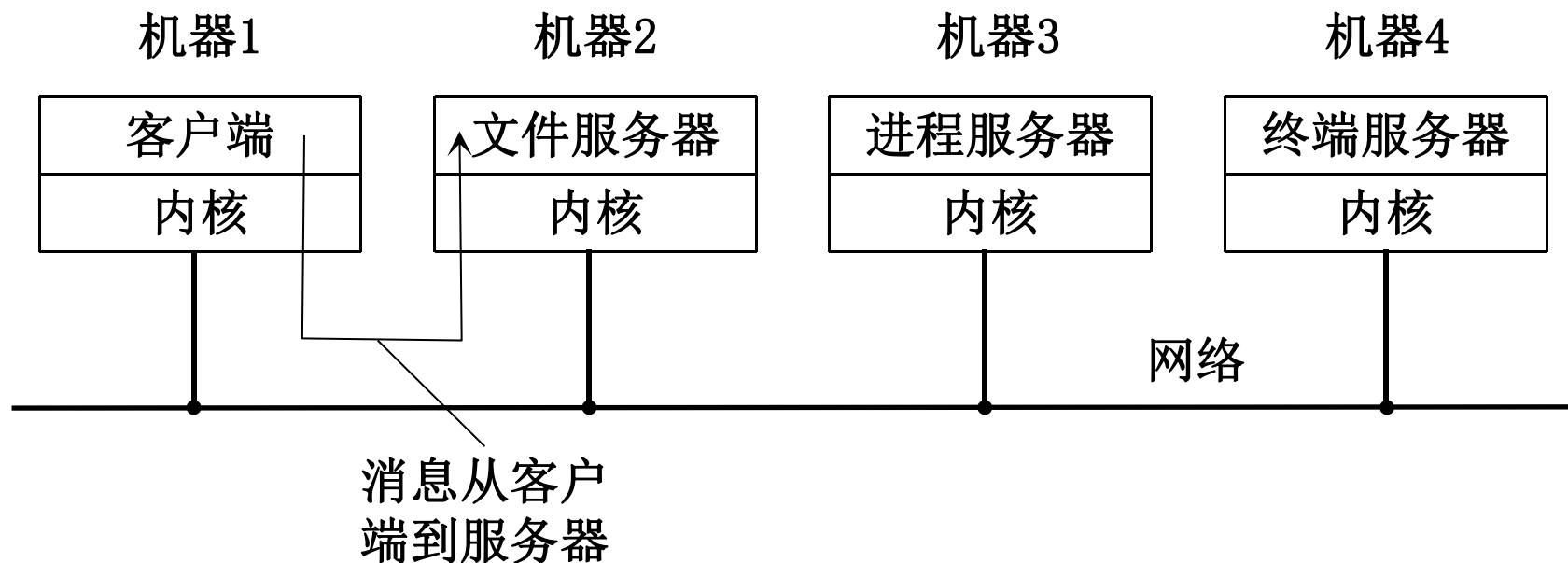
#### (2) 分层结构的缺点

由于层次结构是分层单向依赖的，OS每执行一个功能，通常要自上而下地穿越多个层次，这无疑会增加系统的通信开销，从而导致系统效率的降低。

## 1.5.2 客户/服务器模式 (Client/Server Model) 简介

### 1. 客户/服务器模式的组成

(1) 客户机。 (2) 服务器。 (3) 网络系统。



在网络上的客户端/服务器模型



## 2. 客户/服务器之间的交互

(1) 客户发送请求消息。

(2) 服务器接收消息。

(3) 服务器回送消息。


(4) 客户机接收消息。





### 3. 客户/服务器模式的优点

- (1) 数据的分布处理和存储。
- (2) 便于集中管理。
- (3) 灵活性和可扩充性。
- (4) 易于改编应用软件。



### 1.5.3 面向对象的程序设计技术简介

#### 1. 面向对象技术的基本概念

抽象；封装；继承；多态

#### 2. 面向对象技术的优点

- (1) 通过“重用”提高产品质量和生产率。
- (2) 使系统具有更好的易修改性和易扩展性。
- (3) 更易于保证系统的“正确性”和“可靠性”。

#### 3. OS利用面向对象技术举例：

同一个接口，使用不同的实例而执行不同的操作。

如UNIX系统“一切皆文件”的设计思想。





## 1.5.4 微内核OS结构

### 1. 微内核操作系统的基本概念

#### (1) 足够小的内核

微内核只是将操作系统中最基本的部分放入微内核，通常包含有：

- ① 与硬件处理紧密相关的部分；
- ② 一些较基本的功能；
- ③ 客户和服务端之间的通信。



## (2) 基于客户/服务器模式

将操作系统中最基本的部分放入内核中，而把操作系统的绝大部分功能都放在微内核外面的一组服务器(进程)中实现，运行在用户态。



客户与服务器之间是借助微内核提供的消息传递机制来实现信息交互的。

(3)应用“机制与策略相分离”的原理，将机制放入内核。

**机制**，是指实现某一功能的具体执行机构。

**策略**，则是在机制的基础上借助于某些参数和算法来实现该功能的优化，或达到不同的功能目标。

## (4) 采用面向对象技术





## 2. 微内核操作系统的优点

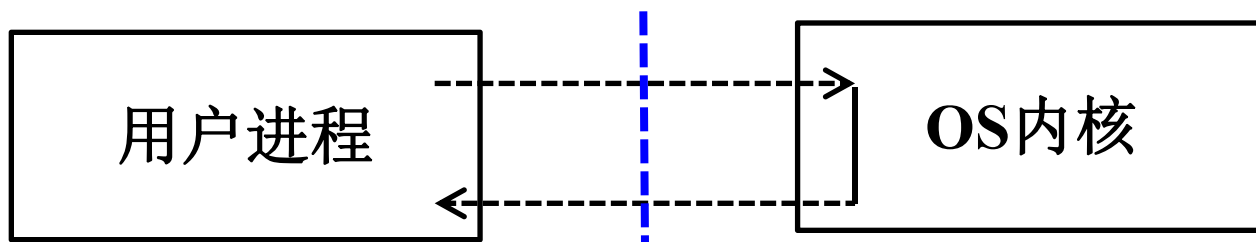
由于微内核OS结构是建立在模块化、层次化结构的基础上的，并采用了客户/服务器模式和面向对象的程序设计技术，因此，微内核结构的操作系统集各种技术优点之大成。



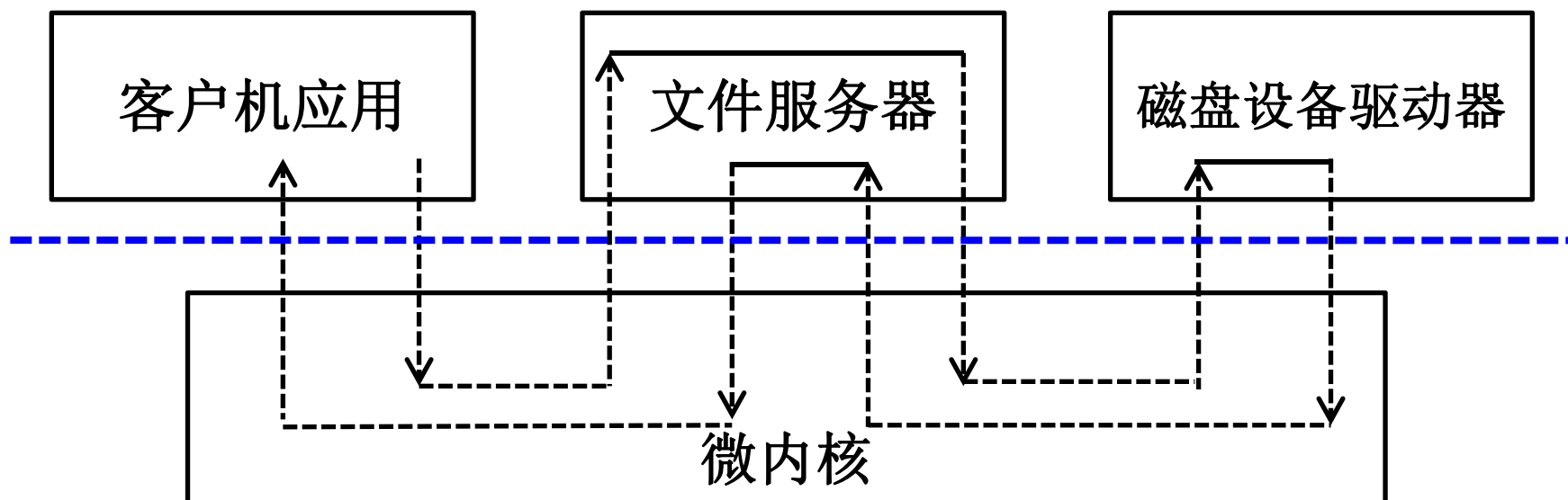
### 3. 微内核操作系统存在的问题

往往会引起更多的模式上下文切换。

例如，当某个服务器自身尚无能力完成客户请求而需要其它服务器的帮助时，如文件服务器还需要磁盘服务器的帮助，这时就需要进行8次模式上下文的切换。

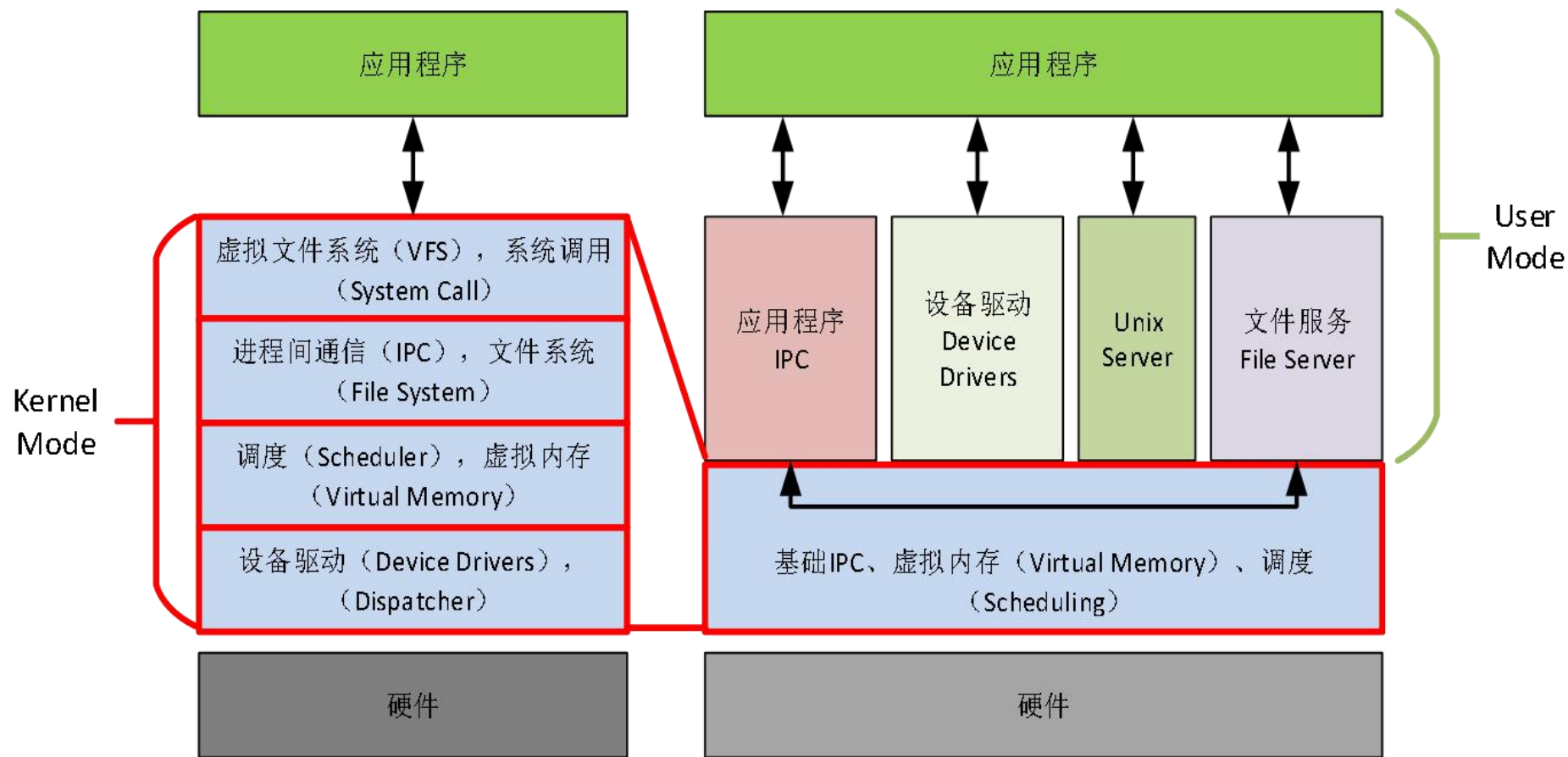


(1) 整体式内核中文件操作的模式上下文切换示意图



(2) 微内核中文件操作的模式上下文切换示意图





宏内核与微内核的对比示意图