**作业3：操作系统主要管理计算机哪些事务？总结其中涉及分配调度的事务，并简介其中算法；总结并介绍涉及虚拟的事务；总结及介绍涉及中间件的事务。**

操作系统主要管理的是计算机中硬件和软件资源。

涉及分配调度的事务有：进程分配调度，处理机分配调度，存储器分配调度，设备调度，磁盘调度，文件调度，接口调度等。

算法：

先来先服务算法：系统按照作业到达的先后次序来进行调度，或者说它优先考虑在系统中等待时间最长的作业，而不管该作业所需执行时间的长短，从后备作业队列中选择几个最先进入该队列的作业，将它们调入内存，为它们分配资源和创建进程。然后把它放入就绪队列。

当进程调度中才有FCFS算法时，每次调度是从就绪的进程队列中选择一个最先进入该队列的进程，为之分配处理机，使之投入运行。该进程一直运行到完成或发生某事件而阻塞后，进程调度程序才将处理机分配给其他进程。

短作业优先算法：SJF算法是以作业的长短来计算优先级，作业越短，其优先级越高。作业的长短是以作业所要求的运行时间来衡量的。SJF算法可以分别用于作业调度和进程调度。在把短作业优先调度算法用于作业调度时，它将从外存的作业后备队列中选择若干个估计运行时间最短的作业，优先将它们调入内存运行。

优先级调度算法：优先级调度算法是一种常见的进程调度算法，它将进程按照优先级进行调度，优先级越高的进程越先被调度。在优先级调度算法中，每个进程都被赋予一个优先级值，优先级值越高的进程越先被调度执行。

高相应比优先调度算法：高相应比优先调度算法是一种优化的优先级调度算法，在传统的优先级调度算法的基础上增加了相应比的概念。相应比是指进程的等待时间与执行时间之比，它反映了进程等待时间与执行时间的比例。相应比越高的进程，优先级越高，越先被调度执行。

轮转调度算法：轮转调度算法是一种简单的进程调度算法，它将进程按照顺序分配时间片，每个进程在分配到的时间片内运行。当时间片用完后，如果进程还没有执行完，则将该进程放到就绪队列的末尾，然后调度下一个进程执行。这个过程一直循环执行，直到所有进程都完成执行。

轮转调度算法的优点是能够公平地分配 CPU 时间，避免长时间运行的进程占用 CPU 资源，从而提高系统的响应速度。但是该算法也存在一些缺点，如时间片大小的设置可能影响系统的性能，同时也容易产生进程上下文切换的开销。

多队列算法：多队列算法是一种进程调度算法，它将就绪队列分成多个队列，每个队列有不同的优先级。通常情况下，优先级高的队列分配的时间片较短，而优先级低的队列分配的时间片较长。每个队列按照轮转调度的方式来执行，当一个队列中的进程执行完毕后，如果没有更高优先级的进程需要执行，则调度下一个队列中的进程执行。

多队列算法的优点是能够根据进程的优先级分配不同的时间片，避免长时间运行的进程占用 CPU 资源，同时也能够充分利用 CPU 资源，提高系统的吞吐量和响应速度。但是该算法也存在一些缺点，如队列的数量和优先级的设置可能影响系统的性能，同时也容易产生进程上下文切换的开销。

多级反馈队列调度算法：多级反馈队列调度算法既能使高优先级的作业得到响应又能使短作业(进程)迅速完成。

基于公平原则的调度算法：

最早截至时间优先算法：最早截至时间优先算法是一种进程调度算法，它将进程按照截至时间进行调度，优先调度截至时间最早的进程。在该算法中，每个进程被分配一个截至时间，当截至时间到达后，进程被强制中断执行。

最低松弛度算法：最低松弛度算法是一种进程调度算法，它将进程按照松弛度进行调度，优先调度松弛度最低的进程。在该算法中，每个进程被分配一个松弛度，松弛度越低的进程优先级越高。

银行家算法：把操作系统看作是银行家，操作系统管理的资源相当于银行家管理的资金，进程向操作系统请求分配资源相当于用户向银行家贷款。操作系统按照银行家制定的规则为进程分配资源，当进程首次申请资源时，要测试该进程对资源的最大需求量，如果系统现存的资源可以满足它的最大需求量则按当前的申请量分配资源，否则就推迟分配。当进程在执行中继续申请资源时，先测试该进程已占用的资源数与本次申请的资源数之和是否超过了该进程对资源的最大需求量。若超过则拒绝分配资源，若没有超过则再测试系统现存的资源能否满足该进程尚需的最大资源量，若能满足则按当前的申请量分配资源，否则也要推迟分配。

FF：首次适应算法是一种磁盘调度算法，它将磁盘上的空闲区域按照地址顺序进行排列，当出现新的磁盘读写请求时，从磁盘的起始位置开始查找，找到第一个能够满足请求的空闲区域，然后将请求分配到该区域中。该算法的优点是简单、快速，但可能会产生大量的碎片。

NF：循环首次适应算法是一种磁盘调度算法，它是首次适应算法的改进版。它将磁盘上的空闲区域按照地址顺序进行排列，当出现新的磁盘读写请求时，从上一个请求位置开始查找，找到第一个能够满足请求的空闲区域，然后将请求分配到该区域中。该算法可以减少碎片，但是可能会产生不均衡的分配问题。

BF：最佳适应算法是一种磁盘调度算法，它将磁盘上的空闲区域按照大小进行排序，当出现新的磁盘读写请求时，从空闲区域中找到能够满足请求的最小空闲区域，然后将请求分配到该区域中。该算法可以减少碎片，但是可能会产生大量的搜索开销。

WF：最坏适应算法是一种磁盘调度算法，它将磁盘上的空闲区域按照大小进行排序，当出现新的磁盘读写请求时，从空闲区域中找到能够满足请求的最大空闲区域，然后将请求分配到该区域中。该算法可以减少搜索开销，但是可能会产生大量的碎片。

快速适应算法：快速适应算法是一种动态内存分配算法，它是基于分离式空闲链表实现的。该算法将内存按照大小划分成多个空闲链表，每个链表对应一定范围内的内存大小。当需要分配内存时，算法会先在相应的链表中查找是否有合适大小的空闲内存块，如果有，则直接分配；否则，算法会向更大的链表中查找。当释放内存时，算法会将这块内存加入到对应的空闲链表中。快速适应算法的优点在于它能够快速地分配内存，同时保证了内存的利用率。但是，由于每个链表的大小范围固定，可能会产生大量的内部碎片。

哈希算法：哈希算法是一种将任意大小的数据映射为固定大小的数据的算法。在计算机科学中，哈希算法通常用于实现数据的快速查找。哈希算法的基本思想是将数据通过一个哈希函数映射到一个固定大小的数组中，然后在数组中查找对应的数据。

最佳置换算法：最佳置换算法是一种页面置换算法，它是一种理论上最优的算法。该算法的基本思想是在每次需要置换页面时，选择距离下一次使用时间最远的页面进行置换。这样可以最大程度地减少缺页率，但是算法需要知道未来的页面使用情况，因此在实际应用中很难实现。

LRU和LFU：最近最少使用淘汰算法（Least Recently Used）。LRU是淘汰最长时间没有被使用的页面。

LRU关键是看数据最后一次被使用到发生替换的时间长短，时间越长，数据就会被置换；最不经常使用淘汰算法（Least Frequently Used）。LFU是淘汰一段时间内，使用次数最少的页面。

Clock置换算法：Clock 置换算法是一种页面置换算法，它是时钟算法的变种。该算法使用一个环形链表来管理页面，每个页面都有一个访问位和一个修改位。当需要置换页面时，在环形链表上顺时针扫描页面，如果当前页面的访问位为 0，则将该页面作为置换目标；如果当前页面的访问位为 1，则将该页面的访问位设置为 0，并继续扫描。当扫描一周后，如果没有找到可置换的页面，则重新开始扫描。

页面缓冲算法：页面缓冲算法是一种用于管理缓存的算法，它可以提高数据访问的效率。在页面缓冲算法中，缓存被划分为多个页面，每个页面可以缓存一个数据块。当需要访问数据时，算法会先在缓存中查找，如果找到则直接返回；否则，算法会从磁盘中读取数据，并将数据块存储到缓存中。常见的页面缓冲算法包括最近最少使用算法、最不经常使用算法和先进先出算法等。这些算法使用不同的策略来决定哪些数据块应该被缓存，从而提高数据访问的效率。

最短寻道时间优先：最短寻道时间优先是一种磁盘调度算法，它的基本思想是优先调度离当前磁道最近的请求。在该算法中，磁盘头会沿着磁盘的某个方向移动，当出现新的磁盘读写请求时，算法会计算当前磁盘头与请求磁道之间的距离，然后根据距离排序，优先调度距离最短的请求。

扫描算法：扫描算法是一种磁盘调度算法，它的基本思想是按照一个方向，将磁盘上的所有读写请求都处理完毕后，再返回到起始位置，重新开始处理请求。在该算法中，磁盘头会沿着磁盘的某个方向移动，当出现新的磁盘读写请求时，算法会将请求按照磁道号排序，然后依次处理请求。当磁盘头到达磁盘的终点时，算法会将磁盘头方向反转，然后重新开始处理请求。

循环扫描算法：循环扫描算法是一种扫描算法的变种，它的基本思想是在扫描过程中始终保持一个方向，不反转磁盘头的方向。当磁盘头到达磁盘的终点时，算法会将磁盘头移动到起始位置，然后重新开始处理请求。

涉及虚拟的任务：

假脱机技术（SPOOLLING）：在假脱机技术中，操作系统会创建一个虚拟机器或容器，该虚拟机器或容器与主机系统共享资源，例如内存和CPU。在执行假脱机任务时，操作系统会将所有I/O请求重定向到虚拟机器或容器中，而不是直接与主机系统交互。这样，假脱机任务就可以在虚拟机器或容器中执行，而不会影响主机系统的运行。

使用虚拟化技术可以使假脱机技术更加灵活和可靠。例如，如果假脱机任务需要访问特定的硬件资源，操作系统可以使用虚拟化来模拟这些资源，而不需要实际存在这些资源。这样，假脱机任务就可以在任何计算机上执行，而不需要考虑硬件差异。

涉及中间件的任务：

IO系统中模块

在文件管理中，中间件事务是指位于操作系统和文件系统之间的软件层，用于提供文件管理的功能和接口。以下是文件管理的中间件事务的一些示例：

文件系统接口：中间件事务可以提供文件系统的抽象接口，使应用程序可以使用统一的方式来操作不同类型的文件系统。它封装了底层文件系统的细节，提供了一致的文件访问接口，使开发人员能够以更高层次的抽象来处理文件操作。

文件缓存管理：中间件事务可以实现文件缓存管理，将常用的文件数据缓存在内存中，提高文件访问的速度和效率。它可以根据文件的使用模式和访问频率，进行缓存策略的管理，如LRU（最近最少使用）算法或LFU（最不经常使用）算法。

文件索引和搜索：中间件事务可以提供文件索引和搜索功能，加快文件的查找和检索过程。它可以建立索引结构，记录文件的元数据和位置信息，使得应用程序可以通过关键字或属性进行文件的快速搜索和定位。

文件传输和同步：中间件事务可以实现文件传输和同步功能，将文件从一个位置或系统传输到另一个位置或系统，并确保文件的一致性和完整性。它可以处理文件传输的错误处理、重试机制和同步策略，以保证文件的安全传输和正确性。

文件权限和访问控制：中间件事务可以提供文件权限和访问控制的管理功能，定义文件的访问权限和安全策略。它可以实现用户身份验证、权限验证和文件加密等机制，保护文件的机密性和完整性