**第一章 操作系统引论**

1.操作系统的定义，操作系统管理着计算机上的( 软、硬件资源 )。

操作系统（Operating System， OS）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；

它是计算机系统中最基本的系统软件

2.单道批处理和多道批处理系统的区别？给几道作业，分别用单道批处理和多道批处理，分别用时多少？

单道批处理：微观的进程调度是从主流在内存的多个作业中选择一个,使它获得处理机并执行。

优点：解决人机矛盾和CPU与IO设备速度不匹配问题，提高系统资源的利用率和系统吞吐量。

缺点：不能充分的利用系统资源，现很少使用。

特征：①自动性

②顺序性

③单道性

多道批处理：多道批处理在内存中可以同时存放多道作业,宏观上这些作业在同时进行,而每一时刻则只有一道作业在运行。

优势：资源利用率高，使CPU始终处于忙碌的状态，提高内存的利用率，提高IO利用率；系统吞吐量大（CPU和其资源始终保持忙碌的状态，仅在作业完成时或者运行不下去的时候才切换，系统开销小）。

缺点：平均周转时间长，无交互能力。

特征：①多道性

②无序性

③调度性（两次调度：作业调度，进程调度）

3.操作系统的特征：并发性、共享性、虚拟性、不确定性

并发：指两个或多个事件在同一时间间隔内发生。这些事件宏观上是同时发生的，但微观上是交替发生的

共享：资源共享，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。

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。物理实体（前者）是实际存在的，而逻辑上对应物（后者）是用户感受到的。

不确定（异步）：也称随机性，是指系统中各种事件的发生顺序是不确定的。（是指，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的，而是走走停停，以不可预知的速度向前推进，这就是进程的异步性。）

4.操作系统的功能有哪些？（进程/处理机管理、存储管理、设备管理、文件管理、操作系统接口）

①进程/处理机管理：通过进程管理协调多道程序之间的关系，可以解决对处理机的调度分配及回收等问题。可分为以下几个方面：

1. 进程控制
2. 进程同步
3. 进程通信
4. 进程调度

②存储管理：方便用户使用内存，提高内存的利用率以及从逻辑上扩充内存。

1. 内存分配
2. 地址映射
3. 内存保护
4. 内存扩充

③设备管理：1）完成用户输入输出的请求

1. 提高外部设备的调用率
2. 尽可能提高输入输出的速度
3. 方便用户使用外部设备

功能： 1、设备分配

2、设备控制

3、设备的无关性

④文件管理：也称信息资源管理，向用户提供一种简便的、统一的存取和管理信息的方法，并同时解决信息的共享、安全保密等问题。

功能：1、文件存储管理空间的管理

1. 目录管理：为文件建立目录项，实行按名存取，实现文件的共享，实现快速查询
2. 文件的读写管理
3. 文件的存取控制

三防止：1）防止未经核准的用户存取文件

1. 防止冒名顶替存取文件
2. 防止以不正确的方式存储文件

三控制：1）系统级存取控制：用口令对口令加密进行

1. 用户级存取控制：对用户分类和分配适当的文件存取权限
2. 文件级存取控制：设置文件属性（只读、只可执行、可读/写）来控制对文件的存取

⑤操作系统接口：方便用户使用操作系统

1. 命令接口：1）联机命令接口
2. 脱机命令接口
3. 图形用户界面接口
4. 程序接口：又称为系统调用，是用户程序取得操作系统的唯一途径

5.操作系统的种类——批处理/分时/实时

①批处理操作系统

1. 单道批处理操作系统：

特征：1、自动性 2、顺序性 3、单道性

1. 多道批处理操作系统：

特征：1、多道性 2、无序性 3、调度性

相对于单道的优点：1、CPU利用率得到提高

2、提高了内存和输入输出设备的利用率

3、增加了系统的吞吐量

②分时操作系统（交互）

需要解决的问题：及时处理，及时解决

实现形式：1、单道分时操作系统（切换作业需要访问内外存）

2、具有”前台“和”后台”的分时操作系统

3、多道分时操作系统（切换作业在内存中进行）

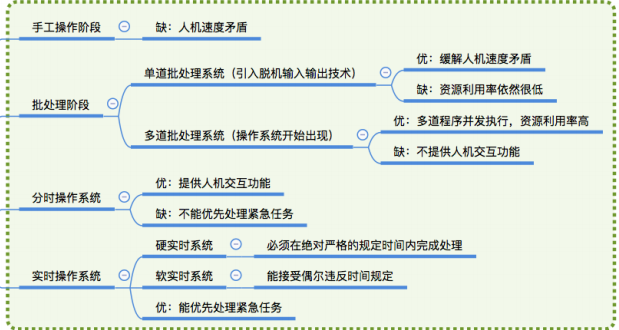
特征：1、多路行 2、独占性 3、交互性 4、及时性

③实时操作系统（规定时间）

类型：1、实时控制系统 2、实时信息处理系统

特殊要求：1、高可靠性 2、过载防护 3、对截至时间的要求

特征（对比分时）：1、多路性 2、独立性 3、交互性 4、及时性 5可靠性



**第二章 进程与线程**

1.程序顺序执行的特征：顺序性、封闭性、可再现性

●顺序性

处理机的操作严格按规定顺序执行

●封闭性

程序执行时，独占系统资源

●可再现性

当初始条件相同时，程序多次执行的结果相同

2.程序并发执行的特征：间断性、失去封闭性、不可再现性

●间断性

并发执行时形成相互制约关系，导致程序“执行—暂停—执行”

●失去封闭性

资源供多个程序共享

●不可再现性

封闭性被打破，多程序运行次序随机

3.可并发执行的条件：Bernstein条件

4.进程的概念：并发执行的程序在一个数据集合上的执行过程。进程与程序的区别和联系

①进程的实质：(1)进程是程序的一次执行

(2)进程是可以和别的计算并发执行的计算

(3)进程可定义为一个数据结构及能在其上进行操作的一个程序

(4)进程是一个程序及其数据在处理机上顺序执行时所发生的活动

(5)进程是程序，在一个数据集合上的运行过程，是系统进行资源分配和调度的一个独立单位。

(6)一个进程就是一个正在执行的程序，包括指令计数器，寄存器和变量的当前值

特征：1、动态性 2、并发性 3、独立性 4、异步性

②区别与联系：(1)进程的动态性和程序的静态性

(2)进程的并发性和程序的顺序性

(3)进程的暂时性和程序的永久性

(4)结构特征。进程是由程序数据和进程控制块三部分组成的，而程序却不是

(5)进程与程序是密切相关的

5.进程是动态的还是静态的？存在的唯一标志是什么？

进程是动态的。进程存在的唯一标志是进程控制块。

6.进程有几种状态？叫什么？基本的状态是哪三个？每个状态具有什么特征？状态之间如何转换？处于就绪态\运行态\阻塞态的进程个数…

①进程有五种状态：1.创建状态 2.运行状态 3.就绪状态 4.阻塞状态 5.退出状态

②三个基本状态：运行，就绪，阻塞

③特征：运行状态：即进程正在处理机上运行的状态

就绪状态：即进程已经获得了除处理器之外的所有必要资源，只要获得处理机就可以运行的状态

阻塞状态：当进程由于等待输入输出操作或某个同步事件而暂停运行时，就处于阻塞状态。

④状态转换：就绪→运行：（1）单处理机：运行状态的进程只有一个.（2）由进程调度程序负责挑选进程，使之获得处理机运行。

运行→规定：时间片用完而被暂停执行，而后根据优先级插入就绪队列，之后重新调度。

运行→阻塞：进程请求IO或者访问某个临界资源而该临界资源正在被其他资源访问。此时系统会调用进程调度程序重新选择一个进程投入运行。

阻塞→就绪：阻塞原因解除，转化为就绪状态。

**例题**： 1、在一个单处理机的系统中，OS的进程有运行、就绪、阻塞三个基本状态。假如某时刻该系统中有10个进程并发执行，在略去调度程序所占用时间情况下试问：

(1)这时刻系统中处于运行态的进程数最多有几个？最少有几个？（1）最多1，最少0

(2)这时刻系统中处于就绪态的进程数最多有几个？最少有几个？（2）最多9，最少0

(3)这时刻系统中处于阻塞态的进程数最多有几个？最少有几个？（3）最多10，最少0

2、在一个N个处理机的系统中，OS的进程有运行、就绪、阻塞三个基本状态。假如某时刻该系统中有10个进程并发执行，在略去调 度程序所占用时间情况下试问(10>N)

(1)这时刻系统中处于运行态的进程数最多有几个？最少有几个？(1)最多N，最少0

(2)这时刻系统中处于就绪态的进程数最多有几个？最少有几个？(2)最多10-N，最少0

(3)这时刻系统中处于阻塞态的进程数最多有几个？最少有几个？(3)最多10，最少0

7.处理机的执行状态：用户态、核心态，特权指令哪个状态不可以执行？

用户态：具有较低特权，只能执行规定的命令，访问指定的寄存器和存储区。

核心态：具有较高的特权，能执行一切命令，访问所有寄存器和存储区。能执行特权指令

8.线程出现的原因，承担进程的哪部分责任？线程的分类

①原因：减少程序并发执行时所付出的时间和空间开销承担进程中的一个控制点，执行一系列指令。

②分类：1、用户级线程

优点：

1)线程切换不需要系统状态的转换；

2)每个进程可以使用专用的线程调度算法来调度线程；

3)用户级线程可以在任何操作系统中运行，不需要对底层OS内核进行修改。

缺点：

1)用户级线程系统调用时，会被认为是进程的行为，所以进程被阻塞，所有线程都不能继续执行。（很多OS中，许多系统调用会引起进程阻塞）

4)只使用用户级线程的系统中，多线程的应用程序不能利用多处理机技术。

2、内核级线程

特点：

有关线程管理的工作都是内核负责的；

若要使用线程，通过API；

OS内核负责保存进程和线程的相关信息；

系统调用是基于线程的（处理机的切换以线程为单位）

系统内核调度线程时，知道线程属于哪个进程，却不需要考虑（线程调度挑选哪个线程与他属于哪个进程无关）

从一个进程内的某个线程切换到另一个进程中的某个线程的开销必定大于一个进程内两个线程的切换

缺点：

同一个进程中两个线程之间的切换都需要内核的状态转换（用户态核心态）

○对比：

切换速度：内核级线程切换需要内核模式的转换，所以速度慢

阻塞：用户级线程因等待I/O等的请求阻塞，会导致整个进程阻塞。

**9. Thread模块里的函数，包括start( )、current\_thread()、 is\_alive()、join() 等的功能**

⚪ start（）

开始线程的活动。

每个线程对象最多只能调用一次。它安排run()在单独的控制线程中调用对象方法。此方法将RuntimeError在同一个线程对象上多次调用if。

⚪ current\_thread()

返回当前Thread对象，对应于调用者的控制线程。如果未通过threading模块创建调用者的控制 线程，则返回具有有限功能的虚拟线程对象。

⚪ is\_alive（）、isAlive（）

返回线程是否存活。

此方法True在run()方法启动之前返回，直到run()方法终止之后。模块函数 enumerate()返回所有活动线程的列表。

⚪join（）

阻塞进程直到线程执行完毕。这将阻塞调用线程，直到调用其join()方法的线程终止 - 正常或通过未处理的异常 - 或直到发生可选的超时。

**第三章 进程同步与通信**

1.临界资源是什么？访问临界资源的4个准则

临界资源：某段时间内仅允许一个进程使用的资源。

4个准则：(1)空闲让进：临界区无进程（临界资源空闲）→允许进入

(2)忙则等待：临界区有进程（临界资源正在被访问）→进程等待

(3)有限等待：进程等待必须保证在有效的时间内进入，以免进入“死等”

(4)让权等待：进程不能进入临界区时，应立即释放处理机，以免进程进入“忙等”

2.临界区是什么？其他几个区？

临界区：每个进程中访问临界资源的那段代码。

进入区：在进入临界区之前，检查可否进入临界区的一段代码。如果可以进入临界区，通常设置相应"正在访问临界区"标志

退出区：用于将"正在访问临界区"标志清除

剩余区：代码中的其余部分

3.信号量的数据结构（value + PCB队列），PV操作原语的具体过程，PV操作的作用

PV操作原语的具体过程: semaphore mutex;

P(mutex);

临界区;

V(mutex);

剩余区;

P就是申请一个资源 V就是释放一个资源

4.信号量的物理意义——互斥/同步，P/V等…

●s. value的初值表示系统中某种资源数目。

● wait(s)表示要申请一个资源。

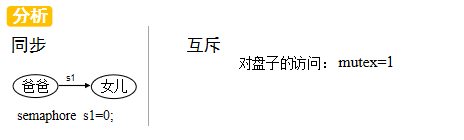
● signal(s)表示要释放一个资源。

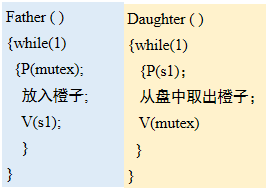
● s.value <0时，|s.value|表示等待队列的进程数。

5.应用到实际问题中的例子：吃水果问题1/2/3

**①吃水果问题1**

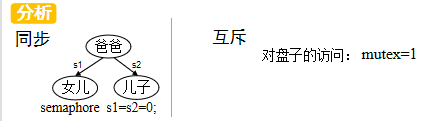
桌上有一空盘，只允许存放一个水果。爸爸向盘中放橙子，女儿从盘中取橙子吃。规定当盘空时一次只能放一个橙子供吃者自用，请用PV操作实现爸爸、女儿两个并发进程。

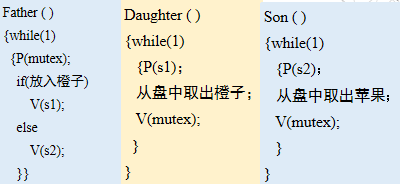




**②吃水果问题2**

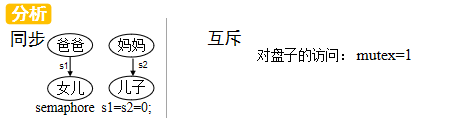
桌上有一空盘，只允许存放一个水果。爸爸可以向盘中放苹果，也可向盘中放橙子，女儿专等吃橙子，儿子专等吃苹果。规定当盘空时一次只能放一个水果供吃者自用，请用PV操作实现爸爸、女儿、儿子三个并发进程的同步。

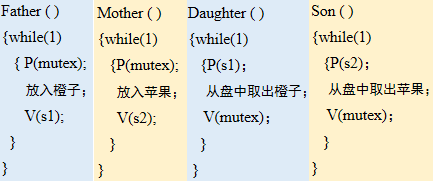




**③吃水果问题3**

桌上有一空盘，只允许存放一个水果。爸爸可以向盘中放橙子，妈妈可向盘中放苹果，女儿专等吃橙子，儿子专等吃苹果。规定当盘空时一次只能放一个水果供吃者自用，请用PV操作实现爸爸、妈妈、女儿、儿子三个并发进程的同步。





6.AND信号量

基本思想：

将整个运行期间所需要的所有临界资源，一次性地全部分配给进程，待该进程使用完后再一起释放。

只要尚有一个资源不能满足进程的要求，其他所有能分配给该进程的资源也都不予以分配。

7.原语是什么？PV操作是高级通信原语

原语是一段用机器指令编写的完成特定功能的程序,在执行过程中不允许中断。

8.进程通信是什么？

进程通信就是指进程之间的信息交换。

9.判断进程之间存在什么制约关系及原因。

进程之间存在着直接制约和间接制约两种制约关系，其中直接制约（同步）是由于进程间的相互合作而引起的，而间接制约（互斥）则是由于进程间共享临界资源而引起的。

10.互斥是什么？同步是什么？

互斥：多个进程不能同时使用同一个资源，当某个进程使用某种资源时，其他进程必须等待

同步：多个进程中发生的事件存在时序关系

11.互斥实现的软件方法：单标志法的做法，缺点

(1)单标志法

算法思想：每个进入临界区的全县只能被另一个进程赋予。如图一

缺点：违背了“空闲让进”的原则。

(2)双标志先检查法

算法思想：每个进程在进入临界区之前先检查当前有没有别的进程想进入临界区，如果没有，则把自身对应的表示设置为true，之后开始访问临界区。如图二

缺点：违反了“忙则等待”的原则。

(3)双标志后检查法

算法思想：先“上锁”后“检查”的方法，避免违反“忙则等待”的则。如图三

缺点：违反了“空闲让进”和“有限等待”的原则。会因为各进程都无法访问临界区而产生“饥饿”现象。

(4)Peterson算法

算法思想：如果双方都争着想进入临界区，那么主动把机会先让给其他想进入临界区的进程，最后一个谦让的进程最后进入临界区。如图四

缺点：违反了“让权等待”的原则。

**第四章 调度与死锁**

1.调度分几种类型，每一种的别名，任务是什么？目的是什么？最基本的调度是什么？

⚪高级调度：又称作业调度、宏观调度

任务：决定将外存上后备队列中的哪些作业调入内存。（多道程序的道数

目的：以使该作业的进程获得竞争处理机的权利

⚪中级调度：又称对换程序

主要作用：内存和外存对换区之间进行进程对换，以解决内存紧张问题。（暂时不能执行的进程不占内存，调出外存）

目的：暂不能运行的进程不再占用宝贵的内存资源，而将它们调至外存上去等待

外存上的进程应具有重新获得处理机的机会

⚪低级调度**（最基本调度）**：又称进程调度、微观调度

任务：决定就绪队列中的哪些进程将获得处理机（分配处理机给进程）

2.可剥夺/不可剥夺——调度方式

(1)可剥夺方式(Preemptive Mode)

某一进程用处理机；有更重要或紧迫的进程进入就绪队列，则立即暂停正在执行的进程，将处理机分配给这个更为重要或紧迫的进程

抢占式调度主要有以下原则

优先权原则:允许高优先权的新到进程抢占当前进程的处理机

短作业(进程)优先原则:允许执行时间短的新到进程抢占当前进程的处理机

时间片原则:时间片用完后停止执行，重新进行调度。

(2)不可剥夺方式(Non-preemptive Mode)

某一进程用处理机；有更重要或紧迫的进程进入就绪队列；该进程仍继续执行，直到其完成或发生某种事件而进入完成或阻塞状态时，才把处理机分配给更为重要或紧迫的进程。

引起进程调度的因素

执行完毕/因发生某事件而不能再继续执行

提出I/O请求；

在进程通信或同步过程中执行了某种原语操作，如wait、Block原语

3.性能准则——系统吞吐量、响应时间、周转时间、多道程序度等

●面向用户的准则

响应时间快

周转时间短

优先权准则

截止时间的保证

●面向系统的准则

系统吞吐量

系统单位时间内完成的作业数。

处理机利用率

各类资源平衡利用：使内存、外存和I/O设备的利用率高

公平

4.调度算法——FCFS、SJF/SPN、RR、优先权（Priority（静态or动态））、MFQ、SRT、HRRN 填写表格注意细心，在考虑了算法思想的同时，要看作业是否已到达系统

●先来先服务调度算法 不可剥夺方式

●短作业（进程）优先调度算法 不可剥夺的调度方式

●时间片轮转调度算法 主要适用于进程调度。是可剥夺式调度算法。

●优先权调度算法 可剥夺式调度和不可剥夺式调度

●最短剩余时间优先算法 可剥夺方式（一个时间片未完不可剥夺）

●响应比高者优先调度算法 不可剥夺方式

●多级反馈队列 各队列的不同处理：不同队列可有不同的优先级、时间片长度等。优先权越高，时间片越短

5.死锁的概念：一组竞争系统资源或相互通信的进程相互的“永久”阻塞。若无外力作用，这组进程将永远不能继续执行。

6.死锁产生的原因：两个

(1)资源不足

(2)进程的推进顺序非法

7.四个必要条件

①互斥条件

②请求和保持

③不可剥夺的条件

④环路条件

8.死锁的预防（打破条件）和避免（银行家算法：资源分配和安全检测）、安全状态

①互斥：不可避免

②去掉请求与保持条件

③去掉不可剥夺条件

④去掉环路条件

死锁避免

1. 资源请求①本次请求是否超过该进程的最大需求量

②检查本次请求是否超过可分配的资源量

③都未超过，进行预分配

1. 安全检测①初始化工作向量

②依次判断每个进程i是否可以顺序执行释放变量

安全序列：是进程执行的顺序，如： <P1, P2, …, Pn>，系统按照这个序列为进程分配资源，直到满足最大需求，每个进程都可顺利完成而不会死锁。

安全状态是指系统至少存在一个安全序列的状态。

若系统不存在这样一个安全序列，（即：进程任何顺序都会死锁）则系统处于不安全状态。

9.死锁的检测（资源分配图的简化）和解除

死锁解除

●当发现死锁时，应立即把它们从死锁中解脱出来，常采用的两种方法是：

●剥夺资源

●从其它进程剥夺足够数量的资源给死锁进程。

●撤消进程：撤消的原则是

●为解除死锁状态所需撤消的进程数目最小。

●撤消进程所付出的代价最小。

●将每个死锁进程恢复到前面定义的检查点

**第五章 存储管理**

1.保存在外存的程序要运行，先要？

调入内存

2.程序从编辑好到运行，经过的大致过程

编译、链接、装入

3.装入方式（重定位方式2种）和链接方式3种，地址重定位的定义

①重定位（地址映射）：把相对地址（逻辑地址）转换为绝对地址（物理地址）过程

●静态重定位：

编译时产生相对地址，装入程序确定要装入模块的地址，并在装入时进行重定位，程序运行中不允许在内存移动。

时机：程序执行之前（装入内存时）；

方法：根据装入模块将要装入的内存起始地址，直接修改装配模块中的有关使用地址的指令。

优点：无需硬件地址变换机构（装入程序负责）

缺点：连续装入、难共享

●动态重定位

编译时产生相对地址，装入程序在把装入模块装入内存，程序要真正执行时才把相对地址转换为绝对地址。

时机：程序执行过程中（每次访问内存单元前）；

方法：使装入模块不加任何修改就装入内存，但是它需要硬件——重定位寄存器的支持。

优点：无需连续存放、便于共享

缺点：需专门硬件机构、软件算法实现复杂

②链接方式

(1)静态链接方式

时机：程序运行之前；

方法：先将各目标模块及它们所需的库函数，链接成一个可执行文件，以后不再拆开。

(2)装入时动态链接

时机：装入内存时

方法：边装入边链接的链接方式。

优点：

　 便于修改和更新软件版本

便于实现对目标模块的共享

(3)运行时动态链接

时机：程序执行需要用时；

方法：将某些模块的链接推迟到执行时进行，即在执行过程中当发现一个调用模块尚未装入内存时，OS立即找到该模块并装入内存，链接到调用模块上。

优点：

　 加快程序的装入过程

可节省大量的内存空间

③地址重定位

要把它装入内存执行，就要确定装入内存的实际物理地址，并修改程序中与地址有关的代码

4.连续分区：单一连续、固定、可变（空闲分区的分配算法（首次适应、循环首次适应、最佳、最差）、回收后空闲分区的变化）、动态重定位

①单一连续：一种简单的存储分配方案，主要用于单用户单任务操作系统。

内存被划分为系统区和用户区。

⚪系统区是操作系统专用区，不允许用户程序直接访问，一般在内存低地址部分，

⚪用户区就是剩余的其它内存区域

物理地址=用户区基地址+逻辑地址

优点：方法简单，易于实现；

缺点：它仅适用于单道程序，因而不能使处理机和内存得到充分利用。

②固定

将内存划分为若干分区，每个分区容纳一个作业。

优点：实现技术简单，适用于作业的大小和多少事先都比较清楚的系统中。

缺点：内存的利用率不高，碎片空间不能充分利用。

③可变

进程装入内存时，把可用的内存空间“切出”一个连续的区域分配给进程

(1)首次适应算法 空闲分区以地址递增的顺序链接，从**链首**开始查找，

条件：

空闲分区链以存储空间地址递增的次序链接。

优点：

释放时，因不改变该区在队列中的位置，因此速度快。

保证高地址有空闲空间，可留给大作业。

缺点：

常用大空闲区适应小作业，从而留下小空闲区，且这些小空闲区在链表的前面，影响分配速度。

(2)下次适应算法

条件：

空闲分区链以存储空间地址递增的次序连接成循环链，为进程分配存储空间时，不是从队首开始找，而是从上次找到的空闲空间的下一个空闲分区开始找。

优点：

存储空间利用均衡。

缺点：

没有了较大空闲空间，使大作业无法运行。

(3)最佳适应算法

条件：

空闲分区链以存储空间大小递增的次序拉链。

优点：

若存储空间中存在与申请大小相等的空闲区，则必然被选中，否则选一个稍大的空闲区，而避免毁掉更大的空闲区。

缺点：

小碎片增加——碎片问题严重。

回收时，将空闲区插入适当的位置费时。从**链首**开始查找，

(4)最差适应算法.

条件：

空闲分区链以存储空间大小递减的次序拉链。

优点：

分配后，剩下的空闲区还好用。

申请时，查找容易，因此速度快。

缺点：

当有大作业时，可能就没有空间可用了。

动态重定位：

方法：移动程序将原来分散的空闲小分区拼接为一个大的分区；

时机：一般情况下，当某进程因为没有足够大的分区，而所有“碎片”之和又大于进程大小时。

评价

首次适应算法是最简单，而且是最快和最好的算法；

循环首次适应算法比首次适应算法稍差一些；

而最佳适应算法虽然名字中有“最佳”，但实际上是性能最差的。

5.页式存储管理：内存分块，进程分页，页离散地装入块

①内存空间划分：

整个内存空间划分为一系列大小相等的存储块（物理块、物理页或页框）。递增顺序连续编号0、1、2、……。

②进程的逻辑地址划分：

地址空间划分成一系列与内存块一样大小的块（逻辑页或页面）。递增顺序连续编号为0、1、2、……。

③页面装入页框：

a.判断作业的总页数>内存中的可用块数?

b.以页为单位分配内存，一页分配一个块;

c.作业所有的页所占的块可以不连续。

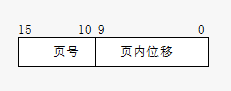
系统同时为这个作业建立一个页号与块号的对照表，称为页表。

6.逻辑地址是一维的、页表、快表

逻辑地址被分为两部分：

●页号

●页内位移



若给定某一个逻辑地址（或相对地址），通过下面式子可以得出页号和页内偏移量：

页号=逻辑地址 /页面大小

页内偏移量=逻辑地址 % 页面大小

页表：记录用户程序的逻辑页与内存物理块之间的对应关系，这通过为每个应用程序建立一张页面映射表来实现

快表：在地址变换机构中增设一个小容量的联想存储器（高速缓冲寄存器），它具有并行查询能力，用来存放当前访问最频繁的少量页表。

把存放在高速缓冲寄存器中的页表叫快表

7.逻辑地址占多少位、物理地址占多少位

给一个十进制逻辑地址，需要懂得转换成二进制与物理地址，详情看P132

8.地址重定位过程 （地址映射）

把相对地址（逻辑地址）转换为绝对地址（物理地址）

9.段式存储管理：进程分段（按逻辑结构），装入过程和可变分区的分配类似

为了满足用户的一系列要求：

方便编程：程序按逻辑关系分为若干个段，每个段从0编址，并有名字和长度，访问的逻辑地址由段名和段内偏移量决定。

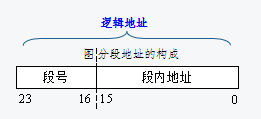
段的共享：共享是以信息为逻辑单位，页是存储信息的物理单位，段却是信息的逻辑单位。

段的保护：保护也是信息为逻辑单位。

动态链接：动态链接以段为单位。

动态增长：实际应用中，某些段（数据段）会不断增长，前面的存储管理方法均难以实现。

10.逻辑地址是二维的、段表



段表  
 为了实现二维地址空间中的逻辑地址到内存空间的物理地址的转换，系统为每个进程建立了一张段映射表，简称“段表”。

11.地址重定位：查段表

|  |  |  |
| --- | --- | --- |
|  | **页式存储管理** | **段式存储管理** |
| **目的** | 实现非连续分配, 解决碎片问题 | 更好满足用户需要 |
| **信息单位** | 页（物理单位） | 段（逻辑单位） |
| **大小** | 固定（由系统定） | 不定（由用户程序定） |
| **内存分配单位** | 页 | 段 |
| **作业地址空间** | 一维 | 二维 |
| **优点** | 有效解决了碎片问题  有效提高内存的利用率 | 更好地实现数据共享与保护  段长可动态增长 便于动态链接 |

物理地址=段内地址+段在内存中的首地址

12.段页式：先分？再分？CPU要取数据要访问几次内存？引入缓存后，又要访问几次内存？

基本原理：

1.先将用户作业**分成若干个段**，每个段都有一个段名；

2.再将每个段划**分为若干页**，再将页离散地装入到内存的块中。

3.内存空间是采用页式存储，但对于用户来说，作业是分段管理的。

4.因此系统要将用户眼中的段式存储转换成内存所需要的页式存储，这种转换是通过段表和页表实现的。

5.此时的逻辑地址结构是由段号、页号、及页内地址三部分组成的，若32位的地址结构采用下面的设置：



CPU要取数据要访问几次内存？

一般需要访问三次以上的内存：

第一次是由段表地址寄存器得段表始址后访问段表，由此取出对应段的页表在内存中的地址。

第二次则是访问页表得到所要访问的物理地址。

第三次才能访问真正需要访问的物理单元。

引入缓存后仍需要3次访问内存。