****作业一****

****进程引入的重要性和必要性**：**

**引入进程用来给程序提供一个抽象的概念，他能申请道系统资源并且独立给程序提供资源，从而解决原来的程序之间因为资源共享而相互限制的问题。 这样就可以提高系统资源的利用率以及系统的处理能力**

**总结有三点：提高资源利用率，实现并发操作，使用户有良好的体验**

****保护进程安全高效的工作：****

****同步：**在多道程序系统中，引入进程同步机制可以保证程序执行的可在现性。进程同步机制的主要任务，是对多个相关进程在执行上进行协调，使并发执行的诸程序之间能按照一定的规则（或时序）共享系统资源，并能很好地相互合作，从而使程序的执行具有可再现性。**通过正确使用同步机制，可以避免多个进程同时访问共享资源导致的数据冲突和错误。

**互斥：**互斥是指在同一时刻只允许一个进程访问共享资源。通过引入互斥机制，如互斥锁，可以确保多个进程不会同时访问共享资源，从而避免数据的不一致性和冲突。互斥锁是一种常见的机制，它确保同一时间只有一个进程可以访问共享资源，其他进程需要等待。通过正确使用互斥机制，可以避免竞态条件和数据的错误修改。

**死锁：**死锁的起因，通常是源于多个进程对资源的争夺，不仅对不可抢占资源进行争夺时会引起死锁，而且对可消耗资源进行争夺时，也会引起死锁。处理死锁的方法有，预防死锁，避免死锁，检测死锁，解除死锁。这四种方法对死锁的防范程度逐渐减弱，但对应的是资源利用率的提高。这些策略可以根据系统的特点和需求来选择和实施，以保证进程的高效安全工作。

**作业二**

**内存分配管理方式：**

**1.连续分配存储管理**

* + 1. 单一连续分区分配
    2. 固定分区分区分配
    3. 动态分区分区分配
    4. 动态可重定位分区分配
  1. **离散分配存储管理** 
     1. 分页存储管理
     2. 分段存储管理
     3. 段页式存储管理

**如何提高内存的使用效率：**

1. 采用合适的算法优化，以最大限度地减少内存碎片问题和提高内存利用率。
2. .内存复用机制：及时回收已经释放的内存空间，防止出现内存碎片。可以采用内存紧缩、合并空闲块等方法来提高内存的利用效率，将已经分配的内存对象重复利用，减少内存分配和释放的开销。
3. 采用缓存技术，将频繁使用的数据缓存到内存中，加快访问速度。
4. 采用虚拟内存技术，将部分不常用的数据存储在硬盘中，以释放内存空间。

**相比于硬盘存储空间分配，内存分配具有以下共性和特性：**

1. **共性：**
2. 都需要进行分配算法设计，可以使用动态分配和静态分配等不同的分配方式。
3. 都存在碎片问题，导致空间利用率下降。硬盘存储空间也存在内部碎片和外部碎片问题。内部碎片是指文件分配的存储空间大于文件实际所需的空间，导致空间浪费；外部碎片是指硬盘上存在一些零散的未分配空间，无法被有效利用。

**2.特性：**

1.内存容量相较于磁盘要小的多，速度也要快得多，且内存可以直接由CPU访问，而磁盘属于IO设备。因此内存分配算法需要更加实时，否则会严重影响效率。硬盘存储空间分配可以容纳更多的数据量，但访问速度较慢，延迟较高。

**作业三**

**操作系统主要管理计算机哪些事务：**

### **1、 控制和管理系统行为：**

操作系统执行基本的任务，比如：识别来自键盘的输入，将输出结果发送给监视器，管理存储设备上的文件和文件夹，控制箱磁盘驱动器和打印机这样的外部设备。操作系统还必须确保同事工作的不同程序和不同用户之间不会互相干扰。此外，还负责安全问题，确保未经授权的用户和程序不能访问这个系统。

### **调度和分配系统资源：**

操作系统负责决定一个程序需要哪些计算机资源（比如：CPU时间，内存空间，磁盘，输入和输出设备）以及调度和分配这些资源来运行这个程序。

### **调度操作：**

### 操作系统负责调度程序的各种行为以充分利用系统资源。现在很多程序都支持多程序设计、多线程以及多进程以提高系统的性能。

**涉及到的算法：**

# **1.先来先服务([FCFS](https://so.csdn.net/so/search?q=FCFS&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_42570192/article/details/_blank))**

**算法思想**：从公平的角度考虑，类似于排队购物、打饭等。  
**算法规则**：按照作业或者进程到达的先后顺序进行服务。  
**方法用于作业/进程调度**：用于作业调度时，考虑的是哪个**作业**先到达**后备队列**；用于进程调度时，考虑的是哪个**进程**先到达**就绪队列**。  
**进程调度的方式**：非抢占式算法。  
  
**优缺点**：优点是算法公平且实现简单；缺点是排队在长作业或长进程后面的短作业或短进程需要等待很长的时间，带权周转时间很大，对短作业或进程的用户体验不好。所以先来先服务算法对长作业有利，对短作业不利。

# **2**.**最短时间优先(**[SJF](https://so.csdn.net/so/search?q=SJF&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_42570192/article/details/_blank)**)**

最短时间优先(SJF, Shortest Job First)  
**算法思想**：追求最少的平均等待时间，最少的平均周转时间，最少的平均带权周转时间。  
**算法规则**：最短的作业或者进程优先得到服务，这里的最短指的是要求服务的时间最短。  
**方法用于作业/进程调度**：即可用于作业调度，也可以用于进程调度，用于进程调度时称为最短进程优先算法(SPF, Shortest Process First)。  
**进程调度的方式**：最短时间优先和最短进程优先是非抢占式算法，但也有抢占式的算法——最短剩余时间优先算法。  
最短剩余时间优先算法(SRTN, Shortest Remaining Time Next)是每当有进程加入就绪队列发生改变时就需要调度，如果新到达的进程剩余时间比当前运行的进程剩余时间更短，则由新进程抢占处理机，当前运行进程重新回到就绪队列，此外，当一个进程完成时也需要重新调度。

**优缺点**：优点是有较短的平均等待时间和平均周转时间；缺点是不太公平，对短作业有利，对长作业不利，可能产生饥饿现象，此外，作业或者进程的运行时间是由用户提供的，并不一定真实，所以不一定能做到真正的短作业优先。

# **3.优先级调度**

**算法思想**：随着计算机的发展，特别是实时操作系统系统的出现，越来越多的应用场景需要根据任务的紧急程度来决定处理顺序。  
**算法规则**：每个作业或进程有各自的优先级，调度时选择优先级最高的作业或进程。  
**方法用于作业/进程调度**：可用于作业调度，也可用于进程调度，还会用于I/O调度。  
**进程调度的方式**：抢占式和非抢占式都有，区别在于非抢占式只需要在进程主动放弃处理机时进行调度，而抢占式的还需要在就绪队列变化时检查是否会发生抢占。  
**合理地设置各类进程的优先级**：  
①系统进程优先级高于用户进程；  
②前台进程优先级高于后台进程；  
③操作系统更偏好 I/O 型进程(I/O 繁忙型进程)。  
**根据优先级是否可以动态地改变，可以将优先级分为静态优先级和动态优先级两种**。静态优先级在创建进程时就已经确定，之后一直不变；动态优先级创建进程时有一个初始值，之后会根据情况动态地调整优先级。  
**动态优先级的调整时机**：从追求公平、提升资源利用率等角度考虑。如果某进程在就绪队列中等待了很长时间，可以适当提升其优先级；如果某进程占用处理机运行了很长时间，可以适当降低其优先级；如果发现一个进程频繁地进行I/O操作，可以适当提升其优先级。  
**优缺点**：优点是用优先级区分紧急程度、重要程度，适用于实时操作系统，可灵活地调整对各种作业或进程的偏好程度；缺点是若源源不断地有高优先级进程到来，则可能导致饥饿。