好的，我已经根据您提供的教材文件和考点，整理出了一份知识清单。内容均来自教材，并按照考点的顺序结构排列。

计算机操作系统知识清单

第一章 概论

1. 操作系统的定义和功能

- 定义：什么是操作系统？
  - 操作系统是一组计算机程序的集合，主要用于控制和管理计算机的硬件和软件资源，合理地组织计算机的工作流程，向应用程序和用户提供方便、快捷、友好的使用接口。
- 功能：
  - 进程管理：主要围绕进程如何使用处理机(CPU)展开讨论，保证CPU正确地同时运行多道程序。
  - 存储管理：主要任务是管理内存资源，为每个程序分配内存、保护用户的程序和数据不被破坏，并实现虚拟存储器。
  - 设备管理：基本任务是管理输入输出(I/O)设备，使多个用户能够共享设备，方便地使用设备。
  - 文件管理：对文件进行组织、管理，向文件用户提供对文件按名称进行存储、检索、查询、保护等的接口。
  - 网络通信与服务：包含主要的标准局域网的网络接口，支持网络进程通信、网络文件服务等分布计算的基本功能。
  - 安全与保护：提供必要的安全与保护机制，如本地与网络用户身份的认证与鉴别、合法用户的使用权限控制、文件系统的访问控制等。

2. 操作系统的四个特征

- 并发性
  - 原因：在单处理器多道程序环境下，在一段时间内有多道程序同时处于活跃状态，为了改善系统资源的利用率。
  - 特点：两个或多个事件在同一时间间隔内发生的、同时处于活动状态的特性。宏观上程序并行运行，微观上程序交替运行。
- 共享性
  - 原因：多道程序系统的自然要求，系统所配备的资源数量有限，需要操作系统提供管理手段协调资源共享使用。
  - 特点：内存中并发运行的多个程序可以共享计算机的硬件和软件资源。主要有互斥共享方式和交替共享方式。
- 虚拟性
  - 原因：为了在资源的使用上更加自由和灵活，不受物理设备数量的限制。

    - 特点：将一个物理实体映射为一个或多个逻辑对象。例如，把一个物理上的CPU虚拟为多个逻辑上的CPU。
- 随机性
    - 原因：由于共享和并发的存在，多道程序相互竞争使用各类有限的物理资源。
    - 特点：也叫异步性，指的是每道程序在何时运行，各个程序运行的顺序，以及每道程序所需的时间都是不确定的，也是不可预知的。

3. 操作系统的发展

- 批处理：
    - 联机批处理：操作员把若干作业合成一批，监督程序把这一批作业从输入设备上逐个输入磁带上，然后逐个装入内存运行。
    - 脱机批处理：增加一台卫星机，首先把输入设备上的成批作业通过卫星机逐个输入输入磁带上，供主机调入内存运行，主机运行后的结果输出到输出磁带上，由卫星机再把输出磁带上的结果打印出来。
    - 多道批处理（特点）：
        - 多个作业同时运行：系统内可同时容纳多个作业，形成后备作业队列，系统按策略挑选作业调入内存运行。
        - 脱机操作：用户预先书写作业控制说明，交给操作员输入计算机，由操作系统控制运行，用户不能干预。
- 分时操作系统（特点与特征）：
    - 特点与特征：
        - 同时性：允许各终端用户同时工作，系统分时响应各用户的请求。
        - 交互性：支持联机的操作方式，用户可以在终端上通过操作系统进行人-机对话。
        - 独立性：各用户之间彼此独立的工作，好像独占一台计算机系统一样，互不干扰。
        - 及时性：用户的请求能在较短的时间内得到响应。
    - 与批处理的区别：
        - 作业控制方式不同：批处理使用作业控制语言，分时系统用户通过命令交互。
        - 适应的作业不同：批处理适应不需交互的大作业，分时系统适应需交互的小作业。
        - 系统追求目标不同：批处理追求资源利用率和吞吐量，分时系统追求及时响应。

4. 操作系统的接口

- 程序接口（包含哪些类型？）
    - 操作系统提供给程序使用操作系统服务和功能的接口，通常通过系统调用和**应用程序编程接口(API)**来实现。
        - 系统调用 **(System Call)**：为了扩充机器功能、增强系统能力、方便用户使用而

建立的。用户程序或其他系统程序通过系统调用就可以访问系统资源，调用操作系统功能。
        - **API (Application Programming Interface)**：是将操作系统的系统调用基础经过规范整理出来的、面向社会公布的唯一的接口方式（特指Windows）。
- 操作接口（包含哪些类型？）
    - 操作系统向用户提供可以操控计算机的接口。
        - 命令界面 **(CUI/CLI)**：用户通过输入命令使用计算机系统。
        - 图形界面 **(GUI)**：使用窗口、图标、菜单、滚动条、按钮和鼠标等各种形象的图形符号，将系统的各项功能直观、逼真地表示出来。
        - 作业控制命令 **(JCL)**：专门为批处理作业的用户提供的，用户利用作业控制语言书写批处理作业控制说明书。

第二章 进程管理

1. 程序的运行方式

- 程序的顺序运行（概念、定义、特征）
    - 概念/定义：一个具有独立功能的程序独占处理器直至最终结束的过程称为程序的顺序运行。
    - 特征：
        - 顺序性：程序运行过程可看作一系列严格按程序规定的状态转移过程。
        - 封闭性：程序运行得到的最终结果由给定的初始条件决定，不受外界因素的影响。
        - 可再现性：只要输入的初始条件相同，则无论何时重复运行该程序都会得到相同的结果。
- 程序的并发运行（概念、定义、特征）
    - 概念/定义：指多个事件在同一时期内发生。在多道程序环境下，并发性是指在一段时间内系统中宏观上有多个程序在同时运行，但在单CPU系统中，每个时刻却仅能有一道程序运行，故微观上这些程序只能是分时地交替运行。
    - 特征：
        - 间断性：各个程序的运行流程可能是"运行→暂停→继续……"这样的模式。
        - 开放/交互性：多个程序运行时可能会相互影响。
        - 不可再现性：程序在不同的情况下运行出现不同的结果，甚至会造成错误。
- 程序的并行运行（概念、定义、特征）
    - 概念/定义：指多个事件在同一时刻发生。如果计算机系统中有多个CPU，则这些可以并发运行的程序便可被分配到多个CPU上，实现并行运行，即利用每个CPU来处理一个可并发运行的程序，这样，多个程序便可以真正地同时运行。
    - 特征：并行是并发的特例，硬件前提是系统中有多个CPU。

2. 进程的概念

- 什么是进程？程序？作业？
  - 进程 **(Process)**：可并发运行的程序在某个数据集合上的一次运行过程，是操作系统资源分配、保护和调度的基本单位。其概念的理解侧重于进程是已装入内存中运行的程序及其相关数据结构。
  - 程序 **(Program)**：特指代码文件，强调其静态性，其代码可以是二进制机器指令，也可以是高级语言。
  - 作业 **(Job)**：一般是指批处理系统要装入系统运行处理的一系列程序和数据，一般由相应的定义语言来描述作业步骤、参数等运行细节。

3. 进程的状态及转换（重点）

- 三态：
  - 就绪状态 **(Ready)**：进程在内存中已经具备执行的条件，等待分配CPU。
  - 运行状态 **(Running)**：进程占用CPU并正在运行。
  - 阻塞状态 **(Blocked/Waiting)**：正在运行的进程由于发生某事件而受到阻塞不能继续运行时，便需要放弃CPU，从运行状态转换到阻塞状态。
- 进程状态转换图：
  - 就绪状态 → 运行状态：当CPU空闲时，操作系统从就绪队列中选中一个就绪进程并分配CPU。
  - 运行状态 → 阻塞状态：当正在运行的进程需要等待某些事件（如I/O请求响应）的发生时。
  - 阻塞状态 → 就绪状态：处于阻塞状态的进程，由于等待的事件到来而不需要再等待时。
  - 运行状态 → 就绪状态：正在运行的进程被操作系统中断运行（如时间片用完，或优先级更高的进程进入就绪队列）。
  - (教材中图2.3为三态模型转换图，图2.4为五态模型转换图，图2.5为具有挂起功能的进程状态转换图)
- 什么是**PCB**？
  - 进程控制块 (Process Control Block, PCB)，或称为进程描述符 (Process Descriptor)，是操作系统为每个进程定义的一个数据结构，它是进程实体的一部分，是操作系统中最重要的数据结构之一。PCB中记录了描述进程的当前状态以及控制进程运行的信息。
- 挂起与激活（特征）阻塞与唤醒挂起（特征）
  - 挂起就绪 **(Ready Suspended)** 状态：进程具备执行条件，但目前不在内存中，需要被系统调入内存才能被执行。
  - 挂起阻塞 **(Blocked Suspended)** 状态：进程在等待某一事件或条件，并且该进程目前不在内存中。
  - 特征：
    - 挂起的进程实际就是被淘汰出内存的进程，不参与低级调度。
    - 平滑系统操作负荷。

        - 在解除挂起之前不能立即被执行。
        - 挂起条件独立于等待事件。
        - 只能由操作系统或其父进程解除挂起。
        - 进程在运行状态也可以被挂起，则转换为挂起就绪状态。
        - 如果就绪状态的进程被挂起，则进程状态转换为挂起就绪状态。处于挂起就绪状态的进程一旦被恢复/激活，将从挂起就绪状态转换为就绪状态。
        - 如果阻塞状态的进程被挂起，则进程状态转换为挂起阻塞状态；如果挂起阻塞状态的进程被恢复/激活，则进程状态转换为阻塞状态；如果挂起阻塞状态进程的阻塞事件或I/O请求完成，则进程状态转换为挂起就绪状态，但仍然是挂起状态。

4. 进程的控制

- 阻塞与唤醒进程：
    - 进程的阻塞是指正在执行的进程，由于期待的某个事件未能发生，则由操作系统执行阻塞原语 (Block)，使自己由运行态变为阻塞态。
    - 进程的唤醒是指当被阻塞进程所期待的事件出现时，则由有关进程（如查找或检测到该事件的进程）调用唤醒原语 (Wakeup)，将等待该事件的进程唤醒。
    - (具体实现在教材P38-P39，涉及原语操作)
- 挂起与激活进程：
    - 进程的挂起是指当出现挂起原语 (Suspend) 调用时，操作系统利用挂起原语将指定的进程或处于某种状态的一批进程挂起。
    - 进程的激活是指当发生激活原语 (Activate) 调用时，操作系统利用激活原语将指定的进程激活。
    - (具体实 Waren 在教材P39，涉及原语操作)

5. 进程的互斥与同步

- 临界资源与临界区（掌握）
    - 临界资源 **(Critical Resource)**：在一段时间内只允许一个进程访问的资源。[1] (教材中此部分在P41)
    - 临界区 **(Critical Section)**：在每个进程中访问临界资源的那段代码。 (教材中此部分在P41)

6. 进程同步机制

- 信号量（含义 ）
    - 信号量 (Semaphore) 是一种用于实现进程同步与互斥的机制。它是一个整型变量，除了初始化外，只能通过两个标准的原子操作P操作和V操作来访问。 (教材中此部分在P45)
- **P**操作与**V**操作（做了什么）(参考内容：p45-46)
    - **P**操作 **(wait**操作**)**：

1. S := S - 1
2. 如果 S < 0，则调用P操作的进程被阻塞，并将其加入该信号量的等待队列中。

- **V操作 (signal操作)**：
  1. S := S + 1
  2. 如果 S ≤ 0，则从该信号量的等待队列中唤醒一个进程，使其进入就绪队列。
- (教材P45-46详细描述了记录型信号量的P、V操作定义)

7. 进程同步经典问题

- 进程控制 (此考点在教材中不明确，通常指进程的创建、终止、阻塞、唤醒、挂起、激活等操作，已在"进程的控制"部分涉及)
- 多生产者多消费者 (教材P46-P48)
  - 问题描述：一组生产者进程生产产品并将它们放入一个有界缓冲区中，一组消费者进程从缓冲区中取出产品消费。需要保证生产者不会在缓冲区满时放入产品，消费者不会在缓冲区空时取出产品，且对缓冲区的访问是互斥的。
  - 解决思路：使用三个信号量，一个互斥信号量mutex用于互斥访问缓冲区，一个资源信号量empty表示缓冲区中空闲单元的数量，一个资源信号量full表示缓冲区中产品的数量。
- 读者写者 (教材P48-P50)
  - 问题描述：允许多个读者同时读一个共享对象；只允许一个写者写共享对象；不允许读者和写者同时访问共享对象。
  - 解决思路：通常有两种策略，读者优先或写者优先。需要信号量来控制读写互斥以及读者计数。

8. 进程调度

- 三级模型：
  - 高级调度 **(**作业调度**)**：又称长程调度，其主要功能是根据某种算法，决定将外存上处于后备队列中的哪几个作业调入内存，为它们创建进程、分配必要的资源，并将它们放入就绪队列。(教材P56)
  - 中级调度 **(**内存调度**/**交换调度**)**：又称中程调度，其主要目的是提高内存利用率和系统吞吐量。它决定将内存中暂时不能运行的进程，或优先级低的进程，调至外存等待，把进程状态变为挂起状态。当这些进程具备运行条件且内存稍有空闲时，由中级调度决定将外存上先前被挂起的、具备运行条件的就绪进程重新调入内存，并修改其状态为就绪状态，挂在就绪队列上等待。(教材P56)
  - 低级调度 **(**进程调度**/CPU**调度**)**：又称短程调度，其主要功能是根据某种算法，决定就绪队列中的哪个进程应获得处理机，并将处理机分配给它。(教材P56)
- 调度算法：(教材P58-P63)
  - 先来先服务 **(FCFS)**：按照进程到达就绪队列的先后次序选择进程投入运行。
  - 短作业优先 **(SJF)**：从就绪队列中选择一个或几个估计运行时间最短的作业，将它们调入内存运行。(非抢占式)

    - 最短剩余时间优先 **(SRTF)**：SJF的抢占式版本。当一个新作业到达时，如果其整个执行时间比当前正在执行的作业的剩余执行时间还短，则SRTF算法会抢占当前作业，运行新作业。
    - 高响应比优先 **(HRRN)**：为每个作业引入动态优先级，优先级随等待时间延长而增加，响应比 = (等待时间 + 要求服务时间) / 要求服务时间。
    - 优先权调度：为每个进程确定一个优先权，每次调度时，总是选择就绪队列中具有最高优先权的进程投入运行。可以是抢占式或非抢占式，优先权可以是静态的或动态的。
- 关键指标：运行时间、平均时间
    - 周转时间：是指从作业被提交给系统开始，到作业完成为止的这段时间间隔。
    - 平均周转时间：多个作业周转时间的平均值。
    - 带权周转时间：作业的周转时间 T 与其服务时间 Ts 之比，W = T / Ts。
    - 平均带权周转时间：多个作业带权周转时间的平均值。
    - 等待时间：进程在就绪队列中等待所花费的时间之和。
    - 平均等待时间：多个进程等待时间的平均值。
    - 响应时间：从用户通过键盘提交一个请求开始，直至系统首次产生响应所需的时间。
    - (教材P57-P58 调度算法的选择/评价准则中提及)

9. 死锁

- 什么是死锁？
    - 死锁是指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。 (教材P65)
- 四个必要条件：(教材P66)
    1. 互斥条件：进程对所分配到的资源进行排他性使用，即在一段时间内某资源只由一个进程占用。
    2. 请求与保持条件：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用，此时请求进程阻塞，但对自己已获得的资源保持不放。
    3. 不剥夺条件：进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。
    4. 环路等待条件：在发生死锁时，必然存在一个进程—资源的环形链，即进程集合{P0, P1, P2, ..., Pn}中的P0正在等待一个P1占用的资源；P1正在等待P2占用的资源，……，Pn正在等待已被P0占用的资源。
- 银行家算法（过程）(参考内容：p69，教材P67-P69)
    - 目的：避免死锁。
    - 数据结构：
        - Available: 可用资源向量。
        - Max: 最大需求矩阵。
        - Allocation: 已分配矩阵。

        - Need: 需求矩阵 (Need[i, j] = Max[i, j] - Allocation[i, j])。
    - 算法过程：当进程Pi发出资源请求向量Request i 时，系统按下述步骤进行检查：
        1. 如果 Request i [j] ≤ Need[i, j]，则转向步骤2；否则认为出错，因为它所需要的资源数已超过它所宣布的最大值。
        2. 如果 Request i [j] ≤ Available[j]，则转向步骤3；否则，表示尚无足够资源，Pi须等待。
        3. 系统试探着把资源分配给进程Pi，并修改下面数据结构中的数值：Available[j] = Available[j] - Request i [j]; Allocation[i, j] = Allocation[i, j] + Request i [j]; Need[i, j] = Need[i, j] - Request i [j];
        4. 系统执行安全性算法，检查此次资源分配后系统是否处于安全状态。若安全，才[1]正式将资源分配给进程Pi，以完成本次分配；否则，将本次的试探分配作废，恢复原来的资源分配状态，让进程Pi等待。
    - [2]安全性算法：
        1. 设置两个向量：工作向量Work，初始时Work = Available；Finish，表示系统是否有足够的资源分配给进程，使之运行完成。开始时Finish[i] = false。
        2. 从进程集合中找到一个能满足下述条件的进程：Finish[i] = false；Need[i, j] ≤ Work[j]。若找到，执行步骤3；否则，执行步骤4。
        3. 当进程Pi获得资源后，可顺利执行，直至完成，并释放出分配给它的资源，故应执行：Work[j] = Work[j] + Allocation[i, j]；Finish[i] = true；go to step 2。
        4. 如果所有进程的Finish[i]都为true，则表示系统处于安全状态；否则，系统处于不安全状态。

第三章 内存管理

1. 覆盖技术与交换技术

- 比较：**p101** (教材P100-P101)
    - 覆盖技术 **(Overlay)**：基本思想是把一个大程序划分为一系列的覆盖，每个覆盖是程序的一部分。程序执行时，先把一部分覆盖调入内存，当需要另一覆盖时，再从外存调入，并覆盖掉内存中不再需要的覆盖。
        - 优点：可以在较小的内存空间运行较大的程序。
        - 缺点：编程复杂，增加了程序员的负担；覆盖的调入调出花费时间。
    - 交换技术 **(Swapping)**：把内存中暂时不能运行的进程或者暂时不用的程序和数据，调到外存上，以便腾出足够的内存空间，再把已具备运行条件的进程或进程所需要的程序和数据调入内存。
        - 优点：增加了并发运行的进程数目，提高了内存利用率。
        - 缺点：交换操作需要较多的时间，影响系统效率。
    - 主要区别：
        - 覆盖技术主要用于解决单个大程序在小内存中运行的问题，发生在同一进程或作业内部。

    - 交换技术主要用于解决多个进程竞争内存的问题，发生在不同进程之间。
    - 覆盖技术对用户不透明，需要程序员指明覆盖结构；交换技术对用户透明，由操作系统自动完成。

2. 可变分区内存管理

- 参考内容：**p106-107** (教材P105-P108)
    - 可变分区管理是一种动态分区分配方法，即在作业装入内存时，根据作业的实际需要，从可用的空闲内存中分割出一个连续的区域分配给该作业。当作业运行结束后，系统收回它所占用的内存区域，并将其与相邻的空闲区合并。
- 内存分配方法：(教材P106-107)
    - 最先适应分配算法 **(First Fit, FF)**：从空闲分区表的第一个表目起查找，直至找到第一个能满足其大小要求的空闲分区为止。然后再按照作业的大小，从该分区中划出一块内存空间分配给请求者，余下的空闲分区仍留在空闲分区表中。
    - 循环首次适应分配算法 **(Next Fit, NF)**：由最先适应算法演变而来。在为进程分配内存空间时，不再是每次都从空闲分区表的第一个表目起查找，而是从上次找到的空闲分区的下一个空闲分区起查找，直至找到第一个能满足其大小要求的空闲分区为止。
    - 最优适应分配算法 **(Best Fit, BF)**：总是把既能满足要求，又是最小的空闲分区分配给作业。为了实现该算法，通常将所有空闲分区按其大小递增的次序链接。
    - 最坏适应分配算法 **(Worst Fit, WF)**：与最优适应算法相反，总是挑选一个最大的空闲分区分割给作业使用。为了实现该算法，通常将所有空闲分区按其大小递减的次序链接。
    - 快速适应分配算法 **(Quick Fit)**：将空闲分区分成若干个队列，每个队列中的分区大小相同。当需要分配内存时，根据所需分区的大小，查找相应队列，从中取出一个分区分配。

3. 页式存储管理

- 逻辑地址到物理地址的转换 (教材P111)
    - 页式存储管理中，逻辑地址由页号P和页内偏移量W组成。
    - 地址转换过程：
        1. 用逻辑地址中的页号P，去检索页表。
        2. 找到该页号对应的页表项，从中读出该页在内存中的物理块号B。
        3. 物理地址 = 物理块号B * 页面大小 + 页内偏移量W。
        4. 访问内存前，会进行地址越界检查（页号是否超出页表长度，页内偏移是否超出页面大小）。
- 多级页表 (教材P113-P114)
    - 为了解决单级页表过大，需要占用大量连续内存空间的问题，引入了多级页表。
    - 将页表本身进行分页，即把页表中的若干个页表项作为一组，构成一个页表页，再为这些页表页建立上一级页表，称为外层页表或页目录表。外层页表的每个条目指

向一个页表页。
    - 两级页表的地址转换：逻辑地址分为外层页号、内层页号和页内偏移量。先用外层页号查找外层页表，得到内层页表的起始地址；再用内层页号查找内层页表，得到物理块号；最后用物理块号和页内偏移量形成物理地址。可以有多于两级的页表。

4. 段式存储管理

- 分段与分页的比较 (教材P117)
    - 目的不同：分页主要是为了提高内存利用率，满足操作系统内存管理的需求，是系统行为，对用户不可见；分段主要是为了满足用户的模块化程序设计需求，方便编程、共享和保护，是用户行为。
    - 单位大小不同：页的大小是固定的，由系统决定；段的长度是不固定的，由用户定义的逻辑段的实际长度决定。
    - 地址空间维度不同：分页的用户程序地址空间是一维的，即单个线性地址空间；分段的用户程序地址空间是二维的，程序员通过段名（段号）和段内地址来访问。
    - 产生碎片不同：页式存储管理会产生内部碎片（最后一个物理块可能未被完全利用）；段式存储管理会产生外部碎片（由于段长不固定，回收后可能产生不连续的小空闲区）。
    - 共享和保护：分段比分页更容易实现共享和保护。可以按逻辑意义上的段进行共享和设置访问权限。

5. 虚拟存储技术（**p118**）

- 请求页式虚拟存储管理（缺页中断次数，缺页率）(教材P120-P124)
    - 基本原理：程序装入时，不必将其全部页面都装入内存，而只将当前需要的一部分页面装入内存便可启动运行。在程序运行过程中，如果发现要访问的页面不在内存（产生缺页），则由操作系统将所需的页面调入内存。
    - 缺页中断：当访问的页面不在内存时，会产生一个缺页中断。操作系统响应中断，执行缺页中断处理程序，将所需的页面从外存调入内存。如果内存已满，还需要进行页面置换。
    - 缺页中断次数：程序运行过程中发生缺页中断的总次数。
    - 缺页率：缺页中断次数 / 总的页面访问次数。缺页率直接影响虚拟存储系统的性能。
- 页面置换算法：(教材P127-P130)
    - 先进先出 **(FIFO)**：选择在内存中驻留时间最久的页面予以淘汰。实现简单，但可能淘汰掉经常访问的页面（Belady异常）。
    - 最佳页面置换 **(OPT)**：选择将来最长时间内不再被访问的页面予以淘汰。这是一种理想算法，无法实际实现，但可以作为评价其他算法的标准。
    - 最近最久未使用 **(LRU)**：选择最近一段时间内最久未被访问过的页面予以淘汰。它认为过去最久未使用的页面，在将来也是最久不会使用的。性能较好，接近OPT，但实现开销较大。

- 页面调入策略与页面分配策略（关联）
  - 页面调入策略 (教材P124)：决定何时将一个页面从外存调入内存。
    - 请求调页 **(Demand Paging)**：当进程运行中需要访问某个不在内存的页面时，才将其调入。
    - 预调页 **(Prepaging)**：预计进程在运行前或运行中不久将要访问某些页面，就预先将它们调入内存。
  - 页面分配策略 (教材P125)：决定给一个进程分配多少个物理块。
    - 固定分配策略：为每个进程分配一组固定数目的物理块，在进程运行期间不再改变。
    - 可变分配策略：先为每个进程分配一定数目的物理块，在进程运行期间，可根据情况适当增加或减少。
  - 关联：页面调入策略和页面分配策略共同影响着虚拟存储系统的性能。例如，如果采用请求调页，当分配给进程的物理块较少时，缺页率可能会很高。预调页可以尝试减少缺页中断，但如果预调的页面不被访问则会浪费内存。可变分配策略可以根据进程的实际缺页情况动态调整分配的物理块数，以期达到较好的性能。

第四章 设备管理

1. 设备的分类

- 按服务功能分类 (教材P151)
  - 存储设备：如磁盘、磁带、光盘。
  - 输入设备：如键盘、鼠标、扫描仪。
  - 输出设备：如显示器、打印机。
  - 网络通信设备：如网卡、调制解调器。
- 按每次信息交换的单位分类 (教材P151)
  - 块设备：以数据块为单位进行信息交换，如磁盘。
  - 字符设备：以字符为单位进行信息交换，如键盘、打印机。
- 按使用特征分类 (教材P151)
  - 独占设备：在一段时间内只允许一个进程访问的设备，如打印机。
  - 共享设备：在一段时间内允许多个进程同时（宏观上）访问的设备，如磁盘。
  - 虚拟设备：通过虚拟技术将独占设备改造成共享设备，如SPOOLing技术中的虚拟打印机。

2. 四种设备的控制方法（输入输出方式）

(教材P154-158)

- 程序循环查询方式 **(Programmed I/O)**：CPU不断地循环检测I/O设备的状态，看数据是否准备好。CPU在整个I/O过程中都处于忙等待状态，效率低。
- 中断驱动方式 **(Interrupt-driven I/O)**：当I/O设备完成数据传输或发生错误时，向CPU发出中断信号，CPU响应该中断，转去执行中断处理程序。CPU在I/O设备工作期

间可以执行其他任务，提高了CPU利用率。

- 直接内存访问 **(DMA)** 方式 **(Direct Memory Access)**：在I/O设备和内存之间开辟直接的数据通路。DMA控制器接管数据传输过程，数据直接在内存和I/O设备之间传送，无需CPU介入（除启动和结束时），大大减轻了CPU的负担。
- 通道方式 **(Channel)**：引入通道处理器，专门负责I/O操作的组织和管理。CPU只需向通道发出I/O指令，通道便会执行一系列的通道程序来完成复杂的I/O操作。进一步解放了CPU，适用于大量数据的高速I/O。

3. SPOOLing 技术

- 假脱机技术 (教材P168-170)
  - SPOOLing (Simultaneous Peripheral Operations On-Line) 技术，即外部设备联机并行操作技术，也称为假脱机技术。
  - 它是在多道程序环境下，利用一道程序（SPOOLing系统）来模拟脱机输入输出功能。它将独占设备（如打印机）虚拟成共享设备，允许每个用户进程都认为自己独占了该设备。
- 输入井与输出井（基本原理、工作原理）(教材P169)
  - 基本原理：在磁盘上开辟出称为“输入井”和“输出井”的存储区域。
  - 工作原理：
    - 输入过程：输入进程将用户要求输入的数据从输入设备（如键盘）预先读入磁盘上的输入井。当某个进程需要输入数据时，直接从输入井中读取。
    - 输出过程：当用户进程要求输出数据时，SPOOLing系统并不立即把数据输出到输出设备（如打印机），而是先将数据送到磁盘上的输出井。待输出设备空闲时，输出进程再将输出井中的数据分批输出到指定的输出设备上。
  - 这样，SPOOLing技术通过在磁盘上设置输入井和输出井作为数据交换的缓冲区，实现了将慢速的I/O操作与CPU的高速计算并行起来，提高了设备的利用率和系统的吞吐量。

第五章 文件系统

1. 文件的逻辑结构和物理结构（**p203**）

- 逻辑结构：(教材P203-206)
  - 指从用户观点出发所观察到的文件组织形式，是用户可以直接处理的数据结构。
  - 流式结构 **(Stream File)**：文件由一串有序的字节流组成，没有明显的记录界限。如文本文件、源程序文件。
  - 记录式结构 **(Record-based File)**：文件由若干逻辑记录组成。
    - 顺序式 **(Sequential Record)**：文件中的记录按其逻辑顺序依次排列。
    - 索引式 **(Indexed Record)**：为文件建立索引表，索引表中的每一项对应一个记录，并指出该记录在文件中的位置。可以通过索引快速查找记录。
    - 索引顺序式 **(Indexed Sequential Record)**：将顺序文件和索引文件结合起

来。文件中的记录分组，组内记录顺序存放，组间通过索引访问。

- 物理结构（区别与优缺点）：(教材P207-213)
  - 指文件在辅存（如磁盘 ）上的存放方式和组织关系。
  - 连续文件 **(Contiguous Allocation)**：为每个文件分配一组连续的磁盘块。
    - 优点：支持顺序存取和直接存取，速度快。
    - 缺点：容易产生外部碎片，文件不易扩展。
  - 链接文件 **(Linked Allocation)**：每个文件占用的磁盘块可以不连续，通过指针将属于同一个文件的磁盘块链接起来。
    - 优点：消除了外部碎片，文件易于扩展。
    - 缺点：只支持顺序存取，可靠性差（指针丢失或损坏会导致文件部分或全部丢失），访问速度慢。
  - 索引文件 **(Indexed Allocation)**：为每个文件建立一个索引块（或索引表），索引块中存放指向该文件所有数据块的指针。
    - 优点：消除了外部碎片，支持直接存取，文件易于扩展。
    - 缺点：索引块本身需要占用存储空间，对于小文件开销较大。
  - 多级索引文件（计算：**p212**）(教材P211-212)
    - 当文件较大，一个索引块不足以存放所有数据块指针时，采用多级索引。将索引块再进行索引，形成多级索引结构。
    - 计算示例参考教材P212。例如，若一个磁盘块大小为1KB，每个地址项占4B，则一个索引块可存放256个地址。若采用一级索引，文件最大为256KB。若采用二级索引，第一级索引块存放256个指向第二级索引块的地址，每个第二级索引块存放256个指向数据块的地址，则文件最大为256 * 256KB = 64MB。
  - 直接文件 (教材中未明确将“直接文件”作为一种独立的物理结构类型与上述几种并列，通常“直接存取”是一种访问方式，可以通过连续分配或索引分配实现。部分教材可能将哈希文件等视为直接文件的一种物理结构实现。)

2. 文件的目录

- 文件目录的基本概念 (教材P216)
  - 文件目录是为了实现对文件“按名存取”而设置的一种数据结构，它存储了文件名及其对应的各种属性信息（如物理地址、创建时间、访问权限等）。
  - 目录本身也常以文件形式存放在磁盘上，称为目录文件。
- 文件控制块的作用与控制方法 (教材P216)
  - 文件控制块 **(File Control Block, FCB)**：存放控制文件所需的各种信息的数据结构，是文件存在的标志。目录项就是文件控制块的集合。
  - 作用：FCB中包含了文件的描述信息和存取控制信息，操作系统通过FCB来对文件进行管理和控制。
  - 控制方法：通过对FCB的查找、读取和修改，实现对文件的打开、读写、关闭、属性修改等操作。

3. 文件的访问方式

(教材P213-214)

- 顺序存取 **(Sequential Access)**：按字节或记录的逻辑顺序依次访问文件内容。这是最简单的访问方式，也是多数文件系统都支持的方式。
- 直接存取 **(Direct Access / Random Access)**：允许按任意顺序读写文件中的记录或字节，而无需经过前面的内容。通常需要提供记录号或字节偏移量。
- 索引存取 **(Indexed Access)**：通过索引来访问文件中的记录。先查找索引，根据索引项中的指针直接定位到记录。

4. 文件存储空间的管理方法（**p214**）

(教材P214-216)

- 磁盘空间的分配：
  - 连续分配 **(Contiguous Allocation)**：为每个文件分配一组连续的磁盘块。（已在文件物理结构中提及）
  - 非连续分配 **(Non-contiguous Allocation)**：允许文件存放在不连续的磁盘块中。主要有链接分配和索引分配。(已在文件物理结构中提及)
- 磁盘空闲空间的有效管理：
  - 空闲区表法 **(Free List / Free Block List)**：将所有空闲的磁盘块记录在一个表中，每个表项记录一个空闲区的起始块号和块数（或结束块号）。适用于连续分配策略。
  - 空闲块链表法 **(Free Block Linked List)**：将所有空闲的磁盘块链接成一个或多个链表。
    - 空闲块链：把所有空闲块链在一起。
    - 空闲盘区链：把所有空闲盘区（连续的空闲块 ）链在一起，每个链结点记录空闲盘区的第一个盘块号和该盘区中的盘块数。
    - (教材中还提到了位示图法 **(Bitmap / Bit Vector)**：用一位二进制来对应一个磁盘块，0表示空闲，1表示已分配。以及成组链接法。)