操作系统

2018年10月10日　　21:31

第一章 操作系统引论

- ### 什么是操作系统

操作系统	是控制和管理计算机系统内各种硬件和软件资源、有效地组织多道程序运行的系统软件（或程序集合），是用户与计算机之间的接口
	OS是什么　是系统软件（一整套程序组成，如UNIX由上千个模块组成） **管什么**　控制和管理系统资源（记录和调度）

- ### 操作系统的主要功能

存储器管理	内存分配，地址映射，内存保护和内存扩充
处理机管理	作业和进程调度，进程控制和进程通信 ，进程同步
设备管理	缓冲区管理，设备分配，设备驱动和设备无关性
文件管理	文件存储空间的管理，文件操作的一般管理，目录管理，文件的读写管理和存取控制
用户接口	命令界面，程序界面，图形界面

- ### 操作系统地位

操作系统是裸机之上的第一层软件，是建立其他所有软件的基础。它是整个系统的控制管理中心，既管硬件，又管软件，它为其它软件提供运行环境。

- ### 操作系统的发展历程

• 最初是手工操作阶段，需要人工干预，有严重的缺点，此时尚未形成操作系统
• 早期批处理分为联机和脱机两类，其主要区别在与I/O是否受主机控制
• 多道批处理系统中允许多道程序并发执行，与单道批处理系统相比有质的飞跃

- ### 操作系统的基本特征

操作系统基本特征：**并发，共享，虚拟**和**异步性**

并发	并发性是指两个或多个活动在同一给定的时间间隔中进行
共享	共享是指计算机系统中的资源被多个任务所共用
虚拟	通过某种技术将一个物理上的设备变成若干个逻辑上的对应物

- ### 操作系统的主要类型

多道批处理系统

特点	多道、成批
优点	资源利用率高、系统吞吐量大
缺点	等待时间长、没有交互能力

分时系统

分时	指若干并发程序对CPU时间的共享。它是通过系统软件实现的。共享的时间单位称为时间片
特征	同时性：若干用户可同时上机使用计算机系统 交互性：用户能方便地与系统进行人--机对话 独立性：系统中各用户可以彼此独立地操作，互不干扰或破坏 及时性：用户能在很短时间内得到系统的响应
优点	• 响应快，界面友好 • 多用户，便于普及 • 便于资源共享

实时系统

特点	响应时间很快，可以在毫秒甚至微秒级立即处理
典型应用形式	过程控制系统、信息查询系统、事务处理系统
与事务系统区别	见下表

	分时系统	实时系统
交互能力	强(通用系统)	弱（专用系统）
响应时间	秒级	及时，毫秒/微妙级
可靠性	一般要求	要求更高

个人机系统

单用户操作系统	特征：个人使用：整个系统由一个人操纵，使用方便 界面友好：人机交互的方式，图形界面 管理方便：根据用户自己的使用要求，方便的对系统进行管理 适于普及：满足一般的工作需求，价格低廉
多用户操作系统	代表是UNIX，具有更强大的功能和更多优点

网络操作系统

计算机网络	=：计算机技术+通信技术 特征：分布性、自治性、互连性、可见性
本机+网络操作系统	本地OS之上覆盖了网络OS，可以是同构的也可以是异构的。
功能	实现网络通信、资源共享和保护、提供网络服务和网络接口等

分布式操作系统

定义	运行在不具有共享内存的多台计算机上，但用户眼里却像是一台计算机（分布式系统无本地操作系统运行在各个机器上）
特征	分布式处理、模块化结构、利用信息通信、实施整体控制
特点	透明性、灵活性、可靠性、高性能、可扩充性

第二章 进程管理

- ## 程序顺序执行与并发执行比较

顺序执行	并发执行
程序顺序执行	间断执行，多个程序各自在“走走停停”种进行
程序具有封闭性	程序失去封闭性
独享资源	共享资源
具有可在现性	失去可再现性
	有直接和简接的相互制约

- ## 多道程序设计概念及其优点

多道程序设计	在一台计算机上同时运行两个或更多个程序
特点	多个程序共享系统资源、多个程序并发执行
优点	提高资源利用率、增加系统吞吐量

- ## 什么是进程，进程与程序的区别和关系

1. 进程的引入

原因	由于多道程序的特点，程序具有了并行、互斥和动态的特征，就使得原来程序的概念已难以刻划和反映系统中的情况了

2. 进程定义

定义	程序在并发环境下的执行过程。
	进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位。

3. 进程与程序的主要区别

程序是永存的，进程是暂时的
程序是静态的概念，进程是动态的观念
进程由三部分组成 **程序** **数据** **进程控制块（描述进程活动情况的数据结构）（PCB）**

进程会程序不是一一对应的

一个程序可对应多个进程即多个进程可执行统一程序
一个进行可以执行一个或几个程序

4. 进程与程序的对比

程　　序	进　　程
唱歌的曲谱或音乐乐器的乐谱	演出或演奏
剧本	演出
火车	列车

5. 进程特征

动态性、并发性、独立性、异步性、结构性

- 进程的基本状态及其转换

运行态（Running）：	进程正在占用CPU；
就绪态（Ready）：	进程具备运行条件，但尚未占用CPU；
阻塞态（Blocked）	进程由于等待某一事件不能享用CPU。

- 进程是由哪些部分组成, 进程控制块的作用

进程控制块的作用：	进程控制块是进程组成中最关键的部分。
1)	每个进程有唯一的PCB。
2)	操作系统根据PCB对进程实施控制和管理。
3)	进程的动态、并发等特征是利用PCB表现出来的。
4)	PCB是进程存在的唯一标志。

- PCB组织方式

线性队列	**链接表**	**索引表**

- 进程的同步与互斥

同步	是进程间共同完成一项任务时直接发生相互作用的关系。

互斥	排它性访问即竞争同一个物理资源而相互制约。

- 什么是临界资源、临界区

临界资源:	一次仅允许一个进程使用的资源。
临界区:	在每个进程中访问临界资源的那段程序。
互斥进入临界区的**准则**:	• 如果有若干进程要求进入空闲的临界区，一次仅允许一个进程进入。 • 任何时候，处于临界区内的进程不可多于一个。如已有进程进入自己的临界区，则 其它所有试图进入临界区的进程必须等待。 • 进入临界区的进程要在有限时间内退出，以便其它进程能及时进入自己的临界区。 • 如果进程不能进入自己的临界区，则应让出CPU，避免进程出现“忙等”现象。

- 信号量

信号量**定义**	信号量（信号灯）= < 信号量的值，指向PCB的指针 >
信号量的**物理意义**	• 大于 0： 表示当前资源可用数量 • 小于 0： 其绝对值表示等待使用该资源的进程个数 • 信号量初值为非负的整数变量，代表资源数 • 信号量值可变，但仅能由 P 、 V 操作来改变

- P，V操作原语

P操作原语P(S)：	i. P 操作一次，S值减 1 ，即S = S - 1 （请求分配一资源）； ii. 如果S≥ 0 ，则该进程继续执行； 如果 S < 0 表示无资源，则该进程的状态置为阻塞态，把相应的PCB连入该信号量队列的末尾，并放弃处理机，进行等待（直至另一个进程执行V（S）操作）
V操作原语V(S)：	i. V操作一次，S值加1，即S = S + 1 （释放一单位量资源）； ii. 如果S > 0，表示有资源，则该进程继续执行； iii. 如果S≤0，则释放信号量队列上的第一个PCB所对应的进程（阻塞态改为就绪态），执行V操作的进程继续执行

第三章 进程同步与通信

- 进程同步

- 基本概念

一组并发进程执行时存在两种相互制约关系：	• 进程互斥—资源共享关系（间接相互制约关系）：进程本身之间不存在直接联系。处于同一系统中的进程，必然是共享着某种系统资源，如共享CPU、I/O设备 • 进程同步—相互合作关系（直接相互制约关）： 进程本身之间存在着相互制约的关系
临界资源	• 在一段时间内只允许一个进程访问 的资源。诸进程间应采取互斥方式，实现对资源的共享； 共享变量，打印机 等均属于此类资源
临界区	把在每个进程中访问临界资源的那段代码称为临界区(critical section)
进入区	在临界区前面增加一段用于进行临界资源检查的代码，称为进入区 。
退出区	将临界区正被访问的标志恢复为未被访问的标志。
剩余区	其余部分。
互斥算法——使用临界区遵循的原则	• **空闲则入**：其他进程均不处于临界区； • **忙则等待**：已有进程处于其临界区； • **有限等待**：等待进入临界区的进程不能"死等"； • **让权等待**：不能进入临界区的进程，应释放CPU（如转换到阻塞状态）

- 信号量

信号量	**代表可用资源实体的数量。** **整型信号量**：除初始化外，仅能通过两个标准的原子操作wait(S)和signal(S)来访问。这两个操作一直被分别称为P、V操作。 注意：S代表资源个数 **记录型信号量**：是一种不存在“忙等”现象的进程同步机制;由于它采用了记录型的数据结构而得名的 **注意**： • 信号量只能通过初始化和两个标准的原语来访问：作为OS核心代码执行，不受进程调度的打断。 • 初始化指定一个非负整数值，表示空闲资源总数（又称为“资源信号量”）： 若为非负值表示当前的空闲资源数 若为负值其绝对值表示当前等待临界区的进程数。
P（s）	i. S：=S-1 ii. 若S≥0，则调用P(S)的进程继续运行； iii. 若S<0，则调用P(S)的进程被阻塞， 并把它插入到等 待信号量S的阻塞队列中。
V（s）	i. S：=S+1; ii. 若S>0，则调用V(S)的进程继续运行； iii. 若S≤0，从等待信号量S的阻塞队列中唤醒头一个进程， 然后调用V(S)的进程继续运行。
AND同步机制的基本思想	将进程在整个运行过程中需要的所有资源，一次性全部地分配给进程，待进程使用完后再一起释放。 只要尚有一个资源未能分配给进程，其它所有可能为之分配的资

	源，也不分配给他。 解决方法：　对若干个临界资源的分配，采取原子操作方式：要么全部分配到进程，要么一个也不分配。 由死锁理论可知，这样就可避免上述死锁情况的发生。 操作步骤　在wait操作中，增加了一个“AND”条件，故称为AND同步，或称为同时wait操作
比较	**整型信号量**　只有一个资源，只能互斥访问这个资源 **记录型信号量**　只可申请一类资源，该资源有n个，一次只可申请一个 **AND型信号量**　可申请n类资源，每类资源有m个，每次可申请每类资源中的一个 **信号量集**　可申请n类资源，每类资源有m个，每次可申请每类资源中的多个

- 管程

引入	用信号量可实现进程间的同步，但由于信号量的控制分布在整个程序中，其正确性分析很困难。管程是管理进程间同步的机制，它保证进程互斥地访问共享变量，并方便地阻塞和唤醒进程。管程可以函数库的形式实现。相比之下，管程比进程信号量好控制

信号量同步的缺点	**同步操作分散**　信号量机制中，同步操作分散在各个进程中，使用不当就可能导致各进程死锁（如P、V操作的次序错误、重复或遗漏） **易读性差**　要了解对于一组共享变量及信号量的操作是否正确，必须通读整个系统或者并发程序 **不利于修改和维护**　各模块的独立性差，任一组变量或一段代码的修改都可能影响全局 **正确性难以保证**　操作系统或并发程序通常很大，很难保证这样一个复杂的系统没有逻辑错误
引入	★　基本思想是**把信号量及其操作原语封装在一个对象内部。**即：将共享变量以及对共享变量能够进行的所有操作集中在一个模块中
定义	管程是关于共享资源的数据结构及一组针对该资源的操作过程所构成的软件模块
优点	可增强模块的**独立性**　系统按资源管理的观点分解成若干模块，按不同的管理方式定义模块的类型和结构，使同步操作相对集中，从而增加了模块的相对独立性 可提高代码的**可读性**　• 便于修改和维护，正确性易于保证 • 采用集中式同步机制。一个操作系统或并发程序由若干个这样的模块所构成
特性	**模块化**　一个管程是一个基本程序单位，可以单独编译； **抽象数据类型**　管程是一种特殊的数据类型，其中不仅有数据，而且有对数据进

		行操作的代码
	信息封装	管程是半透明的，管程中的外部过程（函数）实现了某些功能，至于这些功能是怎样实现的，在其外部则是不可见的
实现要素	• 管程中的共享变量在管程外部是不可见的，外部只能通过调用管程中所说明的外部过程（函数）来间接地访问管程中的共享变量； • 为了保证管程共享变量的数据完整性，规定管程互斥进入； • 管程通常是用来管理资源的，因而在管程中应当设有进程等待队列以及相应的等待及唤醒操作；	
组成	名称：	为每个共享资源设立一个管程
	数据结构说	一组局部于管程的控制变量(生产者放/消费者取)
	操作原语	对控制变量和临界资源进行操作的一组原语过程（程序代码），是访问该管程的唯一途径。这些原语本身是互斥的，任一时刻只允许一个进程去调用，其余需要访问的进程就等待。
	初始化代码	对控制变量进行初始化的代码
管程和进程的异同	• 设置进程和管程的目的不同 • 系统管理数据结构 ★**进程：PCB** ★**管程：等待队列** • 管程被进程调用 • 管程是操作系统的固有成分，无创建和撤消	

★ **管程具体参见PPT**

第四章 处理机调度与死锁

- 处理机调度的基本概念
- 高级、中级和低级调度

一个批处理型作业，通常需要经过高级调度(作业调度)、低级调度(进程调度)、中级调度，方能获得处理机。

作业的**概念**	• 一个作业是指在一次应用业务处理过程中，从输入开始到输出结束，用户要求计算机所做的有关该次业务处理的全部工作。 • 用户的观点：在一次业务处理过程中，从输入程序和数据到输出结果的全过程。作业步：形成中间结果文件 • 系统的观点（针对作业进行资源分配）：作业由程序及数据（作业体）和作业说明书（作业控制语言）作业由不同的顺序相连的作业步组成 • 作业步是在一个作业的处理过程中，计算机所做的相对独立的工作

- *高级调度*

又称作业调度或长程调度。

功能	用于决定把外存上处于后备队列中的那些作业调入内存

具体操作	创建进程、分配资源，排在就绪队列尾端，准备执行

- *中级调度*

  又称中程调度

目的	是为了提高内存利用率和系统吞吐量。
功能	• 暂时不能运行的进程挂起，释放宝贵的内存资源 • 具备条件时：把外存上的就绪进程，重新调入内存，挂在就绪队列上等待进程调度

- *低级调度*

  又称进程调度或短程调度，**即进程3基本状态转换的调度**

功能	用来决定就绪队列中的哪个进程应获得处理机	
调度方式	**非抢占方式**	可能引起进程调度的因素(进程调度的时机)： • 正在执行的进程执行完毕， 选一新进程 • 执行中的进程因提出I/O请求而暂停执行 • 进程调用了P、V操作 • 分时系统中时间片用完
	抢占方式	抢占的原则有： • 优先权原则 • 短作业(进程)优先原则 • 时间片原则。

- 调度队列模型

★ **详见PPT**

- 选择调度方式和调度算法的若干准则

- *面向用户的准则*

1、周转时间短

通常把周转时间作为评价批处理系统的性能、选择作业调度方式与算法的准则

定义	指从作业提交给系统开始，到作业完成为止的这段时间间隔(称为作业周转时间)。
包括	i. 作业在外存后备队列上等待(作业)调度的时间； ii. 进程在就绪队列上等待进程调度的时间； iii. 进程在CPU上执行的时间； iv. 等待I / O操作完成的时间。 其中，第2)、3)、4)项在一个作业的处理过程中，可能发生多次
平均周转时间	$T=\frac{1}{n}\left[\sum_{i=1}^{i} T_i\right]$
带权周转时间	作业的周转时间T与系统为它提供服务的时间TS之比，即W=T/TS

平均带全周转时间	$W=\frac{1}{n}\left[\sum_{i=1}^{n}\frac{T_i}{T_{Si}}\right]$

2、响应时间快

响应时间常常用于评价分时操作系统的性能，是选择分时系统中进程调度算法的重要准则之一

定义	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为止的时间，或说直到在屏幕上显示出结果为止的一段时间间隔。
包括	i. 从键盘输入的请求信息传送到处理机的时间 ii. 处理机对请求信息进行处理的时间 iii. 将所形成的响应回送到终端显示器的时间

3、截止时间的保证

用来评价实时系统性能的重要指标，因而是选择实时调度算法的重要准则

定义	指某任务必须开始执行的最迟时间，或必须完成的最迟时间

- *面向系统的准则*
  - **i. 系统吞吐量高。**
  - **ii. 处理机利用率好**
  - **iii. 各类资源的平衡利用**

- 调度算法

功能	i. 选择占有处理机(CPU)的进程 ii. 记录系统中所有进程的执行情况 iii. 进行进程上、下的转换	• 在OS中调度的实质是一种资源分配 • 不同的系统采用不同的调度算法 批处理系统常采用短作业优先算法 分时系统采用轮转法进行调度

- *进程调度算法*

一、先来先服务（FCFS）调度算法

描述	每次调度是从就绪队列中选择最先进入的进程，为之分配处理机使之执行；直到该进程执行完成或发生阻塞事件
缺点	完全未考虑进程的紧迫程度，因而不能保证紧迫性进程会被及时处理

二、短进程优先调度算法

描述	从就绪队列中选出一估计运行时间最短的进程，将处理机分配给它，使它立即执行，直到执行到完成，或发生某事件而被阻塞放弃处理机时，再重新调度。
缺点	• 对长进程不利。 • 完全未考虑进程的紧迫程度，因而不能保证紧迫性作业(进程)会被及时处理。 • 作业(进程)的长短只是根据用户所提供的估计执行时间而定的，而用户又可能会有意或无意地缩短其作业的估计运行时间，致使该算法不一定能真正做到短作业优先调先

三、高优先权(FPF)优先调度算法

描述	• 非抢占式优先权调度算法 系统一旦把处理机分配给就绪队列中优先权最高的进程后，该进程便一直执行下去，直至完成； 或因发生某事件使该进程放弃处理机时，系统方可再将处理机重新分配给另一优先权最高的进程 • 抢占式优先权调度算法 在执行期间，只要出现了另一个优先权更高的进程，进程调度程序就立即停止当前进程(原优先权最高的进程)的执行，重新将处理机分配给新到的优先权最高的进程
优先权类型	• 静态优先权 静态优先权是在创建进程时确定的，且在进程的整个运行期间保持不变； 利用某一范围内的一个整数来表示 • 动态优先权 在创建进程时所赋予的优先权，是可以随进程的推进或随其等待时间的增加而改变的，以便获得更好的调度性能
优先权依据	• 进程类型(系统优先权高于用户优先权) • 进程对资源的需求。 (对要求少的赋予高优先权) • 用户要求(取决于用户进程的紧迫程度及用户所付费用的多少)

四、高响应比优先调度算法

优先权规律	
优点	• 先来先服务 当要求服务的时间相同时，作业的优先权决定于其等待时间，等待时间愈长，其优先权愈高，因而它实现的是先来先服务 • 短作业 如果作业的等待时间相同，则要求服务的时间愈短，其优先权愈高，因而该算法有利于短作业 • 长作业 作业的优先级可以随等待时间的增加而提高，当其等待时间足够长时，其优先级便可升到很高， 从而也可获得处理机

五、基于时间片的轮转调度算法

描述	就绪进程按先来先服务的原则，把CPU分配给队首进程，并令其执行一个时间片。当执行的时间片用完时，将它送往就绪队列的末尾；依次执行下个进程
优点	保证就绪队列中的所有进程，在一给定的时间内，均能获得一时间片的处理机执行时间

六、多级反馈队列调度算法

描述	• 新进程放入第一队列的末尾，按先来先原则等待调度 • 在规定时间片内能完成，便正常结束；否则降级到下一队列 • 当降到第n队列后，在第n队列中便采取按时间片轮转的方式运行

	• 仅当第一队列空闲时，调度程序才调度第二队列中的进程运行 • 高优先权进程采用抢占原则，被抢占着放回本队列末尾
设计原则	• 设置多个就绪队列 • 各个队列优先级不同 • 各个队列中进程执行时间片的大小也各不相同。优先权愈高，执行时间片就愈小
优点	• 终端型作业用户 终端型作业用户所提交的作业，大都属于交互型作业，作业通常较短小。只要能使这些作业(进程)在第一队列所规定的时间片内完成，便可使终端型作业用户都感到满意 • 短批处理作业用户 对于很短的批处理型作业，开始时像终端型作业一样，如果仅在第一队列中执行一个间片即可完成，便可获得与终端型作业一样的响应时间。对于稍长的作业，通常也只需在第二队列和第三队列中各执行一个时间片即可完成，其周转时间仍然较短 • 长批处理作业用户 对于长作业，它将依次在第1，2，…，直到第n个队列中运行，然后再按轮转方式运行，用户不必耽心其作业长期得不到处理。 而且每向下下降一个队列，其得到的时间片将增加，故可进一步缩短长作业的等待时间

- 死锁

概念	• 多个进程因竞争资源而造成的一种僵局，若无外力作用，这些进程都将永远不能再向前推进 • 两个或两个以上的进程都无限止的等待永不出现的资源而发生的一种状态	
产生原因	竞争资源	系统中供多个进程所共享的资源不足。如打印机
	进程间推进顺序非法	进程在运行过程中，请求和释放资源的顺序不当，导致了进程死锁

- 竞争资源

分类类:	• 可剥夺 • 非剥夺性资源 例: 1. CPU和主存属于可剥夺性资源。 2. 打印机和磁带机属于不可剥夺性资源
竞争非剥夺性资源	资源不够用
竞争临时性资源	指的是由一个进程产生，被另一个进程使用一短暂时间后便无用的资源，也成消耗性资源。

- 进程推进顺序不当引起死锁
- 产生死锁的必要条件

互斥条件	指进程对所分配到的资源进行排它性使用，即在一段时间内某资源只能由一个进程占有。
请求和保持条件	一个进程申请资源得不到满足时处于等待资源的状态，并且不释放占有的资源。
不剥夺条件	指进程获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。
环路等待条件	存在一组进程，其中每一个进程分别等待另一个进程所占有的资源。

- 处理死锁的基本办法

预防死锁	破坏四个条件中的一个、几个；互斥条件由设备的固有条件决定，无法破坏
避免死锁	不事先破坏产生死锁的四个必要条件，而是用某种方法防止系统死锁
检测死锁	事先不采取任何方法预防，而是运行过程中检测死锁，并 处理
解除死锁	同上配套使用，常采用撤销或挂起进程，回收资源的方法解决死锁

- *预防死锁*

方式	**内容**	**缺点**
摒弃“请求和保持”条件	• 摒弃请求条件：预分配运行所需的全部资源，运行时不再请求 • 摒弃保持条件：如有一种不满足则一种资源也不分配给该进程	资源被严重浪费
摒弃“不剥夺”条件	已经占有某些资源进行运行的进程，如又申请得不到的资源， 则释放已经占有的全部资源	有些工作可能反复重新执行，严重浪费机时
摒弃“环路等待”条件	系统将所有的资源按类型进行线性排队，并 赋予不同的序号。所有进程对资源的请求必须严格按资源序号递增的次序提出，因而摒弃了“环路等待”条件	

- *避免死锁*

系统在进行资源分配之前，应先计算此次资源分配的安全性。若此次分配不会导致系统进入不安全状态，则将资源分配给进程； 否则，令进程等待

安全状态	是指系统能按某种进程顺序(P1, P2, …, Pn)(称〈P1, P2, …, Pn〉序列为安全序列)，来为每个进程Pi分配其所需资源，直至满足每个进程对资源的最大需求，使每个进程都可顺利地完成 如果系统无法找到这样一个安全序列，则称系统处于不安全状态

★ 并非所有不安全状态都是死锁状态，但当系统进入不安全状态后，便可能进而进入死锁状态
★ 只要系统处于安全状态，系统便可避免进入死锁状态

! 避免死锁的实质在于：如何使系统不进入不安全状态

银行家算法	采用银行家算法分配资源时，测试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时，就满足该进程的申请，否则就推迟分配
特点	• 允许互斥、部分分配和不可抢占，可提高资源利用率 • 要求事先说明最大资源要求，在现实中很困难
缺点	• 算法必须不断的测试各个进程占有和申请资源的情况，需花费时间较多

- *检测死锁*

资源分配图	• 属于E中的一个边e∈E，都连接着P中的一个结点和R中的一个结点， • e={pi, rj}是资源请求边，由进程pi指向资源rj， 它表示进程pi请求一个单位的rj资源。 • e={rj, pi}是资源分配边，由资源rj指向进程pi, 它表示把一个单位的资源rj分配给进程pi
可完全简化	1. 资源分配图中，找出一个既不阻塞又非独立的进程结点pi。在顺利情况下，pi可获得所需资源而继续执行，直至运行完毕，再释放其所占有的全部资源。这相当于消去pi所有的请求边和分配边，使之成为孤立的结点。 2. p1释放资源后，便可使p2获得资源而继续运行，直到p2完成后又释放出它所占有的全部资源，而形成图(c)所示的情况。 3. 在进行一系列的简化后，若能消去图中所有的边，使所有进程都成为孤立结点，则称该图是可完全简化的
不可完全简化	不能通过任何过程使该图完全简化，则称该图是不可完全简化的
死锁定理	当系统处于某一状态时(称为s状态)，其资源分配图不可完全简化，s一定为死锁状态

- *解除死锁*

剥夺资源	从其它进程剥夺足够数量的资源给死锁进程，以解除死锁状态；
撤消进程	• 最简单的撤消进程的方法，是使全部死锁进程都夭折掉； • 稍微温和一点的方法是按照某种顺序逐个地撤消进程，直至有足够的资源可用，死锁状态消除为止

- 第五章 存储器管理

定义	指存储器资源（主要指内存并涉及外存）的管理
内容	• 存储器资源的组织（如内存的组织方式） • 地址变换（逻辑地址与物理地址的对应关系维护）

	• 虚拟存储的调度算法

- 存储器管理的功能

主存分配和回收	分配和回收算法及相应的数据结构。
地址变换	• 可执行文件生成中的链接技术 • 程序加载(装入)时的重定位技术 • 进程运行时硬件和软件的地址变换技术和机构
存储共享和保护	代码和数据共享地址空间访问权限（读、写、执行）
主存容量扩充	存储器的逻辑组织和物理组织； • 由应用程序控制：覆盖； • 由OS控制：交换（整个进程空间），虚拟存储的请求调入和预调入（部分进程空间） • 提高主存利用率

功能	• 内存分配 • 地址映射 • 内存保护 • 内存扩充

- 程序的装入和连接

- 复习内容

逻辑地址（相对地址，虚地址）	用户的程序经过汇编或编译后形成目标代码，目标代码通常采用相对地址的形式。首地址为0，其余指令中的地址都相对于首地址来编址。不能用逻辑地址在内存中读取信息。
物理地址（绝对地址，实地址）	内存中存储单元的地址。物理地址可直接寻址
地址映射：	将用户程序中的逻辑地址转换为运行时由机器直接寻址的物理地址。 原因： • 当程序装入内存时，操作系统要为该程序分配一个合适的内存空间，由于程序的逻辑地址与分配到内存物理地址不一致，而CPU执行指令时，是按物理地址进行的，所以要进行地址转换

- 程序的装入

即“**重定位方法**”亦即“**地址映射功能**"

重定位	一个经过编译链接的程序存在于自己的虚拟地址空间中，当要运行时才把它装入主存空间。这是需要将程序中的**虚拟地址**转换为主存中的**物理地址**，这个转化过程就是重定位技术

- *程序成为进程前的准备工作:*

编辑	形成源文件(符号地址)
编译	形成目标模块(模块内符号地址解析)
链接	由多个目标模块或程序库生成可执行文件(模块间符号地址解析)
装入	**构造PCB**，形成进程(使用物理地址)

- *重定位方式*

绝对装入方式(直接分配方式)	编写程序时必须采用实际的物理地址，编译器在编译时也必须使用实际地址
优点	装入过程简单。
缺点	过于依赖于硬件结构，不适于多道程序系统。用户编程很不方便

可重定位装入（静态重定位）	• 程序员编写程序时可以采用相对地址 • 作业（用户程序）在装入内存时才确定它的物理地址，并且将相对地址转换为物理地址 • 作业一旦装入就不能移动、改变空间或者被换出主存 • 在程序运行之前，由链接装入程序进行的一次重定位 ★ • 在程序运行之前已经完成了逻辑地址到物理地址的转换(只要完成链接装入)
优点	不需硬件变换机构支持，可以装入有限多道程序（如MS DOS中的TSR）。
缺点	一个程序通常需要占用连续的内存空间，程序装入内存后在运行期间不能移动。不易实现共享

动态运行时装入方式（动态重定位）	★ • 在程序执行的过程中，每当访问指令或数据时，才将要访问的指令或数据的逻辑地址转换成物理地址 ★ • 在程序装入时不对地址做任何操作，也就是保留逻辑地址不变 ★ • 在把程序模块装入内存后，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。因此， **装入内存后的所有地址都仍是相对地址**

- 分区存储管理

原理	把内存分为一些大小相等或不等的分区(partition)，每个应用进程占用一个或几个分区。操作系统占用其中一个分区

特点	• 适用于多道程序系统和分时系统 • 支持多个程序并发执行
缺点	• 难以进行内存分区的共享 • 可能存在内碎片和外碎片。 i. **内碎片**：占用分区之内未被利用的空间 ii. **外碎片**：占用分区之间难以利用的空闲分区（通常是小空闲分区）
数据结构	！• **分区表** i. 空闲分区表 ii. 占用分区表 ！• **分区链表**
内存紧缩	将各个占用分区向内存一端移动。使各个空闲分区聚集在另一端，然后将各个空闲分区合并成为一个空闲分区 特点：• 占用分区进行内存数据搬移占用CPU时间 • 如果对占用分区中的程序进行"浮动"，则其重定位需要硬件支持 时机：• 每个分区释放后 • 内存分配找不到满足条件的空闲分区时

- 分区分配算法

首次适应算法 FF	• 空闲分区链以地址递增的次序链接。 • 内存分配时，从链首开始顺序查找，直至找到一个能满足其大小要求的空闲分区为止。然后按作业大小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链中。
优点	倾向于优先利用内存中低址部分的空闲分区，在高址部分的空闲分区很少被利用，从而保留了高址部分的大空闲区。显然，这为后到的大作业分配大的内存空间创造了条件。
缺点	低址部分不断被划分，形成碎片，另每次查找都从低址部分开始，这增加了查找可用空闲分区的开销

循环首次适应算法	• 内存分配时，不再每次从链首开始查找，而是从上次找到的空闲分区的下一个空闲分区开始查找，直至找到第一个能满足要求的空闲分区，并从中划出一块与请求的大小相等的内存空间分配给作业 • 为实现该算法，应设置一起始查寻指针，以指示下一次起始查寻的空闲分区 • 采用循环查找方式 即如果最后一个(链尾)空闲分区，其大小仍不能满足要求，应返回到第一个空闲分区，比较其大小是否满足要求。找到后，应立即调整起始查寻指针
优点	内存中的空闲分区分布得更均匀，减少查找空闲分区的开销
缺点	缺乏大的空闲分区

最佳适应算法	• 指每次为作业分配内存时，总把既能满足要求、又是最小的空闲分区分配给作业 为

	了加速寻找， • 该算法要求将所有的空闲区，按其大小以递增的顺序形成一空闲区链 • 第一次找到的满足要求的空闲区，必然是最优的
优点	避免了“大材小用”
缺点	最佳适应算法似乎是最佳的，然而在宏观上却不一定。因为每次分配后所切割下的剩余部分，总是最小的，容易产生碎片