第 2 章 操作系统概述

- **中断的作用：**
  （1） 有效运用 CPU，提升 CPU 运行速度
  （2） 实现多道程序

中断定义：中断是指计算机的处理器用来处理 外来请求或内部错误的一种机制，该机制软硬件结合，使得计算机的处理机能够暂停当前指令系列的执行而转向请求指令系列的执行。

处理中断的方法：

（1） 当正在处理一个中断时，禁止再发生中断

（2） 定义中断优先级

1、 **操作系统定义**：控制应用程序执行的程序，并充当应用程序和计算机硬件之间的接口。

操作系统目标：方便、高效、扩展的能力

操作系统发展历史：

（1） 串行处理：20 世界 40 年代后期到 50 年代中期

主要问题：

a. 调度：使用硬拷贝的签约表预订机器时间，导致空置或强制停止

b. 准备时间：若准备期间发生了错误，只能重新开始。导致程序运行前的准备花费大量时间。

（2） 简单批处理系统（见下一条）

（3） 多道批处理（见下一条）

UNIX：

（1） 背景：MIT 的 MAC 项目开发了第一个 CTSS（兼容分时系统）和 Multics

（2） 里程碑：a.将 UNIX 从 PDP-7 移到 PDP-11；b.用 C 程序设计语言重写 UNIX

Linux：

（1） 背景：针对 IBM PC 结构的一个 UNIX 变体

2、 简单**批处理**

（1） 特点：引入监视器

a. 监视器常驻

b. 监视器需主动释放，获取控制权

（2） 硬件功能支持：

a. 内存保护

b. 定时器

c. 特权指令

d. 中断

（3） 缺点：

a. 一部分主程序交付给监控程序

b. 监控程序消耗了一部分机器时间

（4） 模式：

a. 用户模式：有的内存区域受到保护，特权指令不允许执行。

b. 系统模式（内核模式）：可以执行特权指令，受保护内存区域可访问。

多道批处理

（1） 思路：内存中尽量多的加载程序，提高利用率

（2） 所需硬件支持：DMA（直接存储器访问）、资源调度、内存管理

（4）

3、 进程概念：

（1） 一个正在执行的程序

（2） 计算机中正在运行的程序的一个实例

（3） 可分配给处理器并由处理器执行的一个实体

（4） 由单一的顺序的执行线索、一个当前状态和一组相关的系统资源所描述的活动单元

进程的组成：

（1） 一段可执行程序

（2） 程序所需的相关数据（变量、工作空间、缓冲区等）

（3） 程序的执行上下文

4、 模式

非特权模式：通常称为用户模式，用户程序通常在该模式下运行

特权模式：可称为系统模式、控制模式或内核模式，内核模式（内核=操作系统的内核，包含重要系统功能）

引入两种模式的原因：保护操作系统和重要的操作系统数据表（如进程控制块）不受用户程序的干涉。

5、 **挂起态**

概念：主存中进程一部分或全部被暂时调离出内存，转移到磁盘。当主存中的所有进程都处于阻塞态时，操作系统可以把其中一个进程置于挂起态并转移到磁盘，主存中释放的空间可被调入的另一个进程使用。

引入原因：使用“交换”操作，提高处理器的运行效率，减少空闲状态。

挂起态进程特点：

（1） 进程不能立即执行

（2） 进程可能是或不是正在等待一个时间

（3） 为阻止进程执行，可以通过代理把这个进程置于挂起态，代理可以是进程自己，也可以是父进程或操作系统。

（4） 除非代理显式地命令系统进行状态转换，否则进程无法从这个状态中转移。

6、 **引入进程的原因：让程序能够并发执行**

引入线程的原因：区分进程的资源所有权和调度/执行

7、 进程和线程的区别：

（1） 线程：可分派的工作单元。它包括处理器上下文和栈中自己的数据区域。线程顺序执行，并且是可中断的，这样处理器就可以转到另一个线程

（2） 进程：一个或多个线程和相关系统资源的集合。着紧密对应于一个正在执行的程序的概念，通过把一个应用程序分解成多个线程，程序员可以在很大程度上控制应用程序的模块性和应用程序相关事件的时间安排。

8、 **进程切换的步骤：**

（1） 保存处理器上下文，包括程序计数器和其他寄存器。

（2） 更新当前处于运行态的进程的进程控制块，包括把进程的状态改变为另一状态（就绪态、阻塞态、就绪/挂起态或退出态）。还必须更新其他相关域，包括离开运行态的原因和审计信息。

（3） 把进程的进程控制块移到相应的队列（就绪、在事件 i 处阻塞，就绪/挂起）。
（4） 选择另一个进程执行。
（5） 更新所选择进程的进程控制块，包括把进程的状态变为运行态。
（6） 更新内存管理的数据结构。
（7） 回复处理器在被选择的进程最近一次切换出运行态时的上下文，这可以通过载入程序计数器和其他寄存器以前的值来实现。

9、 模式切换和进程切换的关系：发生模式切换可以不改变正处于运行态的进程状态，保存上下文和以后恢复上下文只需要很少的开销。进程状态涉及到状态变化，比模式切换需要做更多的工作。

10、 5 状态模型

五个状态：

（1） 运行态：该进程正在执行
（2） 就绪态：进程做好了准备，只要有机会就开始执行
（3） 阻塞态：进程在某些事件发生前不能执行
（4） 新建态：刚刚创建的进程，OS 还没有将它加入到可执行进程组中
（5） 退出态：OS 从可执行进程组中释放的进程，或者是因为它自身停止了，或者是因为某种原因被取消。

导致进程创建的原因：

（1） 新的批处理作业：通常位于磁带或磁盘中的批处理作业控制流被提供给操作系统。当操作系统准备接纳新工作时，他将读取下一个作业控制命令。
（2） 交互登陆：终端用户登录到系统
（3） 操作系统因为提供一项服务而创建：操作系统可以创建一个进程，代表用户程序执行一个功能，是用户无需等待。
（4） 由现有的进程派生：基于模块化的考虑，或者为了开发并行性，用户程序可以指示创建多个进程。

导致进程终止的原因：

（1） 正常完成：进程自行执行一个操作系统服务调用，表示它已经结束运行
（2） 超过时限：进程运行时间超过规定的时限。
（3） 无可用内存：系统无法满足系统需要的内存空间
（4） 越界：进程试图访问不允许访问的内存单元。
（5） 保护错误：进程试图使用不允许使用的资源或文件，或者试图以一种不正确的方式使用，如：往制度文件中写。
（6） 算术错误：进程试图进行被禁止的计算，如除以零或者存储大于硬件可以接纳的数字
（7） 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。
（8） I/O 失败：在输入或输出期间发生错误，如找不到文件、在超出规定的最多努力次数后仍然读写失败。
（9） 无效指令：进程试图执行一个不存在的指令。
（10） 特权指令：进程试图使用为操作系统保留的指令。
（11） 数据误用：错误类型或未初始化的一块数据。
（12） 操作员或操作系统干涉：由于某些原因，操作员或操作系统终止进程（例如，如果存在死锁）。

（13） 父进程终止：当一个父进程终止时，操作系统可自动终止该进程的所有后代进程。

（14） 父进程请求：父进程通常具有终止其任何后代进程的权利。

11、 7 状态模型

增加的 2 个状态：

（1） 阻塞/挂起态：进程在辅存中并等待一个事件。

（2） 就绪/挂起态：进程在辅存中，但是只要被载入主存就可以执行。

12、 微内核

微内核概念：微内核是一个小型的操作系统核心，它为模块化扩展提供了基础。

微内核结构优点：

（1） 一致接口

（2） 可扩展性

（3） 灵活性

（4） 可移植性

（5） 可靠性

（6） 分布式系统支持

（7） 对面向对象操作系统的支持。

13、 单一内核：一个大内核作为一个进程实现的，所有元素都共享相同的地址空间

微内核性能提升：

（1） 把一些关键的服务程序和驱动程序重新放回操作系统，但将增大微内核。

（2） 减少模式切换（关键）

微内核功能：

（1） 低级存储器管理

（2） 进程间的通信

（3） I/0 和中断管理

14、 基于用户/内核的线程优缺点和对比

用户级线程：通过进程地址空间中的线程库实现，他们对操作系统是不可见的。用户级线程是应用程序的接口。

使用用户级线程代替内核级线程的优点：

（1） 由于所有线程管理数据结构都在一个进程的用户地址空间中，线程切换不需要内核模式特权，因此，**进程不需要为了线程管理而切换到内核模式**，者节省了在者两种模式间进行切换（从用户模式到内核模式；从内核模式返回到用户模式）的开销；

（2） **调用可以是应用程序专用的**。一个应用程序可能倾向于简单的轮询调度算法，而另一个一个用程序可能倾向于基于优先级的调度算法。调度算法可以去适应应用程序，而不会扰乱底层操作系统的调度器；

（3） **用户级线程可以在任何操作系统中运行**，不需要对底层内核进行修改以支持用户级线程。线程库是一组供所有应用程序共享的应用级软件包。

用户级线程相对于内核级线程的两个明显缺点：

（1） **在典型的操作系统中，许多系统调用都会引起阻塞。**因此，当用户级线程执

行一个系统调用时，不仅这个线程会被阻塞，进程中的所有线程都会被阻塞；

（2） **在纯粹的用户级线程策略中，一个多线程应用程序不能利用多处理技术。**内核一次只能一个进程分配给一个处理器，因此一次进程中只有一个线程可以执行。实际上，在一个进程内有应用级的多道程序。虽然多道程序会使得应用程序的速度明显提高，但是同时执行部分代码更会使得应用程序受益。

内核级线程：这是可以调度和分派到系统处理器上运行的基本实体。

使用内核级线程的优点（克服了用户级线程方法的两个基本缺陷）：

（1） 内核可以同时把同一个进程中的多个线程调度到多个处理器中；

（2） 如果进程中的一个线程被阻塞，内核可以调度同一个进程中的另一个线程。

（3） 内核级线程的另一个优点是内核例程自身也可以使用多线程。

相对与用户级线程，内核级线程方法的的主要缺点：同一个进程在把控制从一个线程时传送到另一个线程时需要到内核的模式切换。

15、 互斥

互斥的硬件支持：

（1） 中断禁用

部分情况下禁用中断不能保证互斥（5.2.1 例子）

（2） 专用机器指令

机器指令方法特点：

优点：

A. 适用于在单处理器或共享内存的多处理器上任何数目的进程

B. 非常简单且易于证明

C. 可用于支持多个临界区，每个临界区可以用他自己的变量定义。

缺点：

A. 使用了忙等待

B. 可能饥饿

C. 可能死锁

互斥的要求：

（1） 必须强制实施互斥：在具有关于相同资源或共享对象的临界区的所有进程中，一次只允许一个进程进入临界区

（2） 一个在非临界区挺值得进程必须不干涉其他进程。

（3） 决不允许出现一个需要访问临界区的进程被无限延迟的情况，既不会死锁或饥饿。

（4） 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刻进入。

（5） 对相关进程的速度和处理器的书目没有任何要求和限制

（6） 一个进程驻留在临界区中的时间必须是有限的。

16、 一堆概念：

临界区：是一段代码，在这段代码中进程将访问共享资源，当另外一个进程已经在这段代码中运行时，这个进程就不能在这段代码中进行。

死锁：两个或两个以上的进程因其中的每个进程都在等待其他进程做完某些事情而不能继续执行的情形。

活锁：两个或两个以上的进程为了响应其他进程中的变化而继续改变自己状态但不做有用的工作的情形。

互斥：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共享资源的情形

竞争条件：多个进程或者线程在读写一个共享数据时结果依赖于它们执行的相对时间的情形。

饥饿：一个可运行的进程尽管能继续执行，但被调度器无限期的忽视，而不能被调度执行的情况。

17、 进程和线程的管理

（1） 多道程序设计技术：管理单处理器系统中的多个进程

（2） 多处理技术：管理多处理器系统中的多个进程

（3） 分布式处理技术：管理多台分布式计算机系统中多个进程的执行。

18、 同步

概念：两个或多个进程在某些条件下协同工作的情况

同步和互斥的关系：为实现互斥，进程间需要同步。

19、 条件竞争

条件竞争发生在当多个进程或者线程在读写数据时，其最终的结果依赖于多个进程的指令执行顺序。

20、 信号量

信号量定义：一个可以发送信号的特殊变量

信号量分类 A：

（1） 二元信号量：值只能是 0 或 1

（2） 非二元信号量：也称为计数信号量或一般信号量。

信号量分类 B：

（1） 强信号量：有策略规定进程从队列中移除的顺序的信号量

（2） 弱信号量：没有策略规定进程从队列中移除的顺序的信号量。

生产者/消费者问题：

任何时候只有一个代理（生产者或消费者）可以访问缓冲区。

哲学家问题：

解决方案（网络）：

（1）、使用数组来跟踪一个哲学家的的状态：吃饭，思考或是试图拿起筷子，规定一个哲学家只有在两个邻居都不再进餐时才允许转移到进餐状态。这种方法可以使系统获得最大的并行度，即最多允许两个哲学家同时进餐。

（2）、给每个哲学家编号，规定奇数号的哲学家先拿他的左筷子，然后再去拿他的右筷子；而偶数号的哲学家则相反。这样总可以保证至少有一个哲学家可以进餐。

一般都使用第二种方法，高效而简单。

21、 资源

资源分类：

（1） 可重用资源

（2） 可消费资源

22、 死锁

必要条件：

（1） 互斥。一次只有一个进程可以使用一个资源。其他进程不能访问已分配给其他进程的资源

（2） 占有且等待。当一个进程在等待分配得到其他资源时，其继续占有已分配的到的资源。

（3） 非抢占。不能强行抢占进程中已占有的资源。

充分条件：

（4） 循环等待。存在一个封闭的进程链，使得每个资源至少占有此链中下一个进程所需要的一个资源。

解决方案：

（1） 采用某种策略来消除条件 1-4 中的一个条件来防止死锁。

（2） 基于资源分配的当前状态做动态选择来避免死锁。

（3） 试图检测死锁（满足 1-4 条件）的存在并且试图从死锁中恢复出来。

23、 死锁预防

死锁预防策略概念：试图设计一种系统来排除发生死锁的可能性。

死锁预防方法：

（1） 间接的死锁预防方法：防止前面列出的三个必要条件中任何一个的发生。

（2） 直接的死锁预防方法：防止循环等待的发生。

死锁预防的技术问题：

（1） 互斥：一般来说不可禁止

（2） 占有且等待：为预防可以要求进程一次性地请求所有需要的资源。

（3） 非抢占

a. 要求进程一次性的请求所有需要的资源，并且阻塞这个进程直到所有请求都满足。

b. 分配给一个进程的资源可能有相当长的一段时间处于不可用状态，且在此期间，他们不能被其他进程使用。

（4） 循环等待：通过定义资源类型的线性顺序来预防。

死锁避免与死锁预防不同点：允许三个必要条件，但通过明智选择确保不会到达死锁点，因此允许更多的并发。

死锁避免的两种方法：

（1） 进程启动拒绝：如果一个进程的请求会导致死锁，则不启动此进程。

（2） 资源分配拒绝：如果一个进程增加资源的请求会导致死锁，则不允许此分配。

恢复（复杂度递增）

（1） 取消所有的死锁进程。--------最常采用的方法

（2） 把每个死锁进程回滚到前面定义的某些检查点，并且重新启动所有进程。

（3） 连续取消死锁进程知道不再存在死锁。

（4） 连续抢占资源直到不再存在死锁。

对于 3、4，选择原则可采用下面一种：

（1） 目前为止消耗的处理器时间最少。

（2） 目前为止产生的输出最少
（3） 预计剩下的时间最长
（4） 目前为止分配的资源总量最少
（5） 优先级最低。

第 7 章 内存管理的需求

了解

- 内存管理的需求（5 个）重定位、保护、共享、逻辑组织、物理组织
- 内存管理：细分内存支持多道程序（Subdividing memory to accommodate multiple processes），内存需求管理已确保有适当数目的就绪进程使用处理器（Memory needs to be allocated to ensure a reasonable supply of ready processes to consume available processor time）

理解

- 重定位（Relocation）

  概念：把程序的逻辑地址空间变为物理地址空间的过程，是实现多高程序在内存中同时循行的基础

  实现方式：基址寄存器（base register）的值+相对地址=物理地址

  物理地址与界限寄存器（bounds register）的值相比较，判断是否越界，若越界在生成一个中断，每个进程映像根据基址地址和界限寄存器的内容被隔离开，避免越权访问。
- 覆盖：允许不同模块分配到内存的同一块区域
- 逻辑地址，相关地址，物理地址的概念和关系

  逻辑地址　　与当前数据在内存中的分配无关的访问地址

  相关地址　　表示与基址地址之间的一个偏差。逻辑地址的一个特例，是相对于

某些已知点的存储单元

物理地址　　在内存中的实际地址

- 掌握碎片的概念，清楚区分内部、外部碎片

  内部碎片　　由于被装入数据小于分许大小，从而导致分区内部有空间浪费，这种现象被称为内部碎片

  外部碎片　　由于内存中出现很多小的空洞，随时间的推移，内存中产生了越来越多的碎片，内存利用率随之下降的现象被称为外部碎片

- 什么是压缩

  克服外部碎片，OS 不断移动进程，使进程占用的空间连续，并且所有空闲空间连城一片

- 清楚地指出分区，分页分段的机制与差异特点

技术	说明	优势	劣势
固定分区	在系统生成阶段，内存被划分为许多静态分区，进程可以被装入大于或等于自身大小的分区中	实现简单，只需要极少的系统开销	由于有内部碎片，对内存的使用不充分；活动进程的最大数目固定
动态分区	分区是动态创建的，因而每个进程	没有内部碎片，可以更充分地使用内	产生外部碎片，因此需要压缩外部碎

	可以被装到与自身大小正好相等的分区中	存	片，使得处理器利用率低
简单分段	进程被划分成许多端，段的长度不一定相等。要装入一个进程，则把进程包含的所有段装到内存中不一定连续的动态分区中	没有内部碎片相较于动态分区，提高了内存利用率，减少了开销	存在外部碎片
简单分页	内存被划分成许多大小相等的页框，每个进程被划分成许多与页框大小相等的页，要装入一个进程，则把进程包含的所有页装到内存中不一定连续的页框中	没有外部碎片	有少量的内部碎片

- 能够根据逻辑地址计算物理地址，包括分页机制和分段机制

页：逻辑地址=页号 偏移量　物理地址=页框号 偏移量

段：逻辑地址=段号 偏移量　物理地址=基地址+偏移量

第 8 章 虚拟内存

- 了解：分页，分段以及段页式的计算机制
- 虚存：在存储分配机制中，尽管备用内存是主存的一部分，他也可以被寻址，程序引用内存使用的地址与内存系统用于识别物理存储站点的地址是不同的。程序生成的地址会自动转换成机器地址。虚拟内存大小受计算机系统寻址机制和可以用的备用内存量的限制，而不受内存位置实际数量的限制
- 产生缺页中断时操作系统与硬件对应的处理机制

  将进程置于阻塞态，发出磁盘读取请求，调度另一个进程，载入后重置为就绪态
- 在虚拟存储中用到的页表，段表

  页表：控制域 页号 页框号

  段表：控制域 长度 段基址
- 什么是抖动

  马上需要的进程被唤出，造成处理器用于交换的时间大于执行用户指令的时间
- 什么是反向页表

  虚拟地址的页号部分使用一个简单的散列函数映射到散列表中，散列表中包含一个指向倒排表的指针。倒排表中有也表象。页表大小只和实存有关。（页号 进程标识符 控制位 链表）
- TLB（转换缓冲区/转移后备缓冲区/旁视缓冲器）与 Cache 功能类似，包含最近使用过的页表项。
- 分页地址的翻译过程，看教材流程图（P246）
- 采用分级页表的优缺点：相较于一级页表效率更高，但是页表大小与虚拟空间大小依然成正比

- 页面的置换算法（教材的图）

  最佳（OPT），最近最少被访问（LRU），先进先出（FIFO），时钟（Clock）--有很多优化算法

第 9 章 单处理器调度

- 了解：调度的标准

  处理器分配，响应时间，吞吐率，处理器效率等

- 理解：调度的三种类型以及特点，产生场景

长程调度	决定加入待执行的进程池中	新建态——就绪/就绪挂起
中程调度	决定加入部分或全部在内存中的进程集合中	就绪挂起——就绪，阻塞挂起——阻塞/就绪
短程调度	决定哪一个可运行的进程将被处理器执行	就绪——运行
I/O 调度	决定哪一个挂起的 I/O 请求将被 I/O 设备处理	

- 与调度标准有关的概念，到达时间，服务时间等（P284 页表格）

  到达时间：进程到达系统的时间

  服务时间：处理器执行进程的时间

  周转时间：在这一项系统中花费的总时间

  归一化周转时间=周转时间/服务时间

- 抢占模式，非抢占模式，优先级的概念

  非抢占式：一旦进程进入运行状态，他就不断执行直到终止，或是因为等待 I/O，或者

因而请求某些操作系统服务而阻塞自己。

抢占模式：当前运行的进程可能被操作系统中断，并转移到就绪态。

- 短程调度算法，画甘特图（gantt charts）计算出他的周转时间和归一化周转时间（P287）

第 11 章 I/O 管理和磁盘调度

理解：

- 三种 I/O 实现技术

程序控制 I/O：处理器代表一个进程给 I/O 模块发送一个 I/O 命令；该进程的进入忙等待，直到操作完成才可以继续执行。

中断驱动 I/O：处理器代表 I/O 发出指令，处理器执行其他指令，I/O 完成后产生中断

直接存储器存储 DMA：一个 DMA 模块控制内存和 I/O 模块之间的数据交换。为传输一块数据，处理器请求 DMA 模块，并且只有当整个数据块传送结束后，他才被中断。

- I/O 设备分类

第一种分类方式 人可读：适用于同计算机用户之间的交互，打印机，键盘，鼠标

机器可读：使用与电子设备通信，如 USB 密钥，传感器

通信：适用于与远程设备通信，例如调制解调器

第二种分类方式 面向块：设备将信息保存在块中，块的大小通常是固定的，传送过程中一次传送一块，如磁盘和 USB 智能卡

面向流：以字节流的方式输入输出数据，没有块结构。终端，打印机，通信端口，鼠标和其他指示设备及其他大多数的非辅存设备都属于面向流的设备。

- Buffer 的概念及引入原因

  I/O 完成前必须等待，某些页必须驻留主存

  概念：在输入请求发出前就开始执行输入传送，并且在输出请求发出一段时间后才开始执行输出传送，这项技术被称为缓冲。（预输入，缓输出）

- 磁盘 I/O 操作的五种延迟

寻道时间	磁头定位到磁道所需的时间
旋转延迟	磁头到达扇区开始的位置的时间
存取时间	到达读或写位置需要的时间，寻道+旋转
传输时间	磁头通过相应的扇区读或写的时间
排队延迟	当一个进程发出一个 I/O 请求时，他必须在队列中等待该设备可用

掌握:

- 寻道时间：一般用平均寻道时间表示

  旋转延迟=磁盘转半圈的时间（计算时）=1/2r

  传输延迟=b/rN

  R 是旋转速度（转/秒），N 是一个磁道中的字节数，b 表示要传输的字节数

- 磁盘调度的 8 种算法，画表，计算平均寻道长度（P348）

第 12 章 文件管理

- 文件系统由系统实用程序组成，他们可以作为特权的应用程序来运行。文件作为其输入输出。

- 文件系统的功能：创建，删除，打开，关闭，读，写
- 文件系统的目标

  满足数据管理的要求和用户的需求

  最大限度的保证文件中的数据有效

  优化性能

  支持各类存储设备

  减少或消除丢失或破坏数据的可能性

  标准的 I/O 接口

  为多个用户提供 I/O 支持

理解：

- 什么是文件分配表

  为了跟踪分配给文件的分区而使用的表，一般包括文件名，起始块和长度，在索引分配法中还包括索引块等

- 列举出文件四种组成元素并解释

  域　　是基本的数据单元，一个域包含一个值

  记录　　一组相关域的集合，可以视为应用程序的一个单元

  文件　　一组相似记录的集合，被用户和应用程序视为一个实体，并可以通过名字访问，文件有唯一的文件名。访问控制通常在文件级实施

  数据库　　一组相关数据的集合，数据元素之间存在明确的关系，并且可供不同的应用程序使用

- 列举并解释文件的五种逻辑组织

  设备驱动　　程序直接与外围设备通信。设备驱动程序负责启动该

	设备上的 I/O 操作，处理 I/O 请求完成。
基本文件系统，物理 I/O 层	计算机系统与外部环境的基本接口，他关注的是块在辅助存储和内存缓冲区中的位置。
基本 I/O 管理程序	程序负责所有文件的初始和终止。需要一定的控制结构来维护设备的输入/输出，调度和文件状态。根据所选择的文件来选择执行文件的 I/O 设备，为游湖性能，还参与调度对磁盘和磁带的访问。I/O 缓冲区的指定和辅助存储的分配，也在这一层实现，它是 OS 的一部分
逻辑 I/O	使用户和应用程序能够访问到记录。提供一种通用的记录 I/O 的能力，并维护关于文件的基本数据。
访问方法	文件系统中与用户最近的一层。他在应用程序和文件系统以及板寸数据的设备之间提供了一个标准接口。

- 堆，数据文件，索引数据文件等并清楚说明概念

堆	最简单的文件组织形式，数据按照到达顺序被收集。慕斯仅仅是积累大量的数据并保存数据。记录可以有不同的域，或者域相似但顺序不同。堆文件没有结构，需要穷举搜索以找到文件。
顺序文件	每条记录都使用一种固定格式，所有记录具有相同的长度，并且由相同数目，长度固定的域按照特定的顺序组成。关键域唯一的标示这条记录，记录按照关键域存储。新记录被放在日志文件（事务文件）里，通过周期性地执行一个成批更新，把日志文件合并到主文件，并按照关键字的顺序产生一个新文件。

索引顺序文件　支持随机访问的文件索引和溢出文件。索引提供了快速接近目标记录的查找能力，索引文件中每条记录有两个域组成：关键域和指向主文件的指针，关键域与主文件中的关键域相同。查找关键域值小于或者等于目标关键域的最大记录，然后在该索引的指针所指的主文件中的位置开始查找。

当往文件中插入一个新记录时，他被添加到溢出文件，然后修改主文件中逻辑顺序位于这条新记录之前的记录，使其包含指向溢出文件中新记录的指针。如果新记录之前的文件也在溢出文件中那么修改新记录前面的那条记录的指针，索引顺序文件与顺序文件一样，也会合并文件。

为提供更有效的方式，可以采用多级索引。

索引文件　每条记录有不同的关键域与索引

完全索引：包含主文件中每条记录的索引项，易于查找，索引自身被组织成一个顺序文件。

部分索引：只包含那些感兴趣域的记录索引项

直接或散列文件　直接文件或三列文件开发直接访问磁盘中任何一个地址的已知能力。每条记录中都需要一个关键字。

- 三种文件分配的方式（连续，链表，索引）概念

连续　在创建文件的时候给文件分配一组连续的块，文件分配表中，每个文件只需要一个表项，用于说明起始块和文件长度。会导致外部碎片，要采用压缩算法。

链式　链式分配基于单个块，链表中的每一块都包含指向下一块的指针。每个文件

只需要一个表项（起始块，长度）。无外部碎片，适合顺序文件，局部性原理不再适用

索引　每个文件在文件分配表中有一个一级索引，分配给该文件的每个分区的索引中都有一个表项。每个已分配的部分都用索引表示，文件分配表包括块号作为索引。

- 二级存储空闲空间的管理方式

位表，链式空闲区，索引，空闲块列表，卷

- 链式空间应该重点把握

用链组织，使用头指针和长度描述空闲块

以上是操作系统的期末命题出题范围