第1章 计算机系统概述

**1.1 操作系统的基本概念**

**操作系统：**操作系统是指控制和管理整个计算机系统的硬件和软件资源，并合理地组织调度计算机的工作和资源的分配，以提供给用户和其他软件方便的接口和环境的程序集合。

**并发：**是指两个多多个事件在同一时间间隔内发生，引入进程的目的是使程序能并发执行。

注意：同一时间间隔（并发）和同一时刻（并行）的区别。微观上这些程序还是分时交替执行。

**共享：**是指系统中的资源可供内存中多个并发执行的进程共同使用，可分为互斥共享方式、同时访问方式。

**#并发和共享是操作系统两个最基本的特征。**

**虚拟：**是指把一个物理上的实体变为若干个逻辑上的对应物，如虚拟处理器、虚拟内存、虚拟外部设备。

**异步：**在多道程序环境下，允许多个程序并发执行，但由于资源有限，进程的执行不是一贯到底，而是走走停停，以不可预知的速度向前推进。

**操作系统作为计算机系统资源的管理者**：处理机管理、存储器管理、文件管理、设备管理

操作系统作为用户与计算机硬件系统之间的接口：

1）**命令接口**：联机命令接口又称交互式命令接口，适用于分时或实时系统的接口，由一组键盘操作命令组成；脱机命令接口又称批处理命令接口，即适用于批处理系统，由一组作业控制命令组成。

2）**程序接口**：由一组系统调用命令（简称系统调用，也称广义指令）组成

操作系统用作扩充机器

**1.2 操作系统的发展与分类**

1**手工操作阶段（此阶段无操作系统）**

2**批处理阶段（操作系统开始出现）**：

1）单道批处理系统：自动性、顺序性、单道性

2）多道批处理系统：多道、宏观上并行、微观上串行

3**分时操作系统**：同时性、交互性、独立性、及时性

4**实时操作系统**：及时性、可靠性

5**网络操作系统**：网络中各种资源的共享以及各台计算机之间的通信

6**分布式计算机系统**：分布性、并行性。分布式操作系统与网络操作系统本质上的不同之处在于分布式操作系统中，若干台计算机相互协同完成同一任务。

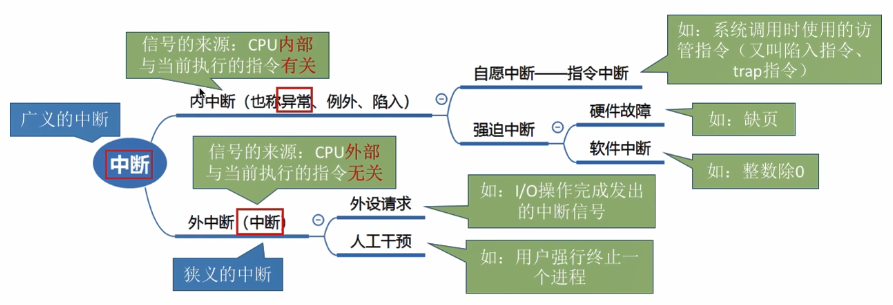
7**个人计算机操作系统**

**1.3 操作系统的运行环境**

操作系统内核包括：时钟管理、中断机制、原语、系统控制的数据结构及处理

**中断**，也称外中断，指来自CPU执行指令以外的事件的发生

**异常**，也称内中断、例外或陷入，指源自CPU执行指令内部的事件



**如果程序的运行由用户态转到核心态，会用到访管指令，访管指令是在用户态使用的，所以它不可能是特权指令。**

**1.4 操作系统的体系结构**

大内核：  
微内核：

第2章 进程管理

**2.1 进程与线程**

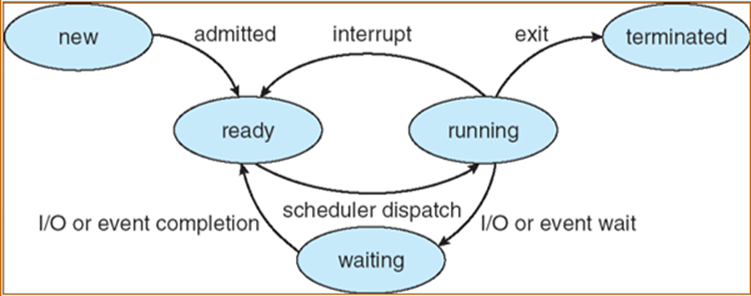
**进程控制块PCB**，描述进程的基本情况和运行状态，进而控制和管理进程。

由程序段、相关数据段和PCB三部分构成了进程映像（进程实体）。

**注意：进程映像是静态的，进程是动态的。PCB是进程存在的唯一标志**

进程是进程实体的运行过程，是系统进行资源分配合调度的一个独立单位，其特征包括：**动态性（最基本）、并发性、独立性、异步性、结构性**。

进程的状态：**运行、就绪、阻塞（等待）、创建、结束**



**就绪->运行**

**运行->就绪**

**运行->阻塞（主动的行为）**

**阻塞->就绪（被动的行为）**

**进程**：

1）进程控制块PCB：进程创建时，操作系统就新建一个PCB，它之后就常驻内存，在进程结束时删除，PCB是进程实体的一部分，是进程存在的唯一标志

2）程序段

3）数据段

**进程的通信**是指进程之间的信息交换

1）**共享存储**：基于数据结构的共享（低级）；基于存储区的共享（高级）

2）**消息传递**：直接通信方式、间接通信方式

3）**管道通信**：管道是指用于连接一个读进程和一个写进程以实现它们之间通信的一个共享文件

引入进程的目的，是为了更好地使多道程序并发执行，以提高资源利用率和系统吞吐量，增加并发程序

引入线程的目的，是为了减小程序在并发执行时所付出的时空开销，提高操作系统的并发性能

**线程**：“轻量级进程”，CPU执行单元，也是程序执行流的最小单元，由线程ID、程序计数器、寄存器集合和堆栈组成，线程自己不拥有系统资源。

线程是独立调度的基本单位，进程是拥有资源的基本单位。在同一进程中，线程的切换不会引起进程切换；在不同进程中进行线程切换，会引起进程切换。

**线程的属性**：

1）线程不拥有系统资源

2）不同的线程可以执行相同的程序

3）同一进程中的各个线程共享该进程所拥有的资源

4）线程是处理机的独立调度单位，多个线程是可以并发执行的

**用户级线程**：有关线程管理的所有工作都由应用程序完成，内核意识不到线程的存在

**内核级线程**：线程管理的所有工作都由内核完成

组合方式的多线程实现：内核级线程数目小于等于用户级线程的数目

多线程模型：

1）**多对一模型**：将多个用户级线程映射到一个内核级线程。效率高，不能并行。

2）**一对一模型**：每个用户级线程映射到一个内核级线程。并发性好，开销大。

3）**多对多模型**：将n个用户级线程映射到m个内核级线程上，要求m≤n。并发性和效率兼顾，但是增加了复杂度。

**2.2 处理机调度**

**调度的层次：**

**作业调度（高级调度）**：选择处于后备状态的作业分配资源，发生频率最低

**内存调度（中级调度）**：选择暂时不能运行的进程调出内存，发生频率中等

**进程调度（低级调度）**：选择就绪队列中合适的进程分配处理机，发生频率最高，最基本，不可或缺

不能进行进程的调度与切换的情况：

能进行进程的调度与切换的情况：

进程调度方式：

剥夺式调度方式，又称**抢占方式：**有更为重要或紧迫的进程需要使用处理，立即分配

非剥夺式调度方式，又称**非抢占方式：**有更为重要或紧迫的进程需要使用处理机，仍让当前进程继续执行

调度的基本准则

周转时间 = 作业完成时间 - 作业提交时间

平均周转时间 = （作业1的周转时间 + … + 作业n的周转时间）/ n

**带权周转时间** = 作业周转时间 / 作业实际运行时间

平均带权周转时间 = （作业1的带权周转时间 + … + 作业n的带权周转时间）/ n

**调度算法：**

**先来先服务FCFS：**选择最先进入队列的，属于不可剥夺算法，对长作业比较有利，但对短作业不利，有利于CPU繁忙型作业，不利于I/O繁忙型作业

**短作业优先SJF：**选择完成时间最短的，对长作业不利，“饥饿”现象，SJF的平均等待时间、平均周转时间最少

**优先级调度：**选择优先级别最高的

1）非剥夺式优先级调度算法

2）剥夺式优先级调度算法

1）静态优先级

2）动态优先级

**高响应比优先：**选择响应比最高的，对FCFS调度和SJF算法的一种平衡，有利于短作业，克服了“饥饿”，兼顾了长作业

响应比Rp=（等待时间+要求服务时间）/ 要求服务时间

**时间片轮转：**总是选择就绪队列中第一个进程，但仅能运行一个时间片，主要适用于分时系统

**多级反馈队列：**时间片轮转调度算法和优先级调度算法的综合和发展，终端型作业用户：短作业优先；短批处理作业用户：周转时间较短；长批处理作业用户：不会长期得不到处理

1）应设置多个就绪队列，并为各个队列赋予不同的优先级

2）赋予各个队列中进程执行时间片的大小也各不相同，在优先级越高的队列中，每个进程的运行时间片就越小

**2.3 进程同步**

进程同步的基本概念：

**临界资源：**一次仅允许一个进程使用的资源。

对临界资源的访问，必须互斥地进行，在每个进程中，访问临界资源的那段代码称为**临界区**，临界资源的访问过程分为四个部分：

1）进入区

2）临界区

3）退出区

4）剩余区

**同步：**需要在某些位置上协调进程之间的工作次序而等待、传递信息所产生的制约关系。

**互斥：**当一个进程进入临界区使用临界资源时，其他要求进入临界区必须等待。

为禁止两个进程同时进入临界区，同步机制应遵循的**准则：空闲让进、忙则等待、有限等待、让权等待**

实现临界区互斥的基本方法：

软件实现

1）单标志法：违背“空闲让进”原则

2）双标志法先检查：违背“忙则等待”原则

3）双标志法后检查：会导致“饥饿”现象

4）皮特森算法：单标志法和双标志法后检查的结合

硬件实现

中断屏蔽法：进区关中断，出区开中断

硬件指令法：设立原子操作指令：TestAndSet指令、Swap指令

**信号量：**只能被两个标准的原语wait(S)和signal(S)来访问，利用PV操作实现互斥。

**原语**是指完成某种功能且不被分割不被中断执行的操作序列。

当信号量K>0时，表示还有K个相关资源可用；

当信号量K=0时，表示临界区外无进程等待；

当信号量K<0时，表示临界区中有一个进程，且临界区外有|K|个进程在等待该资源；

当资源信号量小于0时，表示所有资源已经全部用完，而且还有进程正在等待该资源，等待的进程数就是资源量的绝对值。

整型信号量：

记录型信号量：

**利用信号量实现同步：**

**利用信号量实现互斥：**

**利用信号量实现前驱关系：**

**管程：**由一组数据以及定义在这组数据之上的对这组数据的操作组成的软件模块，这组操作能初始化并改变管程的数据和同步过程。

**管程的组成：**

1）局部于管程的共享结构数据说明

2）对该数据结构进行操作的一组过程

3）对局部于管程的共享数据设置初始值的语句

**管程的基本特性：**

1）局部于管程的数据只能被局部于管程内的过程所访问

2）一个进程只有通过调用管程内的过程才能进入管程访问共享数据

3）每次仅允许一个进程在管程内执行某个内部过程

**经典同步问题：**

**生产者-消费者问题**

**读者-写者问题**

**哲学家进餐问题**

**吸烟者问题**

**2.4 死锁**

**死锁：**多个进程因竞争资源而造成的一种僵局（互相等待），若无外力作用，这些进程都将无法向前推进。（区别于饥饿和死循环）

死锁产生的**原因：**

1）系统资源的竞争：只有对不可剥夺资源的竞争才可能产生死锁，对可剥夺资源的竞争是不会引起死锁的。

2）进程推进顺序非法：请求和释放资源的顺序不当；信号量使用不当也会造成死锁。

3）**死锁产生的必要条件（任意条件不成立则无死锁）：**

**互斥：**在一段时间内某资源仅为一个进程所占有。

**不剥夺：**进程所获得的资源在未使用完毕之前，不能被其他进程强行夺走（只能是主动释放）。

**请求并保持：**进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源已被其他进程占有。

**循环等待：**存在一种进程资源的循环等待链，链中每一个进程已获得的资源同时被链中下一个进程所请求。

**资源分配图**含圈而系统又不一定有死锁的原因是同类资源数大于1。但若系统中**每类资源都只有一个资源**，则资源分配图**含圈**就变成了系统出现死锁的**充分必要条件**。

|  | **资源分配策略** | **各种可能模式** | **主要优点** | **主要缺点** |
| --- | --- | --- | --- | --- |
| 死锁预防 | 保守，宁可资源闲置 | 一次请求所有资源，资源剥夺，资源按序分配 | 适用于做突发式处理的进程，不必进行剥夺 | 效率低，进程初始化时间延长；剥夺次数过多；不便灵活申请新资源 |
| 死锁避免 | 是“预防”和“检测”的折中（在运行时判断是否死锁） | 寻找可能的安全允许顺序 | 不必进行剥夺 | 必须知道将来的资源需求；进程不能被长时间阻塞 |
| 死锁检测 | 宽松，只要允许就分配资源 | 定期检查死锁是否已经发生 | 不延长进程初始化时间，允许对死锁进行现场处理 | 通过剥夺解除死锁，造成损失 |

**死锁预防（事先操作，破坏四个必要中的一个）**

破坏互斥条件：有些资源必须互斥使用，**无法破坏互斥条件**。

#SPOOLing技术

破坏不剥夺条件：增加系统开销，降低吞吐量。

破坏请求和保持条件：可采用预先静态分配法，严重浪费系统资源，还可能导致饥饿现象。

破坏循环等待条件：可采用顺序资源分配法，浪费系统资源，并造成编程不便。

**死锁避免（过程中操作）**

**系统安全状态：**能找到一个分配资源的序列能让所有进程都顺利完成。

**并非所有的不安全状态都是死锁状态**，但当系统进入不安全状态后，便可能进入死锁状态；反之，只要系统处于安全状态，系统便可以避免进入死锁状态。

**银行家算法：**采用**预分配**策略检查**分配完成时**系统**是否处在安全状态**。其中的安全性算法用于查找是否有安全序列，有则分配完成时处于安全状态。银行家算法是保守的算法，系统安全状态才给进程分配资源，虽然非安全状态不一定是死锁状态。

**死锁检测（已经有死锁）**

1）**资源分配图：**圆圈代表一个进程，框代表一类资源，进程到资源的有向边叫**请求边**，资源到进程的边叫**分配边**

2）**死锁定理：**S为死锁的条件是当且仅当S状态的资源分配图是不可完全简化的；若能消去图中所有的边，则称该图是可完全简化的。

**死锁解除（已经有死锁）**

资源剥夺法：挂起某些死锁进程并抢夺它的资源，以便让其他进程继续推进。

撤销进程法：强制撤销部分、甚至全部死锁进程并剥夺这些进程的资源。撤销的原则可以按进程优先级和撤销进程代价的高低进行。

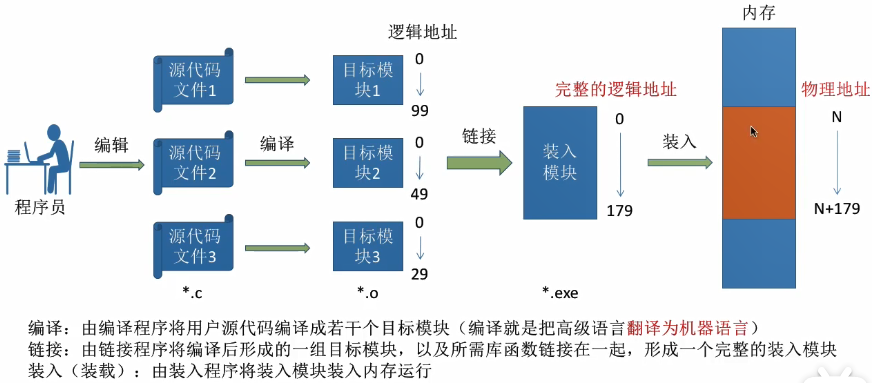
进程回退法：让一个或多个进程回退到足以回避死锁的地步，进程回退时自愿释放资源而不是被剥夺。

第3章 内存管理

**3.1 内存管理概念**

**内存管理的功能**：  
1）内存空间的分配与回收  
2）地址转换：逻辑地址->物理地址  
3）内存空间的扩充：利用虚拟存储技术或自动覆盖技术，从逻辑上扩充内存  
4）存储保护

**编译、链接、装入**



**链接：**

**静态链接**：在**程序运行之前**，先将各目标模块及它们所需的库函数链接成一个完整的可执行程序，**以后不再拆开**。

**装入时动态链接**：将用户源程序编译后所得到的一组目标模块，在装入内存时，采用**边装入边链接**的方式。

**运行时动态链接**：对某些目标模块的链接，是在程序执行中**需要该目标模块时，才对它进行的链接**，便于修改和更新，便于实现对目标模块的共享

**装入：**

**绝对装入**：在编译时，如果知道程序将驻留在内存的某个位置，编译程序将**产生绝对地址的目标代码**。绝对装入程序按照装入模块中的地址，将程序和数据装入内存。**只适用于单道程序环境**。

**可重定位装入**：根据内存当前情况，将装入模块装入到内存的适当位置，装入时对目标程序中指令和数据的修改过程称为重定位，地址变换通常是在装入时一次完成的，所以又称为**静态重定位**。静态重定位的特点是在一个作业装入内存时，**必须分配其要求的全部内存空间**，如果没有足够的内存，就不能装入该作业。作业一旦进入内存后，在整个运行期间**不能在内存中移动，也不能再申请内存空间。**

**动态运行时装入**：也称**动态重定位**，装入程序并不立即把装入模块中的相对地址转换为绝对地址，而是把这种地址转换推迟到程序真正要执行时才进行。装入内存后的所有地址均为相对地址，需要一个**重定位寄存器**的支持。动态重定位的特点是**可以将程序分配到不连续的存储空间**；在程序运行之前可以只装入它的**部分代码**即可投入运行，然后在程序运行期间，根据需要**动态申请分配内存**；便于程序段的**共享**，提供一个比存储空间**大得多的地址空间**。

当装入程序将可执行代码装入内存时，必须通过地址转换将逻辑地址转换成物理地址，这个过程称为地址重定位。

**内存保护**  
1）在CPU中设置一对**上、下限寄存器**，存放用户作业在主存中的下限和上限地址，每当CPU访问一个地址时，分别和两个寄存器的值相比，判断有无越界。  
2）采用**重定位寄存器**（基址寄存器）和**界地址寄存器**（限长寄存器），内存管理机构动态地将逻辑地址与界地址寄存器进行比较，如果未发生地址越界，则加上重定位寄存器的值后映射成物理地址，再送交内存单元。

**扩充内存**  
1）**覆盖**：把用户空间分成**一个固定区**和**若干个覆盖区**，将经常活跃的部分放在固定区，其余部分按调用关系分段，首先将那些即将要访问的段放入覆盖区，其他段放在外存中，**在需要调用前，系统再将其调入覆盖区**，替换覆盖区中原有的段。特点是打破了必须将一个进程的全部信息装入主存后才能运行的限制。  
2）**交换**：把处于等待状态（或在CPU调度原则下被剥夺运行权利）的程序从内存移到辅存，把内存空间腾出来，这一过程叫“**换出**”，把准备好竞争CPU运行的程序从辅存移到内存，这一过程叫“**换入**”。

#交换按各种优先级因素换出进程但PCB常驻内存。

#交换：对换区：速度快，容量小，连续分配；文件区：容量大，利用率高，离散分配。

**交换技术主要是在不同进程（或作业）之间进行，而覆盖则用于同一个程序或进程中。**

#覆盖技术对用户和程序员不透明，已被虚拟内存技术替代。交换技术现在还常用。

**连续分配管理方式**

**单一连续分配**：优点是简单、无外部碎片，可以采用覆盖技术；缺点是只能用于单用户、单任务的操作系统中，有内部碎片，存储器的利用率极低  
**固定分区分配**：无外部碎片，有内部碎片  
**动态分区分配**：可变分区分配，产生外部碎片，可用紧凑技术解决  
1）**首次适应**算法：空闲分区以**地址递增**的次序链接，分配内存时顺序查找，找到大小能满足要求的第一个空闲分区。

2）**最佳适应**算法：空闲分区以**容量递增**形成分区链，找到第一个能满足要求的空闲分区。**性能通常很差，会产生许多的外部碎片，比最坏适应算法还要产生更多的内存碎片**。

3）**最坏适应**算法：又称最大适应算法，空闲分区以**容量递减**的次序链接，找到第一个也即是最大的分区。性能非常差。

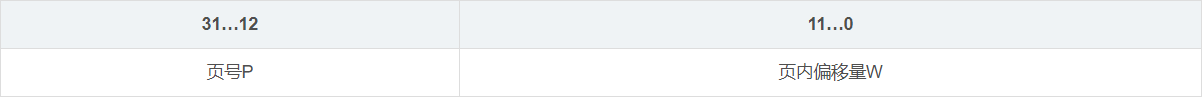
4）**邻近适应**算法：又称循环首次适应算法，分配内存时从上次查找结束的位置开始继续查找。通常不如首次适应算法。

首次适应算法可能比最佳适应算法效果好，而它们两者一定比最大适应算法效果好。

**非连续分配管理方式**

**1）基本分页存储管理：不会产生外部碎片，地址空间是一维的**  
进程中的块称为页，内存中的块称为页框（或页帧），外存也以同样的单位进行划分，直接称为块。

**地址结构**：0~11位为页内地址，即每页大小为4KB；12 ~ 32位为页号，地址空间最多允许有2^20页



页表由页表项组成，页表项包括两部分，第一部分为页号，第二部分为物理内存中的块号，系统为每个进程建立一张页表，记录页面在内存中对应的物理块号，实现从页号到物理块号的地址映射。

**基本地址变换结构**（重要，限于篇幅）：  
页面大小L，逻辑地址A，物理块号b  
页号P=A/L，页内偏移量W=A%L  
物理地址E=b\*L+W

**快表：**具有并行查找能力的高速缓冲存储器，用来存放当前访问的若干页表项，以加速地址变换的过程

建立多级页表的目的在于建立索引，不用浪费主存空间去存储无用的页表项，也不用盲目地顺序式查找页表项

**2）基本分段存储管理：段内要求连续，段间不要求连续，地址空间是二维的**  
在页式系统中，逻辑地址的页号和页内偏移量对用户是透明的；在段式系统中，段号和段内偏移量必须由用户显示提供。

**地址结构**：段号为16位，段内偏移量为16位，则一个作业最多有2^16=65536个段，最大段长为64KB



段表用于实现从逻辑段到物理内存区的映射

地址变换结构

**3）段页式存储管理**：段表只有一个，而页表可能有多个，地址空间是二维的

**地址结构**：

****

**3.2 虚拟内存管理**

局部性原理：时间局部性、空间局部性（高速缓存技术依赖的原理）

虚拟存储器三个特征：多次性、对换性、虚拟性

虚拟内存的实现：需要有一定的硬件支持  
1）请求分页存储管理 2）请求分段存储管理 3）请求段页式存储管理

**页面置换算法**：  
1）**最佳置换OPT**：被淘汰页面是以后永不使用的，或者是在最长时间内不再被访问的页面，该算法无法实现  
2）**先进先出FIFO**：优先淘汰最早进入内存的页面，即在内存中驻留时间最久的页面，当所分配的物理块数增大而页故障数不减反增的异常现象称为Belady异常，LRU和OPT不会出现该异常  
3）**最近最久未使用LRU**：认为过去一段时间内未访问过的页面，在最近的将来可能也不会被访问；需要寄存器和栈的硬件支持，是堆栈类算法，不可能出现Belady异常  
4）**时钟CLOCK置换算法**：又称为**最近未用NRU算法**

**改进型的CLOCK算法**：  
1）从指针的当前位置开始，扫描帧缓冲区，在这次扫描过程中，对使用位不做任何修改，选择遇到的第一个帧（u=0，m=0）用于替换  
2）如果第1步失败，则重新扫描，查找（u=0，m=1）的帧。选择遇到的第一个这样的帧用于替换。在这个扫描过程中，对每个跳过的帧，把它的使用位设置成0（u=0）  
3）如果第2步失败，指针将回到它的最初位置，并且集合中所有帧的使用位均为0，重复第1步，并且如果有必要，重复第2步，这样就可以找到供替换的帧。  
**改进型的CLOCK算法优于简单CLOCK算法之处在于替换时首选没有变化的页。**

**页面分配策略**：

**驻留集**：给一个进程分配的物理页框的集合。

页面分配策略与页面置换策略有组合如下

1）**固定分配局部置换**：分配固定数目的物理块，整个运行期间不变。进程缺页则从该进程在内存的页面中选出一页换出。

2）**可变分配全局置换**：进程缺页就从空闲物理块队列中取出一个物理块分配给该进程。动态增加物理块，但盲目给进程增加物理块，会导致系统的并发能力下降。

3）**可变分配局部置换**：进程缺页则从该进程在内存的页面中选出一页换出。进程频繁缺页则系统再为该进程分配若干物理块，缺页率低就适当减少分配给该进程的物理块。

**调入页面的时机**：  
1）预调页策略  
2）请求调页策略  
预调页实际上就是运行前的调入，请求调页实际上就是运行期间调入。

请求分页系统中的外存分为两部分：用于存放文件的文件区和用于存放对换页面的对换区。对换区通常是采用连续分配方式，而文件区采用离散分配方式，故对换区的磁盘I/O速度比文件区的更快  
1）系统拥有足够的对换区空间：可以全部从对换区调入所需页面  
2）系统缺少足够的对换区空间：凡不会被修改的文件都直接从文件区调入；而当换出这些页面时，由于它们未被修改而不必再将它们换出。但对于那些可能被修改的部分，换出时须调到对换区，以后需要时再从对换区调入  
3）UNIX方式：未运行过的页面，都应从文件区调入，曾经运行过但又被换出的页面，由于是被放在对换区，因此下次调入时应从对换区调入

**抖动（颠簸）**：刚刚换出的页面马上又要换入主存，刚刚换入的页面马上就要换出主存，频繁的页面调度行为。如果一个进程在换页上的时间多于执行时间，那么这个进程就在颠簸。  
频繁的发生缺页中断（抖动），其主要原因是某个进程频繁访问的页面数目高于可用的物理页帧数目。

**工作集**：是指在某段时间间隔内，进程要访问的页面集合。经常被使用的页面需要在工作集中，而长期不被使用的页面要从工作集中被丢弃。基于局部性原理。工作集可由时间t和工作集窗口大小k确定，**t时刻的工作集指该时刻前面访问的k个序列的集合（要去重）**。

工作集模型的原理是：让操作系统跟踪每个进程的工作集，并为进程分配大于其工作集的物理块。

**3.3 本章疑难点**



第4章 文件管理

**4.1 文件系统基础**

**文件**是以计算机硬盘为载体存储在计算机上的信息集合。在系统运行时，计算机以进程为基本单位进行资源的调度和分配，而在用户进行的输入、输出中，则以文件为基本单位。

**无结构文件**：又称流式文件，将数据按顺序组织成记录并积累保存  
**有结构文件**：又称记录式文件，由一组相似记录组成

**文件的属性**：  
1）名称：唯一  
2）标识符：唯一标签，对人不可读的一种内部名称  
3）类型  
4）位置  
5）大小  
6）保护  
7）时间、日期和用户标识  
所有文件的信息都保存在目录结构中，而目录结构保存在外存上，目录条目包括文件名称及其唯一标识符。

文件的基本操作：  
1）创建文件  
2）写文件  
3）读文件  
4）文件重定位（文件寻址）  
5）删除文件  
6）截断文件

系统调用open通常返回一个指向打开文件表中的一个条目的指针，通过使用该指针（而非文件名）进行所有I/O操作。**在open调用完成之后，操作系统对该文件的任何操作，都不再需要文件名，只需要open调用返回的指针。**

**文件的逻辑结构**

**无结构文件**：又称流式文件，将数据按顺序组织成记录并积累保存，如源程序文件、目标代码文件等  
**有结构文件**：又称记录式文件，由一组相似记录组成  
1）顺序文件  
串结构：记录之间的顺序与关键字无关  
顺序结构：记录之间的顺序与关键字有关  
2）索引文件：为变长文件建立索引表，提高查找速度，变长记录文件只能顺序查找，索引表本身是定长记录的顺序文件  
3）索引顺序文件：顺序文件和索引文件的结合  
4）直接文件或散列文件：通过哈希函数直接决定记录的物理地址，没有顺序的特性

**目录结构**

**文件控制块**：用来存放控制文件需要的各种信息的数据结构，以实现“按名存取”。**FCB的有序集合称为文件目录**，一个FCB就是一个文件目录项。为了创建一个新文件，系统将分配一个FCB并存放在文件目录中，成为目录项。  
**FCB主要包含以下信息：基本信息、存取控制信息、使用信息**

单级目录结构：在整个文件系统中只建立一张目录表，每个文件占一个目录项  
两级目录结构：将分件目录分成主文件目录和用户文件目录  
多级目录结构（树形目录结构）：可以很方便地对文件进行分类  
无环图目录结构：实现文件共享

**文件共享**  
1）基于**索引结点**的共享方式（**硬链接**）（count为0时系统才删除文件）

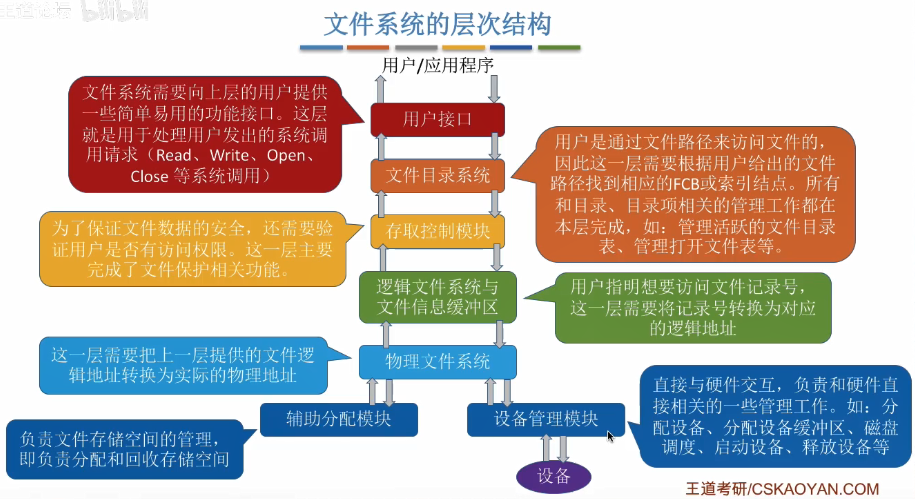
2）利用**符号链**实现文件共享（**软链接**）（快捷方式）

硬链接和软链接都是文件系统中的静态共享方法，硬链接的查找速度会比软链接快。

**文件保护**  
1）口令保护：通过口令访问文件，时间和空间开销不多，缺点是口令直接存在系统内部，不够安全  
2）加密保护：对文件进行加密处理，节省了存储空间，不过编码和译码要花费一定时间  
3）访问控制：根据访问者的身份进行限制

**4.2 文件系统实现**

**文件的层次结构**



**目录实现**  
1）**线性列表**：使用存储文件名和数据块指针的线性表  
无序：查找文件较慢，新建文件较快  
有序：查找文件较快，新建文件较慢  
2）**哈希表**：根据文件名得到一个值，并返回一个指向线性列表中元素的指针  
查找、新建速度都较快，要处理冲突

**文件实现**  
**文件分配方式**：  
1）**连续分配**：在磁盘上连续存放文件，寻道数和寻道时间最小，支持顺序访问和直接访问  
优点：实现简单、存取速度快  
缺点：文件长度不宜动态增加，会产生外部碎片，只适用于长度固定的文件  
2）**链接分配**：采取离散分配的方式，消除了外部碎片，提高磁盘空间的利用率  
隐式：采用类似链表的结构，无法直接访问盘块，只能通过指针顺序访问文件  
显式：把隐式文件中的指针单独抽离出来，存放在内存的一张链接表（文件分配表）中，该表在整个磁盘仅设置一张，每个表项中存放对应块的下一块链接指针，即下一个盘块号  
3）**索引分配**：每个文件所有的盘块号都集中存放，建立索引表，支持直接访问，且没有外部碎片问题，易于文件扩展

|  | **访问第n个记录** | **优点** | **缺点** |
| --- | --- | --- | --- |
| 顺序分配 | 需访问磁盘1次 | 速度快，可根据文件起始地址及记录长度进行随机访问 | 要求连续存储空间，会产生外部碎片，不利于动态扩展 |
| 链接分配 | 需访问磁盘n次 | 可以解决外部碎片问题，提高外存空间利用率，动态增长较方便 | 只能按照文件的指针链顺序访问 |
| 索引分配 | m级需访问磁盘m+1次 | 可以随机访问，易于文件的增删 | 索引表增加存储空间的开销，对文件系统效率影响较大 |

**存储空间管理**：  
**空闲表**：属于连续分配方式，把所有空闲块组织成表  
**空闲链表法**：把所有空闲块组织成链表  
**位示图**：利用二进制的每位记录空闲块  
盘块的分配：盘块号b=n(i-1)+j  
盘块的回收：i=(b-1) DIV n + 1；j=(b-1) MOD n + 1  
**成组链接**：空闲表和空闲链表的结合，适合大的文件系统

**4.3 磁盘组织与管理**

磁盘地址结构：柱面号、盘面号、扇区号

**寻道时间**：将磁头移动到指定磁道所需要的时间，Ts=m\*n+s  
**延迟时间**：磁头定位到某一磁道的扇区所需要的时间，Tr=1/(2r)  
**传输时间**：从磁盘读出或向磁盘写入数据所经历的时间，Tt=b/(rN)  
启动时间（一般忽略）：控制器的启动时间

**调度算法**:  
1）**先来先服务**：根据进程请求访问磁盘的先后顺序进行调度  
2）**最短寻找时间优先**：选择当前磁头所在磁道距离最近的磁道，会产生“饥饿”现象  
3）**扫描算法**：，又称电梯调度，在磁头当前移动方向上选择与当前磁头所在磁道距离最近的请求  
4）**循环扫描**：在扫描算法的基础上规定磁头单向移动来提供服务，回返时直接快速移动至起始端而不服务任何请求

|  | **优点** | **缺点** |
| --- | --- | --- |
| FCFS算法 | 公平、简单 | 平均寻道距离大，仅应用在磁盘I/O较少的场合 |
| SSTF算法 | 性能比FCFS好 | 不能保证平均寻道时间最短，可能出现“饥饿”现象 |
| SCAN算法 | 寻道性能较好，可避免“饥饿”现象 | 不利于远离磁头一端的访问请求 |
| C-SCAN算法 | 消除了对两端磁道请求的不公平 | - |

**磁盘管理**：  
1）磁盘初始化：对磁盘进行低级格式化和逻辑格式化  
**低级格式化（物理分区）：**在磁盘存储数据之前，必须分成扇区以便磁盘控制器能进行读和写操作  
**逻辑格式化（创建文件系统）：**对物理分区进行逻辑格式化，操作系统将初始的文件系统数据结构存储到磁盘上，这些数据结构包括空闲和已分配的空间以及一个初始为空的目录  
2）引导块：存放自举程序  
3）坏块：对于损坏扇区的处理，使系统不去使用坏块，坏块属于硬件故障，操作系统是不能修复的

第5章 输入/输出（I/O）管理

**5.1 I/O管理概述**

**设备分类**  
**按传输速率分**：  
低速：如键盘、鼠标  
中速：如行式打印机、激光打印机  
高速：如磁带机、磁盘机、光盘机  
**按信息交换单位分**：  
块设备：如磁盘  
字符设备：如键盘、打印机

**I/O控制方式**  
**程序直接控制**：程序直接对设备循环测试  
**中断驱动**：引入中断机制，当设备准备完成时发生中断  
**DMA**：在I/O设备与主存之间开辟直接数据通路，彻底解放CPU  
1）基本单位是数据块  
2）所传送的数据，是从设备直接送入内存的，或者相反  
3）仅在传送一个或多个数据块的开始和结束时，才需CPU干预，整块数据的传送是在DMA控制器的控制下完成的  
**中断驱动方式数据传送是在中断处理时由CPU控制完成的，而DMA控制方式则是在DMA控制器的控制下完成的。**  
**通道控制**：引入专门的I/O处理机进行管理，可以实现CPU、通道和I/O设备三者的并行操作  
**I/O通道与一般处理机的区别**是：通道指令的类型单一，没有自己的内存，通道所执行的通道程序是放在主机的内存中的，也就是说通道与CPU共享内存。  
**I/O通道与DMA方式的区别**是：DMA方式需要CPU来控制传输的数据块大小、传输的内存位置，而通道方式中这些信息是由通道控制的。另外，每个DMA控制器对应一台设备与内存传递数据，而一个通道可以控制多台设备与内存的数据交换。

**I/O子系统层次结构**：  
用户层I/O软件：实现与用户交互的接口  
设备独立性软件：实现用户程序与设备驱动器的统一接口、设备命令、设备保护以及设备分配与释放  
设备驱动程序：与硬件直接相关，负责具体实现系统对设备发出的操作指令  
中断处理程序：用于处理中断相关事项  
硬件设备：包括一个机械部件（设备本身）和一个电子部件（控制器）

**5.2 I/O核心子系统**

**I/O调度**：确定一个好的顺序来执行I/O请求

**磁盘高速缓存**：在逻辑上属于磁盘，物理上则是驻留在内存中的盘块，分为两种形式：  
1）在内存中开辟一个单独的存储空间作为磁盘高速缓存  
2）把未利用的内存空间作为一个缓冲池，供请求分页系统和磁盘I/O分时共享

**引入缓冲区的目的**：  
1）缓和CPU与I/O设备间速度不匹配的矛盾  
2）减少对CPU的中断频率，放宽对CPU中断响应时间的限制  
3）解决基本数据单元大小（即数据粒度）不匹配的问题  
4）提高CPU与I/O设备之间的并行性

**单缓冲**：先把被交换数据写入缓冲区，然后需要数据的设备或处理机从缓冲区取走数据，**处理C，输入T，传送M，用时Max(C, T)+M**  
**双缓冲：用时Max(C+M, T)**  
**循环缓冲  
缓冲池**

**高速缓存和缓冲区的区别**：  
**高速缓存**：存放的是低速设备上的某些数据的复制数据，也就是高速缓存上有的低速设备上面必然有；如果高速设备要访问的数据不在高速缓存中，高速设备就需要访问低速设备  
**缓冲区**：存放的是低速设备给高速设备的数据（或者相反），而这些数据在低速设备（或高速设备）不一定有备份，这些数据再从缓冲区传送到高速设备（或低速设备）；高速设备和低速设备的通信要经过缓冲区，高速设备永远不会直接去访问低速设备

**设备的分配与回收**  
**设备分配**是指根据用户的I/O请求分配所需的设备。  
1）独占式使用设备：设备被使用时不再允许其他进程使用该设备  
2）分时式共享使用设备  
3）SPOOLing技术：即假脱机I/O技术，实质上就是对I/O操作进行批处理，是一种以空间换时间的技术，将独占设备改造为共享设备  
**分配的总原则**：即要充分发挥设备的使用效率，又要避免造成进程死锁，还要将用户程序和具体设备隔离开  
分配方式  
静态分配：在用户作业开始执行前，由系统一次性分配该作业所要求的全部设备  
动态分配：在进程执行过程中根据执行需要进行分配

**设备分配的数据结构**  
设备控制表DCT、控制器控制表COCT、通道控制表CHCT、系统设备表SDT  
**设备控制表DCT和控制器控制表COCT是一一对应的关系；  
通道控制表CHCT和控制器控制表COCT是一对多的关系。  
整个系统只有一张系统设备表SDT**

**设备分配的策略**  
1）静态分配：主要用于对独占设备的分配，由系统一次性分配该作业所要求的全部设备、控制器，并不符合分配的总原则  
2）动态分配：在进程执行过程中根据执行需要进行  
设备分配算法：现请求先分配、优先级高者优先  
**设备分配的安全性**  
1）安全分配方式：每当进程发出I/O请求后进入阻塞状态，直到I/O操作完成时才被唤醒，CPU和I/O设备串行工作  
2）不安全分配方式：一个进程可同时操作多个设备，从而使进程推进迅速，但有可能产生死锁

**逻辑设备名到物理设备名的映射**  
1）在整个系统中只设置一张逻辑设备表LUT  
2）为每个用户设置一张逻辑设备表LUT

**SPOOLing技术（假脱机技术）**：外部设备同时联机操作，又称为假脱机输入/输出操作，是操作系统中采用的一项将独占设备改造成共享设备的技术