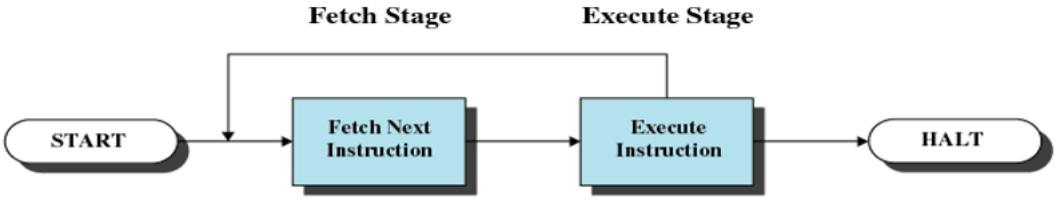
章节重点内容整理

笔记本:	operating_system		
创建时间:	2021/7/6 16:08	**更新时间:**	2021/10/13 16:22
作者:	134exetj717		
URL:	https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&tn=monline_3_dg&wd=%E...		

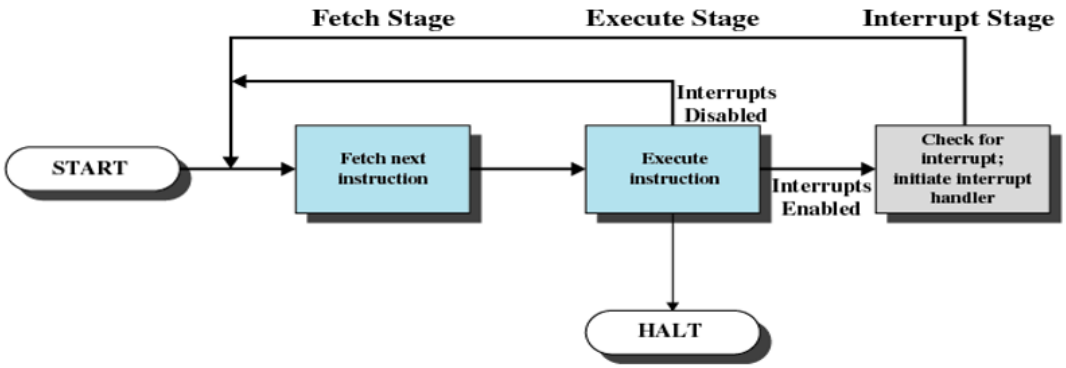
Operating Systems Review

第1章 计算机系统概述

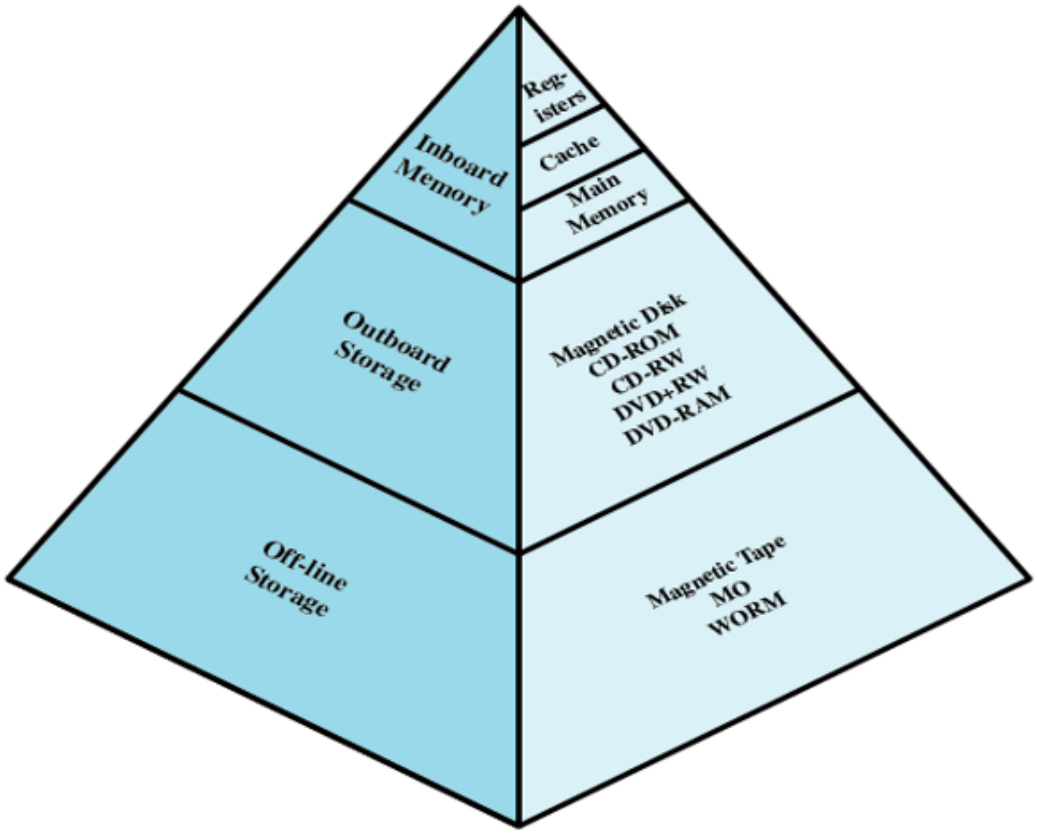
- 指令的执行（指令周期）：1.从主存中取出指令；2.进程执行每个指令



- 加入中断以后指令的执行过程：进程检查中断，如果中断不允许，那么就取指和执行指令；否则，停下当前程序，转去执行中断服务程序（interrupt-handler）



- 存储体系：Memory Hierarchy (存储层次)
    - 组成结构：
        1. 内存：寄存器，缓存，主存
        2. 外存：磁盘、CD
        3. 线下存储：磁带



第2章 操作系统概述

- 操作系统的主要功能
    1. 项目开发：编辑器和调试器
    2. 执行程序
    3. 访问I/O设备
    4. 对文件的受控访问
    5. 错误检测和响应：内部和外部硬件错误、内存错误、设备错误、软件错误等
    6. 会计：收集使用情况统计信息、监视性能、用于预测未来的增强功能
- 坦尼伯母把操作系统的功能归结为两点：

    自顶向下看(Top-down)，操作系统作为虚拟机(Virtual Machine)，为程序员提供统一的编程接口(Programming Interface)

    自底向上看(Bottom-up)，操作系统作为一个资源管理器(Resource Manager)，管理众多设备
- 操作系统的主要目标（Primary Objectives of Operating System）：

    Convenience：让计算机使用更方便

    Efficiency：让系统资源以一种高效的方式被使用

Ability to evolve（进化的能力）：允许有效开发、测试和引入新系统功能，而不干扰服务

- 操作系统的发展历史

    1. 手工操作（Serial Processing）：用户独占全机，CPU等待手工操作；
    2. 批处理系统（Simple Batch Systems，简单批处理）：在他的控制下，计算机能自动地、成批地处理一个或多个用户的作业。缺点是，每次主机中仅能存放一个作业，因此每次输出的时候都需要等待I/O设备完成任务；
    3. 多道程序系统（Multiprogramming ）：允许多个程序同时进入内存并运行。缺点，不提供人机交互能力，作业一旦进入系统，用户就不能干预其作业的运行；
    4. 分时系统（Time Sharing）：把处理机的运行时间分成很短的时间片，按时间片轮流把处理机分配给各联机作业使用。
        1. 多路性：若干个用户同时使用一台计算机。微观上看各用户轮流使用计算机；宏观上看是各用户并行工作；
        2. 交互性：用户可根据系统对请求的响应结果，进一步向系统提出新的请求。因此，又被称为交互式系统；
        3. 独立性：用户之间可以相互独立工作，互不干扰。系统保证了各用户程序运行的完整性；
        4. 及时性：系统可对用户的输入及时做出响应。
    5. 实时系统：（自动控制）

    - 类型：
        1. 实时控制系统：飞机飞行、导弹发射等的自动控制，计算机能够及时处理各类传感器送来的数据，然后控制相应的执行机构；
        2. 实时信息处理系统：预定飞机票、查询票价等，要求计算机能对终端设备发来的服务请求及时予以正确的回答。
    - 特点：
        1. 及时响应。对每个信息接收、分析处理和发送的过程必须在严格的时间限制内完成；
        2. 高可靠性。需采取冗余措施，双机系统前后台工作，也包括必要的保密措施等。

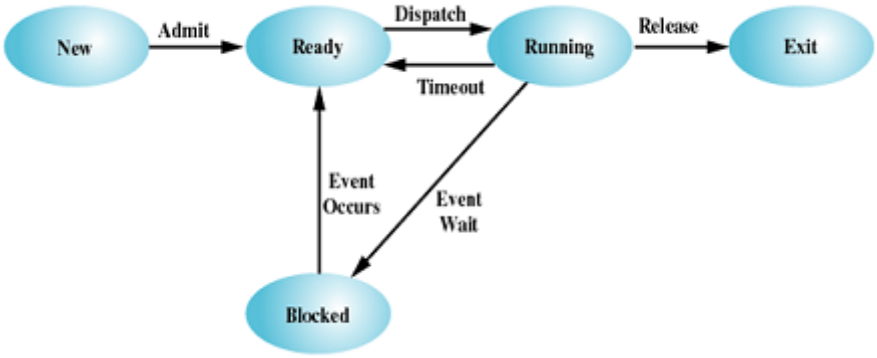
    6.通用操作系统：同时兼有多道批处理、分时、实时处理的功能。

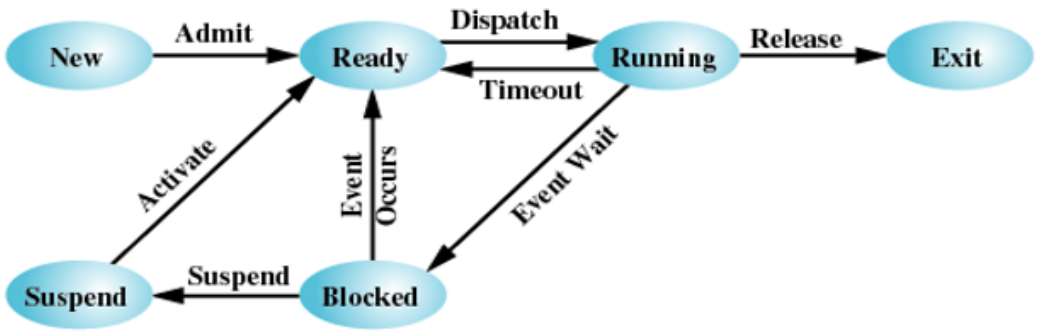
- 程序（Program）： 一组计算机能识别和执行的指令
- 作业（Job）：用户向计算机提交任务的任务实体，一个批处理就是一个作业

- 进程（Process）：一段可执行程序的执行过程，程序对数据集合的处理过程。
- 多道（Multiprogramming）：能够同时执行多个任务。
- 多任务：多个任务（进程）共享处理资源（如CPU）的方法。拓展，单核的并行是多任务的效果；但是多核的并行则可以做到真正意义上的独立工作。
- 操作系统的体系结构（Architecture）
    - 整体式结构、微内核结构、分层结构、客户/服务器结构、分布式结构

第3章 进程描述和控制

- 进程的概念/定义（Process）：一个程序执行的过程
- 进程状态及进程状态转换图（Process Transition Diagram）





(a) With One Suspend State

进程的属性（如何来描述进程）

进程表（Process Table）

进程表表项（Process Table Entry）

- 进程控制块（Process Control Block）：包含流程元素、由操作系统创建和管理、允许支持多个进程、进程表条目(进程表表项-进程表项)

- 进程如何被创建?
    1. 分配唯一的进程标识符
    2. 为进程分配空间
    3. 初始化过程控制块
    4. 建立适当的联系，例如：将新进程添加到用于调度队列的链表中
    5. 创建或扩展其他数据结构，例：维护会计档案
- 进程何时会被切换?
    1. 时钟中断
    2. I/O中断
    3. 内存失效
    4. 陷阱
    5. 系统调用
- 进程何时会终止？：阻塞、死锁

第4章 线程、SMP和微内核

- 引入线程（thread）的目的：
    1. 效率性：创建一个新线程比创建一个进程花费更少的时间，终止线程的时间也比终止进程的时间短，同一进程间线程间切换时间短
    2. 共享性：由于同一进程中的线程共享内存和文件，因此它们可以在不调用内核的情况下互相通信
- 线程和进程的关系：线程是进程的执行单位，进程是系统分配资源的基本单位，包括多个线程，
- 线程的生命期（线程状态）：standby（备用）、transition（过渡）、terminated（结束）

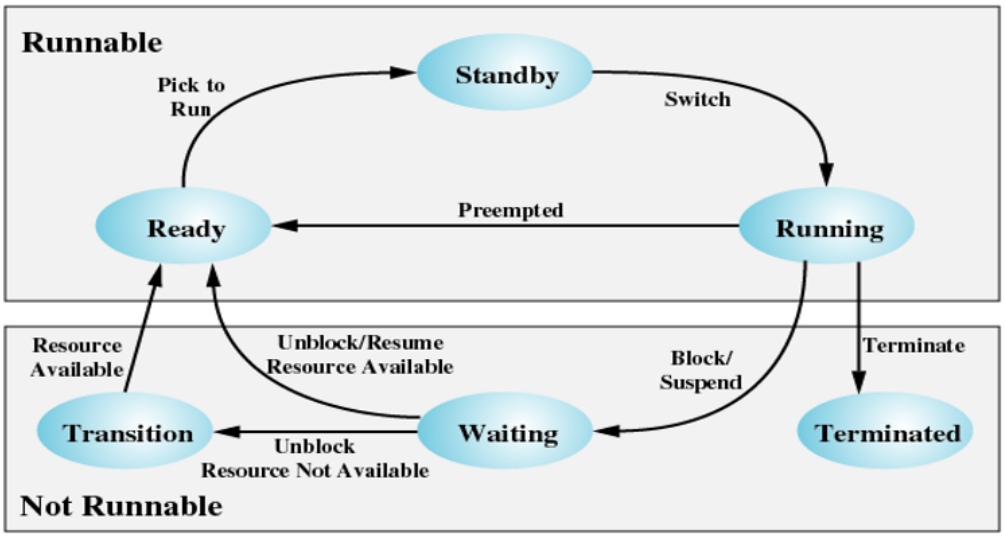
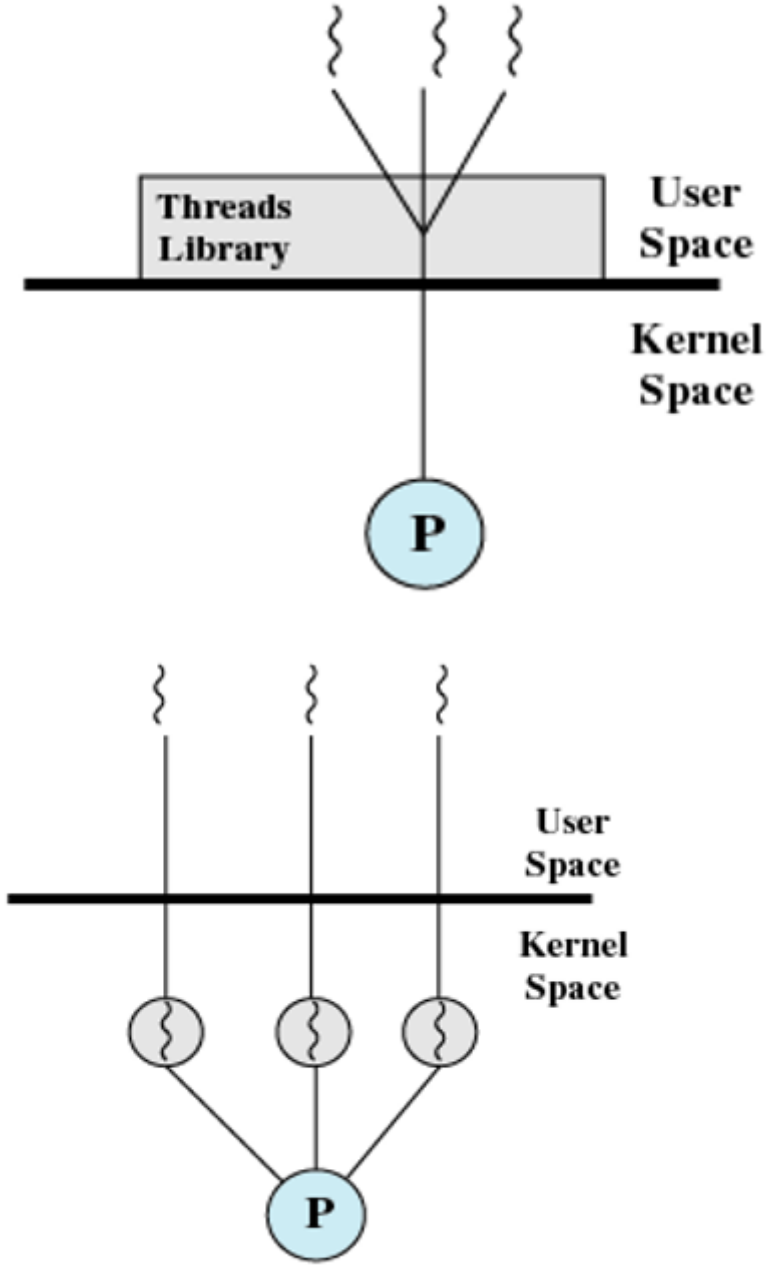
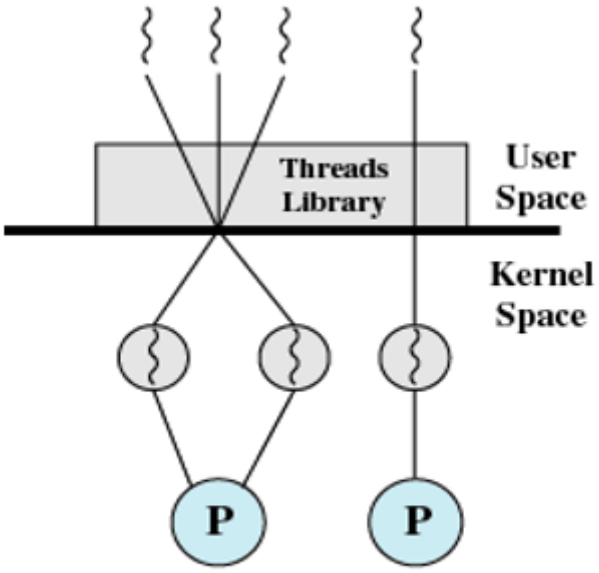


Figure 4.14 Windows Thread States

- 线程的不同实现方式，不同的实现方式对调度的影响
    - 用户级多线程（User-level threads）：所有线程的管理都由应用完成，只能访问受限的内存，且不允许访问外围设备，CPU资源可以被其他程序获取
    - 核心级多线程（Kernel-level threads）：内核包含进程和线程的内容信息，CPU可以访问内存所有数据，包括外围设备，例如硬盘、网卡，CPU也可以将自己从一个程序切换到另一个程序
    - 两者结合：用户空间中完成线程创建，应用程序中线程的大量调度和同步



(b) Pure kernel-level



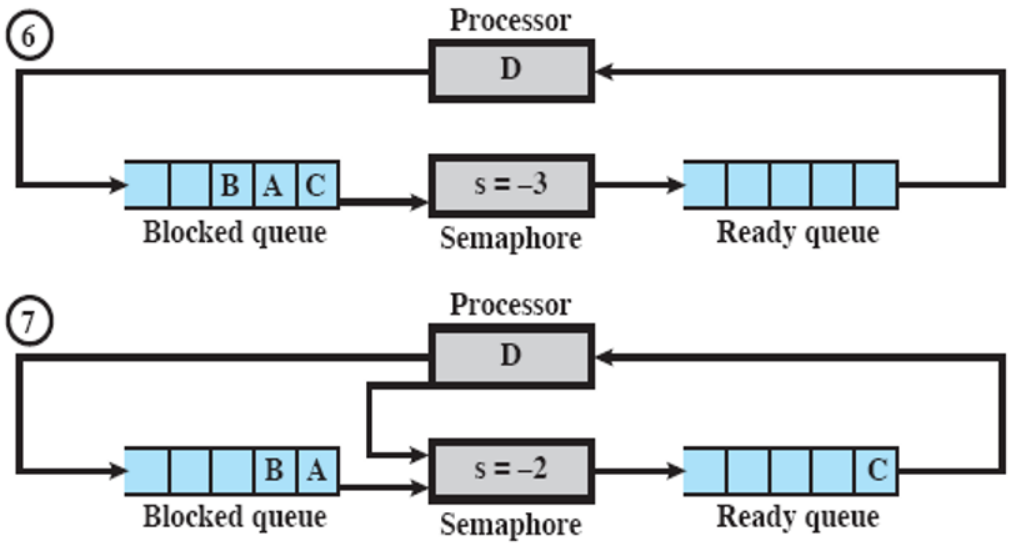
(c) Combined

- 并发与并行、并发的实现机制（interleaving和overlapping）：
    - 并发：在一个CPU上同时运行多个程序，微观上穿行，宏观上并行，比如经典的批处理，并发性是对有限物理资源强制使多用户共享以提高效率；
    - 并行：真正意义上的同时进行，比如多道程序
- 微内核（Microkernel）：小型操作系统的核心，只包含基本的核心操作系统功能。许多传统上包含在操作系统中的服务现在都是外部子系统：设备驱动程序、文件系统、虚拟内存管理器、开窗系统、安全服务
- 操作系统和进程间的关系：前者是计算机资源的管理者，后者归前者管理。

第5章 并发：互斥与同步

- 共享资源（Shared Resource）：一段资源内存可以由多个进程或线程同时使用
- 竞争条件（race condition）：当多个进程或线程访问共享资源时，最终结果取决于多个进程中指令的执行顺序。
- 临界区（critical section/region）： 指的是一个访问共用资源（例如：共用设备或是共用存储器）的程序片段，而这些共用资源又无法同时被多个线程访问的特性。当有线程进入临界区段时，其他线程或是进程必须等待（例如：bounded waiting 等待法），比如，一次只允许一个进程向打印机发送命令
- 硬件解决方案：关中断、TSL和Exchange指令
- 信号量（Semaphore，1965, Dijkstra）：操作系统提供给用户使用的一种机制，帮助用户进程协调使用资源，信号量是解决资源竞争的最有效途径。

- 操作有：wait（会让信号量-1），signal（会让信号量+1）



- 样例：比如记录资源的数量，等待资源的进程数，等待资源的进程阻塞队列在哪，当然最重要的是信号量还代表这个资源是互斥的。比如信号量S=3代表资源目前还有3个，没有进程阻塞；S=-2代表资源已经都被占用，且阻塞队列中等待资源的进程有2个。

例：若有一售票厅只能容纳300人，当少于300人时，可以进入；否则，需在外等候。若将每一个购票者作为一个进程，请用P（wait）、V（signal）操作编程，并写出信号量的初值。（强调：只有一个购票窗口，每次只能为一位购票者服务）

分析：题中有两类资源，售票厅和售票窗口，售票厅可以容纳300人，窗口只能服务1个人。用户到达后要想买到票，首先要进入售票厅，然后申请到窗口才行。因此用户需要获得两类资源才能成功购票。好了，我们定义两个信号量一个是S1，代表售票厅还能容纳的人数，一个是S2，代表窗口，它们的初始值一个是300，一个是1。程序如下：

```
解：semophore S1=330，S2=1；
   用户进程Pi {
            Wait（S1）
            进入大厅
            Wait（S2）
            窗口购票
           退出购票窗口
            Signal（S2）
             退出大厅
            Signal（S1）
}
```

- 生产者-消费者问题（Producer-Consumer Problem）：生产者负责生产资料并将其放进缓冲区中；消费者负责将资料取出缓存缓冲区中，相当于+1，-1；缓冲区只能由其中的一个访问

```
#define  N 100
semaphore mutex = 1;
semaphore empty = N;
semaphore full = 0;
```

```
void producer(void)
{
    int item;
    while(true){
        produce_item(&Item);
        semWait(&empty);
        semWait(&mutex);
        enter_item(item);
        semSignal(&mutex);
        semSignal(&full);
    }
}
```

```
void consumer(void)
{
    int item;
    while(true){
        semWait(&full);
        semWait(&mutex);
        remove_item(&item);
        semSignal(&mutex);
        semSignal(&empty)
        consume_item(item);
    }
}
```

- 原语（Primitive）：等待操作使信号量值递减；信号操作增加信号量值；P/V操作；下行/上行操作

```
struct semaphore {
    int count;
    queueType queue;
}

void semWait (semaphore s)
{
    s.count--;
    if (s.count < 0) {
        place this process in s.queue;
        block this process
    }
}
void semSignal (semaphore s)
{
    s.count++;
    if (s.count <= 0) {
        remove a process P from s.queue;
        place process P on ready list;
    }
}
```

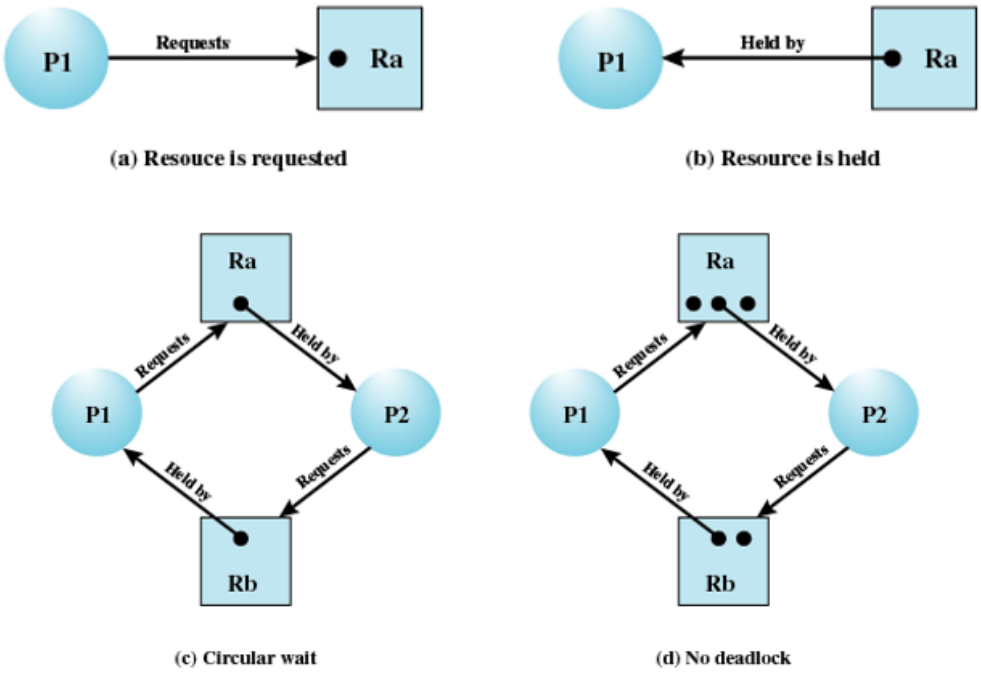
- 生产者-消费者问题中信号量解决方案中信号量的用途（同步、互斥）：用于记录生产者的+1，消费者的-1
- 管程（Monitor）：是由一个或多个程序、初始化序列、本地数据组成的软件模块
    - 其主要特征为：1.本地数据只能被管程内的进程访问，外部程序不能访问；2.进程可以调用它的一个程序进入管程；3.管程中只能有一个进程
- 消息传递（Message Passing）：1.实行互斥；2.交换信息；3.为消息传递提供了一对原语：发送（目的地、消息），接收（源、消息）
- 读者-写者问题：任意数量的读者可以读文件；一次仅有一个写者可以写文件；如果有写者在写文件，那么不允许读者读文件。

第6章 并发：死锁与饥饿

- 死锁的概念（Deadlock）： 一组竞争系统资源或互相通信的进程间相互的“永久”阻塞，没有有效的通用解决方案。
- 资源的类型可重用资源和可消耗资源（Reusable /Consumable）

- 可重用资源：一次只能由一个进程使用，结束以后将被释放的资源；
- 可消耗资源：可以创建（生产）和销毁（消耗），被使用后就不存在的资源。比如，I/O缓冲区中的中断、信号、消息和信息

- 资源分配图（Resource Allocation Diagram）：描述资源和进程系统状态的有向图



资源轨迹图

- 死锁的四个条件（）：1.互斥；2.占有且等待；3.非抢占；4.循环等待
- 死锁的四个处理策略：

1. 忽略（Ignorance）
2. 预防（Prevention）：（1）间接法，预防三个必要条件中的一个发生；（2）直接法，预防循环等待的发生
3. 避免（Avoidance）：
    1. 单种/多种资源银行家算法（Banker algorithm），即资源分配拒绝策略：一个系统有固定数目的进程和资源，任何时候一个进程可能分配到0个或多个资源，该策略确保系统中进程和资源总是处于安全状态。当进程请求一组资源时，假设同意该请求，则改变了系统的状态，然后确定其结果是否还处于安全状态。如果是，同意这个请求；如果不是，阻塞该进程直到同意该请求时系统仍是安全的。
    2. 安全状态（Safe State）：安全状态是指至少有一个进程执行序列不会导致死锁

3. 不安全状态（Unsafe State）：不安全的状态，但不一定导致死锁
4. 银行家算法的不足：缺乏实用价值，因为很少有进程在运行前就能知道其所需要的资源的最大值，而且往往进程数是不会一直不变的，同时有些可用资源在运行的过程中也可能会失效

4. 检测与破坏（Detection and Destroy）：选择一个进程进行销毁，释放其所占有的资源，将资源分配给其他进程

第7章 内存管理

- 存储体系（Storage Hierarchy）：1.内存（寄存器，缓存，主存）；2.外存（硬盘，CD）；3.线下存储（磁带）
- 地址的分类：逻辑地址（由程序产生的与段相关的偏移地址部分，又称绝对地址）、物理地址（实地址）、虚地址
- 内存管理的基本功能：重定位（relocation，把逻辑地址指向实际物理空间的过程）、保护、共享、逻辑管理、实际物理地址管理
- 内存分区（Memory Partitioning）：给进入主存的用户划分一块连续存储区域，把作业装入该连续存储区域，若有多个作业装入主存，则它们可以并发执行
- 固定分区（又称定长分区）：等大小/不等大小，前者容易产生内部碎片，后者产生外部碎片
    - 等大小：事先就将内存分成了若干个等大小的分区
    - 不等大小：事先将内存分成了若干个不等大小的分区
    - 由于作业可能太大，没有分区可以满足，或者作业太小，导致分区浪费，产生”内部碎片“
- 动态分区：当作业进入主存时，根据作业需求的大小和当前内存空间的使用情况来决定如何为该作业分配一个分区，由于分配以后内存的连续存储区域可能越来越少，导致产生”外部碎片“
- 内部、外部碎片/零头（Internal/External Fragmentation）–> compaction
    - 内部碎片指已经分配出去，却不能被利用的内存空间
    - 外部碎片指还未分配出去，但由于太小而不能分配给进程的内存空间。
- 放置/适配算法（Placement algorithms）
    1. 最佳适配：选择与请求大小最接近的块，整体表现最差，因为为进程找到了最小的块，所以剩下的碎片量最小，内存压缩必须经常进行
    2. 首次适配：从头开始扫描寻找第一个足够大的可用块。可能在内存的前端加载了许多进程，在尝试查找空闲块时必须对这些进程进行搜索

3. 下次适配：从最后一个位置扫描内存。更常见的情况是在内存的末尾分配一个内存块，在那里可以找到最大的块，最大的内存块被分成更小的块，为了在内存末尾获得一个大的块，需要进行压缩

- 分页（paging）：将内存划分为大小相等的固定块，称为帧（页框）；将进程划分为大小相同的块，称为页
- 分段（segmentation）：所有程序的所有段不必具有相同的长度，存在最大段长度，寻址由两部分组成一段号和偏移量，由于分段不相等，因此分段类似于动态分段。

第8章 虚拟内存管理

- 虚地址（Virtual Address）:程序访问存储器所用的逻辑地址称为虚拟地址。由页号和偏移量组成。

- 虚地址空间（Address Page Table）：
- 页（page）：进程中的数据的分块
- 页框（page frame）：内存的分块
- 页表（page table）：用于将虚拟地址转换为实际物理地址的表
- 页表表项（page table entry，PTE）的主要内容

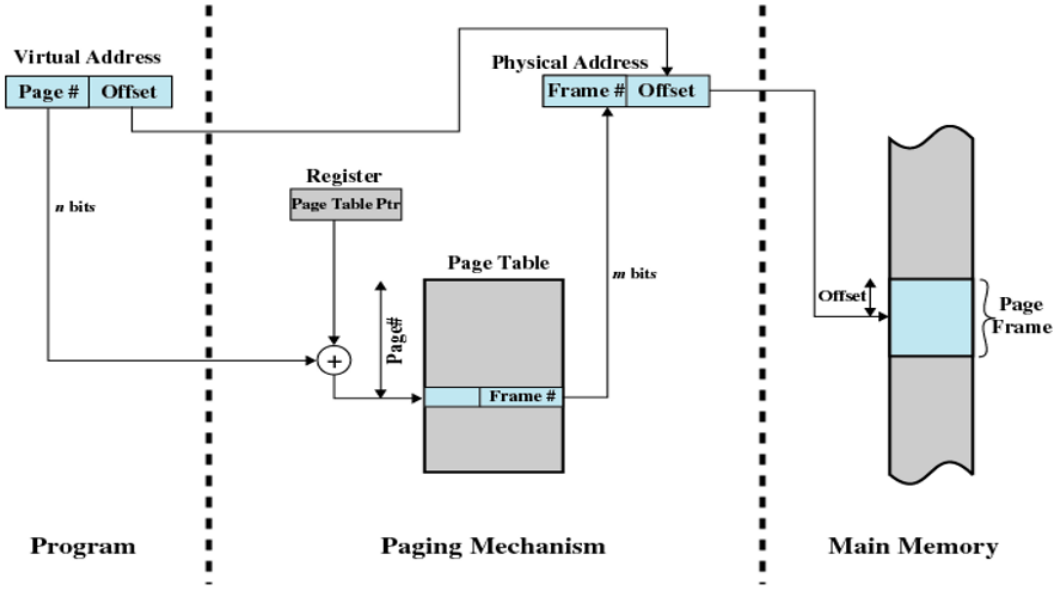
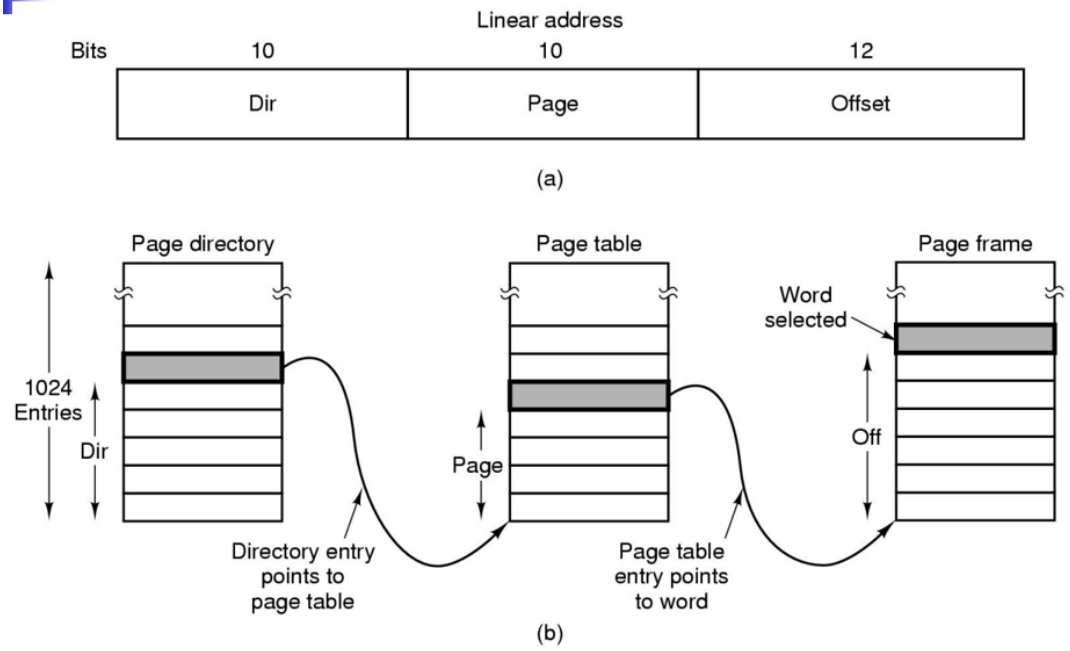


Figure 8.3 Address Translation in a Paging System

Two-Level Scheme



- 虚地址到物理地址的映射/转换（mapping/translation）：虚地址的偏移量+虚地址的页号对应的页表中的实页号
- MMU（Memory Management Unit）
- 虚拟存储器带来的问题：
  - 页表很大 –> 多级页表（multi-level page table）
    - 增加了一次对内存中页表的访问 –> TLB（Translation Lookaside Buffer）

- 地址空间更大 –> 逆向页表（Inverted page table）

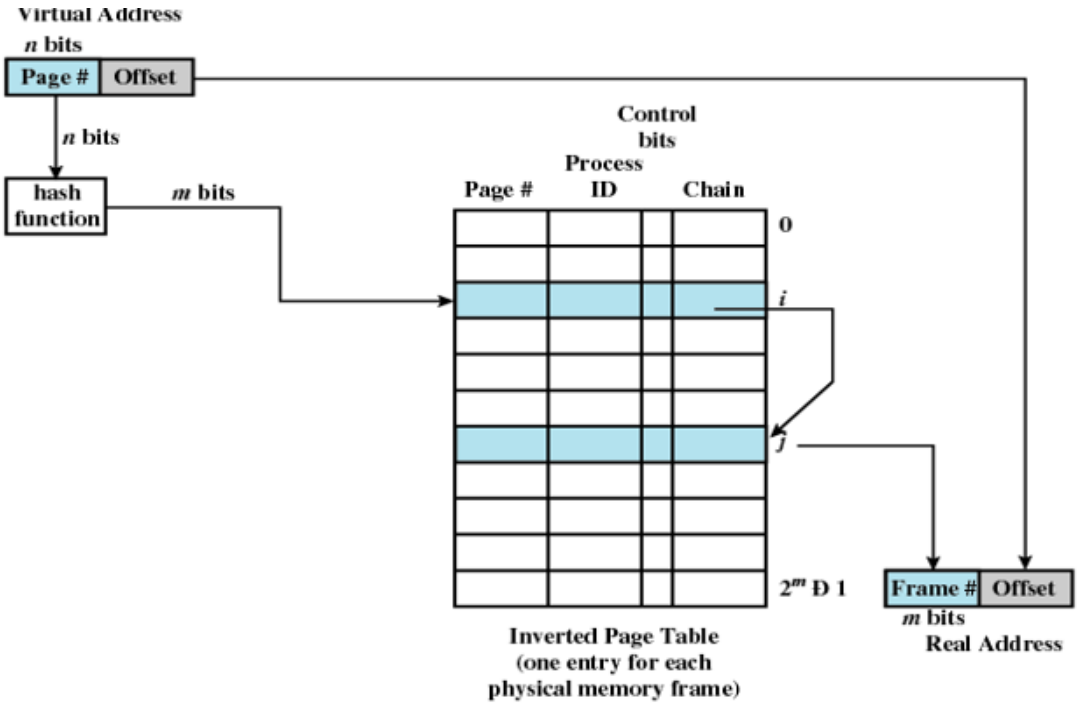


Figure 8.6 Inverted Page Table Structure

访问的局部性原理（Principle of Locality）:最近被访问的以及相邻空间上被访问的内存，很有可能会再次被访问

虚拟存储器的设计问题（Design Issues）

工作集（Working Set）

- 页面的大小（Page Size）:页越小，内部碎片越少，进程就需要更多的页面，每个进程数越多，意味着页表越大，意味着存储页表的空间越大

碎片（Fragmentation）

- 装入页的时机（按需装入和预先装入， Demand Paging and Prepaging）：仅当进程对页上的某个位置进行引用时，请求分页将页带入主存中

全局和局部（Global or Local Scope）

- 颠簸/抖动（Thrashing）：1.在需要某个进程的某个部分之前交换掉该部分；2.处理器花费大量时间交换数据块，而不是执行用户指令
- 页面替换算法
- Optimal、Least Recently Used、FIFO、Clock：
    1. LRU，最近最久未使用：替换主存中上次使用距当钱最远的页。根据局部性原理，这也是最近最不可能访问到的页
    2. CLOCK，时钟置换算法：附加位称为使用位，初始时每个附加位都为 1当某一页首次装入主存时，该帧的使用位设置为 1，然后指

针指向下一个使用位为 0若某一页已经在主存中，那么直接将对应的使用位设置为　1，然后指针指向下一个使用位为　0的地方当所有的使用位都为 1 时，我们将所有使用位都设置为 0 ，然后重新寻找下一个使用位为0 的地方

3. FIFO,优先队列算法：先进先出
4. OPT，最佳算法：接下来最晚被用到、或者不被用到的被替换

- 分段（Segmentation）：可能不相等，动态大小。逻辑地址空间是由一组段构成，地址制定了段名称和段内偏移。

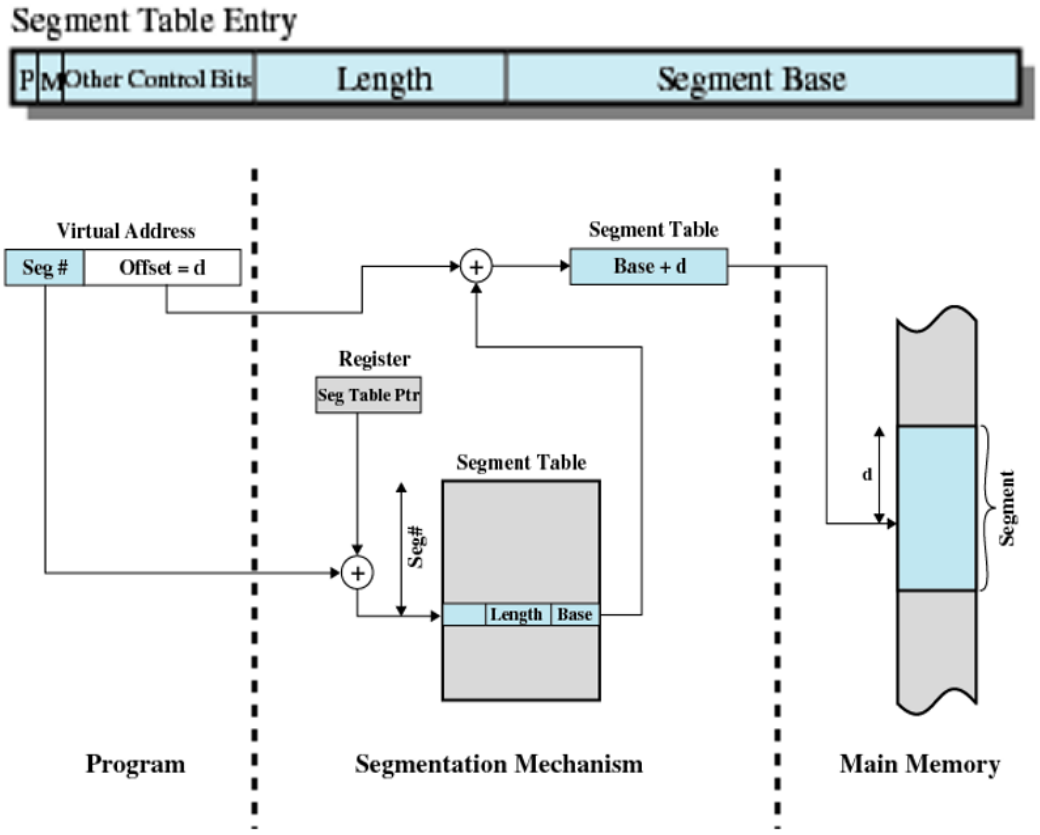


Figure 8.12　Address Translation in a Segmentation System

Combined Paging and Segmentation

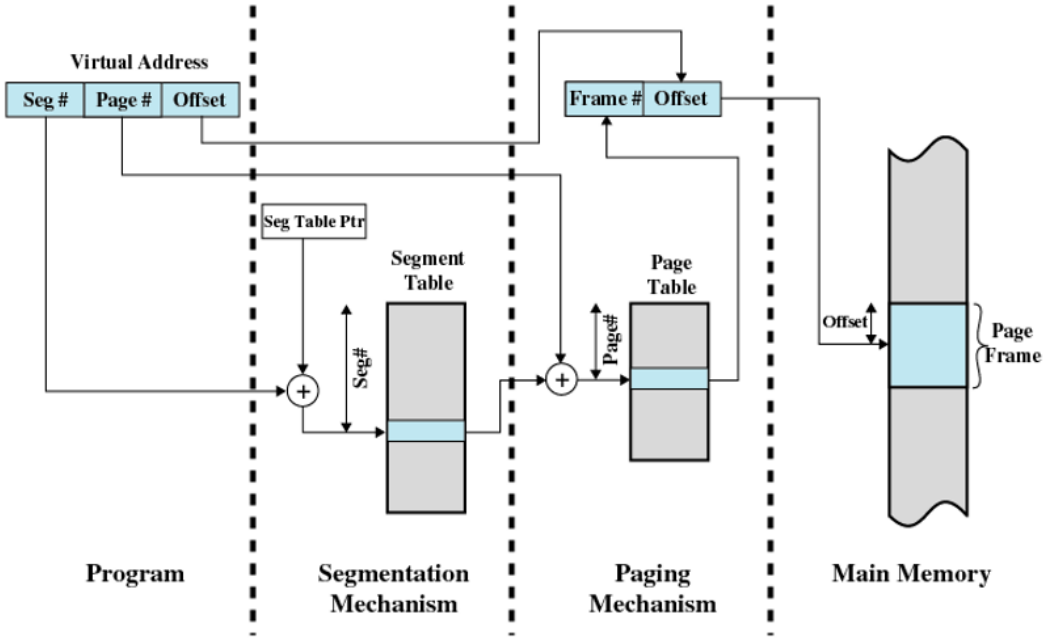


Figure 8.13 Address Translation in a Segmentation/Paging System

第9章 单处理器调度

- 调度类型：
  - 长程（long-term）：决定将哪个进程加入到进程池中执行
  - 中程（Medium-term）：决定将哪个进程加入主存中
  - 短程（Short-term）：决定哪个进程将会被处理器执行
  - I/O：决定I/O设备将满足哪个进程的I/O需求

调度准则（Scheduling Criteria）：响应时间（response time）：进程收到请求并相应的时间、吞吐量（throughput）：单位时间内完成的进程数、进程效率（processor efficiency）：程序运行快慢和所占内存大小的比值

Throughput， Response Time， Turnaround Time， Fairness

- 不同的调度算法
  1. FCFS，先到先服务
  2. SJF/SPF，短进程作业优先
  3. Round-robin，基于时间片轮转算法
  4. SRTF，最短剩余时间优先

5. Priority-based：通用操作系统的调度算法：基于优先级的分时反馈调度，调度算法的实现：队列

几种常用调度算法的比较

算法 / 比较项目	FCFS（先来先服务法）	RR（轮转法）	SJF（短作业优先）	SRTF（最短剩余时间优先）	HRRF（高响应比优先）	MFQ（多级反馈队列）
选择依据	max[w]	常量	min[s]	min[s-e]	max((w+s)/s)	见4.5.7节
调度方式	非抢占式	抢占式（按时间片）	非抢占式	抢占式（进程到达时）	非抢占式	抢占式（时间片）
吞吐量	不突出	如时间片太小，可变低	高	高	高	不突出
响应时间	可能很高	对于短进程提供良好的响应时间	对于短作业（进程）提供较好响应时间	提供较好响应时间	提供较好响应时间	不突出
开销	最小	低	可能高	可能高	可能高	可能高
对进程的作用	不利于短作业（进程）和I/O繁忙作业（进程）	公平对待	不利于长作业（进程）	不利于长进程	较好均衡	偏爱I/O繁忙型作业（进程）
“饥饿”问题	无	无	可能	可能	无	可能

第11章 I/O管理和磁盘调度

I/O设备的特点

- I/O设备的分类：用户可读（打印机）、机器可读、通信
- 块设备/字符设备（流设备）：
    1. 块设备：系统中能够随机访问固定大小数据片（chunks）的设备，比如硬盘
    2. 字符设备（流设备）：按照字符流的方式被有序访问，比如串口和键盘
    3. 两者区别：在对字符设备发出IO请求时，实际的硬件I/O一般就紧接着发生了；而块设备不然，他利用一块系统内存作为缓冲区，当用户对设备请求满足用的要求时，就返回请求的数据，如果不能，就调用请求函数来进行实际的I/O操作。因此，块设备主要是针对磁盘等慢速设备设计的，以免消耗过多的CPU时间来等待
- I/O软件的组织
    1. 编程I/O：进程正忙于等待操作完成
    2. 中断驱动I/O：发出I/O命令，处理器继续执行指令，完成后，I/O模块发送中断
    3. 直接存储器存取（DMA）：DMA模块控制主存储器和I/O设备之间的数据交换，处理器仅在整个数据块传输完毕后中断

- 磁盘调度（Disk arm scheduling）：从磁盘的存储空间中调取内容

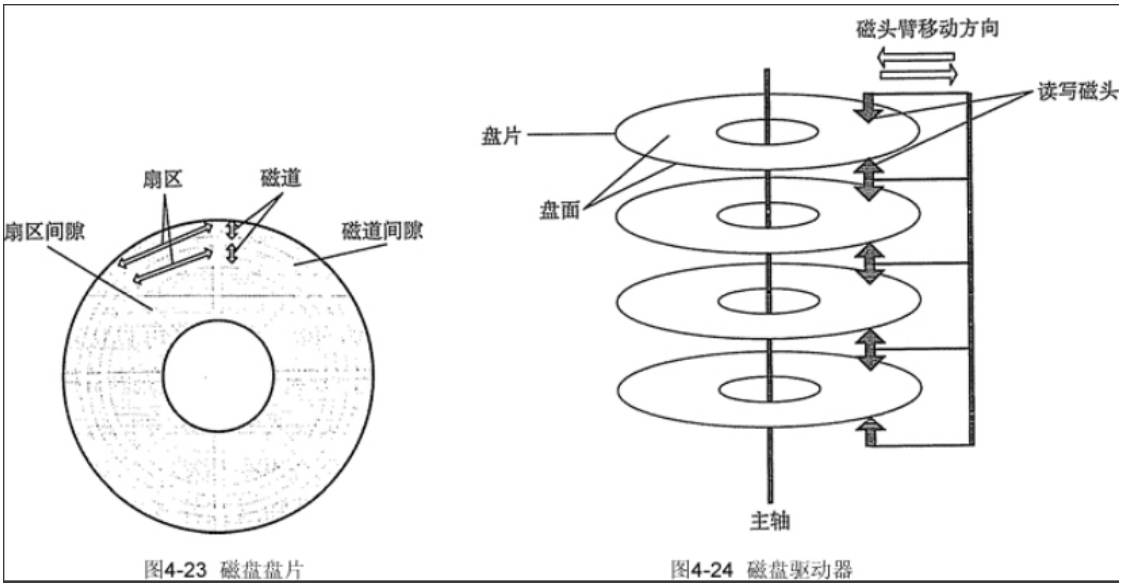
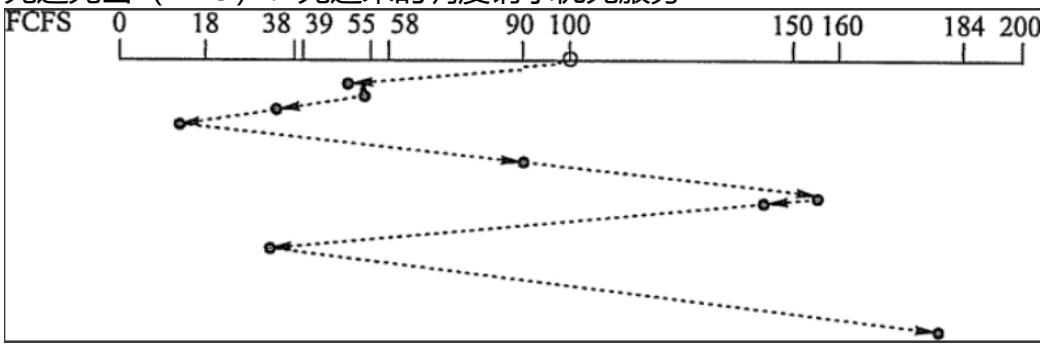


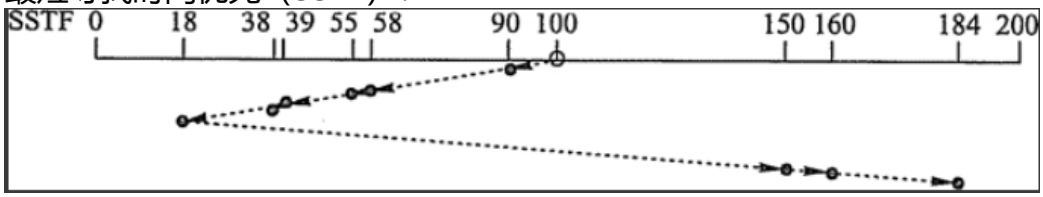
图4-23 磁盘盘片　　图4-24 磁盘驱动器

- 磁盘I/O的过程（Seek time、Rotational Delay、Data Transfer Time）
  - 寻道时间：将头部定位到所需轨迹所需的时间
  - 旋转延迟：扇区到达头部所需的时间
  - 访问时间：寻道时间和旋转延迟之和，读写所需的时间

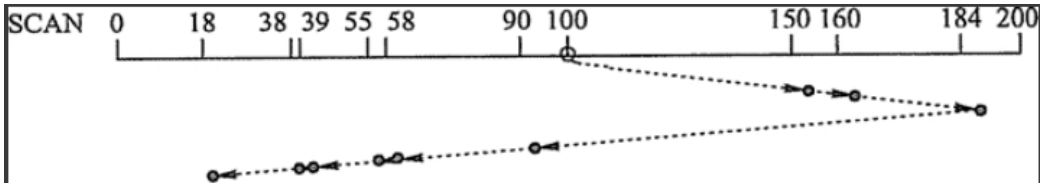
1. 先进先出（FIFO）：先进来的调度请求优先服务



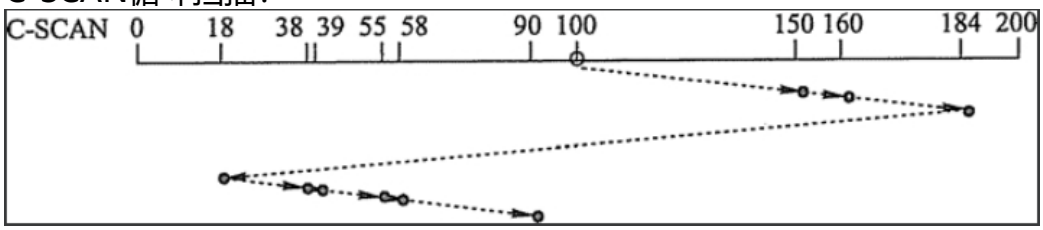
2. 最短寻找时间优先（SSTF）：



3. SCAN扫描：



4. C-SCAN循环扫描：



- RAD，独立磁盘冗余阵列：冗余磁盘容量用于存储奇偶校验信息

第12章 文件管理

- 文件、文件系统：
    - 文件是具有文件名的一组相关元素的集合，在文件系统中是一个最大的数据单位；
    - 文件系统：操作系统中负责管理和存储文件信息的软件机构。
- 文件的分类
    - 正规文件（Regular file）、特殊文件（Device Special File）、目录（Directory）、管道（Pipe：Anonymous Pipe/Named Pipe）等
- 文件的常见操作：none、knowledge、创建文件、删除文件、读文件、写文件、截断文件（将文件删除重新创建）、设置文件的读/写位置
- 树形结构

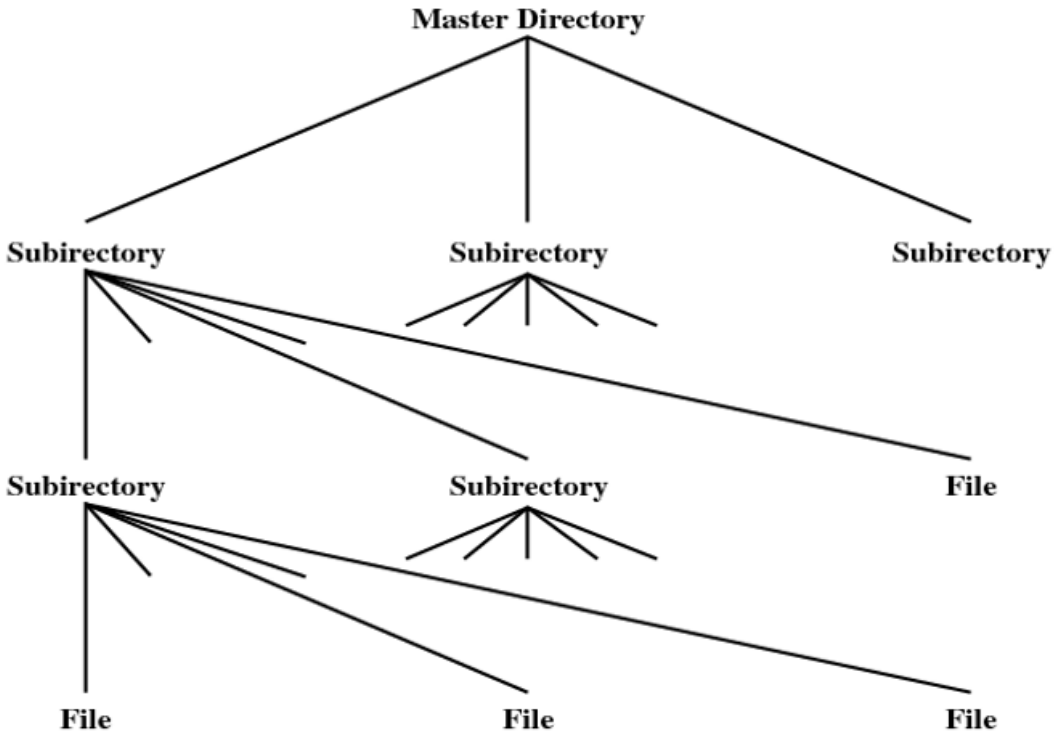


Figure 12.4 Tree-Structured Directory

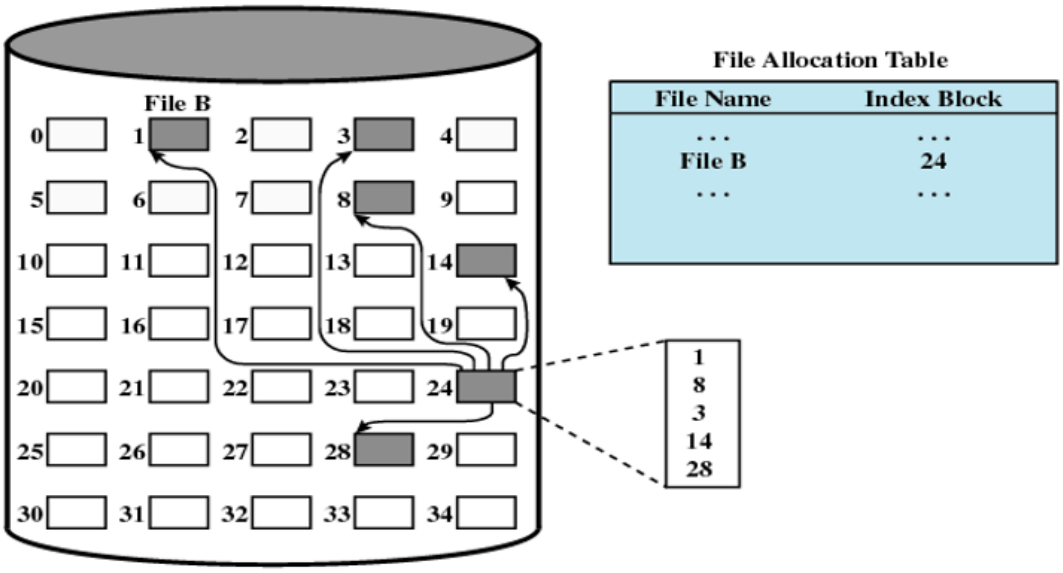
- 路径及分类

  根目录与当前目录（Root Directory and Current Directory/Working Directory）

  绝对路径与相对路径（Absolute Path and Relative Path）：绝对路径是：以根目录为起点的路径；相对路径是：从当前目录为起点出发的路径

- 文件结构（字节流结构）：
  - 组成结构：数据项（基本数据项，组合数据项）组成记录，记录组成文件。
  - 逻辑结构：顺序、索引、索引顺序
- 磁盘空间的分配方法及优缺点：
  1. 预先分配（preallocation）：创建时需要文件的最大大小、难以估计可靠的文件可能最大大小、倾向于高估文件大小，以免耗尽空间
  2. 连续分配（contiguous allocation）：单个块集直接分配给文件，文件分配表只有文件的起始块和长度，容易产生外部碎片，需要压实
  3. 链式分配（chained allocation）：按单个区块分配，每个分块都包含指向链中指向下一个块的指针，文件分配表只包含起始块和长度，没有外部碎片，但是不适应地方性原则

4. 索引分配（index allocation）：文件分配表包含每个文件的单独一级索引，索引对分配给文件的每个部分都有一个条目



磁盘块地址/编号

- 空闲空间的记录：利用二进制的一位来表示磁盘中的一个盘块的使用情况。当其值为“0”时，表示对应的盘块空闲；为“1”时，表示已经分配(或者把"0"作为盘块已分配的标记，把“1”作为空闲标志)。磁盘上的所有盘块都有一个二进制位与之对应，这样，由所有盘块所对应的位构成一个集合，称为位示图。通常可用m*n个位数来构成m行n列的位示图，并使m*n等于磁盘的总块数。（位视图法）

基于i-node的Unix文件系统

文件的同时存取

文件的访问权限

i-node的结构：索引节点，包含了一个特定文件的关键信息

- 基于i-node的路径分析
  - 一级间接地址

disk block size is 1 Kbytes:一块1KB
4 bytes=4*8=32 位，所以一级间接是 2^32 块，二级则平方一次即可。
文件的容量如下表：

级	块数	字节数
直接	10	10KB
一级间接	2^32=256	256KB
二级间接	256*256=64k	64MB

  - 二级间接地址