综合串讲

Chapter 1-12

章节

- 计算机系统/操作系统概述
- 进程与线程
- 并发
- 内存管理与虚拟内存
- I/O与磁盘
- 文件系统

1.计算机系统/操作系统概述

- 计算机系统组成
  - 处理器
    - 控制计算机的操作，执行数据处理功能。
  - 内存
    - 通常也称为实存储器或主存储器
    - 易失的
  - I/O 模块　　在处理器和外部环境之间移动数据
    - 辅助存储器设备
    - 通信设备
    - 终端
  - 系统总线
    - 为处理器、主存储器和输入/出模块间提供通信的设施

1.计算机系统/操作系统概述

- 计算机部件：顶视图

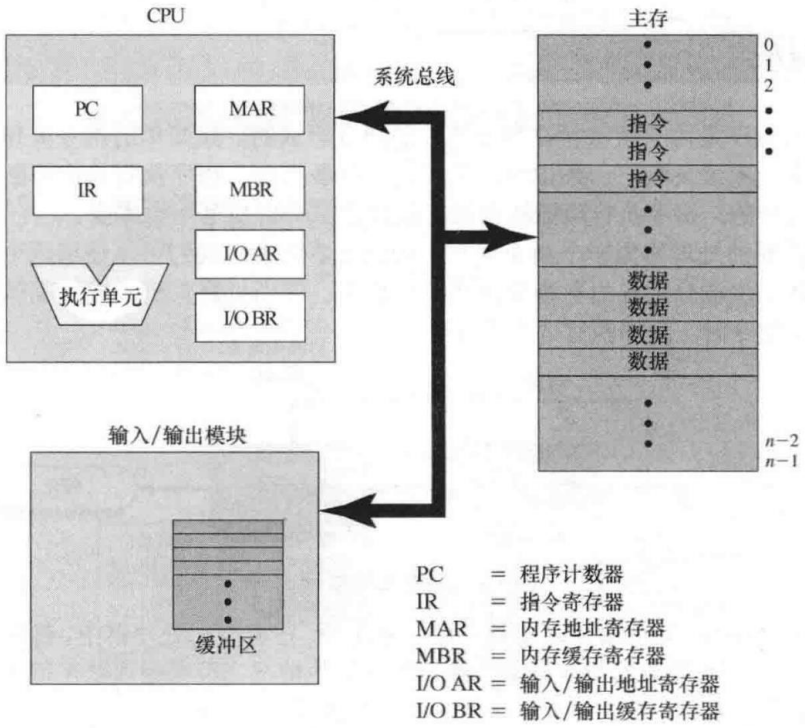


图 1.1　计算机部件：俯视图

1.计算机系统/操作系统概述

- 指令的执行
  - 处理器执行的程序是由一组保存在存储器中的指令组成

两步：

- 处理器从存储器中读取一条指令
- 处理器执行这条指令

1.计算机系统/操作系统概述

- 指令执行
  - 指令的类型
    - 处理器-存储器
      - 在处理器和存储器之间传送数据
    - 处理器-I/O
      - 通过处理器和I/O模块间的数据传送，数据可以输出到外部设备，或者从外部设备输入
    - 数据处理
      - 执行数据相关的算术操作或逻辑操作
    - 控制
      - 改变执行顺序

1.计算机系统/操作系统概述

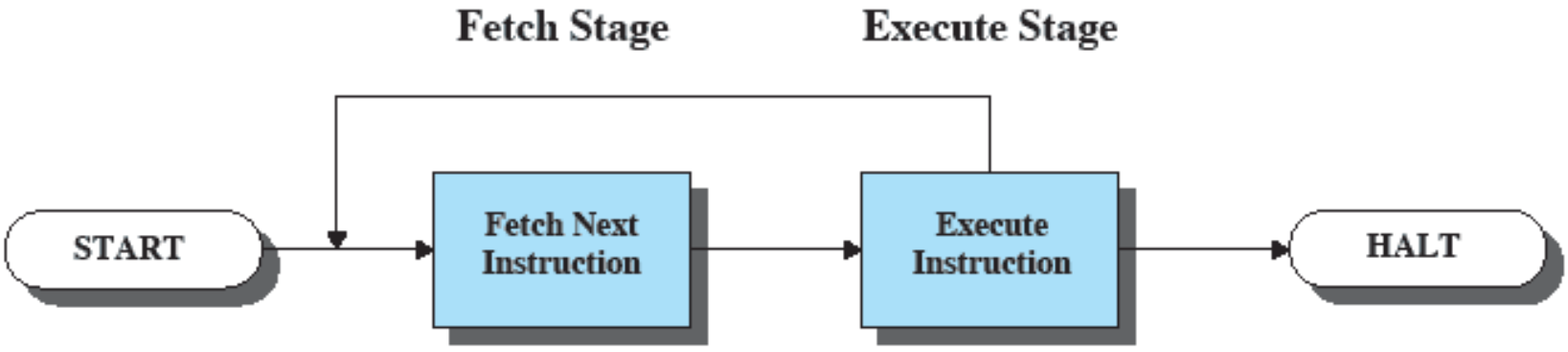


Figure 1.2 Basic Instruction Cycle

1.计算机系统/操作系统概述

- 中断
  - 中断是指CPU对系统发生某事件时作出的这样一种响应：CPU暂停正在执行的程序，在保留现场后自动地转去执行该事件的中断处理程序；执行完后，再返回到原程序的断点处继续执行。

1.计算机系统/操作系统概述

- 中断类型
  - 程序中断
    - 算术溢出
    - 除数为0
    - 执行非法指令
    - 访问到用户不允许的存储器位置
  - 时钟中断
  - I/O中断
  - 硬件失效中断

1.计算机系统/操作系统概述

- 含中断的指令周期

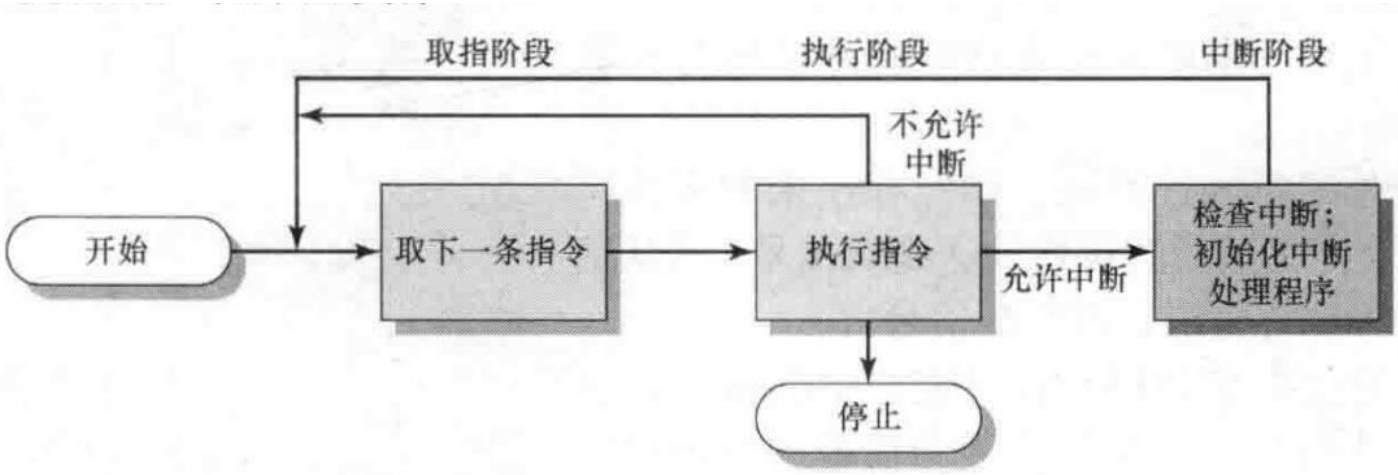


图 1.7　中断和指令周期

1.计算机系统/操作系统概述

- 中断处理流程

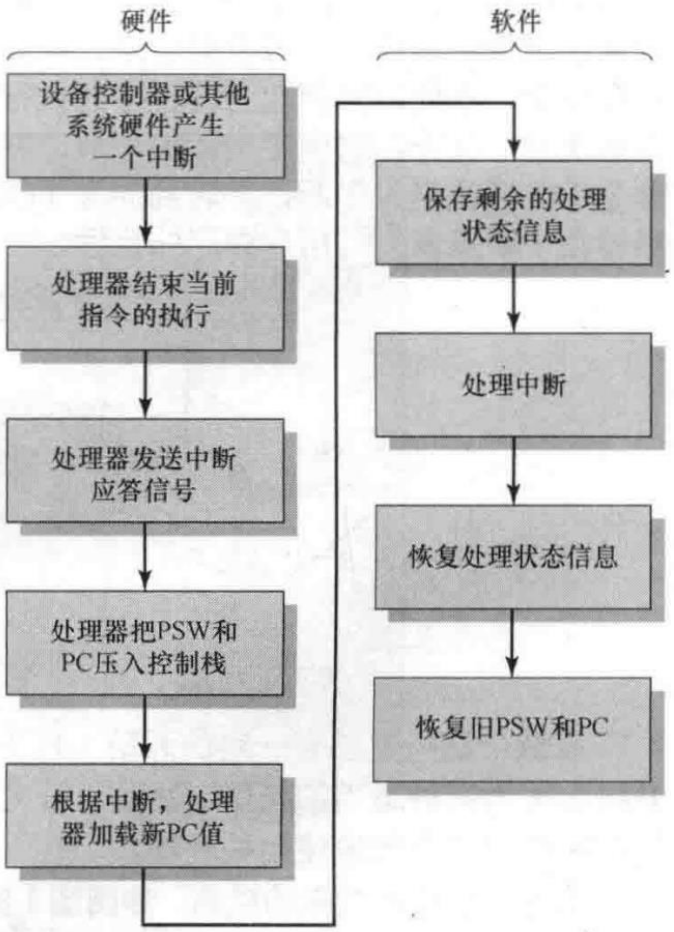


图 1.10　简单中断处理

1.计算机系统/操作系统概述

- 多道程序概念
  - 允许多道用户程序同时处于活动状态
  - 执行顺序取决于它们的相对优先级以及它们是否正在等待 I/O
  - 当一个中断处理完成后,控制可能并不立即返回到这个用户程序，而可能转移到其他待运行的具有更高优先级的程序

1.计算机系统/操作系统概述

- 存储器的层次结构
  - 存储器的三个重要特性
    - 容量
    - 速度
    - 价格
  - 存取时间越快, 每“位”的价格更高
  - 容量越大,每“位”的价格越低
  - 容量越大, 存取速度越慢

1.计算机系统/操作系统概述

- 存储器的层次结构
  - 每“位”的价格递减
  - 容量递增
  - 存取时间递增
  - 处理器访问存储器的频率递减
    - 访问局部性原理
    - 命中率

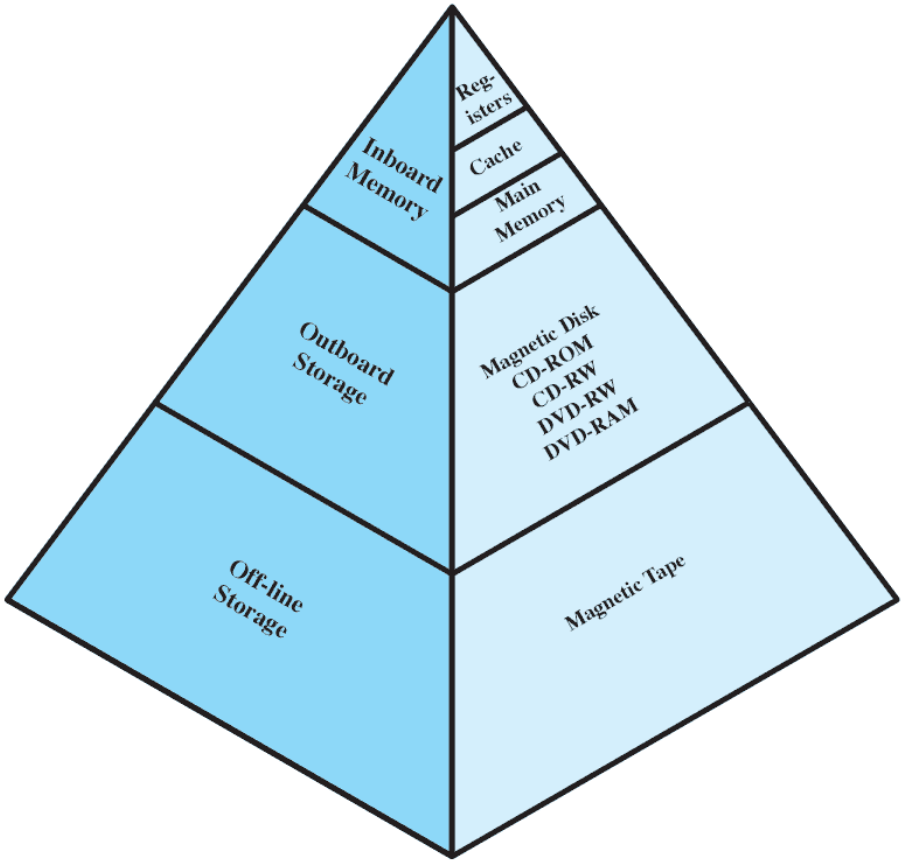


Figure 1.14 The Memory Hierarchy

1.计算机系统/操作系统概述

- 局部性原理
  - 处理器的指令访存和数据访存呈现“簇”状
    - 程序中包含许多循环和子程序
    - 对表和数组的操作涉及存取“一簇”数据
  - 按层次组织数据，使得随着组织层次的递减，对各层次的访问比例也依次递减
  - 可应用于多层存储器组织结构

1.计算机系统/操作系统概述

- 操作系统概念
  - 操作系统是管理计算机系统的软硬件资源、控制程序执行、改善人机界面和为应用软件提供支持的一种系统软件。

1.计算机系统/操作系统概述

- 操作系统目标
  - 方便
    - 使计算机更易于使用

      （这是OS产生的根本原因）
  - 有效
    - 允许以更有效的方式使用计算机系统资源

      （以优化算法合理分配、调度资源和提高效率）
  - 扩展的能力
    - 允许在不妨碍服务的前提下有效的开发、测试和引进新的系统功能
    - 原因：（1）硬件的升级，（2）用户新的服务要求

      （3）OS纠正错误

1.计算机系统/操作系统概述

- OS作为用户/计算机接口
  - 程序开发
    - 编辑器和调试器
  - 程序运行（进程管理）
  - 访问I/O设备（设备管理）
  - 控制访问文件（文件管理）
  - 系统访问
  - 统计
    - 收集各种资源的统计
    - 监控如响应时间之类的性能参数
    - 用于将来对增加功能的需求
    - 对多用户系统，还可用于记账
  - 错误检测和响应
    - 内部和外部硬件错误
      - 存储器错误
      - 设备失败或故障
    - 软件错误
      - 算术溢出
      - 访问被禁止的存储器单元
    - OS无力确认应用程序的请求

1.计算机系统/操作系统概述

- **OS作为资源管理器**
  - 控制机制
    - 操作系统与普通的计算机软件相同
      - 由处理器执行的一段程序或一组程序
    - 操作系统经常会释放控制，而且必须依赖处理器才能恢复控制
  - 管理对象：CPU、存储器、外部设备、信息和软件
  - 管理内容：资源的当前状态（数量和使用情况）、资源的分配、回收和访问操作，相应管理策略（包括用户权限）

1.计算机系统/操作系统概述

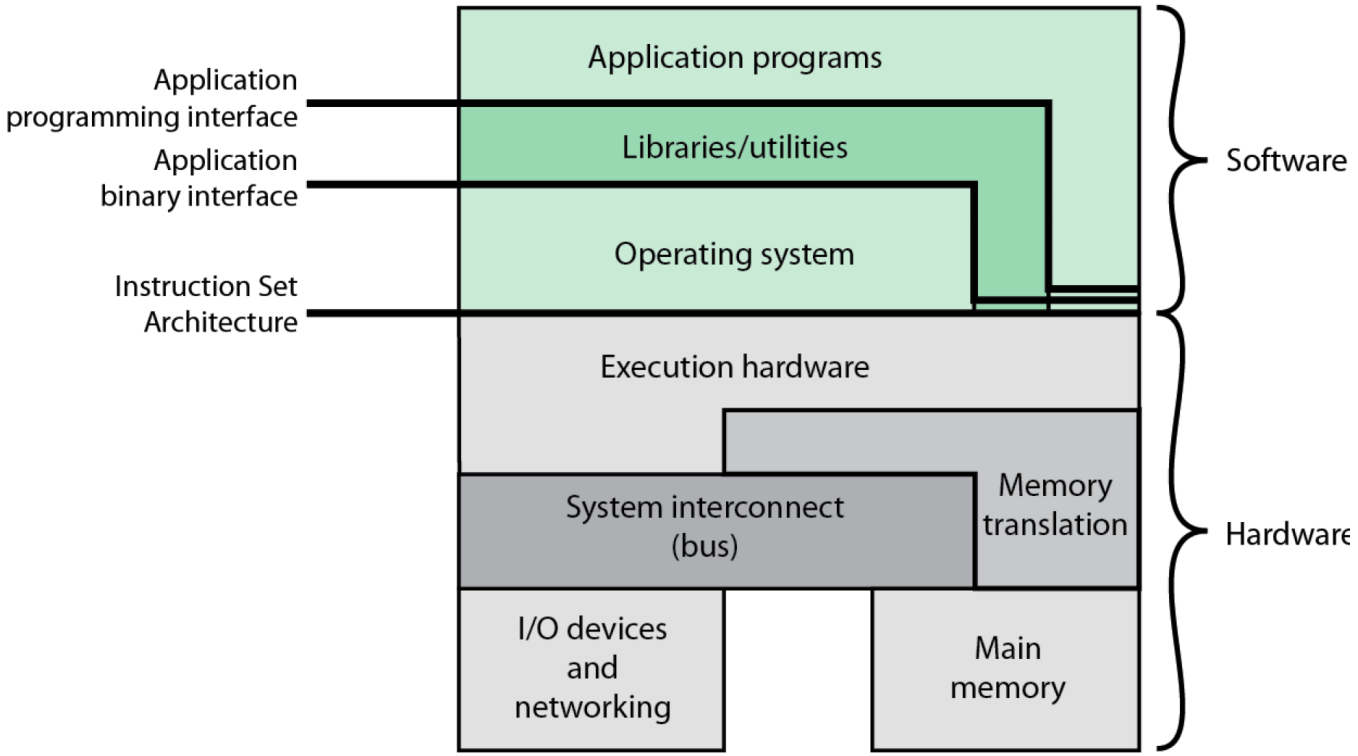


Figure 2.1 Computer Hardware and Software Infrastructure

操作系统地位：紧贴系统硬件之上，所有其它软件之下（是其它软件的共同环境）

1.计算机系统/操作系统概述

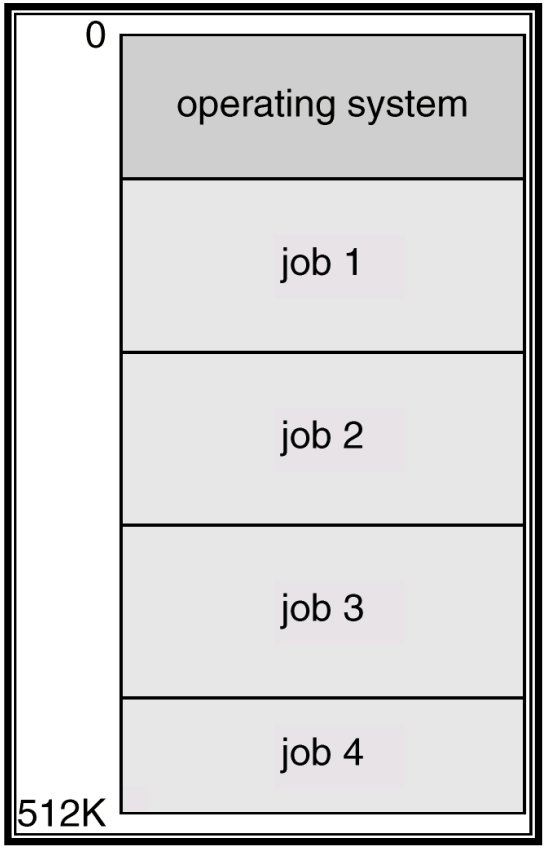
- 简单批处理系统
  - 监控程序(单道批处理，OS雏形)
    - 控制用户程序的运行的软件
    - 控制整批作业
    - 用户程序结束后，程序分支返回监控程序
    - 常驻监控程序总处于主存储器中并且可以执行

1.计算机系统/操作系统概述

- 多道程序批处理系统
  - 标志现代意义上OS的出现
  - 背景：60年代中 ~ 70年代中（集成电路），计算机进入第三代
    - 主存、辅存容量增大，可以同时装入多道程序到主存
    - 出现替代CPU管理者的“DMA通道”，使得I/O操作与CPU并行成为可能

1.计算机系统/操作系统概述

- 利用多道批处理提高资源的利用率。



Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.

1.计算机系统/操作系统概述

- 并发(concurrency)
- 共享(sharing)
- 虚拟(virtual)
- 异步性(asynchronism)

1.计算机系统/操作系统概述

并行性(parallel)是指两个或多个事件在同一时刻发生。
并发性（concurrence）是指两个或多个事件在同一时间间隔内发生。

操作系统是一个并发系统，各进程间的并发，系统与应用间的并发。操作系统要完成这些并发过程的管理。

- 在多道程序处理时，宏观上并发，微观上交替执行（在单处理器情况下）。
- 程序的静态实体是可执行文件，而动态实体是进程（或称作任务），并发指的是进程。
- 引入线程后，独立运行的单位变为线程。

1.计算机系统/操作系统概述

共享(sharing)

多个进程共享有限的计算机系统资源。操作系统要对系统资源进行合理分配和使用。资源在一个时间段内交替被多个进程所用。

- 互斥共享（如音频设备）：资源分配后到释放前，不能被其他进程所用。
- 同时访问（如可重入代码，磁盘文件）

1.计算机系统/操作系统概述

虚拟是指通过某种技术（分时或分空间）把一个物理实体映射为若干个对应的逻辑实体。虚拟是操作系统管理系统资源的重要手段，可提高资源利用率。

- CPU——每个用户（进程）的"虚处理机"
- 虚拟存储器——每个进程都占有的地址空间（指令＋数据＋堆栈）
- 虚拟设备技术——多窗口或虚拟终端(virtual terminal)，SPOOLINg技术，虚拟信道

1.计算机系统/操作系统概述

也称不确定性，指进程的执行顺序和执行时间的不确定性；

- 进程的运行速度不可预知：分时系统中，多个进程并发执行，"时走时停"，不可预知每个进程的运行推进快慢
- 判据：无论快慢，应该结果相同——通过进程互斥和同步手段来保证
- 难以重现系统在某个时刻的状态（包括重现运行中的错误）

2.进程描述和控制

进程的定义有很多种：

- 一个正在执行的程序
- 计算机中正在运行的程序的一个实例
- 可以分配给处理器并有处理执行的一个实体
- 由一个顺序的执行线程、一个当前状态和一组相关的系统资源所描述的活动单元

2.进程描述和控制

进程同程序的比较

- 程序是指令的有序集合，其本身没有任何运行的含义，是一个静态的概念。而进程是程序在处理机上的一次执行过程，它是一个动态的概念。
- 程序可以作为一种软件资料长期存在，而进程是有一定生命期的。程序是永久的，进程是暂时的。
- 进程更能真实地描述并发，而程序不能
- 进程是由程序和数据、进程控制块等组成的
- 进程具有创建其他进程的功能，而程序没有
- 同一程序同时运行于若干个数据集合上，它将属于若干个不同的进程。也就是说同一程序可以对应多个进程

2.进程描述和控制

进程控制块（PCB）

- OS维持的用于管理进程的数据结构
- 用于中断与恢复
- 由OS创建与管理
- 支持多道程序设计、多处理的关键工具

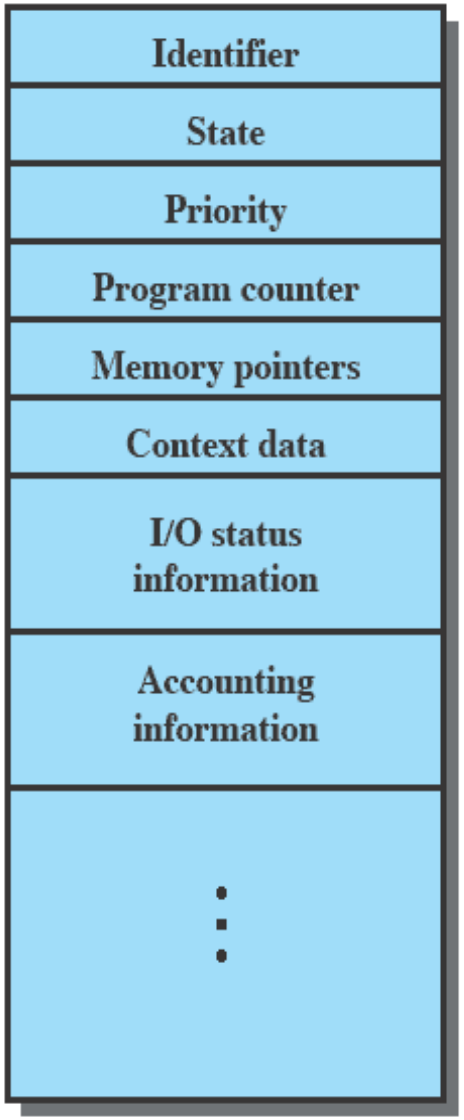


Figure 3.1 Simplified Process Control Block

2.进程描述和控制

进程的特征

- 动态性：有一定的生命期
- 并发性：多个进程实体，同存于内存中，能在一段时间内同时运行
- 独立性：进程实体是一个能独立运行的基本单位，同时也是系统中独立获得资源和独立调度的基本单位
- 异步性：进程按各自独立的、不可预知的速度向前推进，i.e. 进程按异步方式运行
- 结构特征：进程实体是由程序段、数据段及进程控制块等部分组成--进程映像

2.进程描述和控制

两状态进程模型

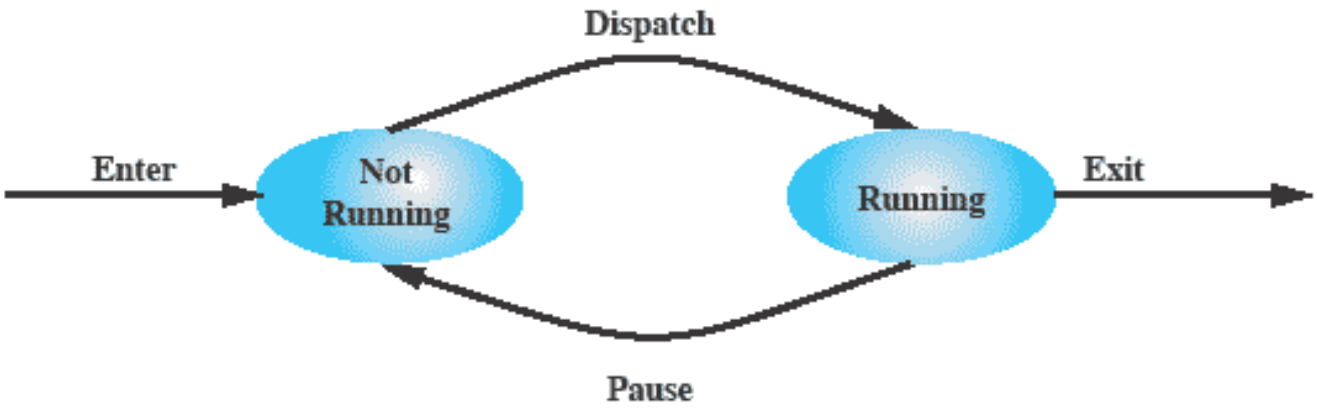
OS为了描述进程的各个阶段的特征，控制和管理进程，定义了各种进程状态，理论上可分为二状态、三状态、五状态等

- 一个进程可以处于两种状态之一：
  - 运行
  - 未运行

2.进程描述和控制

两状态进程模型

- 为了控制进程，需要描述它们的行为
- 进程可以处于以下两种状态之一
  - 运行态
  - 未运行态



(a) State transition diagram

2.进程描述和控制

三状态进程模型

- 在两状态进程模型实现中，非运行状态的进程存在着：
  - 已经就绪等待执行的进程
  - 处于阻塞状态等待I/O操作结束的进程
- 分派程序不能只考虑队列中最老的进程，而应该查找那些未被阻塞且在队列中时间最长的进程
- 将非运行状态分成两个状态：
  - 就绪（ready）
  - 阻塞（blocked）

2.进程描述和控制

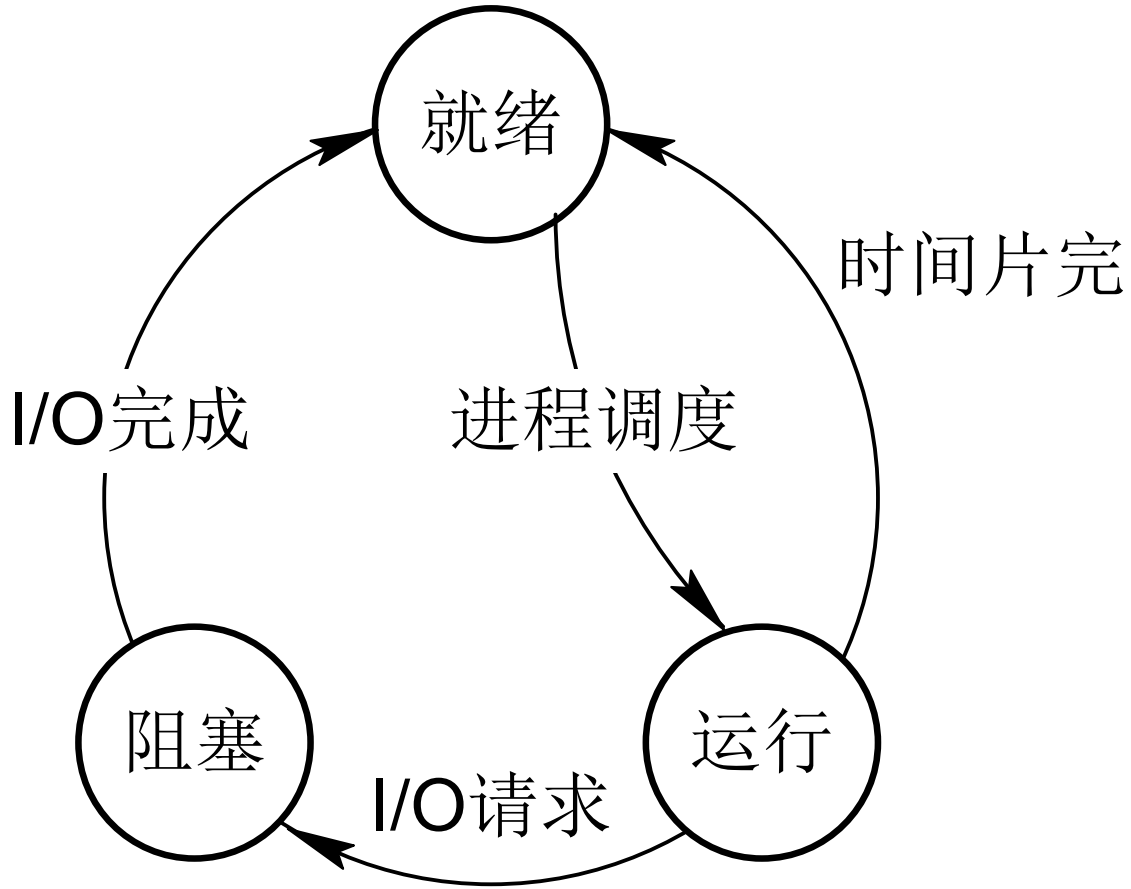
进程的三种基本状态

1) 就绪(Ready)状态

2) 运行状态

3) 阻塞状态

2.进程描述和控制



进程的三种基本状态及其转换

2.进程描述和控制

进程何时创建

- 创建的原因
  - 新的批作业
  - 交互登录
  - 提供一项服务而创建，如打印
  - 由现有的进程生成
    - 进程派生：OS为另一个进程的显示请求创建一个进程
- 动作
  - 建立管理该进程的数据结构
  - 在主存中给它分配地址空间

2.进程描述和控制

进程何时终止

- 批处理作业中发出Halt指令
- 用户退出系统
- 结束一个应用程序
- 错误和故障条件

2.进程描述和控制

进程终止的原因

- 正常完成
- 超过时限
- 无可用存储器
- 越界
- 保护错误
  - 例如写入一个只读文件
- 算术错误
- 时间超出
  - 进程等待某一事件发生的时间超过了规定的最大值

2.进程描述和控制

- I/O 失败
- 无效指令
  - 试图执行一个不存在的指令
- 特权指令
- 数据误用
- 操作员或操作系统干涉
  - 例如，如果存在死锁
- 父进程终止
- 父进程请求

2.进程描述和控制

五状态模型

- 运行态
  - 占用CPU运行，运行态进程数目不大于CPU数目
- 就绪态
  - 已占有除CPU之外的所有请求资源，只等待分配CPU即可运行
- 阻塞态
  - 进程需要等待I/O操作或其它相关事件无法继续运行
- 新建态
  - PCB已经创建，但还没有加载到主存中的新进程，
- 退出态
  - 进程完成或致命错误而结束

2.进程描述和控制

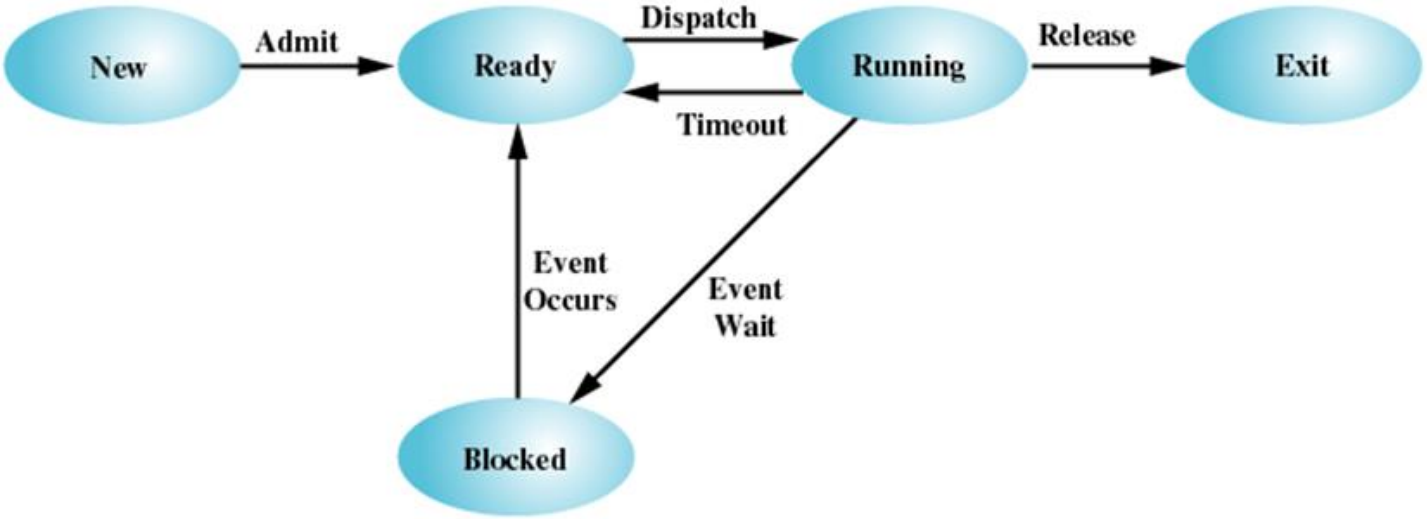


Figure 3.6 Five-State Process Model

进程状态转换的事件类型

1、空→新建：创建执行一个程序的新进程，原因见表3.1

2、新建→就绪：OS准备好再接纳一个进程时，为其分配资源

3、就绪→运行：从就绪进程表中选择一个进程进入运行态

4、运行→退出：进程完成或取消

2.进程描述和控制

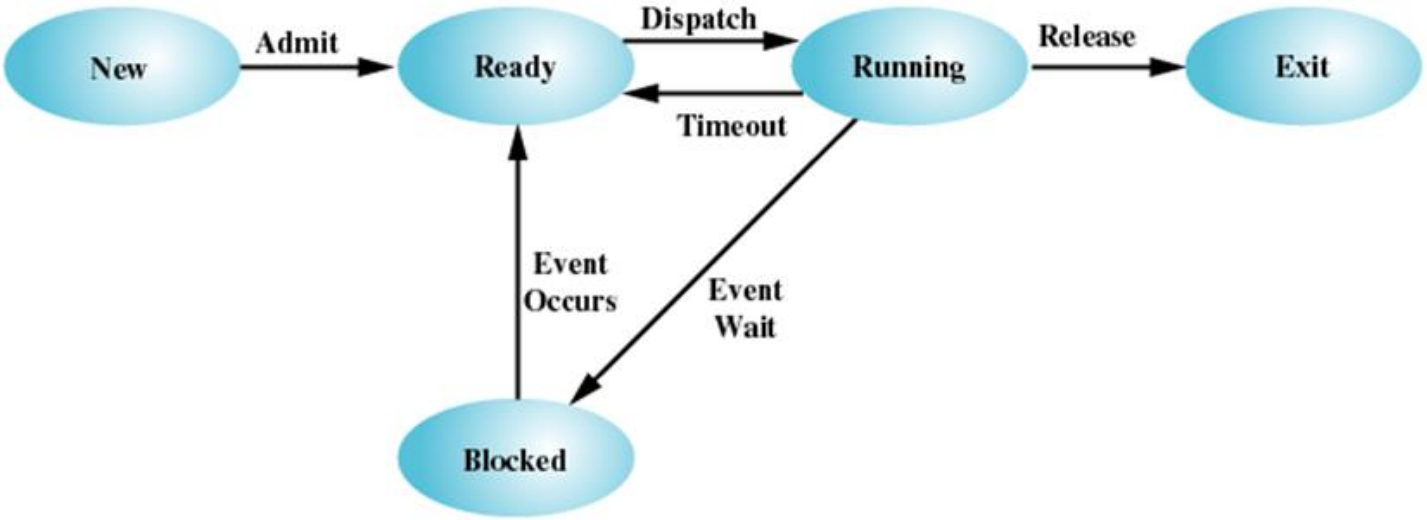


Figure 3.6 Five-State Process Model

5、运行→就绪：运行的进程到达了“允许不中断执行”的最大时间（时间片）或高优先级进程从阻塞状态变为就绪（抢占）

6、运行→阻塞：进程请求的事件未出现，如I/O未完成

7、阻塞→就绪：当所等待的事件发生时，如I/O完成，申请资源成功等

8、就绪/阻塞→退出：父进程终止一个子进程，或者父进程终止导致相关的所有子进程终止

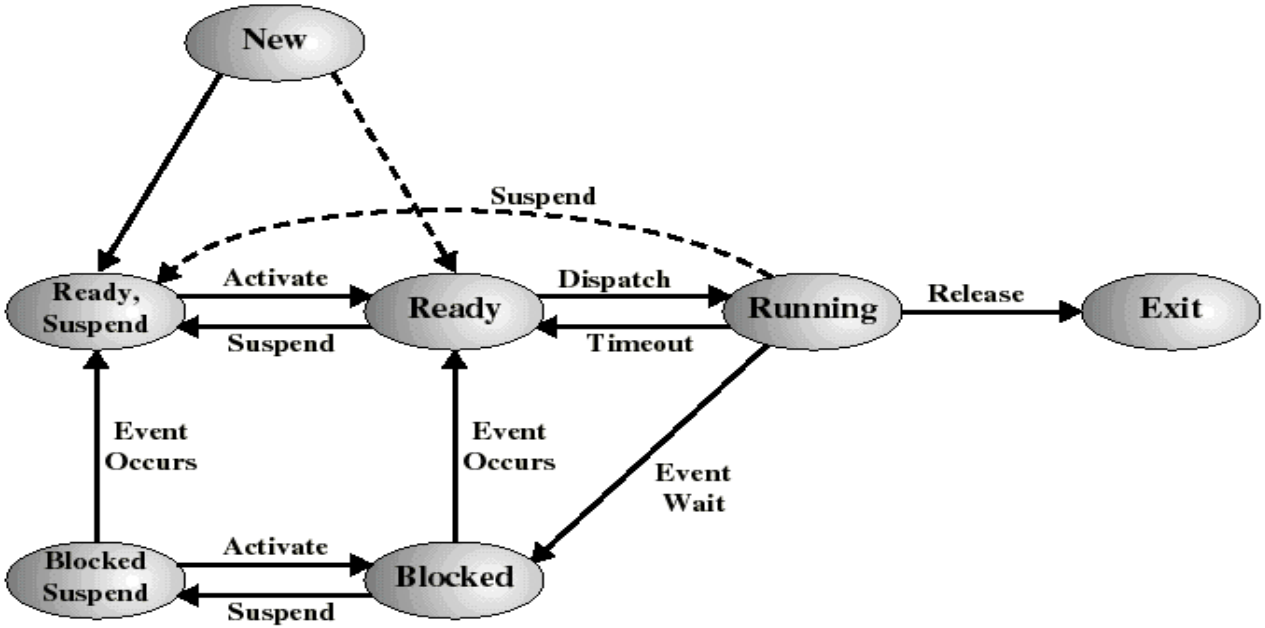
2.进程描述和控制

七状态进程模型

-有两个挂起状态

阻塞/挂起：进程在辅存中并等待一个事件

就绪/挂起：进程在辅存中，但是只要被载入主存就可以执行



2.进程描述和控制

- 阻塞 -->阻塞挂起
  - 当所有进程都阻塞，OS会安排空间让一就绪进程进入内存
- 阻塞挂起 --> 就绪挂起
  - 当等待的事件发生时（状态信息已在OS中）
- 就绪挂起-->就绪
  - 当内存中没有就绪进程时
- 就绪-->就绪挂起（较少见）
  - 当没有被阻塞的进程，而为了性能上的考虑，必须释放一些内存时

2.进程描述和控制

- 操作系统必须掌握关于每个进程和资源当前状态的信息
- 方法：构造并维护它所管理的每个实体的信息表
  - 内存
  - I/O设备
  - 文件
  - 进程

2.进程描述和控制

存储表

- 用于跟踪主存和辅存
- 包括的信息：
  - 分配给进程的主存
  - 分配给进程的辅存
  - 主存块或虚存块的任何保护属性，如哪些进程可以访问某些共享存储器区域
  - 管理虚存所需要的任何信息

2.进程描述和控制

I/O 表

- 管理计算机系统中的I/O设备和通道
- 包括的信息
  - I/O 设备可用或者已分配
  - I/O 操作的状态
  - 作为I/O传送的源和目标的主存单元

2.进程描述和控制

文件表

- 管理文件
- 包括的信息：
  - 文件是否存在
  - 文件在辅存中的位置
  - 当前状态
  - 其它属性

大部分信息可能由文件管理系统维护和使用

2.进程描述和控制

进程表

- 管理进程
- 包括的信息:
  - 进程的位置
  - 管理时所必需的属性
    - 进程 ID
    - 进程状态
    - 存储器中的位置

2.进程描述和控制

进程控制块(PCB)

系统利用PCB来控制和管理进程，所以PCB是系统感知进程存在的唯一标志

可以把进程控制块信息分成三类：

- ✓ 进程标识号
- ✓ 进程状态信息
- ✓ 进程控制信息

2.进程描述和控制

执行模式

区分与OS相关联以及与用户程序相关联的处理器执行模式

- 用户模式
  - 较少特权模式
  - 用户程序通常在该模式下运行
- 系统模式、控制模式或内核模式
  - 较多特权模式
  - 操作系统的内核
  - 典型功能（表3.7）
- 当发生中断或系统调用时，用户模式切换到系统模式。

2.进程描述和控制

进程创建

- 给新进程分配一个惟一的进程标识号
- 给进程分配空间
  - 进程映像中的所有元素
- 初始化进程控制块
  - 基于标准默认值和为该进程请求的特性来初始化
- 设置正确的连接
  - 例如，如果操作系统把每个调度队列都保存成链表，则新进程必须放置在就绪或就绪/挂起链表中
- 创建或扩充其它数据结构
  - 例如，为每个进程保存着一个计账文件，用于编制账单和性能评估

2.进程描述和控制

进程切换

- 在某一时刻，中断正在运行的进程，OS指定另一个进程为运行状态，并把控制权交给它

1）中断（与当前正在运行的进程无关的某种类型的外部事件相关）

- 时钟中断
  - 正在运行的进程的执行时间是否已经超过了最大允许时间段，如果超过了，进程切换到就绪状态
- I/O 中断
- 存储器失效
  - 引用的虚存地址不在主存中，则必须从辅存中把包含这个引用的存储器块调入主存中，发生存储器失效的进程被置为阻塞状态

2.进程描述和控制

进程切换

2）陷阱（内中断，与当前正在运行的进程所产生的错误和异常条件相关）

- 错误发生
- 可能导致进程被转换到退出状态

3）正在执行程序的系统调用

- 如打开文件
- 通常会导致把用户进程置为阻塞状态

3.线程

进程基本特点

进程的两个基本特点：

- 资源所有权
  - 给每个进程分配一虚拟地址空间，保存进程映像，控制一些资源（文件，I/O设备），有状态、优先级、调度
- 调度/执行
  - 沿着执行路径，其执行过程可能与其它进程的执行过程交替

3.线程

对进程系统必须完成的操作：

- 创建进程
- 撤消进程
- 进程切换

缺点：

时间空间开销大，限制并发度的提高

3.线程

- 在操作系统中，进程的引入提高了计算机资源的利用效率。但在进一步提高进程的并发性时，人们发现进程切换开销占的比重越来越大，同时进程间通信的效率也受到限制
- 线程的引入正是为了简化进程间的通信，以小的开销来提高进程内的并发程度
- 线程：有时称轻量级进程，进程中的一个运行实体，是一个CPU调度单位，资源的拥有者还是进程或称任务

3.线程

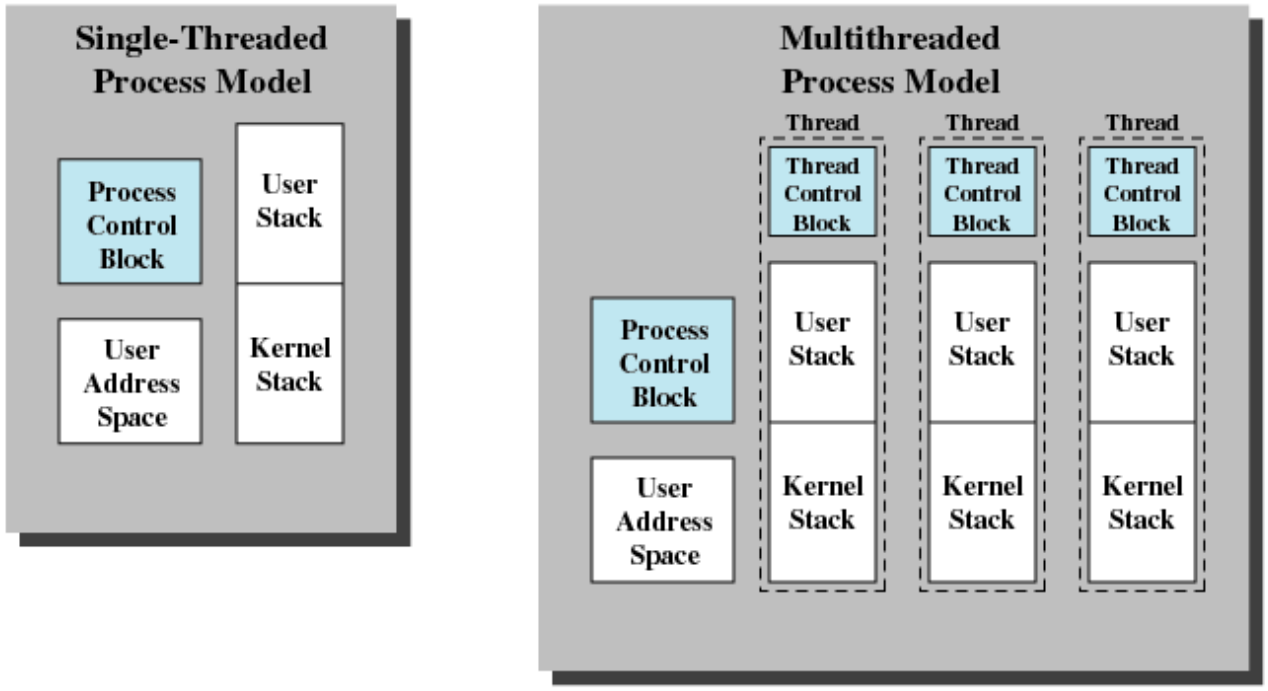


Figure 4.2 Single Threaded and Multithreaded Process Models

3.线程

- 在操作系统中，进程的引入提高了计算机资源的利用效率。但在进一步提高进程的并发性时，人们发现进程切换开销占的比重越来越大，同时进程间通信的效率也受到限制
- 线程的引入正是为了简化进程间的通信，以小的开销来提高进程内的并发程度
- 线程：有时称轻量级进程，进程中的一个运行实体，是一个CPU调度单位，资源的拥有者还是进程或称任务

3.线程

线程与进程的比较

- 调度
  - 传统的OS，作为拥有资源的基本单位和独立调度、分派的基本单位是进程。
  - 在引入线程的OS中，把线程作为调度和分配的基本单位。
  - 线程基本上不拥有资源，可以轻装上阵，显著提高了系统的并发程度。

3.线程

线程与进程的比较

- 并发性
  - 在引入线程的OS中，不仅进程之间可以并发执行，而且在一个进程内的多个线程间也可以并发执行，使得OS具有更好的并发性。

3.线程

线程与进程的比较

- 拥有资源
  - 进程比线程拥有更多的资源，是系统中拥有资源的一个基本单位。
  - 除了一点必不可少的资源外，线程自己一般不拥有资源，但它可以访问其隶属进程的资源

3.线程

线程与进程的比较

- 系统开销
  - 创建或撤销进程时，OS要付出的开销明显大于线程创建或撤销时的开销。
  - 进程切换时的开销也远大于线程切换时的开销。

3.线程

进程对线程的影响

- 一个进程的挂起会导致挂起这个进程的所有线程，因为进程中的所有线程共享同一个地址空间
- 进程的终止会导致进程中所有线程的终止

3.线程

线程状态

- 与线程状态变化相关的基本操作
  - 派生
    - 派生另一个线程
  - 阻塞
  - 解除阻塞
  - 结束
    - 释放其寄存器上下文和栈

3.线程

线程同步

- 原因
  - 共享同一个地址空间和其他资源
  - 一个线程对资源的任何修改影响其它线程的环境
- 同步各种线程的活动，互不干涉且不破坏数据结构

3.线程

线程分类

- 用户级线程
- 内核级线程

3.线程

用户级线程

- 有关线程管理的所有工作都由应用程序完成
  - 通过线程库（用户空间）
  - 一组管理线程的过程
- 内核没有意识到线程的存在
- 线程切换不需要核心态特权
- 调度是应用特定的

3.线程

用户级线程优点

优点:

- 线程切换不调用核心，没有模式切换。
- 调度是应用程序特定的：可以选择最好的算法
- ULT可运行在任何操作系统上（只需要线程库），可以在一个不支持线程的OS上实现

缺点:

- 大多数系统调用是阻塞的，因此核心阻塞进程，故进程中所有线程将被阻塞
- 核心只将处理器分配给进程，同一进程中的两个线程不能同时运行于两个处理器上

3.线程

内核级线程

- 内核为进程以及进程中的各个线程保存上下文环境信息
- 线程之间的切换需要内核来支持
- 调度是由内核基于线程的基础完成的
- 没有线程库，但对核心线程工具提供API
- Windwos, Linux, and OS/2 都有这种线程例子

3.线程

内核级线程优点

- 优点
  - 对多处理器，内核可以同时调度同一进程的多个线程
  - 阻塞是在线程一级完成，进程中一个线程被阻塞，内核可以调度同一进程的另一个线程
  - 核心例程是多线程的

4.并发：互斥与同步

并发的概念

- 在单处理器多道程序设计中，进程会被交替的执行，因而表现出一种并发执行的外部特征。

4.并发：互斥与同步

竞争条件

- 竞争条件发生在多个进程或线程读写数据时，其最终的结果依赖于多个进程的指令执行顺序。
- 竞争条件不解决，则程序的执行将具有不可再现性。

4.并发：互斥与同步

操作系统关注的问题

- 跟踪各种不同的进程
  - 使用进程控制块来实现
- 为每个活跃进程分配和释放资源
  - 处理器时间
  - 存储器
  - 文件
  - 输入／输出设备
- 保护每个进程的数据和物理资源
- 进程的功能和输出结果必须与其执行速度无关

4.并发：互斥与同步

进程的交互

- 进程之间不知道对方
  - 资源竞争
  - 互斥、死锁、饥饿
- 进程间接知道对方
  - 共享合作
  - 互斥、死锁、饥饿、数据一致性
- 进程直接知道对方
  - 通信合作
  - 死锁、饥饿

4.并发：互斥与同步

进程的交互

- 同步：对多个相关进程在执行次序上进行协调，以使并发执行的诸进程之间能有效地共享资源和相互合作，从而使程序的执行具有可再现性。

- 互斥：同步的特例，当一个进程在临界区访问共享资源时，其他进程不能进入该临界区访问任何资源。

4.并发：互斥与同步

进程的交互

- 临界资源(critical resource)：一次仅允许一个进程访问的资源。

  例如：进程A、B共享一台打印机，若让它们交替使用则得到的结果肯定不是我们希望的。

  临界资源可能是硬件，也可能是软件：变量，数据，表格，队列等。

  并发进程对临界资源的访问必须作某种限制，否则就可能出与时间有关的错误，如：联网售票。

- 临界区(critical section)：临界段，在每个程序中，访问临界资源的那段程序。

4.并发：互斥与同步

进程间通过共享的合作

- 共享对象
  - 如变量、文件和数据库
- 涉及的控制问题：互斥、死锁和饥饿
- 写操作必须保证互斥
- 新问题：数据一致性
  - 例子：**p134**
  - 临界区用来保护数据的完整性

4.并发：互斥与同步

进程间通过通信的合作

- 消息的传递
  - 不需要互斥
- 可能存在死锁问题
  - 每个进程都在等待来自对方的通信
- 可能存在饥饿问题
  - 两个进程传送信息给彼此而另外的一个进程等待一个信息

4.并发：互斥与同步

互斥的要求

- 强制实施互斥：在多个想要同时访问临界资源的进程中，一次只允许一个进程进入临界区（忙则等待）
- 一个在非临界区停止的进程必须不干涉其它进程
- 有限等待：不会死锁或饥饿（不允许出现需要访问临界区的进程被无限延迟的情况）

4.并发：互斥与同步

互斥的要求

- 空闲让进：当没有进程在临界区中时，任何需要进入临界区的进程必须能够立即进入
- 对相关进程的速度和处理器的数目没有任何要求和限制
- 一个进程驻留在临界区中的时间必须是有限的
- 让权等待：当进程不能进入自己的临界区时，应立即释放处理机，以免陷入“忙等”

4.并发：互斥与同步

互斥的实现

- 由并发的进程利用软件的方法解决
  - 实现过于复杂，需要较高的编程技巧
- 利用硬件的支持解决
- 利用操作系统或程序设计语言提供的某种支持（例如信号量机制）解决

4.并发：互斥与同步

互斥的实现

- 硬件的支持
  - 中断禁用
  - 专门的机器指令

4.并发：互斥与同步

信号量

- 基本原理
  - 两个或多个进程可以通过简单的信号进行合作，一个进程可以被迫在某一位置停止，直到它接收一个特定的信号。
  - 通过信号量**s**，进程可以执行**semSignal(s)**原语操作（或称为**V**操作）来发送信号；
  - 进程也可以执行**semWait(s)**原语操作（或称为**P**操作）来接收信号。
  - 如果相应的信号没有发送，则接收信号的进程被阻塞，直到信号发送完为止。

4.并发：互斥与同步

一般信号量上的三个操作

- 初始化操作
  - 一个信号量可以初始化成非负整数
- **semWait**操作
  - **semWait**操作使信号量减**1**。如果值变成负数，则执行**semWait**的进程被阻塞
- **semSigal**操作
  - **semSignal**操作使信号量加**1**。如果值不是正数，则因**semWait**操作阻塞的进程被解除阻塞

4.并发：互斥与同步

生产者/消费者问题

- 生产者/消费者问题是一种同步问题的抽象描述。计算机系统中的每个进程都可以消费（使用）或生产（释放）某类资源。这些资源可以是硬件资源，也可以是软件资源。
- 当某一进程使用某一资源时，可以看作是消费，称该进程为消费者。而当某一进程释放某一资源时，它就相当于生产者。

4.并发：互斥与同步

生产者/消费者问题

- 一个或多个的生产者产生某种类型的数据，并放置在缓冲区中
- 一个或多个消费者从缓冲区中取数据，每次取一项
- 任何时候只有一位生产者或消费者可以访问缓冲区

4.并发：互斥与同步

一般信号量解决无限缓冲区生产者/消费者问题

- 为解决生产者消费者问题，应该
  - 设一个同步信号量n，表明缓冲区中已放入产品数量，初值置为0；
  - 由于在此问题中有M个生产者和N个消费者，它们在执行生产活动和消费活动中要对缓冲区进行操作。由于缓冲区是一个临界资源，必须互斥使用，所以，另外还需要设置一个互斥信号量s，其初值为 1 ；

4.并发：互斥与同步

生产者/消费者问题

- 在每个程序中用于实现互斥的semWait(s)和semSignal(s)必须成对地出现。
- 对资源信号量e和n的semWait和semSignal操作，同样需要成对地出现，但它们分别处于不同的程序中。
- 在每个程序中的多个semWait操作顺序不能颠倒。应先执行对资源信号量的semWait操作，然后再执行对互斥信号量的semWait操作，否则可能引起进程死锁
- semSignal操作的次序倒是无关紧要

4.并发：互斥与同步

信号量的实现

- **Wait和signal操作必须作为原子原语实现**
- 解决方案：
  - 软件方法
  - 硬件方法
  - 单处理器系统可用禁止中断方法

4.并发：互斥与同步

管程

- 信号量机制功能强大，但使用时对信号量的操作分散，而且难以控制，读写和维护都很困难。
- 因此后来又提出了一种集中式同步进程——管程。其基本思想是将共享变量和对它们的操作集中在一个模块中，操作系统或并发程序就由这样的模块构成。这样模块之间联系清晰，便于维护和修改，易于保证正确性。

4.并发：互斥与同步

管程

- 管程是一个软件模块
  - 由一个或多个过程、一个初始化序列和局部数据组成
  - 描述共享资源的数据结构和在数据结构上的共享资源管理程序
- 主要的特性
  - 局部数据变量只能被管程的过程访问
  - 一个进程通过调用管程的一个过程进入管程
  - 任何时候只能有一个进程在管程中执行(互斥)

4.并发：互斥与同步

进程通信类型

- 1、消息传递系统
- 2、共享存储器系统
- 3、管道通信（共享文件方式）

4.并发：互斥与同步

消息传递

- 进程交互的两个基本要求：
  - 同步（为实施互斥）
  - 通信（交换数据）
- 以一对原语的形式提供：

```
send (destination, message)
receive (source, message)
```

5.并发：死锁和饥饿

死锁的基本概念

- 定义：一组相互竞争系统资源或进行通信的进程间的“永久”阻塞
- 没有有效的通用解决方案
- 涉及到两个或更多的进程之间因对资源的需求所引起的冲突

5.并发：死锁和饥饿

死锁的条件

- 互斥
  - 一次只有一个进程可以使用一个资源
- 占有且等待
  - 当一个进程等待其它资源时，可以占有已分配的资源
- 不可抢占
  - 不能强行抢占进程拥有的资源
- 循环等待
  - 存在一个封闭的进程链，使得每个进程至少占有链中下一个进程所需要的一个资源

5.并发：死锁和饥饿

预防死锁

通过设置某些限制条件，去破坏死锁四个充要条件中的一个或多个，来防止死锁。

较易实现，广泛使用，但由于所施加的限制往往太严格，可能导致系统资源利用率和系统吞吐量的降低。

5.并发：死锁和饥饿

避免死锁

- 允许三个必要条件，但通过明智的选择，确保永远不会到达死锁点
  - 不启动一个进程，如果它的请求会导致死锁
  - 不允许一个进程增加的资源请求，如果这个分配会导致死锁

- 比预防允许更多的并发

5.并发：死锁和饥饿

避免死锁

系统的状态是当前给进程分配的资源情况

- 安全状态
  - 系统按某种顺序并发进程，并使它们都能达到获得最大资源而顺序完成的序列为安全序列。
  - 能找到安全序列的状态为安全状态，安全状态不会导致死锁
- 不安全状态指不安全的一个状态
  - 不安全状态不一定导致死锁

5.并发：死锁和饥饿

避免死锁

- 银行家算法

银行家对每一个请求进行检查，看他是否有足够的资源满足一个距最大需求最近的客户。如果可以，则这笔投资认为是能够收回的，然后接着检查下一个距最大需求最近的客户，如此反复下去。直到所有投资最终都被收回。

5.并发：死锁和饥饿

银行家算法

当一进程提出资源申请时，银行家算法执行下列步骤以决定是否向其分配资源：

（1）检查该进程所需要的资源是否已超过它所宣布的最大值。

（2）检查系统当前是否有足够资源满足该进程的请求。

（3）系统试探着将资源分配给该进程，得到一个新状态。

（4）执行安全性算法，若该新状态是安全的，则分配完成；若新状态是不安全的，则恢复原状态，阻塞该进程。

5.并发：死锁和饥饿

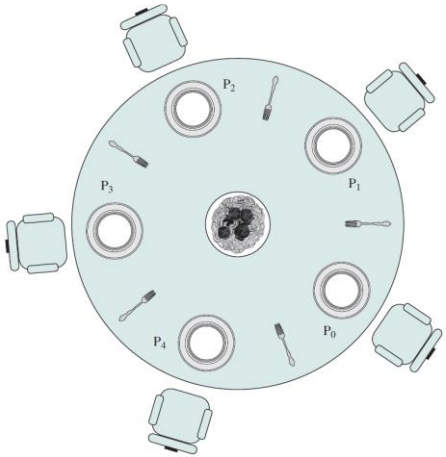
检测死锁

事先并不采取任何限制，也不检查系统是否进入不安全区，允许死锁发生，但可通过检测机构及时检测出死锁的发生，并精确确定与死锁有关的进程和资源，然后采取适当措施，将系统中已发生的死锁清除掉

5.并发：死锁和饥饿

哲学家进餐问题

- 五个哲学家共用一张圆桌，桌上有五个盘子、五把叉子和一大盘面条。
- 平时哲学家进行思考，想吃面时便试图取其左右最近的叉子，只有拿到两把叉子时他才能进餐。进餐完毕后，放下叉子继续思考。
- 要求互斥使用叉子，并且没有死锁与饥饿。



6.内存管理与虚拟内存

内存管理需求

- 重定位
- 保护
- 共享
- 逻辑组织
- 物理组织

6.内存管理与虚拟内存

重定位

- 程序员不可能事先知道程序执行期间，它被放置在主存中的哪个地方，当程序从辅存调入主存时，必须进行程序在主存中的定位
- 当程序执行期间，通过使用交换技术可能会被换入或换出主存。换入时，可以把该进程重定位到存储器的不同区域
- 处理器硬件和OS软件必须能够把程序代码中的存储器访问（以相对地址形式表示）转换到实际的物理存储器地址

6.内存管理与虚拟内存

保护

- 其它进程中的程序不能为未经授权的读操作或写操作访问这个进程的内存单元
- 程序编译时不可能为了保护而检查程序的绝对地址
- 进程运行时必须检查产生的所有内存访问
  - **Operating system** 不能预测一个程序可能产生的所有存储器访问
  - 由处理器硬件在访问指令执行评估
    - 不能访问内核空间的任何代码和数据
    - 未经许可不能访问其他进程的数据区

6.内存管理与虚拟内存

共享

- 任何保护机制要允许多个进程访问主存的同一部分

  进程之间共享/通信的合作

- 允许每个进程访问该程序的同一个副本要优于让它们有自己单独的副本

  节省存储空间

共享和保护两者相互配合实现某一区域或某种级别的共享

6.内存管理与虚拟内存

逻辑组织

- 内存被组织成线性(或一维)的地址空间
- 大多数程序被组织成模块(用户程序结构)
- 模块化组织的好处:
  - 可以独立的编写或编译模块
  - 可以给不同的模块不同的保护级别(只读、只执行)
  - 共享模块

6.内存管理与虚拟内存

物理组织

存储器至少有两级(主存+辅存)，一般被组织成线性的地址空间，物理组织着重解决主存和辅存之间的信息流动及重定位问题，

这由程序员负责是不切实际的：

- 可供程序和数据使用的主存可能是不够的
  - 覆盖允许不同的模块可以指派给存储器中同一区域
- 程序员在编写代码时不知道有多少空间可用以及可用空间在哪里

一般由处理机硬件(MMU单元)和OS软件两者配合解决

6.内存管理与虚拟内存

内存分区

单一连续分配方式（单分区分配）

分区式分配方式

a. 固定分区分配方式

b. 动态分区分配方式（可变分区）

c. 可重定位分区分配方式（紧缩技术解决外碎片）

6.内存管理与虚拟内存

固定分区

- 大小相等的分区
  - 只要存在可用分区，进程就能装入分区。
- 大小不等的分区
  - 最简单的方法：
    - 把每个进程指定到适应它的最小分区
    - 每个分区一个队列，保存为这个分区换出的进程
    - 优点：可以使一个分区内部浪费的空间最少
    - 问题：有的分区可能一直都不会使用
    - 解决：所有进程使用一个队列

6.内存管理与虚拟内存

动态分区

- 最佳适配算法(最佳适应算法，**best-fit)**
- 首次适配算法
- 下次适配算法

6.内存管理与虚拟内存

非连续内存分配

- 分页
  - 不具备页面对换功能的称为基本分页存储管理方式
- 分段
- 段页

6.内存管理与虚拟内存

非连续内存分配

- 页框（帧、页帧、物理页面, Frame, Page Frame）
  - 把物理地址空间划分为大小相同的基本分配单位
  - 2的n次方，如512, 4096, 8192
- 页（页面、逻辑页面、Page）
  - 把逻辑地址空间也划分为相同大小的基本分配单位
  - 页框和页的大小必须是相同的
- 页表（操作系统为每个进程维护一个页表）
  - 页表给出了该进程的每一页对应的页框位置
- 逻辑地址包括一个页号和页的偏移量。CPU使用页表产生物理地址（页框号，偏移量）

6.内存管理与虚拟内存

非连续内存分配

- 页框（帧、页帧、物理页面, Frame, Page Frame）
  - 把物理地址空间划分为大小相同的基本分配单位
  - 2的n次方，如512, 4096, 8192
- 页（页面、逻辑页面、Page）
  - 把逻辑地址空间也划分为相同大小的基本分配单位
  - 页框和页的大小必须是相同的
- 页表（操作系统为每个进程维护一个页表）
  - 页表给出了该进程的每一页对应的页框位置
- 逻辑地址包括一个页号和页的偏移量。CPU使用页表产生物理地址（页框号，偏移量）

6.内存管理与虚拟内存

分页的地址变换机构

基本任务：借助页表完成逻辑页号-物理块号的映射。

1、基本地址变换机构：

- 页表寄存器：页表始址，页表长度
- 平时每个进程未执行时将其信息（如页表长度、始址）放在PCB中，执行时将它们装入页表寄存器。
- 地址变换机构自动将逻辑地址分为页号和页内地址
- 越界保护
- 检索页表
- 生成物理地址

6.内存管理与虚拟内存

分段

- 分段技术细分用户程序，可以把程序和相关数据划分为几个段
  - 所有程序的所有段不要求具有相同的长度。
  - 有一个最大段长度（段长度的最大限度）
- 逻辑地址由两部分组成：
  - 段号
  - 偏移量

6.内存管理与虚拟内存

虚拟内存的概念

- 虚存：把内存与外存有机的结合起来使用，从而得到一个容量很大的“内存”，这就是虚存
- 实现：当进程运行时，先将程序的一部分装入内存，另一部分暂时留在外存，当要执行的指令不在内存时，由系统自动完成将它们从外存调入内存工作（请求调入、置换）
- 目的：提高内存利用率

6.内存管理与虚拟内存

虚拟内存的概念

- 具有请求调入功能和置换功能，能从逻辑上对内存容量进行扩充的一种存储系统。
- 从用户的观点看，虚拟内存具有比实际内存大得多的容量，其逻辑容量由逻辑地址结构及其内存和外存之和决定，其运行速度接近于内存速度，而每位成本接近于外存。

6.内存管理与虚拟内存

虚拟内存的实现方法

- 简单的分页系统 → 请求分页系统.
- 简单的分段系统 → 请求分段系统
- 简单的段页式系统 → 请求段页式系统

6.内存管理与虚拟内存

请求式分页

- 页表项：除了页号对应的页框号外，增加存在位、修改位、控制位等

6.内存管理与虚拟内存

请求式分页

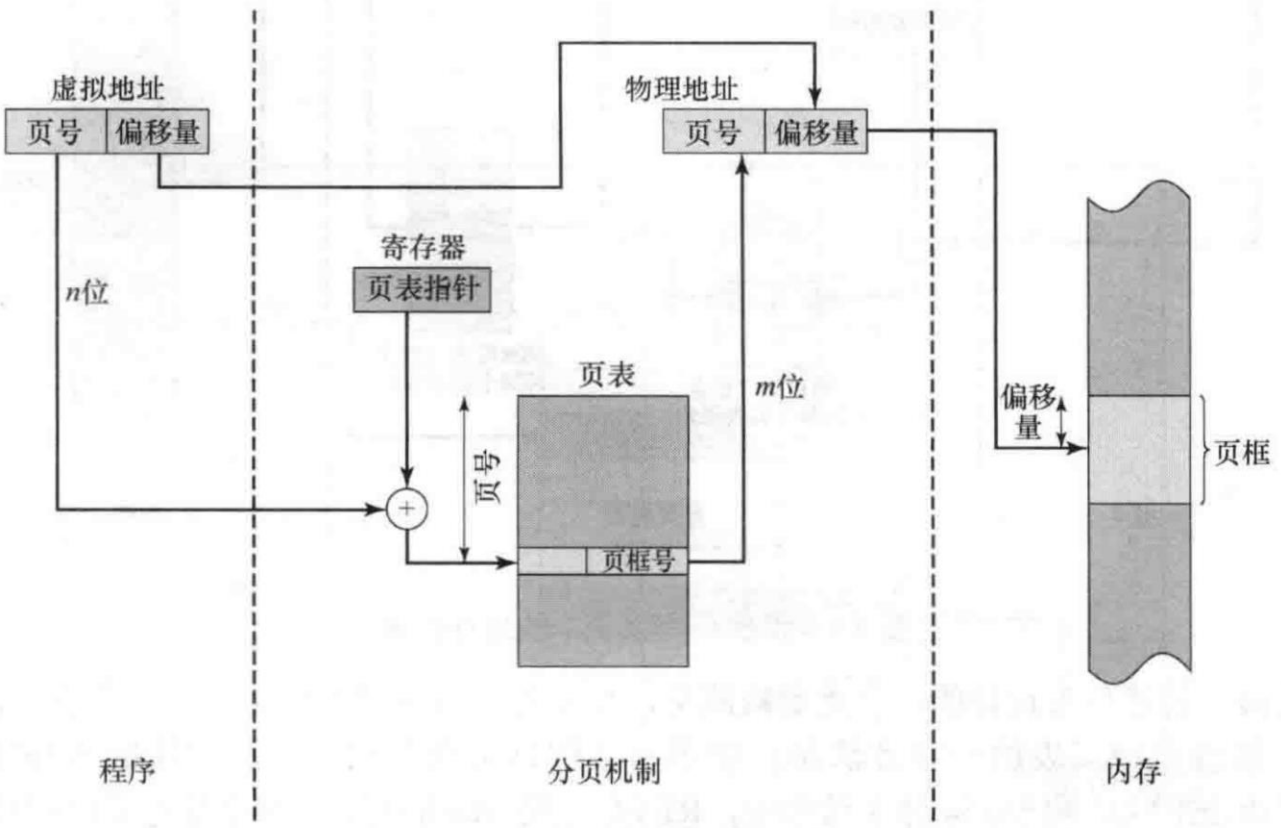


图 8.2　分页系统中的地址转换

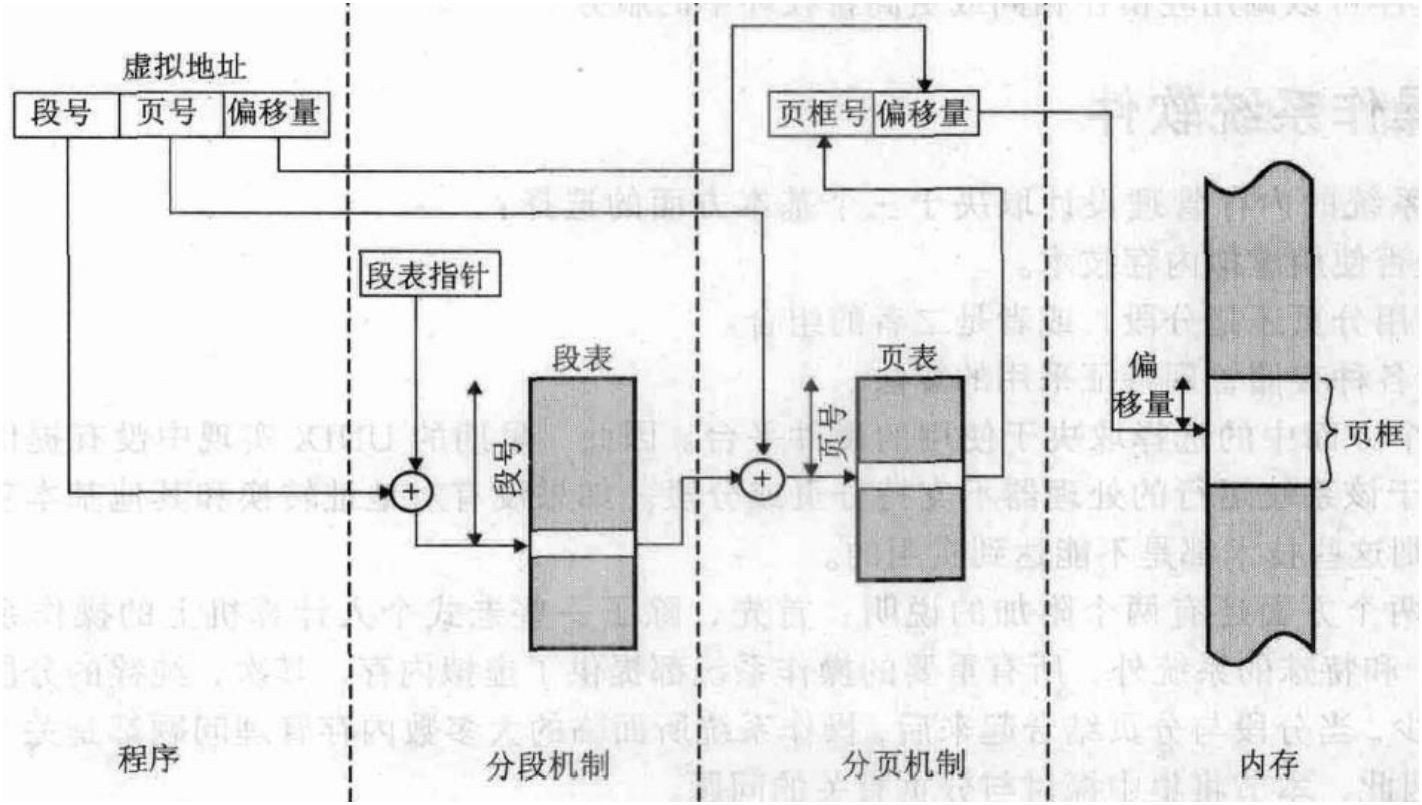
6.内存管理与虚拟内存

请求式分段

- 在简单段式存储管理的基础上，增加请求调段和段置换功能
  - 每个段表项中需要有一位**P**表明相应的段是否在主存中
  - 需要另一个修改位**M**，用于表明相应的段从上一次被装入主存到目前为止，其内容是否被改变

6.内存管理与虚拟内存

请求式分段



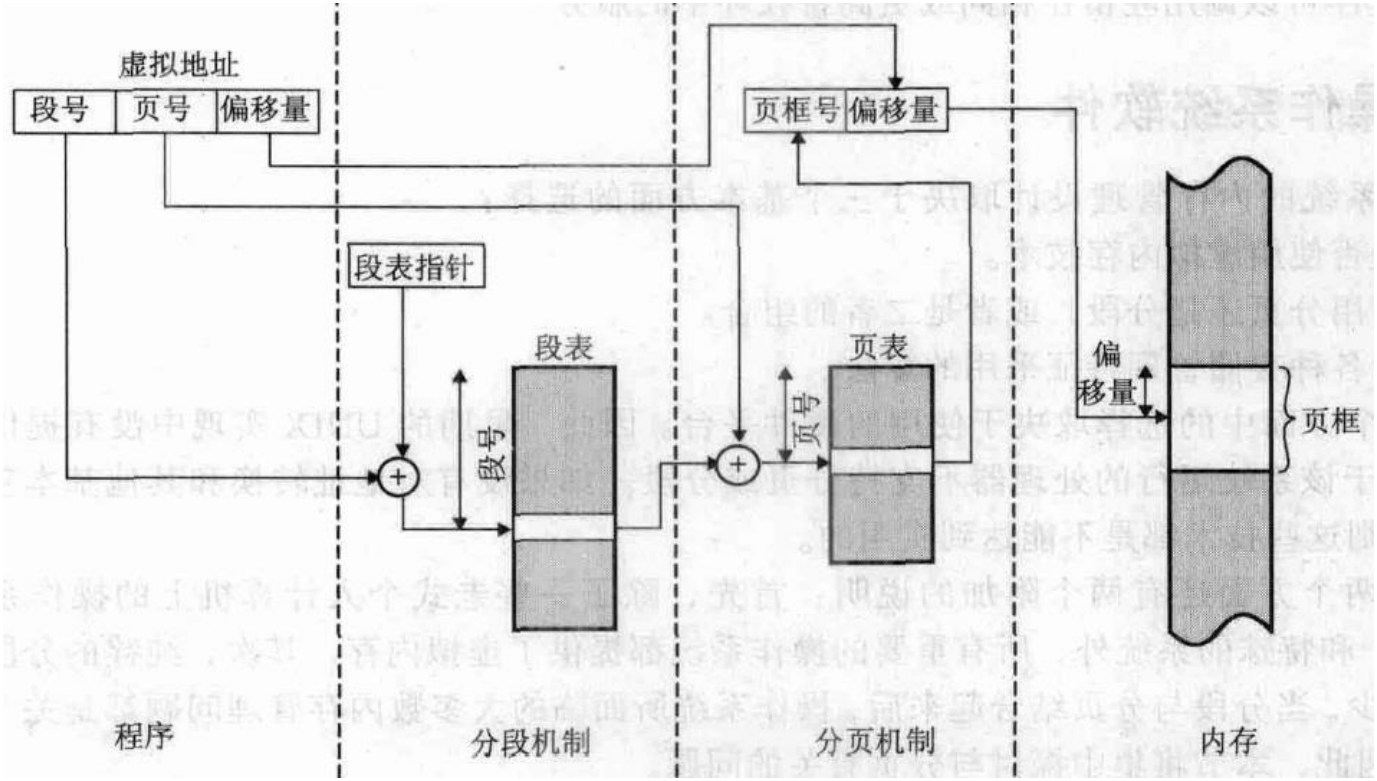
6.内存管理与虚拟内存

请求式分段

- 在简单段式存储管理的基础上，增加请求调段和段置换功能
  - 每个段表项中需要有一位**P**表明相应的段是否在主存中
  - 需要另一个修改位**M**，用于表明相应的段从上一次被装入主存到目前为止，其内容是否被改变

6.内存管理与虚拟内存

段页式



6.内存管理与虚拟内存

页面置换算法

- 通过一个进程运行时的页面访问(引用)顺序（又称为页地址流，或引用串）来检验置换算法性能，计算其缺页率
- 追求最低的缺页率

6.内存管理与虚拟内存

页面置换算法

- 最佳置换算法 (OPT, optimal)
- 最近最少使用 LRU
- 先进先出算法(FIFO)
- 时钟算法 CLOCK

6.内存管理与虚拟内存

Belady现象

- 采用FIFO等算法时，可能出现分配的物理页面数增加，缺页次数反而升高的异常现象

6.内存管理与虚拟内存

驻留集

- 驻留集指虚拟页式管理中给进程分配的物理页面数目

6.内存管理与虚拟内存

驻留集策略

- 固定分配+局部置换
- 可变分配+局部置换
- 可变分配+全局置换

6.内存管理与虚拟内存

工作集策略

- 工作集是一个进程执行过程中所访问页面的集合，可用一个二元函数**W(t, Δ)**表示

6.内存管理与虚拟内存

工作集置换算法

- 换出不在工作集中的页面
- 当前时刻前S个内存访问的页引用是工作集，S被称为窗口大小
  - 访存链表：维护窗口内的访存页面链表
  - 访存时，换出不在工作集的页面；更新访存链表
  - 缺页时，换入页面；更新访存链表

6.内存管理与虚拟内存

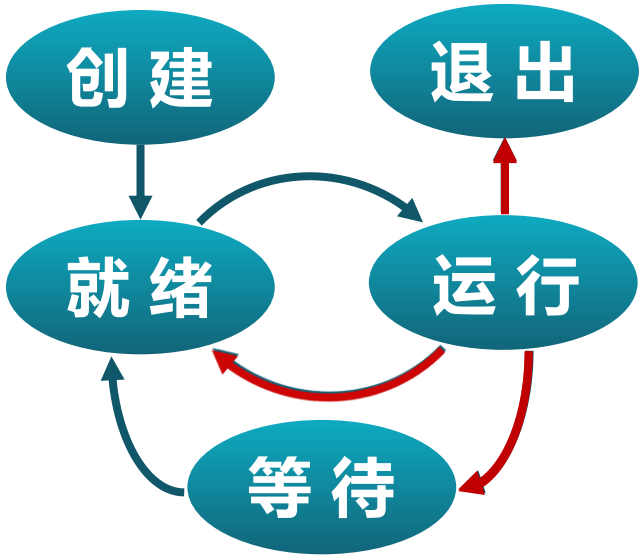
缺页率算法

- 页面被访问时的处理：每个页面设立使用位(use bit)，在该页被访问时设置use bit=1；
- 缺页时的处理：每次缺页时，由操作系统计算与上次缺页的“虚拟时间”间隔t。
- 缺页时对驻留集的调整：定义一个缺页时间间隔的阈值(threshold)F。依据t和F来修改驻留集。
  - 如果t小于F，则所缺页添加到驻留集中；
  - 否则，将所有use bit=0的页面从物理内存清除并缩小驻留集；随后，对驻留集中的所有页面设置use bit=0；

7.处理机调度

调度时机

- 在进程/线程的生命周期中的什么时候进行调度
  - 进程从运行状态切换到等待状态
  - 进程被终结了
- 非抢占系统
  - 当前进程主动放弃CPU时
- 可抢占系统
  - 中断请求被服务例程响应完成时
  - 当前进程被抢占



7.处理机调度

调度目标

- CPU使用率
- 吞吐量
- 周转时间
- 等待时间
- 响应时间

7.处理机调度

调度策略

- 先来先服务（FCFS）
- 轮转法（Round Robin，RR）
- 最短进程优先（SPN）
- 最短剩余时间（SRT）
- 最高响应比优先（HRRN）
- 反馈法/多级队列反馈
- 公平共享算法

8.I/O与磁盘调度

I/O种类

字符设备

- 适合与计算机用户通信，如：打印机，终端（显示器 ，键盘 ， 鼠标 ）

块设备

- 适合与电子设备通信，如磁盘驱动器， 磁带驱动器，光驱等

网络设备

- 适合与远程设备通信，以太网，无线网，蓝牙

8.I/O与磁盘调度

I/O方式

- 程序控制 I/O
  - 处理器代表进程给I/O模块发送一个I/O命令，该进程进入忙等待，等待操作的完成，然后才可以继续执行
- 中断驱动 I/O
  - 处理器代表进程发送一个I/O命令,然后继续执行后续指令，当I/O模块完成工作后，处理器被该模块中断。
- 直接存储器存取 (DMA)
  - DMA 模块控制主存和 I/O模块之间的数据交换

8.I/O与磁盘调度

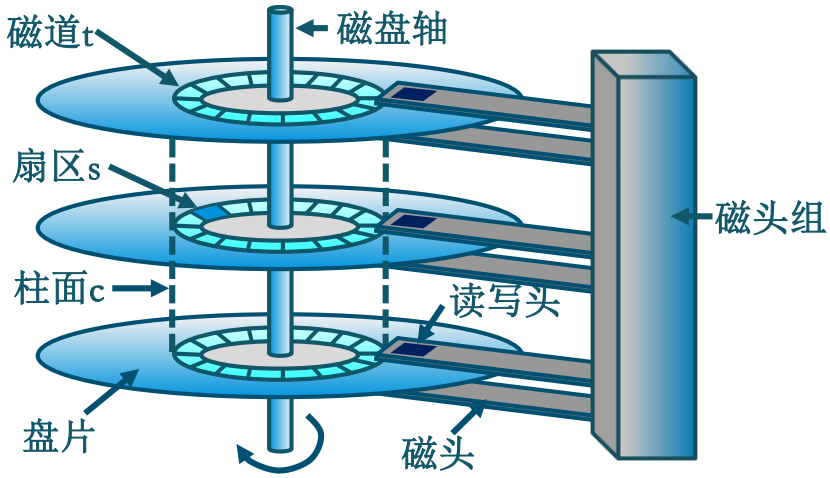
I/O缓冲

- 单缓冲
- 双缓冲
- 循环缓冲

8.I/O与磁盘调度

磁盘

- 信息记录在磁道上，多个盘片，正反两面都用来记录信息，每面一个磁头
- 所有盘面中处于同一磁道号上的所有磁道组成一个柱面
- 每个扇区大小为512字节
- 物理地址形式：
  - 柱面号
  - 磁头号
  - 扇区号



8.I/O与磁盘调度

磁盘传送的时序

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

访问时间

寻道时间

旋转延迟

1/r＝旋转一周的时间

传输时间

b＝传输的比特数

N＝磁道上的比特数

r＝磁盘转数

8.I/O与磁盘调度

磁盘调度策略

- 先进先出 (First-in, first-out， FIFO)
- 最短服务时间优先(Shortest Service Time First, SSTF)
- SCAN（电梯算法）
  - C-SCAN（循环扫描）
  - C-LOOK（循环扫描）
  - N-step-SCAN
  - FSCAN

8.I/O与磁盘调度

磁盘缓存置换算法

- **Least Recently Used**
- **Least Frequently Used**
- 访问频率置换算法（**Frequency-based Replacement**）

9. 文件与文件系统

文件与文件系统概念

- 文件
  - 由用户创建的一种数据集合
  - 文件系统是操作系统一个重要部分
  - 文件拥有的理想属性:
    - 长期存在
    - 进程间可共享
    - 结构

9. 文件与文件系统

文件与文件系统概念

- 文件系统
  - 提供一种手段去存储数据(被组织为文件)
  - 提供一系列对文件进行操作的功能接口

9. 文件与文件系统

术语概念

- 域（field）
- 记录(record)
- 文件（file）
- 数据库（database）

9. 文件与文件系统

文件系统架构

- 访问方法层
  - 文件格式 – 向用户提供访问方法接口
- 逻辑 I/O
- 基本 I/O
- 基本文件系统
- 设备驱动程序

9. 文件与文件系统

文件组织

主要介绍5种文件组织：

1) 堆 （最简单的）
2) 顺序文件
3) 索引文件
4) 索引顺序文件
5) 直接文件或散列文件

9. 文件与文件系统

目录管理

- 目录包含有关文件的信息
  - 属性
  - 位置
  - 所有权
- 目录自身是操作系统拥有的一个文件，并且可以被各种文件管理例程访问
- 从用户的角度看，目录在用户所知道的文件名、应用和文件自身之间提供了一种映射
- 文件目录：文件控制块的有序集合
- 目录通常为系统中的每个文件保存的信息：
  - 基本信息、地址信息、访问控制信息、使用信息

9. 文件与文件系统

文件共享

- 在多用户系统中，要求允许文件在多个用户间共享
- 两个问题
  - 访问权限
  - 对同时访问的管理

9. 文件与文件系统

记录组块

- 记录是访问文件的逻辑单元，而块是与辅存进行I/O的单位，为执行I/O，记录必须组织成块
- 对于给定的块大小，有三种组块方法：

- 固定组块：记录长度固定，若干完整的记录放在一块，块后部可能会有内碎片。

- 可变长度跨越式组块：记录长度可变，尽可能占满一块。有些记录可能跨越两块，则用指针链接。

- 可变长度非跨越式组块：记录长度可变，但不跨快，有块内碎片。一个记录的长度不能超过块尺寸

9. 文件与文件系统

文件分配方法

- 连续分配
- 链接分配
  - 显式链接：把用于链接的指针显式存放在内存的一张表中，查找在内存中进行。
- 索引分配

9. 文件与文件系统

空闲空间管理

- 位表
- 链式空闲区
- 索引
- 空闲块列表

计算机系统/操作系统概述

- 习题：

计算机系统/操作系统概述

- 计算机系统/操作系统概述
- 进程与线程
- 内存管理与虚拟内存
- I/O与磁盘
- 文件系统