



第一章 操作系统引论

1.1 操作系统的目标和作用

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

操作系统的概念

操作系统（OS）是配置在计算机硬件上的第一层软件，是对硬件系统的首次扩充。其主要作用是管理好这些设备，提高它们的利用率和系统的吞吐量，并为用户和应用程序提供一个简单的接口，便于用户使用。



1.1 操作系统的目标和作用

操作系统的目标与应用环境有关。

- 在查询系统中所用的OS，希望能提供良好的人—机交互性；
- 对于应用于工业控制、武器控制以及多媒体环境下的OS，要求其具有实时性；
- 而对于微机上配置的OS，则更看重的是其使用的方便性。

1.1.1 操作系统的目标

1. 方便性 微型计算机的OS更注重方便性
2. 有效性 大、中型计算机的OS更注重有效性
3. 可扩充性 采用新的微内核结构和客户服务器模式便于增加新的功能和模块
4. 开放性 遵循世界标准规范

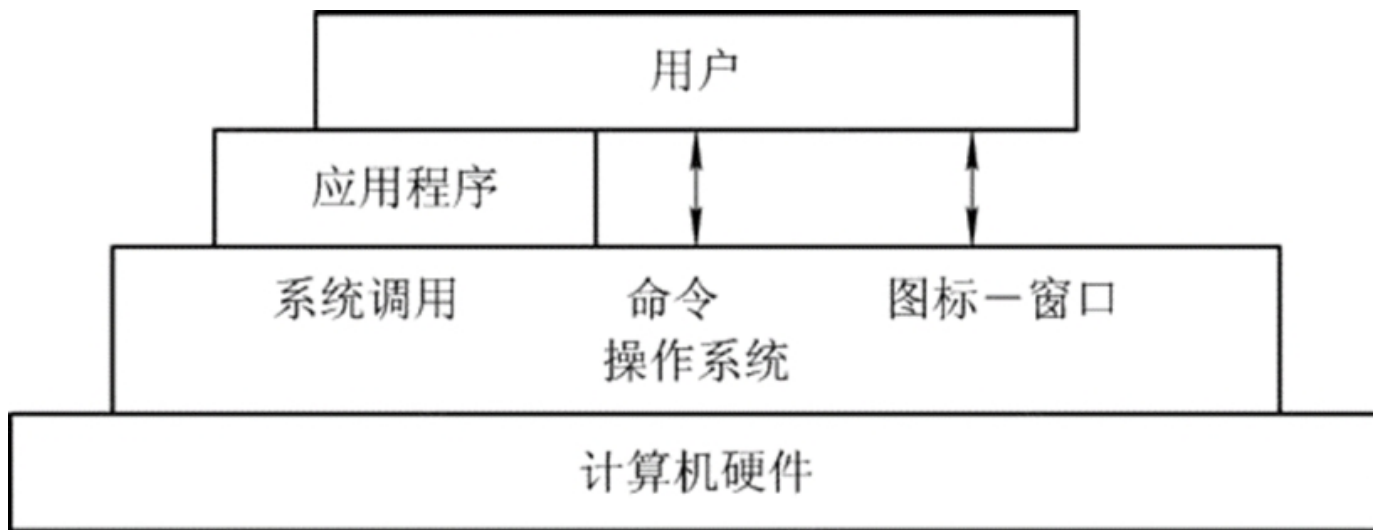
1.1.2 操作系统的作用



1. OS作为用户与计算机硬件系统之间的接口

OS处于用户与计算机硬件系统之间，用户通过OS来使用计算机系统。

用户在OS帮助下能够方便、快捷、可靠地操纵计算机硬件和运行自己的程序。



(1) 命令方式

由OS提供了一组联机命令（语言）。用户可通过键盘输入有关命令，来直接操纵计算机系统。

- DOS命令
- UNIX命令
- Linux命令

图 1-1 OS作为接口的示意图

(2) 系统调用方式

OS提供了一组系统调用，用户可在自己的应用程序中通过相应的系统调用，来操纵计算机。□

- DOS系统调用
- Windows系统调用
- UNIX系统调用
- Linux系统调用

图 1-1 OS作为接口的示意图

(3) 图标、窗口方式

用户通过屏幕上的窗口和图标来实现与操作系统的通信。

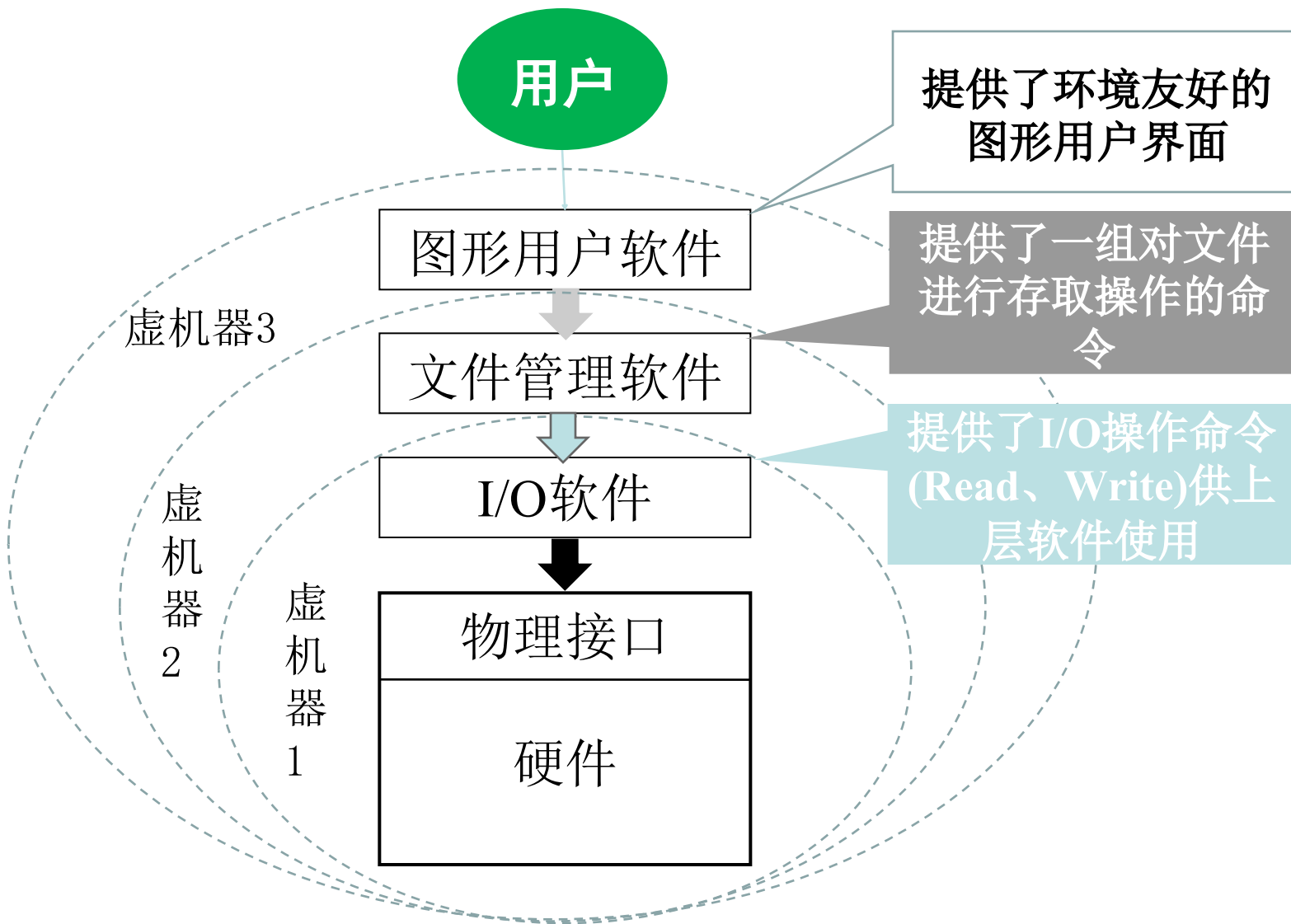


图 1-1 OS作为接口的示意图

2. OS作为计算机系统资源的管理者

资源	OS的资源管理的 功能	描述
处理器	处理器管理	分配和控制处理机
存储器	存储器管理	内存的分配与回收
I/O设备	I/O设备管理	I/O设备的分配与操纵
信息（数据和程序）	文件管理	文件的存取、共享与保护

3. OS实现了对计算机资源的抽象



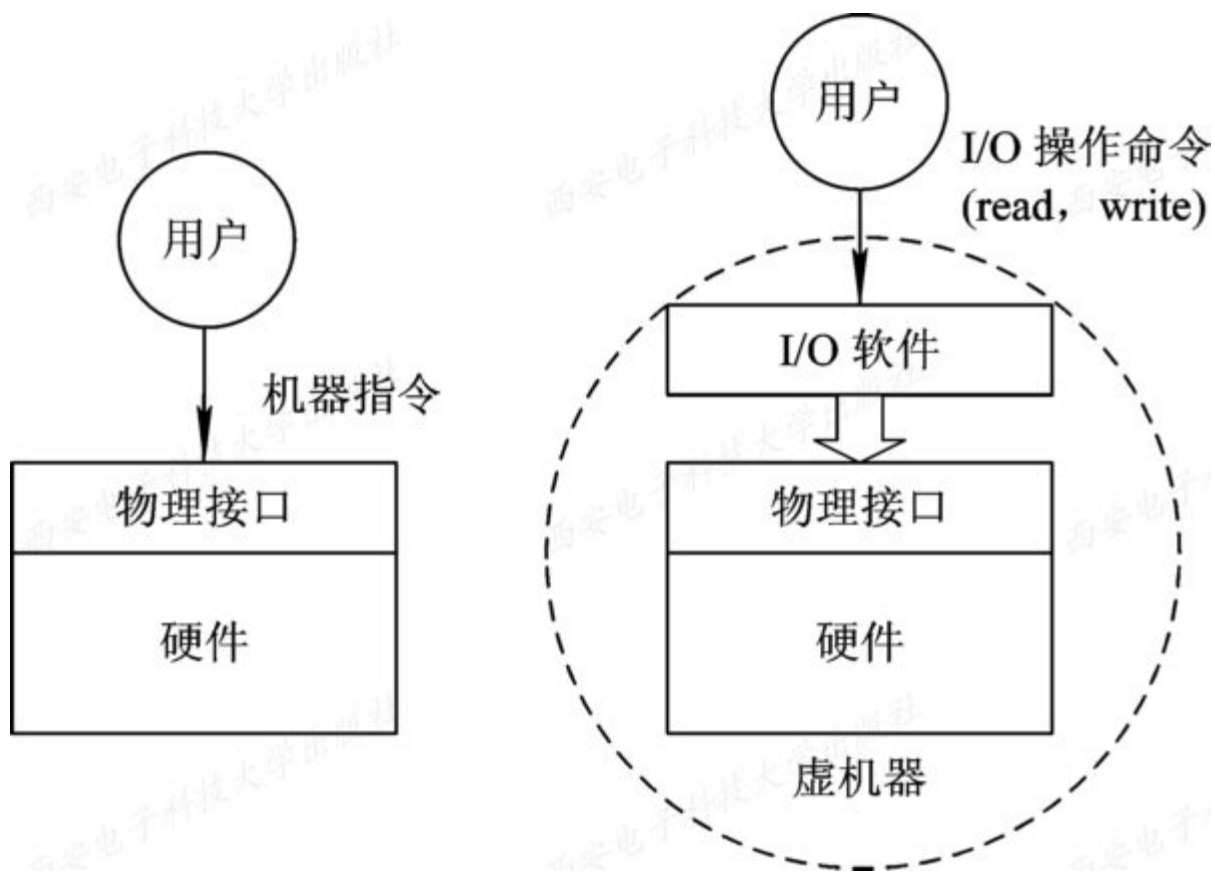




图1-2 I/O软件隐藏了I/O操作实现的细节



1.1.3 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
5. 不断提出新的应用需求

1.2 操作系统的发展过程

在20世纪50年代中期，出现了第一个简单的批处理OS；60年代中期开发出多道程序批处理系统；不久又推出分时系统，与此同时，用于工业和武器控制的实时OS也相继问世。20世纪70到90年代，是VLSI和计算机体系结构大发展的年代，导致了微型机、多处理机和计算机网络的诞生和发展，与此相应地，也相继开发出了微机OS、多处理机OS和网络OS，并得到极为迅猛的发展。



1.2.1 未配置操作系统的计算机系统

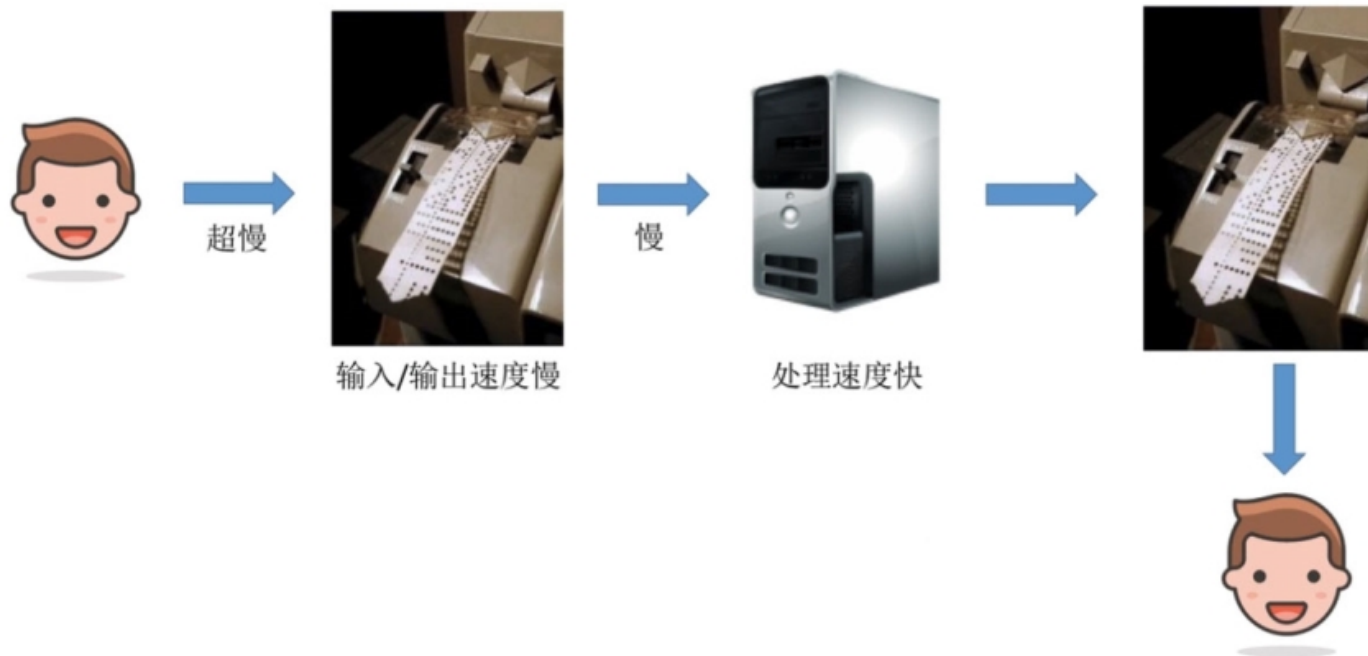
1. 人工操作方式

早期的操作方式是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), 再启动它们将纸带(或卡片)上的程序和数据输入计算机, 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。这种人工操作方式有以下两方面的缺点:

(1) 用户独占全机, 即一台计算机的全部资源由上机用户所独占。

(2) CPU等待人工操作。当用户进行装带(卡)、卸带(卡)等人工操作时, CPU及内存等资源是空闲的。

人工操作方式

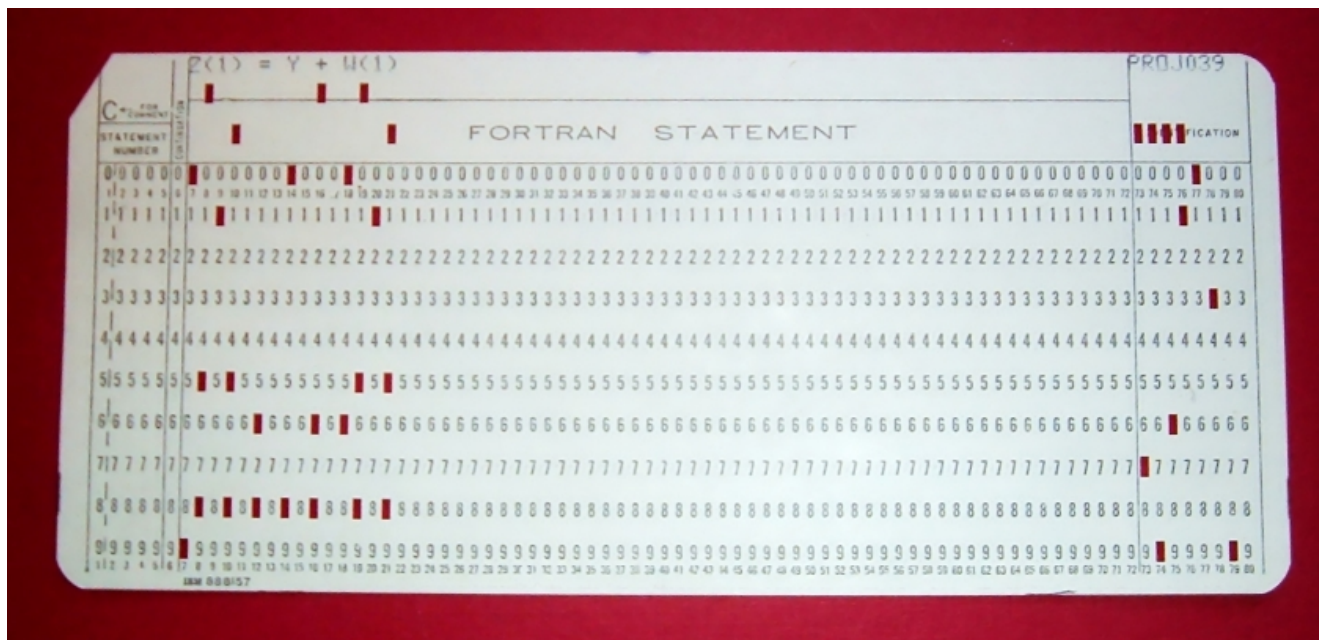




打孔纸带（5孔和8孔）



打孔纸带输入机



打孔卡片 (Fortran程序: $Z(1)=Y + W(1)$)



打孔卡片程序

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。该技术是事先将装有用户程序和数据的数据带装入纸带输入机，在一台外围机的控制下，把纸带(卡片)上的数据(程序)输入到磁带上。当CPU需要这些程序和数据时，再从磁带上高速地调入内存。

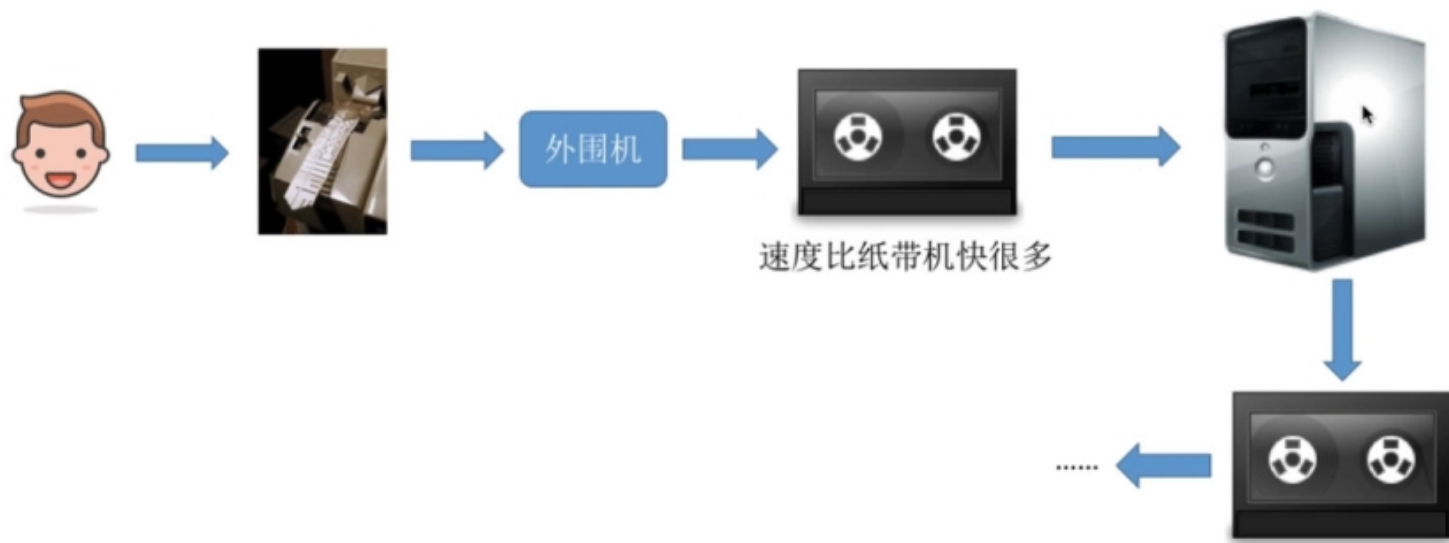


图1-3 脱机I/O示意图

说明：

一盘磁带上只有一个作业，通过装带和卸带顺序完成多个作业。



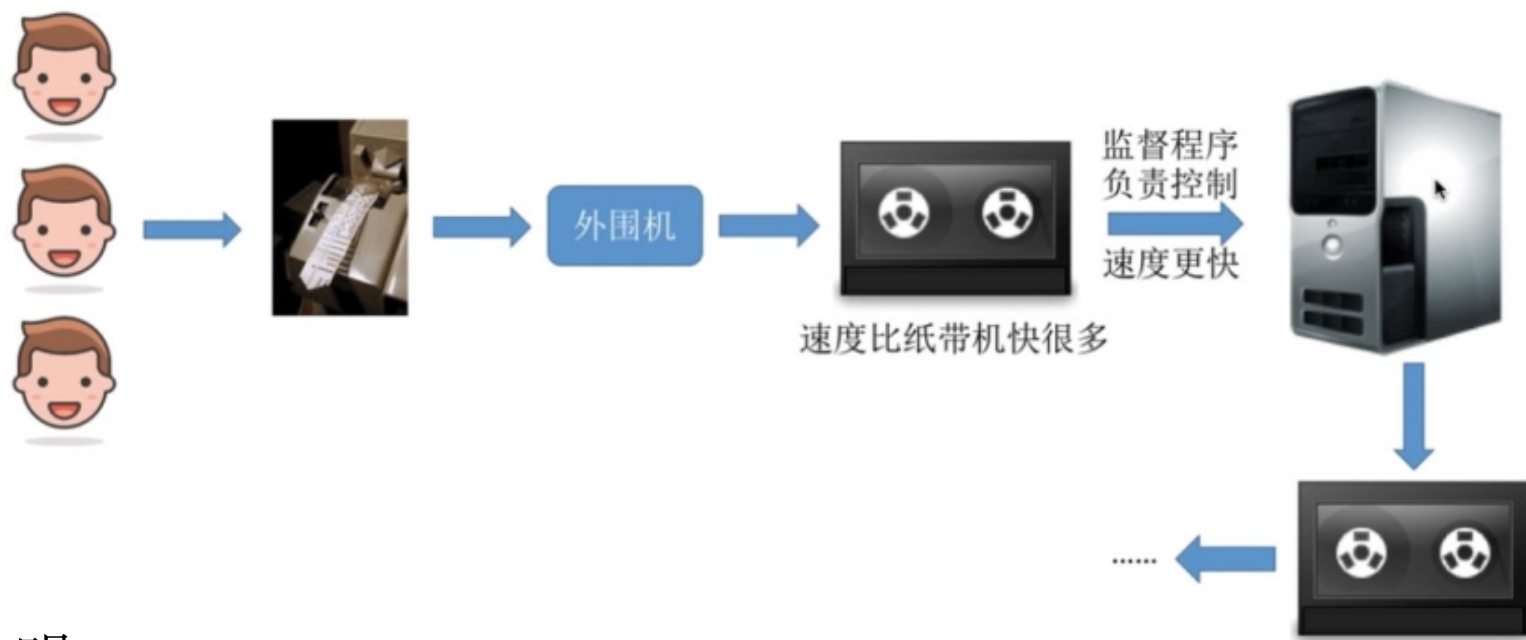
1.2.2 单道批处理系统

1. 单道批处理系统(Simple Batch Processing System)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**监督程序**(Monitor)，在它的控制下，使这批作业能一个接一个地连续处理。



单通道批处理系统示意图



说明：

一盘磁带上有多作业，通过监督程序控制作业的连续处理。

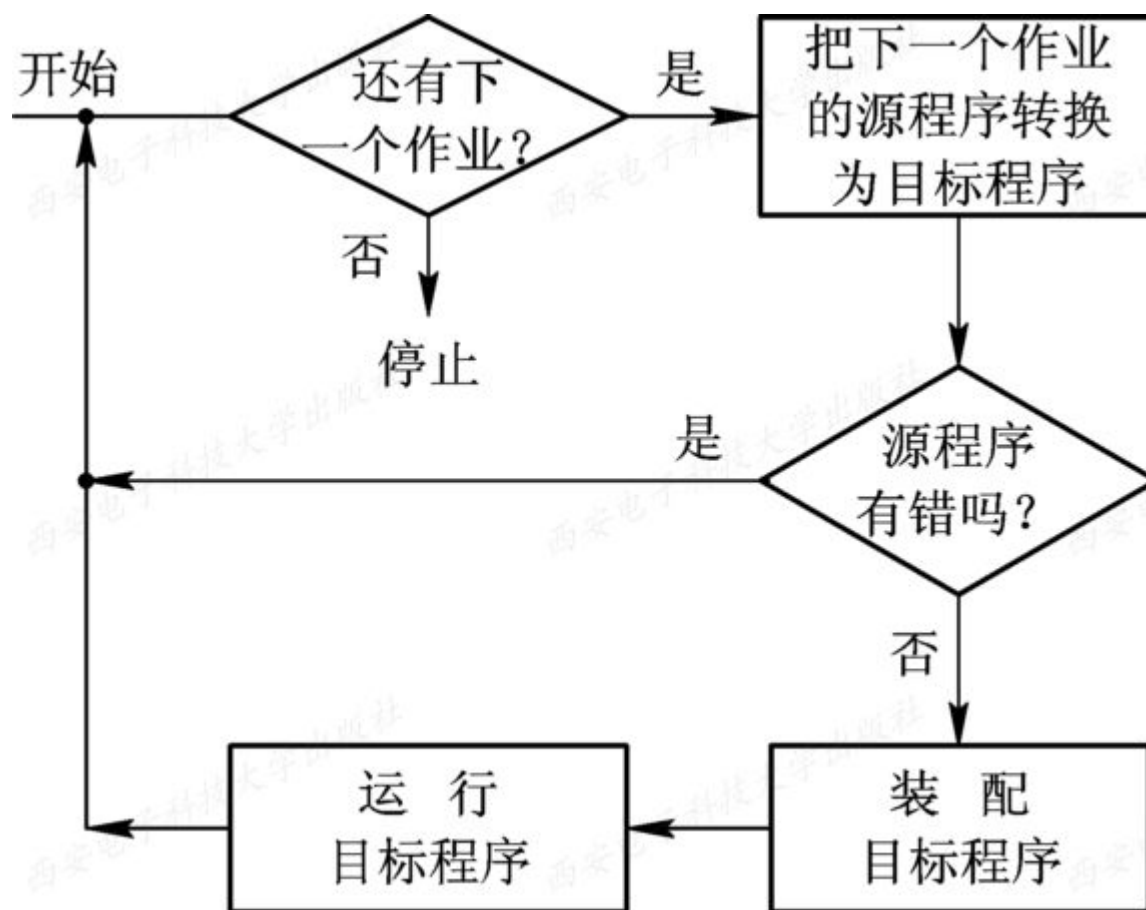


图1-4 单道批处理系统的处理流程

2. 单道批处理系统的缺点

单道批处理系统最主要的缺点是，系统中的资源得不到充分的利用。

这是因为在内存中仅有一道程序，每逢该程序在运行中发出I/O请求后，CPU便处于等待状态，必须在其I/O完成后才继续运行。又因I/O设备的低速性，更使CPU的利用率显著降低。图1-5示出了单道程序的运行情况，从图可以看出：在 $t_2 \sim t_3$ 、 $t_6 \sim t_7$ 时间间隔内CPU空闲。

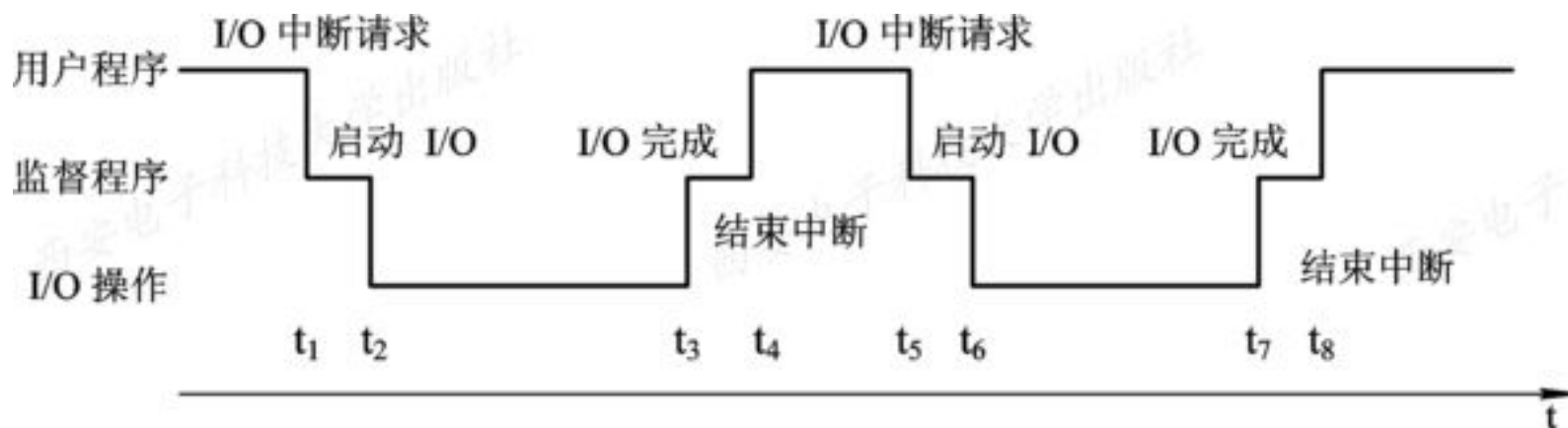


图1-5 单道程序的运行情况

举例

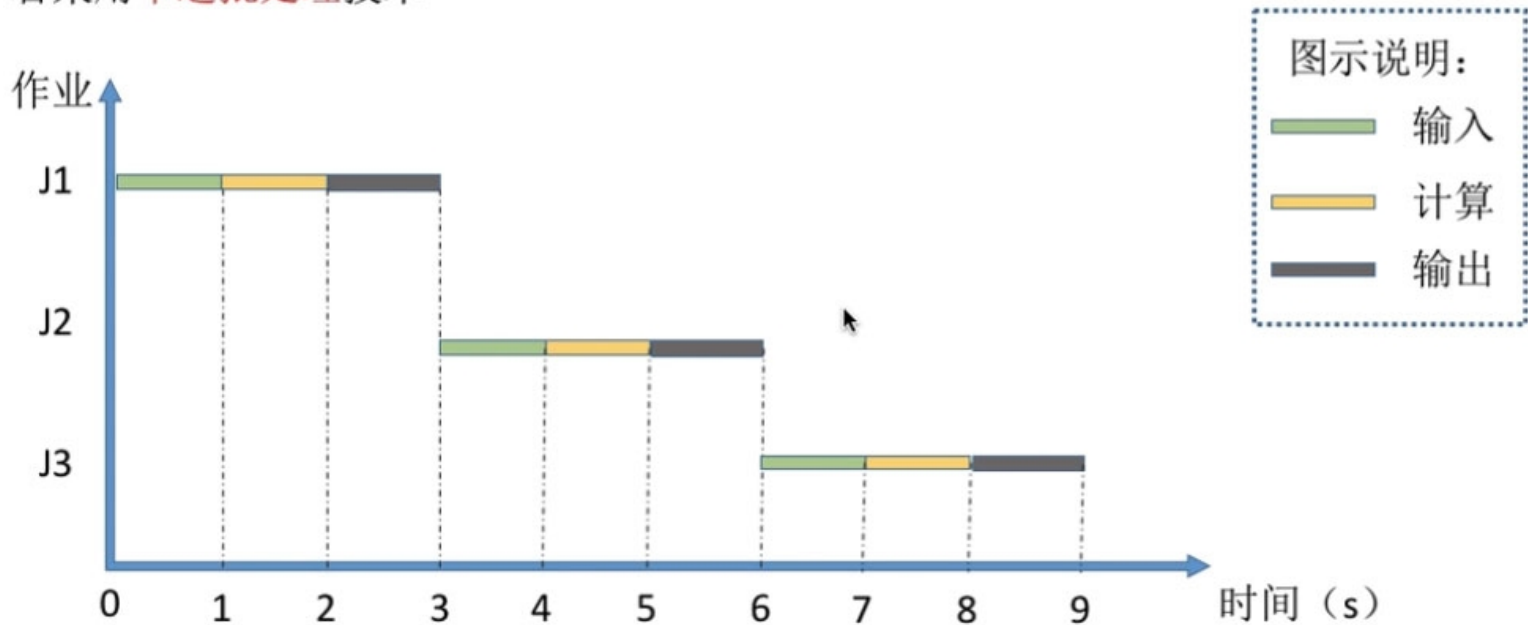
假设计算机需要处理三个作业

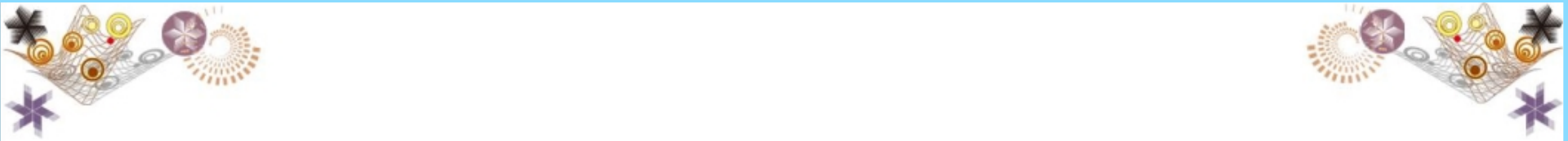
作业一：输入1秒，计算1秒，输出1秒

作业二：输入1秒，计算1秒，输出1秒

作业三：输入1秒，计算1秒，输出1秒

若采用单道批处理技术





1.2.3 多道批处理系统 (Multiprogrammed Batch Processing System)

1. 多道程序设计的基本概念

为了进一步提高资源的利用率和系统吞吐量，在20世纪60年代中期引入了多道程序设计技术，由此形成了多道批处理系统。

在该系统中，用户所提交的作业先存放在外存上，并排成一个队列，称为“后备队列”。然后由作业调度程序按照一定的算法，从后备队列选择若干个作业调入内存，使它们共享CPU和系统中的各种资源。内存中的多道程序可以交替地运行，使得CPU处于忙碌状态。



IBM System/360 Model 65 的操作台，其中有寄存器数值指示灯、调节开关（中间）与紧急红色开关（右上方）



IBM System/360 Model 30: CPU（红色中间部分），左侧磁带驱动器，右侧磁盘驱动器



-
- 吐量
- 磁盘
- CPU
- 作业1
作业2
⋮
作业n
- 磁盘
- 输入设备
- 读卡机
- 后备作业
- 输出设备
- 打印机
- 后备队列
- 不一定按顺序执行，

(1) 提高CPU的利用率

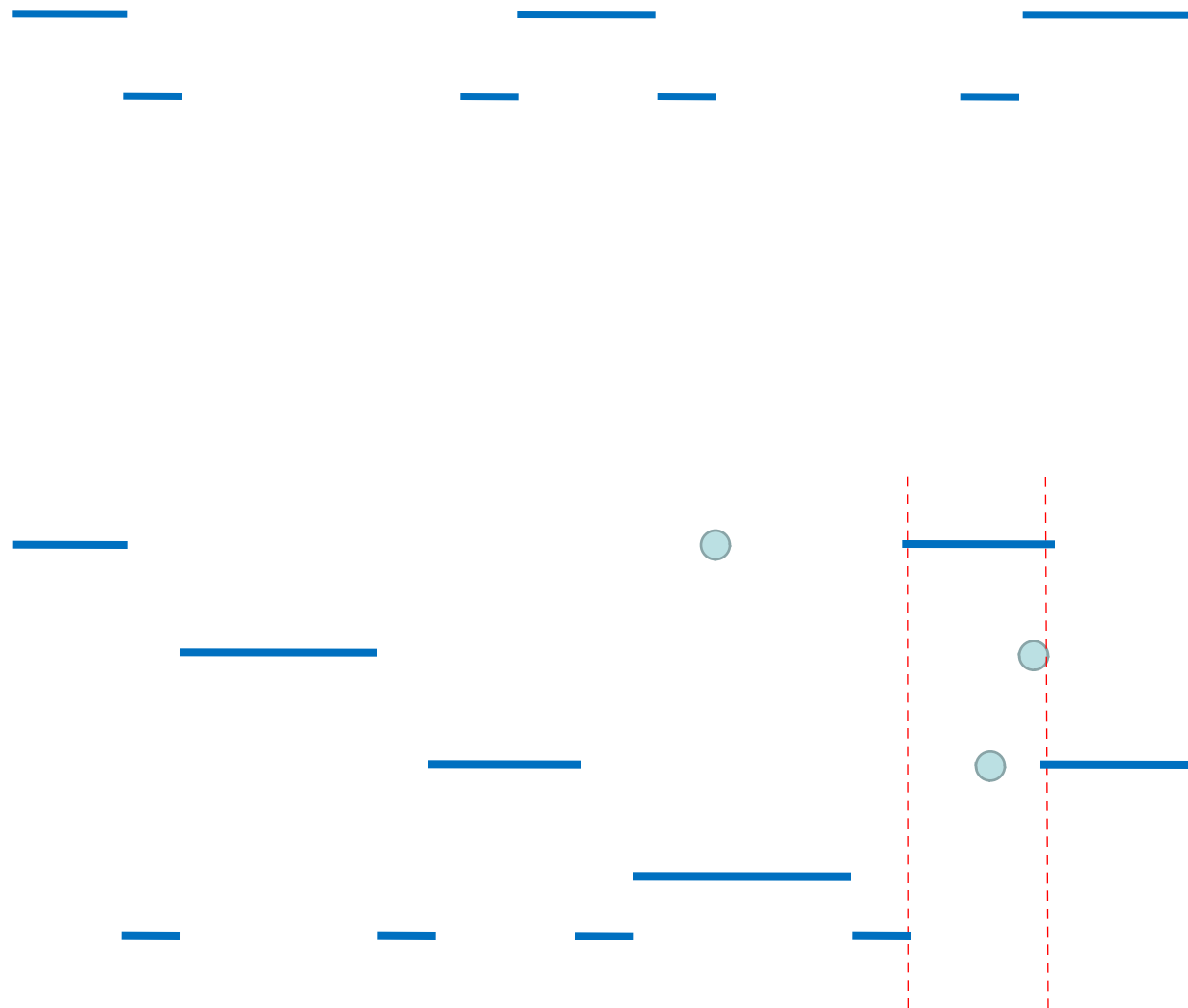


图 1-4 单道和多道程序运行情况



(2) 可提高内存和I/O设备利用率

为了能运行较大的作业，通常**内存都具有较大容量**，但由于80%以上的作业都属于**中小型**，因此在单道程序环境下，也必定造成**内存的浪费**。

对于系统中所配置的**多种类型的I/O设备**，在**单道程序**环境下也不能充分利用。

如果允许在内存中装入多道程序，并允许它们并发执行，会大大提高内存和I/O设备的利用率。□



(3) 增加系统吞吐量

在保持CPU、I/O设备不断忙碌的同时，也必然会大幅度地提高系统的吞吐量，从而降低作业加工所需的费用。

举例

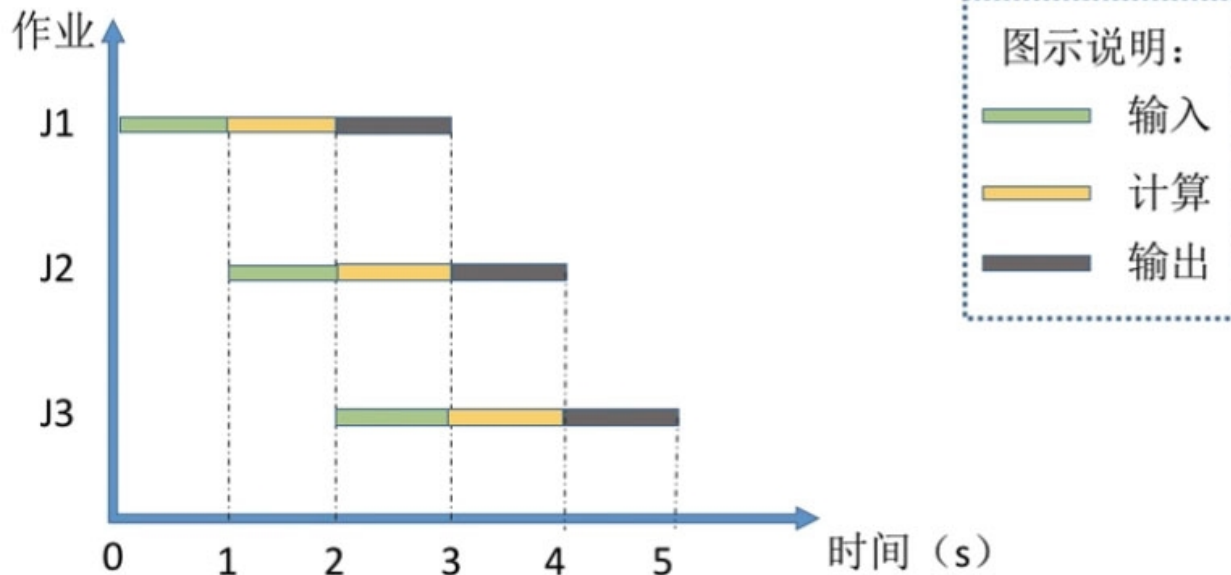
假设计算机需要处理三个作业

作业一：输入1秒，计算1秒，输出1秒

作业二：输入1秒，计算1秒，输出1秒

作业三：输入1秒，计算1秒，输出1秒

若采用**多道批处理**技术



2. 多道批处理系统的优缺点

多道批处理系统的优缺点如下：

(1) 资源利用率高。引入多道批处理能使多道程序交替运行，以保持CPU处于忙碌状态；在内存中装入多道程序可提高内存的利用率；此外还可以提高I/O设备的利用率。

(2) 系统吞吐量大。能提高系统吞吐量的主要原因可归结为：① CPU和其它资源保持“忙碌”状态；② 仅当作业完成时或运行不下去时才进行切换，系统开销小。

系统吞吐量： 系统在单位时间内完成的总工作量



(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。

作业周转时间：作业从进入系统开始，直至其完成并退出系统为止所经历的时间。







3. 多道批处理系统需要解决的问题

为使系统中的多道程序间能协调地运行，系统必须解决下述一系列问题：

(1) 处理机争用问题。既要能满足各道程序运行的需要，又要能提高处理机的利用率。

(2) 内存分配和保护问题。系统应能为每道程序分配必要的内存空间，使它们“各得其所”，且不会因某道程序出现异常情况而破坏其它程序。

(3) I/O设备分配问题。系统应采取适当的策略来分配系统中的I/O设备，以达到既能方便用户对设备的使用，又能提高设备利用率的目的。



(4) 文件的组织和管理问题。系统应能有效地组织存放在系统中的大量的程序和数据，使它们既便于用户使用，又能保证数据的安全性。

(5) 作业管理问题。系统中存在着各种作业(应用程序)，系统应能对系统中所有的作业进行合理的组织，以满足这些作业用户的不同要求。

(6) 用户与系统的接口问题。为使用户能方便的使用操作系统，OS还应提供用户与OS之间的接口。

1.2.4 分时系统

1. 分时系统(Time-Sharing System)的产生□

推动分时系统形成和发展的主要动力是**用户的需求**。它与**多道批处理系统**之间，有着**截然不同的性能差别**。用户的需求具体表现在以下几个方面：□

- (1) **人一机交互**
- (2) 共享主机
- (3) 便于用户上机

使用终端传送和控制
作业



2. 分时系统实现中的关键问题

为实现分时系统，其中，最关键的问题是如何使用户能与自己的作业进行交互，即当用户在自己的终端上键入命令时，系统应能及时接收并及时处理该命令，再将结果返回给用户。此后，用户可继续键入下一条命令，此即人一机交互。应强调指出，即使有多个用户同时通过自己的键盘键入命令，系统也应能全部地及时接收并处理。



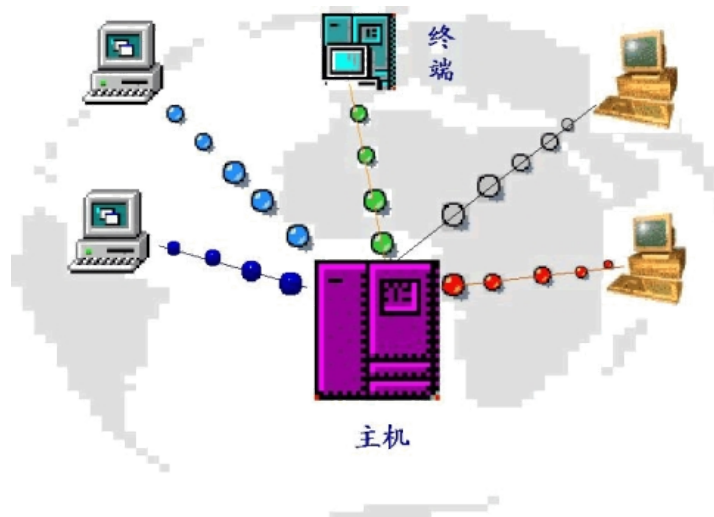
(1) 及时接收

(2) 及时处理

作业直接进入内存，否则用户作业无法运行，用户不能和主机交互，不允许一个作业长期占用处理机，否则其他用户无法和主机交互，解决方法——设置时间片。

3. 分时系统的特征

- (1) 多路性
- (2) 独立性
- (3) 及时性
- (4) 交互性



1.2.5 实时系统

所谓“实时”，是表示“及时”，而**实时系统(Real-Time System)**是指系统能**及时(或即时)响应外部事件的请求**，在**规定的时间内**完成对该事件的处理，并控制所有**实时任务协调一致地运行**。

1. 应用需求

(1) 实时控制 自动控制、自动驾驶、导弹制导等

(2) 实时信息处理 飞机票、火车票订票系统

(3) 多媒体系统 多媒体视听系统

(4) 嵌入式系统 智能仪器和设备



2. 实时任务

1) 按任务执行时是否呈现周期性来划分：□

(1) 周期性实时任务

(2) 非周期性实时任务

外部设备所发出的激励信号并无明显的周期性，但都必须联系着一个截止时间(Deadline)。它又可分为：

① **开始截止时间**——任务在某时间以前必须**开始**执行；

② **完成截止时间**——任务在某时间以前必须**完成**。



2) 根据对截止时间的要求来划分□

(1) **硬实时任务**(hard real-time task)。系统必须满足任务对**截止时间**的**严格**要求，否则可能出现难以预测的结果。

(2) **软实时任务**(Soft real-time task)。它也联系着一个**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. 实时系统与分时系统特征的比较

	实 时 系 统	分 时 系 统
多路性	体现在对多路的现场信息进行采集、对多个对象或多个执行机构进行控制。	按分时原则为多个终端用户服务。
独立性	每个终端用户向系统提出服务请求时，彼此独立操作，互不干扰；对信息的采集和对象的控制也彼此互不干扰。	每个用户各占一个终端，彼此互不干扰，独立操作。
及时性	以控制对象所要求的开始截止时间或完成截止时间来确定，一般为秒级到毫秒级。	用户的请求能在短时间（用户能接受的）内获得响应。
交互性	人与系统的交互，仅限于访问系统中某些特定的专用服务程序。	用户与系统进行广泛的人机对话，系统能向终端用户提供数据处理服务，资源共享等服务。
可靠性	高度可靠	可靠程度较低



1.2.6 微机操作系统的发展

1. 单用户单任务操作系统：只允许一个用户上机，且只允许用户程序作为一个任务运行。

1) CP/M: Digital Research公司1975年推出，具有较好的体系结构，可适应性强，可移植性以及易学易用等优点，在8位微机中占据了统治地位。

2) MS-DOS: Microsoft公司开发，用于1981年IBM公司推出的IBM-PC机上（16位微机），后又推出多个版本，可以用在Intel 80286, 80386, 80486等微机上。



2. 单用户多任务操作系统

单用户多任务操作系统的含义是，只允许一个用户上机，但允许用户把程序分为若干个任务，使它们并发执行，从而有效地改善了系统的性能。

代表性的操作系统是Windows系列。

3. 多用户多任务操作系统

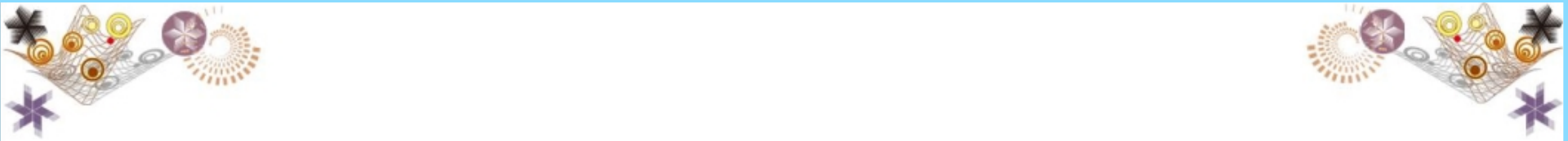
多用户多任务操作系统的含义是，允许多个用户通过各自的终端，使用同一台机器，共享主机系统中的各种资源，而每个用户程序又可进一步分为几个任务，使它们能并发执行，从而可进一步提高资源利用率和系统吞吐量。

在大、中和小型机中所配置的大多是多用户多任务操作系统，而在32位微机上，也有不少配置的是多用户多任务操作系统，其中最有代表性的是UNIX OS。



1.3 操作系统的基本特性

- 并发
- 共享
- 虚拟
- 异步



1.3.1 并发 (Concurrence)

1. 并行与并发

并行性和并发性是既相似又有区别的两个概念。并行性是指两个或多个事件在**同一时刻**发生。而并发性是指两个或多个事件在**同一时间间隔**内发生。在多道程序环境下，并发性是指在一段时间内，**宏观上有多个程序**在同时运行。

单个处理机（单核CPU）同一时刻只能执行一个程序，各个程序只能并发地执行。多个处理机（多核CPU）同一时刻可以同时执行多个程序，多个程序可以并行地执行。

2. 引入进程

在为计算程序和I/O程序分别建立一个**进程(Process)**后，这两个进程便可并发执行。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。



进程是指在系统中**能独立运行并作为资源分配的基本单位**，它是由**一组机器指令、数据和堆栈**等组成的，是一个能独立运行的**活动**实体。多个进程之间可以并发执行和**交换信息**。一个进程在运行时需要一定的资源，如**CPU、存储空间及I/O设备**等。

3. 引入线程

通常在一个**进程**中可以包含**若干个线程**，它们可以利用进程所拥有的资源。在引入线程的OS中，通常都是把进程作为**分配资源的基本单位**，而把线程作为**独立运行和独立调度的基本单位**。

由于线程比进程更小，基本上不拥有系统资源，故对它的调度所付出的**开销**就会**小**得多，能更高效地**提高**系统内多个程序间**并发执行的程度**。

近年来推出的通用操作系统都引入了线程，以便进一步提高系统的并发性，并把它视作**现代操作系统的一个重要标志**。



1.3.2 共享(Sharing)

一般情况下的共享与操作系统环境下的共享其含义并不完全相同。

1. 互斥共享方式

系统中的某些资源，如打印机、磁带机等，虽然可以提供给多个进程(线程)使用，但应规定在一段时间内，只允许一个进程访问该资源。为此，在系统中应建立一种机制，以保证多个进程对这类资源的互斥访问。



2. 同时访问方式

系统中还有另一类资源，允许在一段时间内由多个进程“同时”对它们进行访问。

这里所谓的“同时”，在单处理机环境下是宏观意义上的，而在微观上，这些进程对该资源的访问是交替进行的。典型的可供多个进程“同时”访问的资源是磁盘设备。

一些用重入码编写的文件也可以被“同时”共享，即允许若干个用户同时访问该文件。



1.3.3 虚拟(Virtual)

在OS中，把通过某种技术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

利用设备为某一用户服务的空闲时间，转去为其他用户服务。

(1) 虚拟处理机技术。

(2) 虚拟设备技术。





2. 空分复用技术

利用存储器的空闲空间分区域存放和运行其他的多道程序，以此来提高内存的利用率。

单纯的空分复用存储器只能提高内存的利用率，并不能实现在逻辑上扩大存储器容量的功能，还必须引入**虚拟存储技术**才能达到此目的。

1.3.4 异步(Asynchronism)

在多道程序环境下，系统允许多个进程并发执行。在单处理机环境下，由于系统中只有一台处理机，因而每次只允许一个进程执行，其余进程只能等待。

当正在执行的进程提出某种资源要求时，如打印请求，而此时打印机正在为其它进程打印，由于打印机属于临界资源，因此正在执行的进程必须等待，并释放出处理机，直到打印机空闲，并再次获得处理机时，该进程方能继续执行。

可见，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。



1.4 操作系统的主要功能

引入OS的主要目的是，为多道程序的运行提供良好的运行环境，以保证多道程序能有条不紊地、高效地运行，并能最大程度地提高系统中各种资源的利用率，方便用户的使用。

在传统的OS中应具有处理机管理、存储器管理、设备管理和文件管理等基本功能。此外，为了方便用户使用OS，还需向用户提供方便的用户接口。



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。

(2) 进程调度。



1. 进程控制□

在**传统**的多道程序环境下，进程控制的主要功能：

- (1) 为作业**创建进程**，为进程**分配资源**。
- (2) 控制**进程**在运行过程中的**状态转换**。
- (3) **撤消**已结束的**进程**，**回收资源**。

在**现代OS**中，进程控制还应具有：

- (1) 为一个进程**创建**若干个**线程**的功能
- (2) **撤消** (终止) 已完成任务的**线程**的功能。

2. 进程同步

进程同步的主要任务是为多个进程(含线程)的运行进行协调，使多个进程能有条不紊地运行。两种协调方式：

- ① **进程互斥方式**。各进程(线程)在对临界资源进行访问时，应采用互斥方式；
- ② **进程同步方式**。指在相互合作去完成共同任务的各进程(线程)间，由同步机构对它们的执行次序加以协调。

为了实现进程同步，系统中必须设置进程同步机制。实现进程同步最常用的机制是**信号量机制**。

3. 进程通信□

进程通信是指在多道程序环境下，并发的进程(线程)如果相互合作去完成一个共同的任务，那么相互之间又往往需要交换信息。

例如，有三个相互合作的进程，它们是输入进程、计算进程和打印进程。输入进程负责将所输入的数据传送给计算进程；计算进程利用输入数据进行计算，并把计算结果传送给打印进程；最后，由打印进程把计算结果打印出来。

4. 调度

在**传统的**操作系统中，包括作业调度和进程调度两步。

(1) **作业调度**的基本任务，是：

多道批处理系统中有作业调度，而分时系统中没有作业调度

①从**后备队列**（在**外存**）中按照一定的算法，选择出若干个**作业**。

②为它们**分配**其**必需的资源**（首先是**分配内存**）。

③在将它们调入内存后，便分别为它们**建立进程**，使它们都成为可能获得处理机的**就绪进程**。

④然后按照一定的算法将它们**插入就绪队列**。

多道批处理系统和分时系统中都有进程调度。

(2) **进程调度**的任务，是：

- ①从进程的**就绪队列**（在**内存**）中选出一新**进程**。
- ②把**处理机**分配给它，并为其**设置运行现场**（**处理剂的各寄存器的值**），使进程**投入执行**。

在**多线程OS(现代OS)**中，通常是把线程作为独立运行和分配处理机的基本单位，为此，须把就绪线程排成一个队列，每次调度时，是从**就绪线程队列**中选出一个**线程**，把**处理机**分配给它。□

1.4.2 存储器管理功能

存储器管理的主要任务：

- (1) 为多道程序的运行提供良好的环境，方便用户使用存储器
- (2) 提高存储器的利用率
- (3) 能从逻辑上扩充内存

存储器管理的主要功能：

- 1. 内存分配
- 2. 内存保护
- 3. 地址映射
- 4. 内存扩充

1. 内存分配

OS在实现内存分配时，可采取静态和动态两种方式。

(1) 静态分配方式

每个作业的基本内存空间是在作业装入时确定的；在作业装入后的整个运行期间，①不允许该作业再申请新的内存空间，②也不允许作业在内存中“移动”；

(2) 动态分配方式

每个作业所要求的基本内存空间，也是在作业装入时确定的，但①允许作业在运行过程中，继续申请新的附加内存空间，以适应程序和数据的动态增长，②也允许作业在内存中“移动”。



为了实现内存分配，在内存分配的机制中应具有这样的结构和功能：

① **内存分配数据结构**：用于记录内存空间的使用情况，作为内存分配的依据；

② **内存分配功能** **内存分配算法**

③ **内存回收功能** **内存回收算法**





2. 内存保护□

内存保护的主要任务，是确保每道用户程序都只在**自己的内存空间**内运行，彼此互不干扰。

为了确保每道程序都只在自己的内存区中运行，必须设置**内存保护机制**。



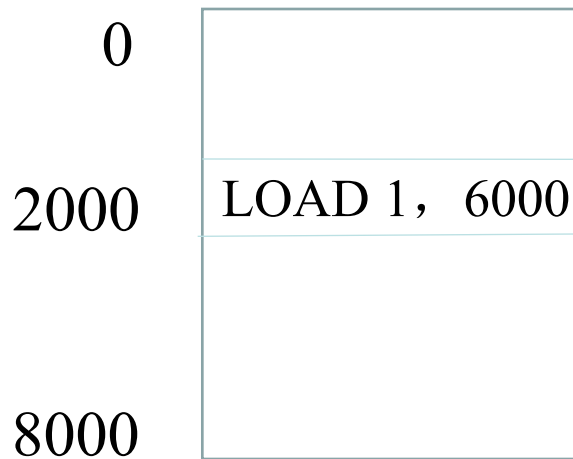
3. 地址映射□

一个应用程序(源程序)经编译后，通常会形成若干个目标程序；这些目标程序再经过链接便形成了可装入程序。

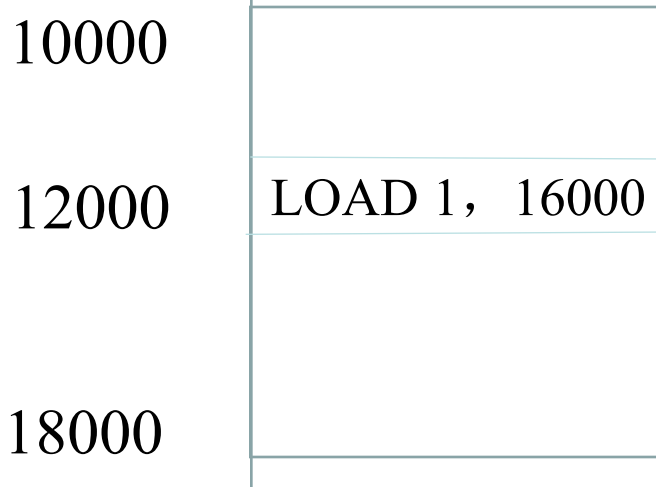
■ 地址空间与内存空间、逻辑地址与物理地址

这些程序的地址都是从“0”开始的，程序中的其它地址都是相对于起始地址计算的；由这些地址所形成的地址范围称为“地址空间”，其中的地址称为“逻辑地址”或“相对地址”。

由内存中的一系列单元所限定的地址范围称为“内存空间”，其中的地址称为“物理地址”。

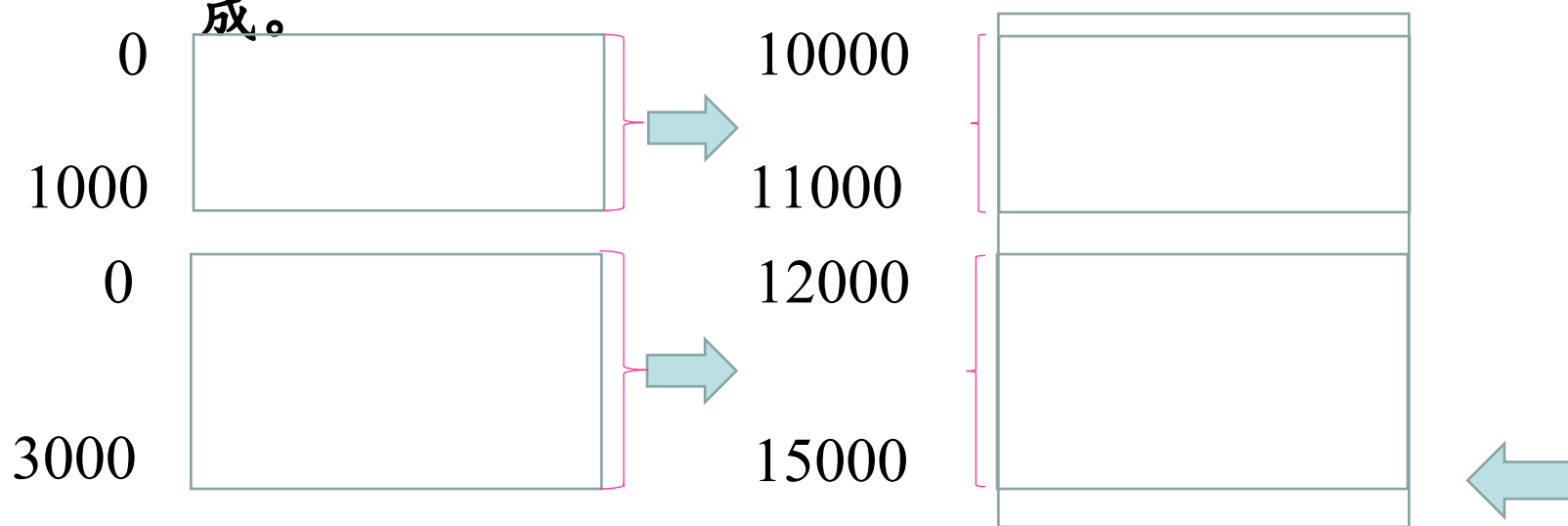


这个程序的**地址空间**
是0——8000，其中0、
2000、8000等都是**逻辑
地址**



内存空间的一部分：
10000——18000，其中
10000、12000、18000等都
是**物理地址**

在多道程序环境下，每道程序不可能都从“0”地址开始装入(内存)，这就致使地址空间内的逻辑地址和内存空间中的物理地址不相一致。使程序能正确运行，存储器管理必须提供地址映射功能，以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。该功能应在硬件的支持下完成。



4. 内存扩充□

存储器管理中的内存扩充任务，并非是去扩大物理内存的容量，而是借助于虚拟存储技术，从逻辑上去扩充内存容量。

为了能在逻辑上扩充内存，系统必须具有内存扩充机制，用于实现下述各功能：

- (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，设备管理应具有缓冲管理、设备分配和设备处理以及虚拟设备等功能。

1. 缓冲管理
2. 设备分配
3. 设备处理

1. 缓冲管理□

缓冲管理是为了解决CPU运行的高速性和I/O低速性间的矛盾。 □

最常见的缓冲区机制有：

- (1) 单缓冲机制
- (2) 双缓冲机制
- (3) 公用缓冲池机制

2. 设备分配

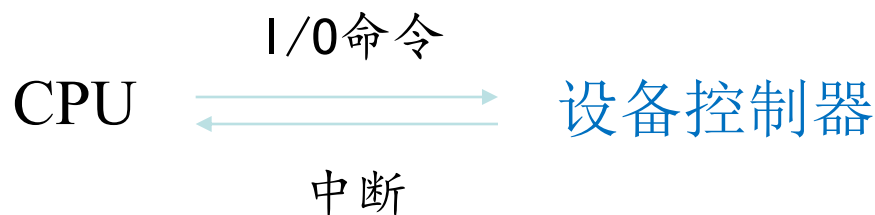
设备分配的基本任务，是根据用户进程的I/O请求、系统的现有资源情况以及按照某种设备分配策略，为之分配其所需的设备。如果在I/O设备和CPU之间，还存在着设备控制器和I/O通道时，还须为分配出去的设备分配相应的控制器和通道。

为了实现设备分配：

- (1)系统中应设置相应的数据结构。
- (2)针对不同的设备类型而采用不同的设备分配方式。
- (3)对于独占设备(临界资源)应考虑到该设备被分配出去后，系统是否安全。
- (4)设备的回收。

3. 设备处理□

设备处理程序又称为设备驱动程序。其基本任务是用于**实现CPU和设备控制器之间的通信**。





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) 文件的读/写管理。
 - (2) 文件保护。

1. 文件存储空间的管理□

由文件系统对诸多文件及文件的存储空间，实施统一的管理。其主要任务是：

- (1) 为每个文件分配必要的外存空间。
- (2) 提高外存的利用率。
- (3) 能有助于提高文件系统的运行速度。□

为此：

- (1) 系统应设置相应的数据结构
- (2) 对存储空间(外存)进行分配
- (3) 对存储空间(外存)进行回收

2. 目录管理

由系统为每个文件建立一个目录项。目录项包括文件名、文件属性、文件在磁盘上的物理位置等。

由若干个目录项又可构成一个目录文件。

目录管理的主要任务：

- (1) 实现方便的按名存取。
- (2) 实现文件共享。
- (3) 提供快速的目录查询。





3. 文件的读/写管理和保护□

(1) 文件的读/写管理

文件读写的过程：

- ① 根据用户给出的文件名去检索文件目录，从而获取文件在外存中的位置。
- ② 利用文件读写指针进行读写，读写完成后修改文件的读写指针。

(2) 文件保护



1.4.5 操作系统与用户之间的接口

1. 用户接口

(1) 联机用户接口：由一组键盘操作命令及命令解释程序组成。特点是“用户说一句，系统跟着做一句”。

(2) 脱机用户接口：为批处理的用户提供。特点是“用户做一堆，系统跟着做一堆”

(3) 图形用户接口：采用图形化的界面（GUI），用户通过键盘和鼠标进行操作。



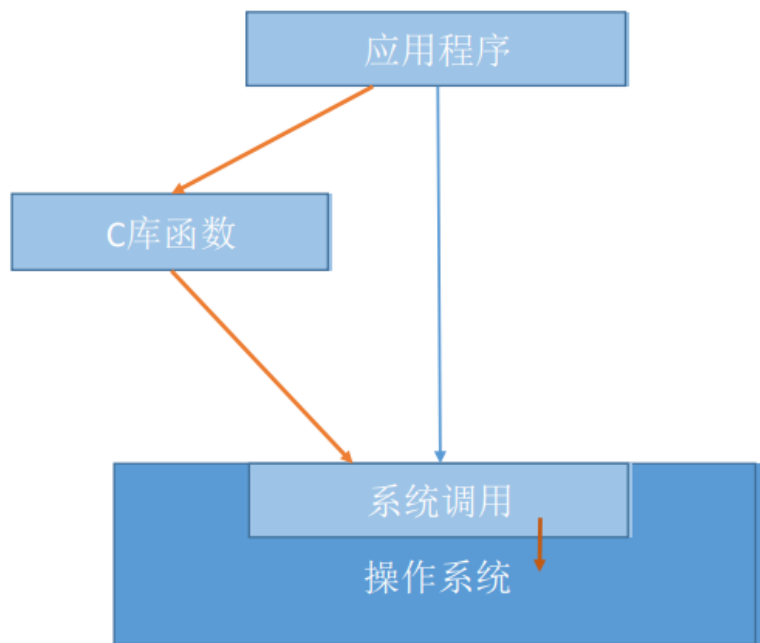
2. 程序接口

程序接口是为用户程序在执行中访问系统资源而设置的，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。每当应用程序要求OS提供某种服务(功能)时，便调用具有相应功能的系统调用(子程序)。

但在高级语言以及C语言中，往往提供了与各系统调用一一对应的库函数，这样，应用程序便可通过调用对应的库函数来使用系统调用。

系统调用与库函数的区别



普通应用程序	可直接进行系统调用，也可使用库函数。有的库函数涉及系统调用，有的不涉及
编程语言	向上提供库函数。有时会将系统调用封装成库函数，以隐藏系统调用的一些细节，使程序员编程更加方便。
操作系统	向上提供系统调用，使得上层程序能请求内核的服务
裸机	

不涉及系统调用的库函数：如的“取绝对值”的函数
涉及系统调用的库函数：如“创建一个新文件”的函数




1.4.6 现代操作系统的新功能

现代操作系统是在传统操作系统基础上发展起来的，它除了具有传统操作系统的功能外，还增加了面向安全、面向网络和面向多媒体等功能。

1. 系统安全

- (1) 认证技术。
- (2) 密码技术。
- (3) 访问控制技术。
- (4) 反病毒技术。





2. 网络的功能和服务

(1) 网络通信。

(2) 资源管理。

(3) 应用互操作。



3. 支持多媒体

- (1) 接纳控制功能。
- (2) 实时调度。
- (3) 多媒体文件的存储。



1.5 OS结构设计

随着OS规模的愈来愈大，其所具有的代码也愈来愈多，往往需要由数十人或数百人甚至更多的人参与，分工合作，共同来完成操作系统的设计。这意味着，应采用工程化的开发方法对大型软件进行开发。由此产生了“软件工程学”。



1.5.1 传统操作系统结构

1. 无结构操作系统

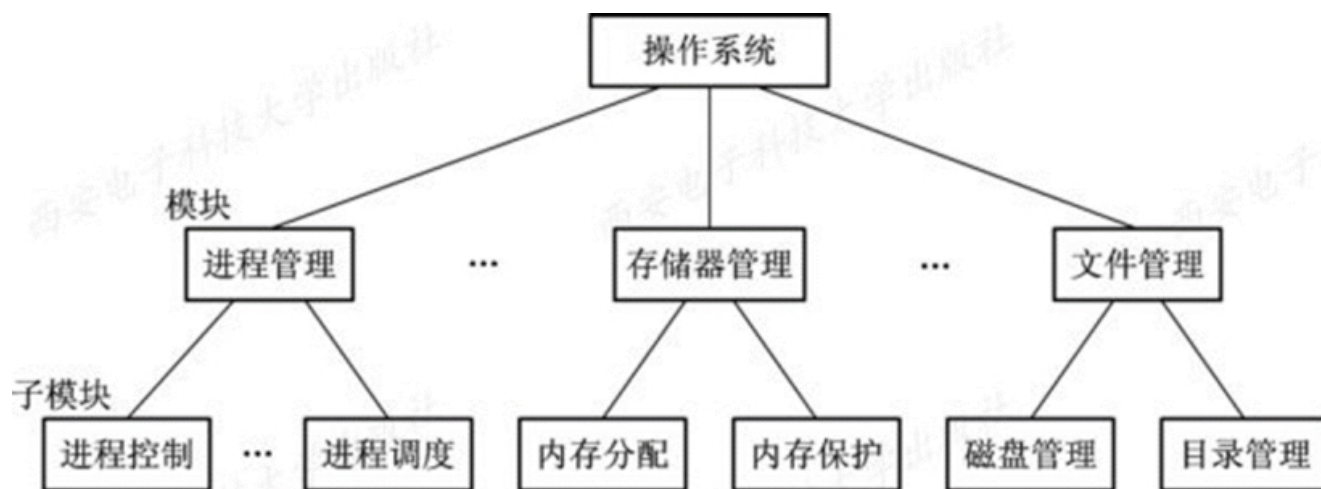
在早期开发操作系统时，设计者只是把他的注意力放在功能的实现和获得高的效率上，缺乏首尾一致的设计思想。此时的OS是为数众多的一组过程的集合，每个过程可以任意地相互调用其它过程，致使操作系统内部既复杂又混乱，因此，这种OS是无结构的，也有人把它称为**整体系统结构**。



2. 模块化结构OS

1) 模块化程序设计技术的基本概念

该技术基于“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

在模块-接口法中，**关键问题是模块的划分和规定好模块之间的接口。**

衡量模块的独立性有以下两个标准：



- a. 内聚性：指模块内部各部分间的紧密程度。内聚性越高，模块独立性越强。
- b. 耦合度：指模块间相互联系和相互影响的程度。耦合度越低，模块独立性越好。



3) 模块接口法的优缺点

利用模块-接口法开发的OS，较之无结构OS具有以下明显的优点：

- (1) 提高OS设计的正确性、可理解性和可维护性。
- (2) 增强OS的可适应性。
- (3) 加速OS的开发过程。



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) 在OS设计阶段，设计者必须做出一系列的决策，每一个决定必须建立在上一个决定的基础上，但模块化结构设计中，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”，这将使程序人员很难做到“设计中的每一步决定”都是建立在可靠的基础上，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

为了将模块-接口法中“决定顺序”的无序性变为有序性，引入了有序分层法，分层法的设计任务是，在目标系统 A_n 和裸机系统(又称宿主系统) 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。在操作系统中，常采用**自底向上**法来铺设这些中间层。



2) 分层结构的优缺点

分层结构的主要优点有：

- (1) 易保证系统的正确性。
- (2) 易扩充和易维护性。

分层结构的主要缺点是**系统效率降低**。由于层次结构是分层单向依赖的，必须在每层之间都建立层次间的通信机制，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/服务器模式的由来、组成和类型

客户/服务器系统主要由三部分组成。

(1) 客户机：通常在一个LAN网络上连接有多台网络工作站（简称客户机），每台客户机都是一个自主计算机，具有一定的处理能力，客户进程在其上运行，平时它处理一些本地业务，也可以发送一个消息给服务器，以请求某项服务。

(2) 服务器：通常是一台规模较大的机器，其上驻留有网络文件系统或数据库系统等，它能为网上的所有用户提供一种或多种服务。

(3) 网络系统：用于连接所有客户机和服务器，实现它们之间通信和网络资源共享的系统。



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 (Object-Oriented Programming) 技术简介

1. 面向对象技术的基本概念

面向对象技术是20世纪80年代初提出并很快流行起来的。





1) 对象

在面向对象的技术中，是利用被封装的数据结构(变量)和一组对它进行操作的过程(方法)来表示系统中的某个对象的。

对象中的变量(数据)也称为属性，它可以是单个标量或一张表。

面向对象中的方法是用于执行某种功能的过程，它可以改变对象的状态，更新对象中的某些数据值或作用于对象所要访问的外部资源。



数据结构

过程 1

过程 2

过程 3

图1-8 一个对象的示意图



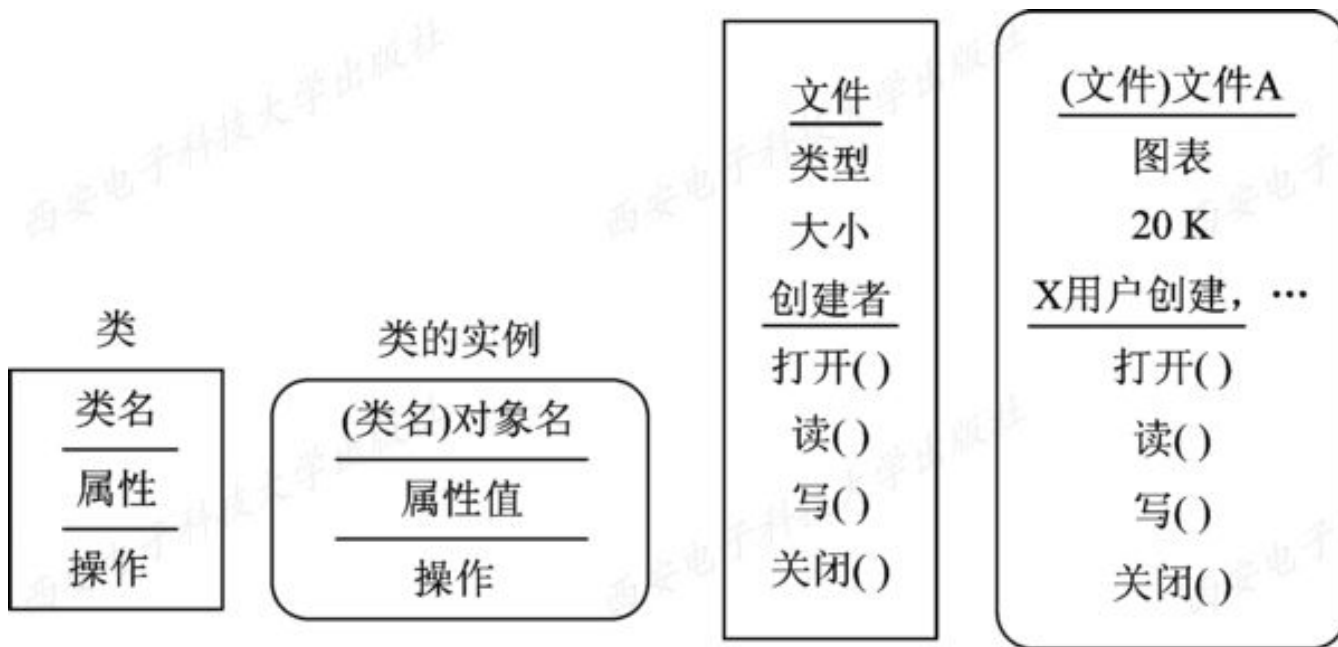



图1-9 类和对象的关系



2) 对象类

利用“对象类”来定义一组大体相似的对象。一个类同样定义了一组变量和针对该变量的一组方法，用它们来描述一组对象的共同属性和行为。类是在对象上的抽象，对象则是类的实例。对象类中所定义的变量在实例中均有具体的值。



3) 继承

在面向对象的技术中，可以根据已有类来定义一个新的类，新类被称为子类(B)，原来的类被称为父类(A)，见图1-10所示。

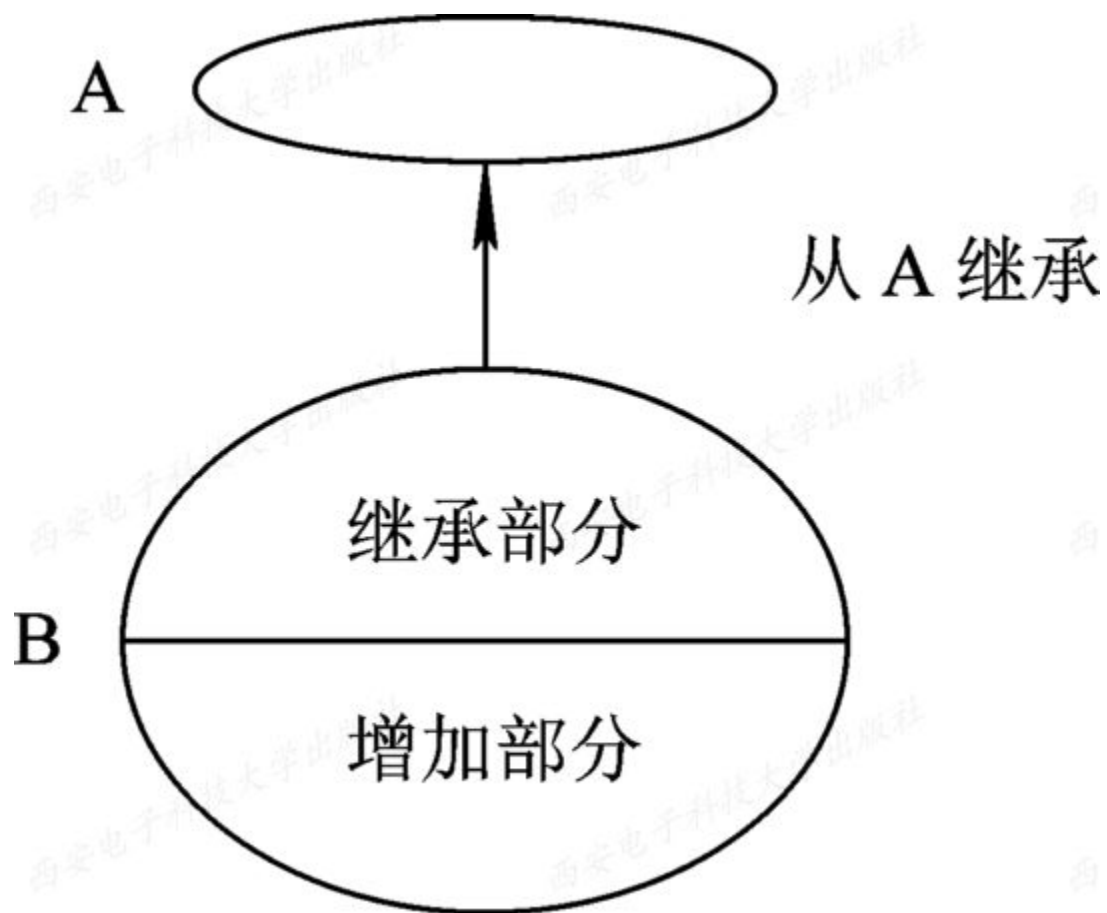


图1-10 类的继承关系



2. 面向对象技术的优点

在操作系统设计时，将计算机中的实体作为对象来处理，可带来如下好处：

- (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”。

1.5.4 微内核OS结构

现代OS大多采用基于客户/服务器模式的微内核结构，将操作系统划分为两部分：**微内核**和**多个服务器**。

1. 微内核操作系统的基本概念

1) 足够小的内核

在微内核操作系统中，内核是指精心设计的、能实现现代OS最基本核心功能的小型内核，微内核并非是一个完整的OS，而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 **客户和服务服务器之间的通信。**

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“机制与策略分离”原理

在现在操作系统的结构设计中，经常利用“机制与策略分离”的原理来构造OS结构。

机制：指实现某一功能的具体执行机构。

策略：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。

在传统的OS中，将机制放在OS的内核的较低层。在微内核操作系统中，将机制放在OS的微内核中。

4) 采用面向对象技术

操作系统是一个极其复杂的大型软件系统，我们不仅可以
通过结构设计来分解操作系统的复杂度，还可以基于面向
对象技术中的“抽象”和“隐蔽”原则控制系统的复杂性，再进
一步利用“对象”、“封装”和“继承”等概念来确保操作系统的“
正确性”、“可靠性”、“易修改性”、“易扩展性”等，并提高操
作系统的设计速度。正因为面向对象技术能带来如此多的好
处，故面向对象技术被广泛应用于现代操作系统的设计中。



2. 微内核的基本功能

现在一般都采用“机制与策略分离”的原理，将机制部分以及与硬件紧密相关的部分放入微内核中。微内核通常具有如下几方面的功能：

- 1) 进程(线程)管理
- 2) 低级存储器管理
- 3) 中断和陷入处理

(1) 进程（线程）管理

例

进程管理服务器

如何确定优先级

如何修改优先级

策略

微内核

进程调度：

设置优先级队列

取出一个进程投入运行

进程通信

进程切换

线程调度

多处理机之间的同步

机制

(2) 低级存储器管理

例

存储器管理服务器

采用何种页面置换算法

采用何种内存分配与回收策略

微内核

页表机制

地址变换机制

策略

机制

(3) 中断和陷入管理

例

服务器

调用相应的处理程序进行后期处理

策略

微内核

中断现场保护

识别中断和陷入类型

机制



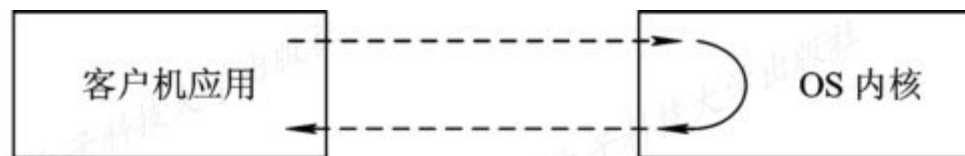
3. 微内核操作系统的优点

- (1) 提高了系统的可扩展性。
- (2) 增强了系统的可靠性。
- (3) 可移植性强。
- (4) 提供了对分布式系统的支持。
- (5) 融入了面向对象技术。

4. 微内核操作系统存在的问题

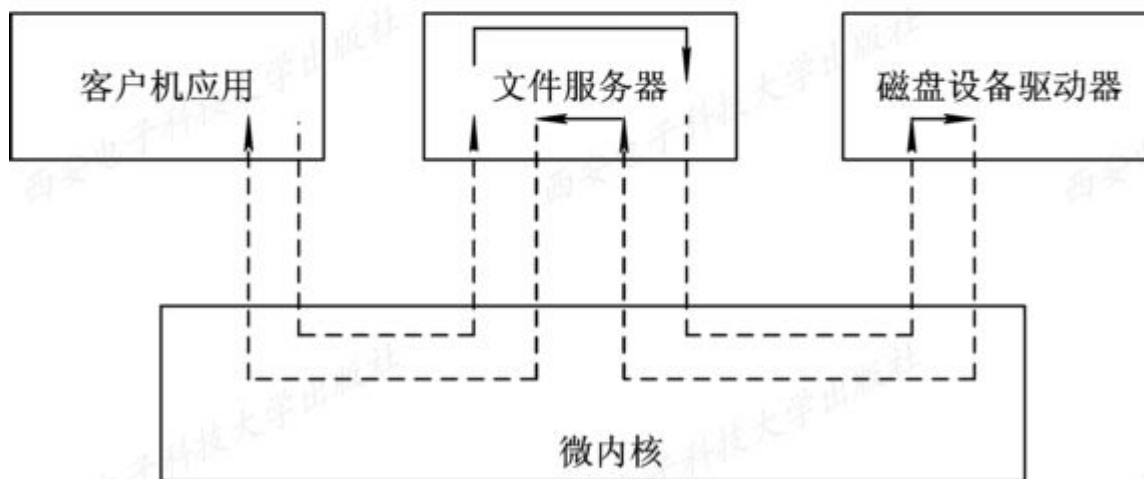
较之早期的操作系统，微内核操作系统的运行效率有所降低。

实际情况是往往还会引起更多的上下文切换。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，如图1-12所示，其中的文件服务器还需要磁盘服务器的帮助，这时就需要进行8次上下文的切换。



返回

(a) 在整体式内核文件操作中的上下文切换



(b) 在微内核中等价操作的上下文切换

图1-12 在传统OS和微内核OS中的上下文切换



计算机系统的层次结构

注意：
操作系统内核需要运行在内核态
操作系统的非内核功能运行在用户态

大内核和微内核

操作系统的体系结构

大内核

将操作系统的主要功能模块都作为系统内核，运行在核心态

优点：高性能

缺点：内核代码庞大，结构混乱，难以维护

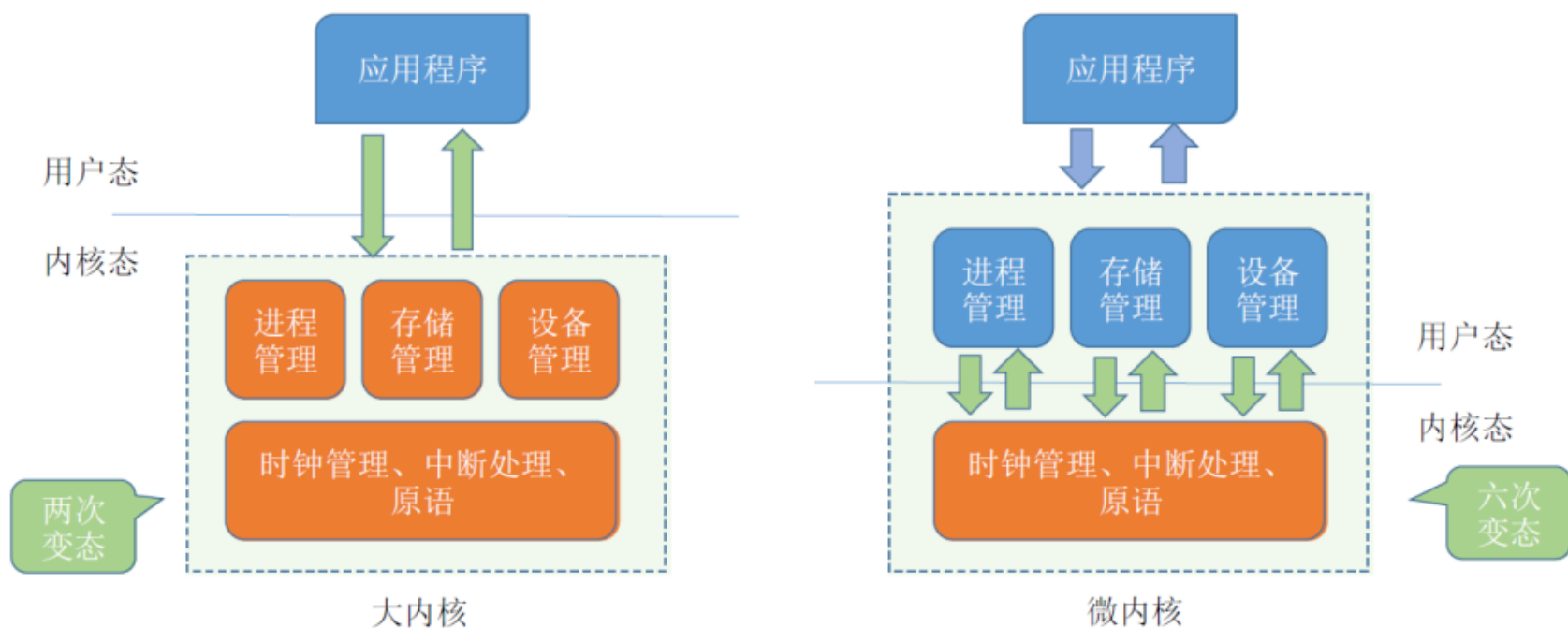
微内核

只把最基本的功能保留在内核

优点：内核功能少，结构清晰，方便维护

缺点：需要频繁地在核心态和用户态之间切换，性能低

操作系统的体系结构



一个故事：现在，应用程序想要请求操作系统的服务，这个服务的处理同时涉及到进程管理、存储管理、设备管理

注意：变态的过程是有成本的，要消耗不少时间，频繁地变态会降低系统性能