

第一章：操作系统概述

操作系统的定义、作用、特性

操作系统定义：操作系统是位于硬件层之上、所有其他系统软件层之下的一个系统软件，通过它管理系统中的各种软件和硬件资源，使它们能被充分利用，方便用户使用计算机系统。

操作系统作用：（1）管理系统的各种资源（2）为用户提供友好的界面

操作系统的特性：并发性、共享性、异步性、虚拟性

程序并发与程序并行的区别：（1）程序并行要求微观上的同时，即在绝对的同时时刻有多个程序同时向前推进，而程序并发并不要求微观上的同时，只需从宏观上看多个程序都在向前推进。（2）实现程序并行必须要有多个处理器，但是在单处理器环境下可以实现程序并发，此时这些并发执行的程序是按照某种次序交替地获得处理器并运行的。

硬件环境：

定时装置：（1）绝对时钟（2）间隔时钟

堆与栈的差别：栈是一块按后进先出规则访问的存储区域，用来实现中断嵌套和子程序嵌套（保存调用参数和返回断点）。堆虽然是一块存储区域，但是对堆的访问是任意的，没有后进先出的要求，堆主要用来为动态变量分配存储空间。

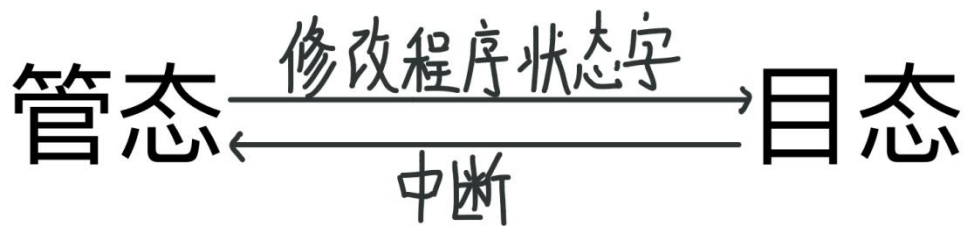
系统栈是内存中属于操作系统空间的一块固定区域，其主要用途为（1）保存中断现场，对于嵌套中断，被中断程序的现场信息被依次压入系统栈，中断返回时逆序弹出；（2）保存操作系统子程序之间相互调用的参数、返回值、返回点，以及子程序（函数）的局部变量。

用户栈是用户进程空间中的一块区域，用于保存用户进程的子程序之间相互调用的参数、返回值、返回点以及子程序（函数）的局部变量。

硬件将处理器分为两种：**管态和目态**

管态：也称为系统态、核心态，是操作系统运行时所处的状态。计算机处于管态时可以执行硬件所提供的全部指令，包括特权指令和非特权指令。

目态：也称为用户态，是一般用户程序运行时所处的状态。处理器在处于目态时只能执行硬件机器指令的一个子集，即非特权指令。



第二章：进程、线程与作业

多道程序设计

单道程序设计缺点：（1）处理器资源利用率低（2）设备资源利用率低（3）内存资源利用率低

多道程序设计：根本目标是提高整个计算机系统的效率。衡量系统效率的尺度为吞吐量

$$\text{吞吐量} = \frac{\text{作业道数}}{\text{全部处理时间}}$$

增加同时运行程序的道数可以提高资源利用率，从而提高系统效率，但道数应与系统资源数量相当。道数过少：系统资源利用率低。道数过多：系统开销增大，程序响应速度下降。

多道程序设计对于系统的资源利用率所带来的影响包括：（1）设备资源利用率提高（2）内存资源利用率提高（3）处理器资源利用率提高（4）自然地表达一个程序内在的并行性。

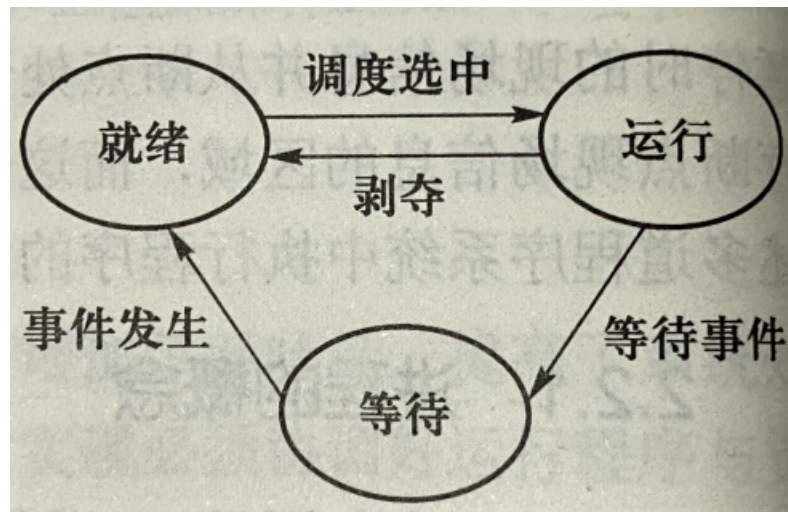
多道程序设计的问题：（1）处理器资源管理问题（2）内存资源管理问题（3）设备资源管理问题

进程的引入

进程是具有一定独立功能的程序关于一个数据集合的一次运行活动。

进程的状态：运行（run）态、就绪（ready）态、等待（wait）态

状态转换：



进程控制块（PCB）是标志进程存在的数据结构，其中包含系统对进程进行管理所需要的全部信息。

进程 = 进程控制块（PCB） + 程序

程序 = 代码 + 数据

进程块存放在系统空间中，只有操作系统才能对其进行存取，用户程序不能访问。进程控制块属于操作系统空间，而程序则属于用户空间。

进程的程序（代码和数据）被称为**进程映像**。

将进程的物理实体与支持进程运行的物理环境合称为**进程上下文**。

系统开销一般是指运行操作系统程序、对系统进行管理所花费的时间和空间。

进程的队列：（1）就绪队列 一般一个系统一个（2）等待队列 每个等待事件一个（3）运行队列 每个CPU一个

进程的类型：系统进程和用户进程

系统进程属于操作系统的一部分，运行于管态，可以执行包括特权指令在内的所有机器指令。

用户进程运行用户程序，直接为用户服务。

进程的特性：并发性、动态性、独立性、交互性、异步性、结构性。

进程与程序的联系：

程序是构成进程的组成部分之一，一个进程存在的目的就是执行其所对应的程序。

进程与程序的差别：

(1) 程序是静态的，而进程是动态的

(2) 程序可以写在纸上或者在某种存储介质上长期保存，而进程具有生存周期，创建后存在，撤销后消亡。

(3) 一个程序可以对应多个进程，但一个进程只能对应一个程序。

线程与轻进程

线程：又称轻进程，是进程内的一个相对独立的执行流。一般认为，进程是资源的分配单位，线程是 CPU 的调度单位。

与进程相比，线程具有如下优点：

(1) 上下文切换速度快 (2) 系统开销小 (3) 通信容易

线程控制块 (TCB) 是标志线程存在的数据结构，其中包含系统对线程进行管理所需要的全部信息。（线程控制块可能属于操作系统空间，也可能属于用户进程空间）

线程的实现：用户级别线程、核心级别线程

用户级别线程的系统调度以进程为单位。若同一进程中的多个线程中至少有一个处于运行态，则该进程的状态为运行态；若同一进程中的多个线程均不处于运行态，但是至少有一个线程处于就绪态，则该进程的状态为就绪态。若同一进程中的多个线程均处于等待态，则该进程的状态为等待态。

优点：（1）不依赖于操作系统，调度灵活（2）同一进程中多线程切换速度快（不需进入操作系统）

缺点：（1）同一进程中多个线程不能真正并行（2）一个进程进入系统受阻，进程中其它线程不能执行

核心级别线程的系统调度以线程为单位。基于系统调用，创建、撤销、状态转换均由操作系统完成。

优点：（1）同一进程内多个线程可以并行执行（2）一线程进入核心等待，其它线程仍可执行

缺点：（1）系统开销大，同一进程内多线程切换速度慢（2）调度算法不能灵活控制

混合线程：

引入多线程程序设计的原因：

（1）某些应用具有内在的多个控制流结构，这些控制流结构具有合作性质，需要共享内存。

（2）在需要多控制流的应用中，多线程比多进程在速度上具有更大优势。

（3）采用多线程可以提高处理器与设备之间的并行性。

（4）在有多个处理器的硬件环境中，多线程可以并行执行，既可提高资源利用效率，又可提高进程推进速度。

作业

用户要求计算机系统为其完成的计算任务的集合称为**作业**（job）。

作业中的一个相对独立的处理步骤称为一个**作业步**。作业步之间具有顺序或并发关系。一个作业步通常可以由一个进程来完成，作业与进程之间具有一对多的关系。

作业控制块是标志作业存在的数据结构，其中包含系统对作业进行管理所需要的全部信息。

第三章：中断与处理器调度

中断与中断系统

在程序运行过程中出现某种紧急事件，必须中止当前正在运行的程序，转去处理此事件，然后再恢复原来运行的程序，这个过程称为**中断**。

中断装置（硬件部分）和中断处理程序（软件部分）统称为**中断系统**。中断装置是中断系统的硬件部分，它的职能是发现并响应中断。中断处理程序为软件部分，根据中断码进一步分析中断源，进行相应的处理，最后根据情况决定是否需要切换进程。

中断响应过程：（1）识别中断源 （2）保存现场 （3）引出中断处理程序

引起中断的事件称为**中断源**。用于保存与中断事件相关信息的寄存器称为**中断寄存器**。中断寄存器中的内容称为**中断字**。

中断可以分为两大类：强迫性中断和自愿性中断。

因为对同类事件的处理方法是相同的，所以不是每个中断源都有一个中断处理程序，每类中断事件有一个中断处理程序。每个中断处理程序有一个入口地址(PC)及其运行环境(PSW)，PSW 和 PC 被称为**中断向量**。

如果系统在处理一个中断事件的过程又响应了新的中断，则称发生**中断嵌套**。

由于中断发送时，被中断程序的程序状态字和指令计数器的值是由硬件中断装置压入系统栈中的，因而**系统栈区的位置实际上是由硬件确定的**。

根据引起中断事件的重要性和紧迫程度，硬件将中断源分为若干个级别，称为**中断优先级别**。中断优先级别是由硬件规定的。

访管指令由指令码和访管中断号组成，即 SVC n，其中 SVC 为指令码，n 为访管中断号

系统调用命令：准备参数、SVC n、取返回值

广义指令：返回值=系统调用名称（参数 1，参数 2，...，参数 m）

处理器调度

处理器调度：指 CPU 资源在可运行实体之间的分配。线程是 CPU 资源分配的基本单位。

对处理器的一次连续使用称为 **CPU 阵发期**，对设备的一次连续使用称为 **I/O 阵发期**。

处理器调度算法：

(1) **先到先服务算法 (FCFS)**：按照进程申请 CPU 的次序，即进入就绪态的次序来调度。

优点：公平，不会出现饿死

缺点：短进程（线程）的等待时间长，从而平均等待时间较长

(2) **最短作业优先算法 (SJF)**：按照 CPU 阵发时间递增的次序调度，易于证明其平均周转时间最短。

优点：平均等待时间最短

缺点：不公平，长作业可能被饿死

(3) **最短剩余时间优先算法 (STRN)**：剥夺式调度算法，当 CPU 空闲时，选择剩余时间最短的进程或线程。当一个新进程或线程到达时，比较新进程所需时间与当前运行进程的估计剩余时间。如果新进程所需的运行时间短，则切换运行进程。

(4) **最高响应比优先算法 (HRN)**：先到先服务算法和最短作业优先算法的折中。

$$RR = \frac{BT + WT}{BT} = 1 + \frac{WT}{BT}$$

RR 为响应比，BT 为 CPU 阵发时间，WT 为等待时间。显然，对于同时到达的任务，处理时间较短的任务将被优先调度，处理时间较长的任务将随其等待时间的增加而动态提升其响应比，因而不会出现饥饿现象。

(5) **最高优先数优先算法 (HPF)**：在每一个进程的进程控制块有一个由数字表示的优先数。

静态优先数：优点：简单、开销较小 缺点：公平性差，可能会造成低优先数进程长期等待

动态优先数：优点：资源利用率高，公平性好 缺点：开销较大，复杂

(6) **循环轮转算法 (RR)**：系统为每一个进程规定一个时间片，所有进程按照时间片的长短轮流地运行。

需要认真考虑时间片长度，若过长，影响系统的响应速度；若过短，则会增加系统开销。循环轮转算法特别适用于分时系统，具有公平、响应及时等特点。

(7) **分类排队算法 (MLQ)**：以多个就绪进程队列为特征

(8) **反馈排队算法 (FB)**：以多个就绪进程队列为特征，在每个队列中通常采用循环轮转算法，所不同的是，进程可以在不同的就绪队列之间移动。这些就绪队列的优先级依次递减，而其时间片的长度则依次递增，

特点：（1）短进程优先处理（2）设备资源利用率高（3）系统开销较小

处理器调度过程（1）保存下降过程现场（2）选择将要运行的进程（3）恢复上升进程现场

调度级别与多级调度

低级调度：（处理器调度）将处理器资源分配给进程或线程使其真正向前推进。

中级调度：是系统控制并发程度的一个调度级别。当系统并发度过高时，将内存中的某些进程暂时交换到外存储器，待以后系统并发度较低时再调回内存。

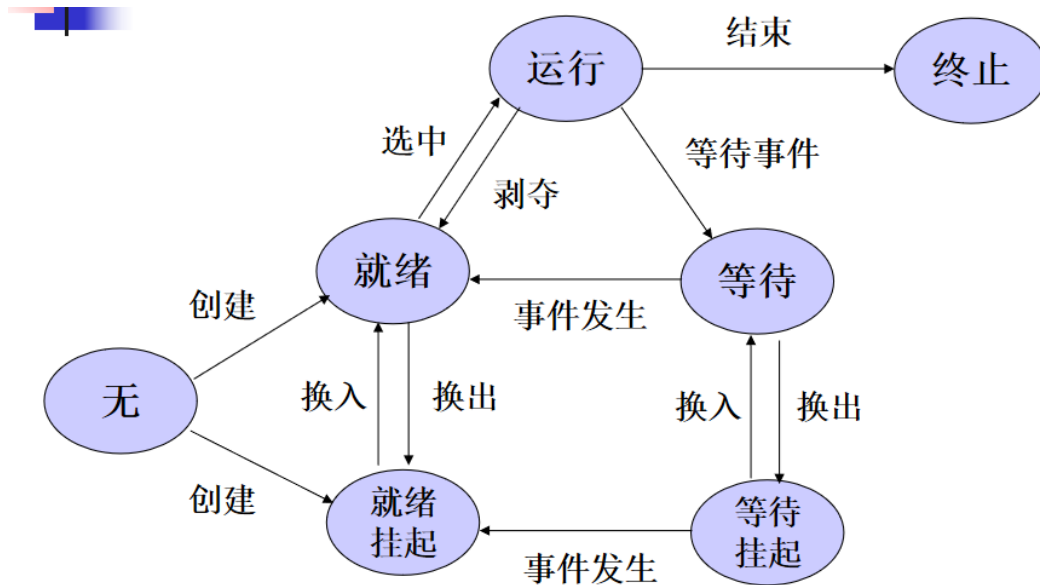
交换是进程在内存和外存储器之间的调度，交换的目标一般有两个：一是缓解内存空间系统等资源紧张的矛盾，二是减小并发度以降低系统开销。

并发度过高时：

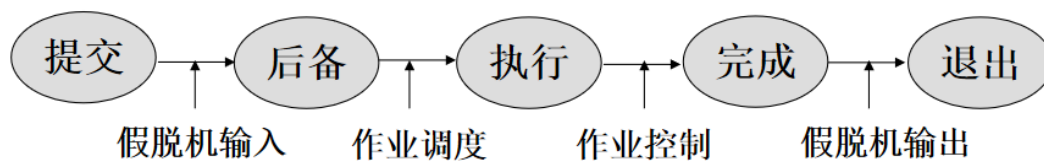
（1）系统开销大（2）响应速度慢

（3）内存等资源紧张（4）进程（线程）频繁进入等待状态

高级调度：（作业调度）其职能是将一个作业由输入并调入内存，并为其建立相应的进程，使其具有运行的资格。



具有中级调度的进程状态转换关系



作业状态转换关系

提交: 输入机向输入井传送

后备: 在输入井,尚未进入内存

执行: 分解为进程,在内存处理

完成: 处理完毕,结果在输出井

退出: 由输出井向打印机传送

实时调度

满足实时任务各自时间约束条件的调度称为**实时调度**。

实时任务按其发生规律可分为两类：随机性实时任务、周期性实时任务。

对于周期性实时任务来说，令 C_i 为任务 P_i 的处理时间， T_i 为任务 P_i 的发生周期，则任务

$$P_1, P_2, \dots, P_n \text{ 可调度的必要条件为 } \sum_{i=1}^n \frac{C_i}{T_i} \leq 1$$

最早截止期优先 (EDF) 调度优先选择完成截止期最早的实时任务, 对于新到达的实时任务,

如果其完成截止期先于正在运行任务的完成截止期, 则重新分派处理器, 即剥夺。

单调速率调度 (RMS) 面向周期性实时任务, 属于非剥夺式调度的范畴。

速率单调调度算法可调度的条件:
$$\sum_{i=1}^n \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$$

第四章：互斥、同步与通信

并发进程

前驱图是一个有向无环图, 图中的每个结点表示一条语句、一个计算步骤或一个进程。

顺序程序的特性: (1) 连续性 (2) 封闭性 (3) 可再现性

并发程序的特性: (1) 间断性 (2) 非封闭性 (3) 不可再现性

程序并发执行的条件 (Bernstein 条件) :

$R(p_i) = \{a_1, a_2, \dots, a_m\}$ 表示程序 p_i 在执行期间所需读取的所有变量的集合, 称为“**读集**”;

$W(p_i) = \{b_1, b_2, \dots, b_n\}$ 表示程序 p_i 在执行期间所需改变的所有变量的集合, 称为“**写集**”。

$R(p_1) \cap W(p_2) \cup R(p_2) \cap W(p_1) \cup W(p_1) \cap W(p_2) = \Phi$

并发进程的执行实际上是进程活动的某种交叉, 某些交叉次序可能得到错误结果。由于具体

交叉的形成与进程的推进速度有关, 而速度又是时间的函数, 因而将这种错误称为**与时间有**

关的错误。

进程互斥

多个进程均需访问的变量称为**共享变量**。

访问共享变量的程序称为临界区, 也称为**临界段**。

一次只允许一个进程使用的资源称为**临界资源**。(打印机、绘图仪、一个临界区)

共享变量: shared <一组变量>

临界区: region <一组变量> do <语句>

两个或两个以上的进程不能同时进入关于同一组共享变量的临界区, 否则可能发生与时间有关的错误, 这种现象称为**进程互斥**。其中有两层含义:

- (1) 任何时刻最多只能有一个进程处于同一组共享变量的相同的临界区域;
- (2) 任何时刻最多只能有一个进程处于同一组共享变量的不同的临界区域。

进程互斥原则: (1) 互斥性原则 (2) 进展性原则 (3) 有限等待性原则

互斥性: 只需证明当一个进程进入其临界区后, 另一进程不能进入其临界区。

进展性: 两种情况: 一个想进, 两个想进的情况下, 满足条件的必定有一个进入临界区

有限等待性: 在前一个进程再次进入临界区之前, 当前进程一定能够进入临界区。

不进入等待状态的等待称为**忙式等待**。

进程同步

一组进程, 为协调其推进速度, 在某些关键点处需要相互等待与相互唤醒, 进程之间这种相互制约的关系称为**进程同步**, 简称同步。

一组进程, 如果它们单独不能正常进行, 但并发可以正常进行, 称这种现象为**进程合作**。

一段不可间断执行的程序称为原语。

■ PV操作:

- 分散式同步机制: 共享变量操作, **PV**操作, 分散在整个系统中或各个进程中。
- 缺点:
 - 可读性差;
 - 正确性不易保证;
 - 不易修改。
- 优点:
 - 高效, 灵活。

■ 管程:

- 集中式同步工具: 共享变量及其所有相关操作集中在一个模块中。
- 优点:
 - 可读性好;
 - 正确性易于保证;
 - 易于修改。
- 缺点:
 - 不甚灵活, 效率略低。

第五章：死锁与饥饿

一组进程中的每个进程均等待此组进程中其他进程所占有的、因而永远无法得到的资源，这种现象称为进程死锁，简称**死锁**。

死锁的有关结论：

- (1) 参与死锁的进程数目至少为 2
- (2) 参与死锁的所有进程均等待资源
- (3) 参与死锁的进程至少有两个占有资源
- (4) 参与死锁的进程是系统中当前正在运行的进程集合的一个子集

死锁的条件： (1) 资源独占 (2) 不可剥夺 (3) 保持申请 (4) 循环等待

死锁的处理： (1) 死锁预防 (2) 死锁避免


资源分配图定义： $G=(V,E)$, $V=P \cup R$, $P=\{p_1, p_2, \dots, p_n\}$, $R=\{r_1, r_2, \dots, r_m\}$,

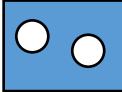
$E=\{(p_i, r_j)\} \cup \{(r_j, p_i)\}$, $p_i \in P$, $r_j \in R$.

申请边(p_i, r_j): p_i 申请 r_j ;

分配边(r_j, p_i): r_j 分配 p_i ;

图示：

进程: 

资源: 

申请边：由进程到资源类；

分配边：由资源实例到进程

资源分配图的约简：

- (1) 寻找一个非孤立且没有请求边的节点 p_i , 若无算法结束

- (2) 去除 p_i 的所有分配边使其成为一个孤立节点;
- (3) 寻找所有请求边都可满足的进程 p_j , 将 p_j 的所有请求边全部改为分配边;
- (4) 转步骤 1

算法结束时, 若所有结点均为孤立结点, 则称资源分配图是可以完全约简的, 否则称为不可完全约简。

死锁定理: S 为死锁状态的充分必要条件是 S 的资源分配图不可完全约简。

死锁的预防: 预先分配策略、有序分配策略

采用**预先分配策略**, 进程在运行前一次性地向系统申请它所需要的全部资源。如果系统当前不能满足进程的全部资源请求, 则不分配资源, 此进程暂不投入运行。如果系统当前能够满足进程的全部资源请求, 则一次性地将所申请的资源全部分配给申请进程。破坏了保持申请条件来避免死锁发生。

缺点: (1) 资源利用率低 (2) 进程在运行前可能并不知道它所需要的全部资源。

采用**有序分配策略**规定进程必须按照资源编号由小到大的次序申请资源。破坏了循环等待条件来避免死锁发生。

缺点: (1) 资源编号困难 (2) 为保持按序申请, 某些暂时不用的资源也需提前申请, 牺牲资源利用率。

银行家算法: 资源分配算法、安全性检测算法

死锁检测时刻: 定时监测、资源利用率降低时检测

当等待时间给进程的推进和响应带来明显的影响时, 就称发生进程**饥饿**。当饥饿到一定程度的进程所赋予的任务即使完成也不再具有实际意义时, 称该进程被**饿死**。

饿死与死锁的联系: 都是由竞争资源引起的。

饿死与死锁的区别:

(1) 从进程状态考虑，死锁进程都处于等待态。忙式等待的进程并非处于等待态，但是有可能被饿死。

(2) 死锁进程等待永远不会被释放的资源，饿死进程等待会被释放却不会分配给自己的资源，其等待时限没有上届。

(3) 死锁一定是发生了循环等待，而饿死则不然。这也表明通过资源分配图可以检测死锁存在与否，但却不能检测是否有进程饿死。

(4) 死锁一定涉及多个进程，而饥饿或被饿死的进程可能只有一个。

忙式等待条件下发生的饥饿称为**活锁**。

第六章：主存储器管理

存储管理功能

存储分配和去配的对象：内存、外存(相同方法)

存储分配和去配时刻：进程创建、撤销、交换、长度变化(栈溢出, exec)

存储共享是指两个或者多个进程共用内存中的相同区域

存储共享的目的：节省内存、相互通讯

存储共享的内容：代码、数据

存储保护：防止地址越界、防止操作越权

将逻辑地址转换为物理地址的过程称为**地址映射**。完成地址映射的软件机构称为存储管理部件。

内存资源管理

内存分区

根据分区时刻划分：静态分区和动态分区

静态分区：在系统运行之前就将内存空间划分为若干个区域。

动态分区：在系统运行的时刻中划分内存空间

根据分区大小划分：等长分区和异长分区

等长分区：将存储空间划分为若干个长度相同的区域。

异长分区：将存储空间划分为若干个长度不同的区域。

动态+异长（段式、界地址）

内存分配

静态等长分区的分配常用于页式存储管理方式和段页式存储管理方式中。

静态等长分区的分配算法：（1）位图 （2）空闲页表 （3）空闲页链

动态异长分区的分配常用于界地址存储管理方式和段式存储管理方式中。

动态异长分区的分配算法：

（1）**最先适应算法**（FF）是指对于存储申请命令，选取满足申请长度要求且起始地址最小的空闲区域。

优点：尽量使用低地址空间，高地址空间可能形成较大的空闲区域

缺点：可能分割较大的空闲区

（2）**下次适应算法**（NF）自上次分配空闲区域的下一个位置开始，选取第一个可满足的空闲区域。

优点：空闲区域分布和分配更均匀

缺点：可能分割大空闲区域

（3）**最佳适应算法**（BF）分配时取满足申请要求且长度最小的空闲区域

优点：尽量不分割大的空闲区域

缺点：可能会形成很小以致以后无法利用的空闲区域，称为碎片

(4) **最坏适应算法** (WF) 分配时取满足申请要求且长度最大的空闲区域。

优点：避免产生碎片

缺点：分割大的空闲区域，当遇到较长存储空间的命令时，无法满足要求的可能性较大

解决碎片问题的方法是移动所有的占有区域，以使所有的空闲区域连成一片，这个过程称为**紧凑**。

界地址管理方式（单对界）

内存空间划分：采用动态异长分区方法，整个内存被动态地划分为若干个长度各异区域。

进程空间划分：一个进程由连续的区域构成，假设进程的长度为 l ，则逻辑地址 $0 \sim l-1$

所需表目：内存分配表和空闲区域表

所需寄存器：

(1) 基址寄存器 b ：保存运行进程起始地址 (2) 限长寄存器 l ：保存运行进程长度

地址映射的步骤：

(1) 由程序确定逻辑地址；

(2) 由逻辑地址与限长寄存器的内容相比较以判断是否满足 $0 \leq a \leq l-1$ 。如果不满足则为越界，发生越界中断

(3) 由逻辑地址与首址寄存器的内容相加得到物理地址。

单对界限制一个进程在内存中只占有一个连续区域，双对界则允许一个进程在内存中占有两个连续的区域。这两个区域一个可用于保存代码，另一个可用于保存数据。

交换是指进程在内存空间与外存空间之间的动态调度，它是缓解内存空间使用矛盾的一种有效方法。

覆盖是将较大程序装入较小进程空间的一种技术。其思想是只将全局代码和数据静态地放在内存中，其他部分则分阶段地动态装入，后装入的对象重复使用以前对象所占用的空间。

页式存储管理

内存空间划分：被静态地划分为若干个等长的区域，每个区域称为一个页框 (frame)，每个页框通常有 2^i 个单元。

物理地址=页框首址+页内地址=页框号 $\times 2^i$ + 页内地址

进程空间划分：被静态地划分为若干个等长的区域，每个区域称为一个逻辑页面

逻辑地址=逻辑页首址+页内地址=逻辑页号 $\times 2^i$ + 页内地址

所需表目：页表、总页表

所需寄存器：页表首址寄存器、页表长度寄存器、快表

访问页号在快表中的次数与总访问次数之比称为命中率。

地址映射的步骤：

- (1) 由指令产生逻辑地址 (p,d) 。
- (2) 由逻辑页号 p 查快表得到页框号 f

如果找不到：

(a) 由逻辑页号 p 与页表长度寄存器的内容 l 相比较以判断是否满足 $0 \leq p \leq l-1$ 。如果不满足则为越界，发生越界中断。

(b) 由逻辑页号 p 与页面首址寄存器中的内容 b 查页表得到页框号 f。

(c) Parbegin

由页框号 f 与页内地址 d 合并得到物理地址 (f,d)

将 (p,f) 送入快表中，若快表已满，则按置换算法淘汰一个。

Parend

(3) 由页框号 f 与页内地址 d 合并得到物理地址 (f,d)

有效访问时间 EAT=快表命中率*(快表访问时间+内存访问时间)+快表不中率*(快表访问时间

+ (n+1) × 内存访问时间) ns

反置页表是面向内存物理页框的, 即对应内存的每个物理构架设置一个表项, 表项的序号就是物理页框号 f, 表项的内容则为进程标识 pid 与逻辑页号的有序对。(使用散列技术)

段式存储管理

内存空间划分: 被动态地划分为若干个长度各异区域, 每个区域成为一个物理段

物理地址=段首址+段内地址

进程空间划分: 被静态地划分为若干个长度各异区域, 每个区域成为一个逻辑段。

所需表目: 段表、空闲表

所需寄存器: 段表首址寄存器、段表长度寄存器、快表

地址映射的步骤:

- (1) 由指令产生逻辑地址 (s,d)
- (2) 由段号 s 查找快表得到段首址 b' 和段长 l'

如果找不到:

(a) 将段号 s 与段表长度寄存器的内容 l 相比较以判断是否满足 $0 \leq s \leq l-1$ 。如果不满足, 则为越界, 发生越界中断。

(b) 由段号 s 与段表首址寄存器的内容 b 查找内存段表得到段首址和段长

(c) parbegin

由段首址 b' 和段内地址 d 相加得到物理地址 $b' + d$

将 (s, b' , l') 送入快表。如果此时快表已满, 则按照置换算法淘汰一个。

parend

(3) 将段内地址与段长相比较以判断是否满足 $0 \leq d \leq l'-1$ 。如果不满足, 则为越界, 发生越界中断。

(4) 由段首址 b' 和段内地址 d 相加得到物理地址 $b' + d$

段页式存储管理

内存空间划分：被静态地划分为若干个长度相同的区域

物理地址=(页架号,页内地址)=(f, d)

进程空间划分：被静态地划分为若干个长度不同的区域。

所需表目：段表、页表、总页表

所需寄存器：段表首址寄存器、段表长度寄存器、快表

地址映射的步骤：

(1) 由指令产生逻辑地址 (s, p, d)

(2) 由段号和逻辑页号 (s, p) 查找快表得到页框号 f

如果找不到：

(a) 将段号 s 与段表长度寄存器中的内容 l 相比较以判断是否满足 $0 \leq s \leq l - 1$ 。如果不满足则为越界，发生越界中断。

(b) 由段号 s 与段表首址寄存器中的内容 b 查找段表得到页表首址与页表长度 (b', l')。

(c) 将逻辑页号与页表长度相比较以判断是否满足 $0 \leq p \leq l' - 1$ 。如果不满足则为越界，发生越界中断。

(d) 由逻辑页号 p 与页表首址 b' 查找页表得到页框号 f 。

(e) parbegin

由页框号 f 与页内地址 d 合并得到物理地址 (f, d)

将 (s, p, f) 送入快表。如果此时快表已满，则按照某种置换算法淘汰一个。

Parend

(3) 由页框号 f 与页内地址 d 合并得到物理地址 (f, d)

第七章：虚拟存储系统

虚拟内存管理的工作基于**局部性原理**。

局部性原理：

(1) 时间上的局限性。一条指令的执行与该指令的下一次执行，以及一个数据的一次访问与该数据的下一次访问，在时间上是相对集中的。

(2) 空间上的局限性。程序中相邻的指令与相邻的数据在执行时通常会被集中用到。

虚拟存储是一种借助外存空间，允许一个进程在其运行过程中部分装入内存的技术。

外存储器管理技术

外存空间是指磁盘等存储型设备上的存储区域，通常在逻辑上可以划分为 4 个主要部分：

输入井、输出井、文件空间、交换空间。块是外存空间分配的基本单位，也是数据传输的基本单位。

外存空间分配：空闲块链、空闲块表、位图

虚拟页式存储管理：进程运行之前，部分页面被装入内存，另一部分页面（或全部页面）

被装入外存储器。

内存页框分配策略（静态策略）

(1) 平均分配 (2) 按进程长度比例分配 (3) 按进程优先级比例分配 (4) 按进程长度和优先级别比例分配

外存块的分配策略：静态分配、动态分配

静态分配：外存保持进程的全部页面

优点：速度快，淘汰时不必写回(未修改情况) 缺点：外存浪费

动态分配：外存仅保持进程不在内存的页面

优点：节省外存 缺点：速度慢--淘汰时必须写回

页面调入时机:

(1) 请调: 当页故障发生时进行调度, 即当访问某一页面而该页面不在内存时由动态调页系统将其调入内存。

(2) 预调: 又称先行调度, 是在页故障发送之前进行调度, 即当一个页面即将被访问之前就由动态调页系统将其调入内存。

置换算法: 全局置换算法、局部置换算法

最佳算法 (OPT) 是淘汰以后不再需要的或者在最长的时间以后才会用到的页面。(不可实现)

先进先出算法 (FIFO) 淘汰最先进入内存的页面。(会出现 Belady 异常)

最近最少使用算法 (LRU) 淘汰最后一次访问时间距离当前时间间隔最长的页面。

最近不用的先淘汰算法 (NUR) 淘汰最近一段时间内未用过的页面。设置访问位和修改位, 每隔固定的一段时间将所有引用位清零, 按照下面次序进行淘汰。

(1) 引用位 = 0, 修改位 = 0

(2) 引用位 = 0, 修改位 = 1

(3) 引用位 = 1, 修改位 = 0

(4) 引用位 = 1, 修改位 = 1

最不经常使用的先淘汰(LFU): 淘汰访问次数最少的页面。

最频繁使用算法(MFU): 淘汰访问次数最多的页面。

二次机会算法 (second chance) : 淘汰装入最久且最近未被访问的页面

时钟算法(clock algorithm): 将页面组织成环形, 有一个指针指向当前位置。每次需要淘汰页面时, 从指针所指的页面开始检查。如果当前页面的访问位为 0, 即从上次检测到目前, 该页没有访问过, 则将该页替换。如果当前页面的访问位为 1, 则将其清 0, 并顺时针移动

指针到下一个位置。重复上述步骤直至找到一个访问位为 0 的页面。

改进的时钟算法：考虑修改标志 m

颠簸(thrashing)：又称抖动，是指页面在内存与外存储器之间频繁地调度，以致系统用于调度页面的时间比进程实际运行所占用的时间还要长。

颠簸的原因：（1）分给进程的页框数过少（2）页面置换算法不合理（3）程序结构

颠簸的处理：（1）增加分给进程的页框数（2）改进页面置换算法（3）改进程序结构

工作集：进程在一段时间之内活跃地访问页面的集合。工作集模型的精确度与窗口尺寸 (Δ) 密切相关。如果 Δ 值过小，程序的活动页面不能全部装入内存，页故障率高。如果 Δ 值过大，允许并发执行的进程的个数降低。

虚拟段式存储管理：进程运行前，全部装入外存，部分装入内存，访问段不再内存时，发生缺段中断。

静态连接：在程序运行之前将各段连接在一起，该连接是由连接装配程序完成的。

动态连接：在程序运行过程中需要某一段时才能将该段连接上，该连接是由操作系统来完成的。

静态连接的缺点：

（1）连接时间长 （2）所产生的目标代码长 （3）所连接的段在程序运行过程中不一定都能用得到

虚拟段页式存储系统：

所需表目：段表、页表、总页表、共享段表、段名-段号对照表

特性 管理方式	地址空间	存储分配	存储碎片	存储共享	存储保护	动态扩充	动态连接
虚拟页式	一维	简单	无	不便	不便	不可	不可
虚拟段式	二维	复杂	有	方便	方便	可以	可以
虚拟段页式	二维	简单	无	方便	方便	可以	可以

各种虚拟存储管理系统特性比较

第八章：文件系统

文件是具有符号名且在逻辑上具有完整意义的信息项的有序序列。

文件与管理信息资源的程序集合称为**文件系统**。

文件的访问方式：（1）顺序访问：从文件起始位置开始顺序访问、从文件中间某处开始顺序访问（2）随机访问：按记录编号随机访问、按关键字(key)随机访问

文件的组织

逻辑组织：流式和记录式。流式文件以字节为基本单位。记录式文件以记录为基本单位。

记录式文件分为等长记录（优点：处理方便，速度快；缺点：空间浪费）和不等长记录（优点：省空间；缺点：处理不便，速度慢）

物理组织：顺序结构、链接结构、索引结构、散列结构、倒排结构

顺序结构：一个文件占有若干个连续的物理块，其首块号和块数被记录在文件控制块中。

优点：访问速度快 缺点：文件长度增加起来比较困难

链接结构：一个文件占有若干个不连续的存储块，各块之间以指针相连，其首块号和块数倍记录于该文件的文件控制块中。

优点：长度易于变化 缺点：随机访问的速度很慢

索引结构：一个文件占有若干个不连续的存储块，这些块的块号被记录于一个索引块中。

优点：访问速度快，长度变化容易 缺点：存储开销大

散列结构：只适用于定长记录和按键随机查找的访问方式，常用于构造文件目录。

倒排结构：以键值和记录地址构成的索引结构

优点：适合不同的查找方式，速度很快 缺点：索引会带来较大的系统开销

文件控制块是标志文件存在的数据结构，其中包含系统对文件管理所需要的全部信息。

用于检索文件的目录称为**文件目录**。保存文件目录的文件称为**目录文件**。

多级目录的优点：

- (1) 便于文件分类，可以为每类文件建立一个子目录
- (2) 查找速度快，因为每个目录下的文件数目较少
- (3) 可以实现文件的连接

文件目录的改进：将文件控制块分为两部分，FCB 主部和 FCB 次部。文件的目录项中仅保存文件控制块的次部。

优点：（1）提高查找速度（2）实现文件连接

根目录：根节点对应的目录。

当前目录：目前正在使用的工作目录。

文件目录的查找算法：（1）顺序查找（2）散列查找（3）对分查找

文件共享的目的：（1）节省存储空间（2）进程相互通信

文件共享的模式：（1）异步使用同一文件（2）同步使用同一文件

文件共享的实现：（1）公共目录（2）连接（3）共享说明

文件的保护是要防止用户对于文件实施非法的或者不适宜的访问。

文件保护的方法：（1）存取控制矩阵（2）访问权限说明（3）分级目录

文件的保密方法：设置口令、加密

外存管理：（1）空闲块表（2）空闲块链（3）位图

保证文件安全的常用措施是**备份**。

将文件由磁盘复制到磁带上的过程称为**转储**。转储可以采用 3 种策略：完全转储、增量转储、差分转储。

原子事务：一组动作或者全部执行完，或者一个也不做。

第九章：设备与输入输出管理

设备管理的功能：（1）设备分配（2）设备处理（3）缓冲管理

设备管理的目标：（1）方便性（2）并行性（3）均衡性（4）独立性

设备的分类：（1）输入输出型设备与存储型设备（2）块型设备与字符型设备（3）独占型

设备与共享型设备

输入输出型设备都是以字符为数据传输的基本单位的,每传送一个字符发生一次输入输出中断。

存储型设备数据传输必须以完整的块为基本单位,即块是存储分配的基本单位,也是数据传输的基本单位。

磁盘的地址转换：设柱面数为 l , 盘面数为 m , 扇区数为 n ; 又设柱面号为 i , 盘面号为 j , 扇区号为 k , 块号为 b

三维转一维

$$b = imn + jn + k$$

一维转三维

$$i = b / (mn)$$

$$j = b \text{ MOD}(mn) / n$$

$$k = b \text{ MOD}(mn) \text{ MOD } n$$

由于读写延迟，扇区编号是不连续的，交错的。

数据传输 (I/O) 方式

- (1) 程序控制查询方式 缺点：忙式等待
- (2) 中断驱动方式 CPU 与设备并行工作，设备多时对 CPU 干扰多
- (3) 内存映射方式
- (4) DMA 方式
- (5) 通道方式

通道连接方式主要有两种：单通道的连接方式和多通道的连接方式。

通道有三种类型：字节多路通道、数组选择通道、数组多路通道。

通道程序执行过程：一条通道指令可以传送一组数据，一个通道程序可以传送多组数据。

多组数据全部传输完毕后才向处理器发一次中断，因而大大地减轻了主机的负担。

对于**同步 I/O**，引发 I/O 操作的进程需要等待物理 I/O 传输完成后才返回；对于**异步 I/O**，引发 I/O 操作的进程在安排好 I/O 操作后即可返回，而并不等物理 I/O 传输完成。

设备无关性：进程申请设备资源时，应当指定所需设备的类别，而不是指定某一具体的设备编号。系统根据当前请求情况以及系统资源分配情况在相应类别的设备中选择一个空闲设备，并将其分配给申请者。优点：（1）提高设备资源利用率（2）程序与设备无关

共享型设备的分配与去配分为有通道形式和无通道形式。

设备调度应考虑的两个因素：公平性、高效性

磁盘引臂调度算法：

- (1) 先到先服务算法 FCFS (2) 最短查找时间优先算法 SSTF (3) 扫描算法（电梯算法）
- (4) 循环扫描算法 (5) N 步扫描算法 (6) 冻结扫描算法

磁盘输入输出参数：

寻道时间 $T_s = nm + s$ 其中 n 为跨越磁道数, m 为跨越一个磁道所用的时间, s 为启动时间

旋转延迟 $T_r = \frac{1}{2r}$ 其中, r 为磁盘转速。

传输时间 $T_t = \frac{b}{rN}$ 其中 b 为读写字节数, r 为磁盘转速, N 为一条磁道上的字节数

访问时间 $T_a = T_s + T_r + T_t = mn + s + \frac{1}{2r} + \frac{b}{rN} = mn + s + \frac{1}{2r} + \frac{1}{rM}$ 其中 M 为一个磁道上的扇区的个数

缓冲: 处理数据到达与离开速度不一致所采用的技术

缓存: 将慢速存储器上活动信息缓冲到快速存储设备上的技术

根据系统设置的缓冲区个数, 可以将缓冲技术分为单缓冲、双缓冲、循环缓冲以及缓冲池 4 种。

私用缓冲: 一个缓冲区与一个固定设备相联系, 不同设备使用不同的缓冲区

公共缓冲: 缓冲区由系统统一管理, 按需要动态分派给正在进行 I/O 传输的设备

缓冲池: 系统中缓冲区的集合。

输入输出进程是基于客户服务器模型的输入输出操作形式。

RAID 级别: 行业标准规定的数据在多个磁盘上的存放方法。

RAID 分条(striping)数据存储方式: 位级分条、块级分条

RAID 衡量指标: 速度、可靠性、成本

RAID 级别的比较

Level	分条粒度	读并发性	写并发性	冗余(容错/开销)
0	块	支持	支持	无
1	块	支持	不支持	镜像
2	位	不支持	不支持	汉明纠错码奇偶校验与恢复
3	位	不支持	不支持	单个奇偶校验
4	块	支持	不支持	块级异或校验
5	块	支持	支持	块级分布式异或校验

利用共享型设备实现的数量较多、速度较快的独占型设备称为虚拟设备。其基本思想是在独占型设备与内存进程之间加入一个共享型设备作为过渡。

不丢失信息的存储器称为稳定存储器。