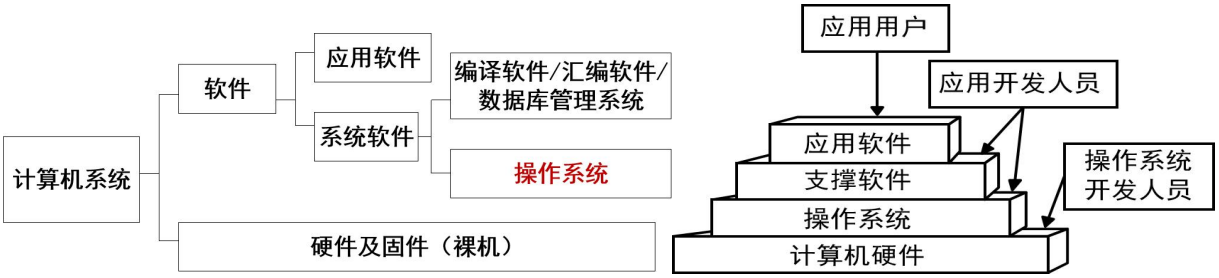
第一章 操作系统概述 **21071003 高立扬**

1. OS 地位、作用和定义

①地位：紧贴系统硬件之上，所有其他软件之下，是各种软件的基础运行平台，是计算机系统的内核和基石，是一种系统软件。**PPT7-8**



②作用：

- OS 作为用户/计算机接口（用户观点）
- OS 作为资源管理者（系统观点）

③定义:操作系统是计算机系统中的一个系统软件，是一些程序模块的集合——它们能以尽量有效、合理的方式组织和管理计算机的软、硬件资源，合理的组织计算机的工作流程，控制程序的执行并向用户提供各种服务功能，使得用户能够灵活、方便、有效的使用计算机，使整个计算机系统能高效地运行。是计算机与用户之间的接口。

2. OS 分类和发展历史

①分类：

- 传统&现代
- 按硬件规模：微机、小型机、大型机 OS
- 按资源共享级别：单任务、多任务、单用户、多用户、单道、多道 OS
- 按交互类型：批处理、分时处理、个人机、实时

②发展历史

- 第一代(1950)电子管时代，无操作系统
- 第二代(1960)晶体管时代，批处理系统
- 第三代(70-80)集成电路时代，多道程序系统
- 第四代(80-)超大规模集成电路时代，速度、容量、稳定性、可靠性、性能都得到逐步提高

③批处理&产生背景

手工操作阶段无操作系统，人机交互效率低，出错误要从头执行，需要管理程序管理用户提交的多程序，需要专业操作员将作业分为若干批作业，有管理程序自动执行。

★单道批处理系统分为联机&脱机批处理，特点为批量性、自动性、顺序性、单道性。但是系统吞吐量低，CPU 常空闲，无人机交互。

★多道批处理系统（多道程序系统），特点为多道运行、宏观并发、微观串行，提高吞吐量和资源利用率，但是仍无交互，作业平均周转时间长。

④分时系统&产生背景

事务性任务的涌现、支持多用户&多任务、多终端计算机、中断技术&通道技术。

★时间片流转，多终端用户共享一台计算机资源，例子：LINUX、UNIX、CTSS、MULTICS 等

特点	描述
多路性	多个用户同时使用一台计算机工作。共享系统资源，提高了资源利用率。
及时性	用户能在很短时间内得到系统响应。
独立性	各用户独立操作，互不干扰，好像独占主机。
交互性	系统能及时对用户的操作进行响应，加快调试过程，缩短了周转时间。

⑤实时系统&产生背景

实时任务出现，实时处理需求。

★软&硬实时系统，相应时间短，系统可靠性高（及时性和可靠性）

3. 现代 OS 特征&功能

①特征：

• **并发**：多个事件在同一时间段内发生

并发(Concurrency)：在单处理器环境下，同一时间段内。

并行(Parallel):在多处理器环境下，两个或多个事件在同一时刻发生。

• **共享**：互斥共享、同时访问

• **虚拟**：将一个物理实体**映射**为若干个对应的逻辑实体，是操作系统**管理系统资源**的重要手段，可提高资源利用率。**时分复用、空分复用**

• **异步**：也称不确定性，指进程的执行顺序和执行时间的不确定性，进程的运行速度不可预知“走走停停”，难以重现系统在某个时刻的状态

②功能：用户/计算机接口、资源管理者、处理器管理、存储器管理、设备管理、文件管理

4. 一些概念

概念名称	解释
监控程序	通过植入目标主机的程序，动态地监视目标主机的屏幕，实现将鼠标和键盘事件传输过去，进行一般操作的程序
多道程序系统（多道批处理）	多个作业同时进入主存 ①共享资源，资源利用率高　②提高吞吐率　③CPU 和 IO 并行工作
多处理系统（多处理器系统）	多台功能相近的处理器，处理器之间彼此可以交换数据，共享内存，I/O 设备，控制器，及外部设备，整个硬件系统由统一的操作系统控制，在处理器和程序之间实现作业、任务、程序、数组极其元素各级的全面并行。
批处理	对作业进行批量处理
分时系统	切换频率高，用户可以在多个程序之间进行交互。多路性（同时性）、交互性、独立性、及时性

（1）引入多道程序设计的目的：**在内存中同时存放多道已开始运行且尚未结束的程序。它们交替运行，共享系统中的各种资源，从而使处理机得到充分利用。**

（2）多道程序设计的优点：**资源利用率高、系统吞吐量大**

第二章 操作系统结构

1. 什么是双重操作模式？为什么要引进它？

为了确保 OS 和所有其它程序和数据**不受任何故障程序影响**，根据运行程序**对资源和机器指令的使用权限**将处理器设置为不同状态。CPU 至少需要两重独立的操作模式。如 2.所述

2. 系统态与用户态的相互转化

（1）系统模式/系统态/管态/特权状态：OS 管理程序运行时的状态，较高级别，可以执行全部指令，使用所有资源，能改变处理器的状态

（2）用户模式/用户态/目态/常态：用户程序运行时的状态，较低级别，只能执行非特权指令

（3）目态→管态：系统调用，如 INT，Trap，Syscall

（4）管态→目态：设置 PSW（程序状态字：寄存 CPU 运行状态的寄存器）

3. 特权指令和非特权指令

（1）特权指令：只能由 OS 程序使用，比如启动 I/O 设备、控制中断屏蔽位、清理内存、加载 PSW 等

（2）非~：用户程序所使用的指令

（3）特权指令一般会引起处理器状态的切换

（4）硬件保护机制：I/O 指令都是特权指令，用户只能通过 OS 进行 I/O 系统调用。确保用户不获得管理模式

4. 用户与 OS 的两种接口，各自的定义和功能

接口口类	细分	定义&功能
命令接口	联机接口（交互式）	使用系统提供的操作命令，交互地控制程序执行和管理计算机系统。如系统管理、环境设置、权限管理、文件管理等。
	脱机接口	以作业说明书的方式提交给系统(批方式）。执行过程中，用户无法干涉。

程序接口（系统调用）	-	操作系统提供给编程人员的唯一接口，编程人员利用系统调用，可以在程序中调用操作系统所提供的子功能。

5. 系统调用：定义、功能

（1）定义：OS 核心中都有一组实现系统功能的过程（子程序），系统调用就是对上述过程的调用。编程人员利用系统调用，向 OS 提出服务请求，由 OS 代为完成。**也是进程与 OS 之间的接口**。

（2）功能：**允许用户级进程请求 OS 服务**

（3）分类：进程控制、文件管理、信息管理、硬件管理、通信

6. 操作系统结构

结构	描述
整体结构（无结构）	①定义：是一组过程的集合，过程之间相互调用 ②特点：关注功能的高效实现，侧重服务功能 ③缺点：错误多，调试难，阅读理解难，增加了维护的难度，忽略了软件工程的设计思想
模块化结构	①定义：OS 按功能进行设计，划分成了若干个具有一定独立性和大小的模块，如：进程管理、内存管理、设备管理、文件管理模块等 ②特点：模块化设计、编码和调试独立完成，调用自由 ③优点：开发效率提高、加速开发过程、提高 OS 可理解性&可维护性、一定的灵活性 ④缺点：模块划分和接口定义不明确，难满足实际要求、模块依赖关系复杂、模块设计工作量随着数量增加而增大
分层式结构	①定义：从资源管理观点出发，把 OS 划分为**若干层次**，某一层代码只能调用低层代码，使得模块之间调用无序到**有序** ②优点 • 功能明确，调用关系清晰，保证设计和实现的正确性 • 低层和高层分别实现，便于扩充；高层错误不影响低层；避免递归调用 • 整体问题局部化，更容易保证 OS 的正确性 • 简化 OS 设计和实现，有利于维护、扩充&移植 ③缺点 • 层定义困难 • 效率差。每执行一个功能，通常要自上而下穿越多层
微内核结构	①定义：从操作系统中去掉尽可能多的东西，而只留一个最小的核心（存放核心功能、管态）。操作系统=微内核+多个服务器（存放大部分功能、目态），微内核提供客户程序和运行在用户空间的各种服务器之间的通信 ②优点： • 良好扩充性：只需利用率添加新功能服务进程，并加入到用户空间，不必修改内核 • 可靠性好：服务器以目态运行，不在管态(不直接访问硬件)，故服务器崩溃不导致 OS 崩 • 便于移植：稳定，更少代码运行在管态，更安全 ③缺点：消息传递比直接调用效率低，但是可以通过提高硬件性能补偿

第三章 进程

1. 进程的定义、特征、作用

（1）定义：一个具有一定独立功能的**程序**在一个**数据集合上**的一次**动态执行**过程。简言之，**进程是程序的一次执行活动。在内存中存放**

结构：代码段 + 数据段 + PCB

（2）特征

特征	描述
动态性	①进程对程序的执行②动态产生&消亡 ③进程在其生命周期内，三态转换
独立性	各进程的地址空间相互独立，除非采用进程间通信手段

并发性	任何进程都可以同其他进程一起向前推进
异步性	每个进程都以其相对独立的不可预知的速度向前推进。

（3）作用：描述了程序的**动态执行**过程、对**资源**的**分配和回收**，反映了 OS 中程序执行的并发性、随机性和共享，提高了对硬件资源的**利用率**，但是带来了空间&时间**开销**，提高了 OS **复杂**度

2. 程序顺序执行、并发执行的特点

执行	定义	特点
顺序执行	具有独立功能的程序独占 CPU 直到得到最终结果	顺序性、封闭性、可再现性
并发执行	一组逻辑独立的程序（段）在**执行时间上客观重叠**	异步性(间断性)、无封闭性（资源共享）、不可再现

3. 进程与程序的区别与联系

（1）进程动态，程序静态：进程是程序的执行；程序是有序代码的集合，通常对应着文件、静态和可复制

（2）进程暂时，程序永久：进程是一个状态变化的过程，会消亡；程序可以长久保存

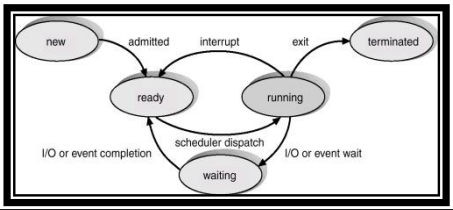
（3）组成不同：进程的组成包括程序、数据和 PCB(进程控制块)

（4）对应关系：通过多次执行，一个程序可对应多个进程；通过调用关系，一个进程可包括多个程序。

4. 进程状态（ 三状态，五状态）及其转换条件

（1）三态模型：运行态、就绪态、等待态

（2）五态模型



状态介绍		状态转换	
状态	介绍	转换	介绍
创建态 new	进程创建：分配资源、PCB 创建	创建->就绪	创建新进程，且有 CPU 资源
就绪态 ready	已有资源、等待 CPU	就绪->运行	进程被调度
运行态 running	已有资源、已有 CPU，执行中	运行->就绪	时间片到 or 被抢占
		运行->等待	等待资源 or 等待某事件发生(**主动**)
等待态 waiting	等待资源、等待 CPU	等待->就绪	资源分配到位，等待事件发生（**被动**）
终止态 terminated	撤销 PCB，回收资源；被迫中止	运行->终止	进程结束 or 发生错误

5. 进程控制块 PCB 作用及其内容

（1）定义：进程在 OS 内的**表示**，是进程**属性**之一。它使一个在多道程序环境下不能独立运行的程序（包含数据），成为一个能独立运行的基本单位，一个能与其它进程并发执行的进程。是 OS 管理控制进程所用的多信息集合。提供了进程调度所用的信息

（2）内容

①进程描述信息：进程标示符、进程名、用户标示符、进程组等

②进程控制信息:当前状态、优先级、代码执行入口地址、程序外存地址、运行统计信息、进程阻塞原因等

③资源占用信息：占用的系统资源列表

④处理器现场保护结构：保存寄存器值等

第四章 进程同步与互斥

1. 进程的同步与互斥

①临界资源：系统中某些资源一次只允许一个进程使用，这样的资源称为临界资源/互斥资源/共享变量。

②临界区：进程中访问临界资源的一段代码

③信号量：信号量就是 OS 提供的管理公有资源的有效手段。是解决并发进程问题的第一个重要进展

④同步：指系统中一些进程需要相互合作，共同完成一项任务。具体说，一个进程运行到某一点时要求另一伙伴进程为它提供消息，在未获得消息之前，该进程处于等待状态，获得消息后被唤醒进入就绪态。

⑤互斥：由于各进程要求共享资源，而有些资源需要互斥使用，因此各进程间竞争使用这些资源

⑥进程间的制约关系：**直接制约：同步（合作）；间接制约：互斥（共享）**

⑦死锁：指多个进程无限等待一个事件，而该事件只能由这些等待进程之一产生。

⑧饥饿：指一个进程一直得不到资源（如：其他进程可能轮流占用资源）。

2. 互斥问题

（1）使用临界区的准则

准则	描述
有空让进	当无进程在临界区时，任何有权使用临界区的进程可进入。
无空等待	不允许两个以上的进程同时进入临界区
多中择一	若进程在临界区，而同时有多个进程要进入临界区，只能其中之一进入临界区，其他进程必须等待
有限等待	任何进入临界区的要求应在有限的时间内得到满足。
让权等待	处于等待状态的进程应放弃占用 CPU。
平等竞争	任何进程无权停止其它进程的运行进程之间相对运行速度无硬性规定。

（2）实现方法：信号量（3.中介绍）、硬件

①关中断：关中断原语、临界区、开中断原语、剩余指令

②专门机器指令

③优点：适用于单处理器 or 共享主存的多处理器系统、进程数目任意、简单易证、借助多个 bool 变量支持多个临界区，一个临界区只需要一个 bool 变量

④缺点：忙等待（实现不了让权等待）、可能饥饿、可能死锁

3. 信号量

（1）定义：信号量就是 OS 提供的管理公有资源的有效手段。是解决并发进程问题的第一个重要进展 。它是一个结构体，存放空闲资源数和阻塞队列。

（2）含义

①空闲资源数：初始为非负整数，非负代表可用资源数，负数代表无可用资源

②阻塞队列：阻塞在该信号量的各个进程的标识

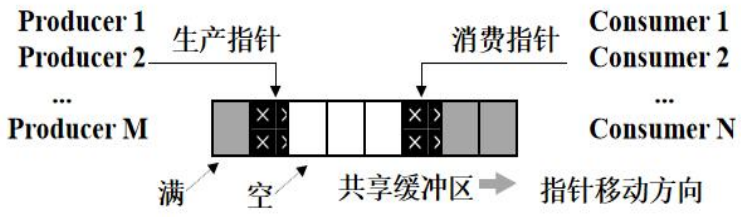
（3）操作

①初始化：将信号量初始化

②P 操作：若信号量非负则减 1，否则阻塞进程

③V 操作：信号量加一，解除一个被阻塞的进程的阻塞态

4. ★★★生产者消费者问题★★★ 10 分大题！！！！！！！



第五章 死锁

1. 定义、发生原因、4 个必要条件

（1）定义：一组进程中，每个进程都**无限等待**被该组进程中**另一进程所占有的资源**，因而**永远无法得到**资源，这种现象称为进程死锁，这一组进程就称为死锁进程。

（2）发生原因

①资源不足导致的资源竞争　　②并发执行的顺序不当（P 操作顺序不当）

（3）必要条件

①互斥条件：一个资源某一时刻只能让一个进程使用

②占有且等待：占有一个资源的同时等待另一个资源

③非抢占（非剥夺）条件：资源不可以抢占，只能由占有它的进程释放

④循环等待条件：一组资源循环被占有

2. 解决死锁的方法：预防、避免、检测、恢复

（1）预防：通过限制如何申请资源的方法来确保至少有一个条件不成立

（2）避免：根据有关进程申请资源和使用资源的额外信息，确定对于一个申请，进程是否应该等待

（3）检测和和恢复：相关算法

（4）忽视死锁：UNIX 认为死锁不可能在系统内发生

死锁检测算法：举例

1. 由于P4没有已分配的资源，标记P4；
2. 令W=（0 0 0 0 1）；
3. 进程P3的请求小于或者等于W，因此标记P3，并令W=W+（0 0 0 1 0）=（0 0 0 1 1）；
4. 没有其他未标记的进程在Q中的行小于或者等于W，因此终止算法。

算法的结果是P1和P2未标记，表示这两个进程是死锁的。

3. 掌握银行家算法：如何判别安全性

（1）设计思想：当用户申请一组资源时，系统必须做出判断：如果把这些资源分出去，系统是否还处于安全状态。若是，就可以分配这些资源；否则，暂时不分配。

（2）数据结构

available 向量	系统中尚未分配的每种资源的总量	available[j]：尚未分配的资源 j 的数量
max 矩阵	各个进程对每种资源的最大需求量(事先声明)	max[i, j]：进程 i 对资源 j 的最大需求量
allocation 矩阵	当前资源分配状况	Allocation[i, j]：进程 i 获得的资源 j 的数量
need 矩阵	每个进程还需要的剩余资源的数量	need[i, j]：进程 i 尚需的资源 j 的数量

（3）**★★★判别安全性算法举例★★★　必考且送分！！！！！**

	最大资源需求M			已分配资源数量U			尚需资源N		
	A	B	C	A	B	C	A	B	C
P1	5	5	9	2	1	2	3	4	7
P2	5	3	6	4	0	2	1	3	4
P3	4	0	11	4	0	0	2	0	6
P4	4	0	5	2	0	4	2	0	1
P5	4	2	4	2	3	1	4	1	0

剩余向量A = （2，3，3）

1. 是否安全？

（2，3，3）→ P4 →（4，3，7）→ P5 →（7，4，11）→ P1 →（9，5，13）→ P2 →（13，5，15）→ P3

（2，0，4）　（3，1，4）　（2，1，2）　（4，0，2）

安全序列：P4，P5，P1，P2，P3

思考： 安全序列是否唯一？

	最大资源需求M			已分配资源数量U			尚需资源N		
	A	B	C	A	B	C	A	B	C
P1	5	5	9	2	1	2	3	4	7
P2	5	3	6	4	0	2	1	3	4
P3	4	0	11	4	0	5	0	0	6
P4	4	0	5	2	0	4	2	0	1
P5	4	2	4	3	1	4	1	1	0

剩余向量A = （2，3，3）

2. 在T0时刻若进程P2请求资源（0，3，4），是否能实施资源分配？为什么？

解：假设能够分配，则P2还需（1，0，0），

但因为 R（0，3，4）> A （2，3，3）

所以不能满足

	最大资源需求M			已分配资源数量U			尚需资源N		
	A	B	C	A	B	C	A	B	C
P1	5	5	9	2	1	2	3	4	7
P2	5	3	6	4	0	2	1	3	4
P3	4	0	11	4	0	5	0	0	6
P4	4	0	5	2	0	4	2	0	1
P5	4	2	4	3	1	4	1	0	1

剩余向量A = （2，3，3）

③ 在②的基础上，若进程P4请求资源（2，0，1），是否能实施资源分配？为什么？

解：因为R（2，0，1）<=N4(2，0，1) 且 < A （2，3，3）

假设可以满足，则N4=(0，0，0)，A=（0，3，2），在此基础上，

(0，3，2) → P4 →(4，3，7) → P5 →(7，4，11) → P1 →(9，5，13) → P2 →(13，5，15) → P3

所以是安全的，因此可以实施分配。

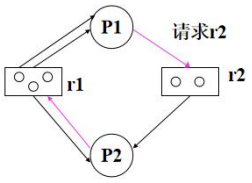
注意！！第二张图和第三张图需要两步判断！！！：①申请资源≤need　②申请资源≤向量 A

4. **★★★银行家算法与资源分配图相结合★★★**

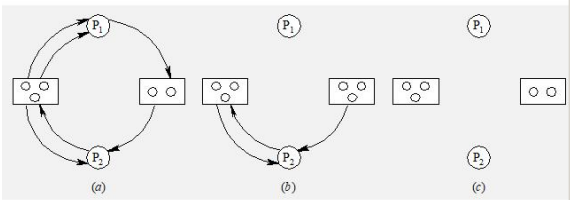
（1）资源分配图 RAG：一个有向图，用于表示某时刻系统资源与进程之间的状态

RAG图中有两种节点和两种边

- 圆圈代表进程，方块代表一类资源。
- 方框中的点：由于一种类型的资源可能有多个，可用方框中的一个点代表一类资源中的一个资源
- 分配边：从资源节点(圆圈)到进程节点(方块)的有向弧表示资源已经分配给进程；
- 请求边：从进程到资源的有向弧表示进程当前正处于阻塞状态，等待资源变为可用

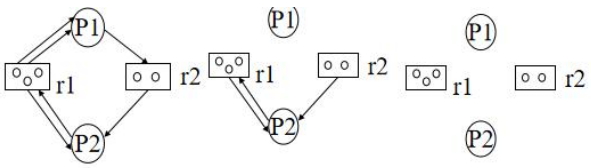


资源分配图



资源分配图的简化

- 资源分配图的化简方法：假设某个RAG中存在一个进程Pi，此刻Pi是非封锁进程，那么可以进行如下化简：
  - 当Pi有请求边时，首先将其请求边变成分配边(即满足Pi的资源请求)，而一旦Pi的所有资源请求都得到满足，Pi就能在有限的时间内运行结束，并释放其所占用的全部资源，此时Pi只有分配边，删去这些分配边（实际上相当于消去了Pi的所有请求边和分配边），使Pi成为孤立结点（反复进行）。



死锁定理：

系统中某个时刻S为死锁状态的充要条件是S时刻系统的资源分配图是不可完全简化的。

在经过一系列的简化后，若能消去图中的所有边，使所有的进程都成为孤立结点，则称该图是可完全简化的；反之的是不可完全简化的。

（2）**封锁进程：指某个进程由于请求了超过了系统中现有的未分配资源数目的资源，被系统封锁的进程。**

（3）非封锁进程：即没有被系统封锁的进程。

（4）死锁检测的办法：由软件检查系统中由进程和资源构成的资源分配图是否构成一个或多个环路。若有环，则存在死锁，否则不存在。一张图就是一个时刻，判断封锁进程时每个进程都要同时判断，不能化简一个判一个

第六章 线程

1. 什么是线程？为什么要引进线程？

（1）线程定义

①进程中的一个实体 ②调度和分派的基本单位（CPU）

③与同进程内的其它线程共享进程所拥有的资源：线程必须在某个进程内执行，它所需的其它资源，如代码段、数据段、打开的文件和信号等，都由它所属进程拥有。

④只拥有运行所必须的资源: 如 PC、寄存器、 栈

（2）引进原因：进程负载很重，局限性大，不能很好地利用多处理器系统。如果将进程理解为在逻辑上 OS 所完成的任务，则线程表示完成该任务的许多可能的子任务之一。

2. 线程和进程的区别？

（1）**（机制）**一个进程可以有多个线程，但至少有一个线程；而一个线程只能在一个进程的地址空间内活动。资源分配给进程，处理机分配给线程。

（2）**（资源）**进程间相互独立；同一进程的各线程间共享（某进程内的线程在其他进程不可见）。

（3）**（通信）**进程间通信采用 IPC；线程间可以直接读写进程数据段来进行通信（共享进程的主存）

（4）**（调度）**线程上下文切换比进程上下文切换要快得多

3. 内核级&用户级的线程

（1）用户级线程

①定义

- 它的维护由应用进程通过线程库来完成
- 线程库：应用进程利用线程库提供创建、同步、调度和管理线程的函数来控制用户线程，无需内核支持。

②特点

- 内核不了解用户线程的存在
- 用户线程切换不需要内核特权
- 调度由应用软件内部进行，通常采用非抢先式和更简单的规则，也无需用户态/核心态切换，所以速度很快。

③缺点

- 一个线程发起系统调用而阻塞，同一个进程的其它线程都被阻塞。
- 一个多线程应用程序不能利用多处理技术。内核一次只把一个进程分配给一个处理器，因此一次进程中只有一个线程可以执行。

（2）内核级线程

①定义

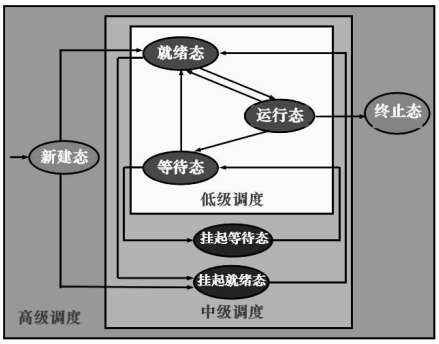
- 有关线程的所有管理工作都在内核完成，应用程序部分没有线程管理的代码，只有一个到内核线程的 API。
- 线程切换由内核完成，内核维护进程和线程的上下文信息

②优点

- 内核可以将同一进程中的多个线程调度到多个处理器上
- 一个线程发起系统调用而阻塞，不会影响其它线程的运行，内核可以调度同一个进程中的另一个线程。

③缺点

- 由于有内核的参与，需要额外开销
- 线程之间的切换需要内核的模式切换。



第七章 CPU 调度 ★★★★★重点★★★★★

1. CPU 调度三类型 **★简答★**

调度类型	描述
高级调度（作业、长程调度）	负责将**作业**调入**内存**，为其**创建进程分配资源**并善后。决定了**多道程序的道数** ①调度对象：作业 ②主要功能：决定外存那写作业调入（接纳多少、哪些作业） ③特点：只用于批处理系统，作业调度运行频率低，几分钟一次

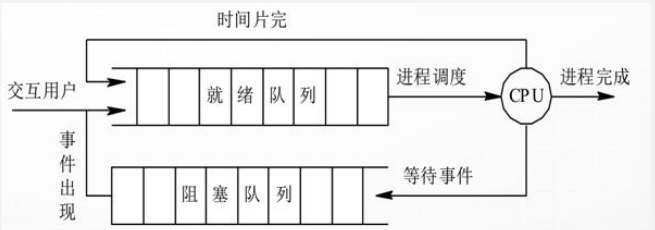
中级调度（中程调度）	①根据存储资源量和进程的当前状态来决定**外存和主存中进程的对换（挂起在外存）** ②决定**主存储器中所能容纳的进程数**，这些进程将允许参与竞争处理器资源。 ③目的：便于内存管理扩充；提高内存利用率和系统吞吐量 ④运行频率介于高级、低级调度之间。
低级调度（CPU 调度）	①动态地把处理器分配给进程。调度对象：进程②决定就绪队列中哪个进程获得 CPU。 ③运行频率高，几十毫秒一次，算法不能太复杂。

2. 调度队列

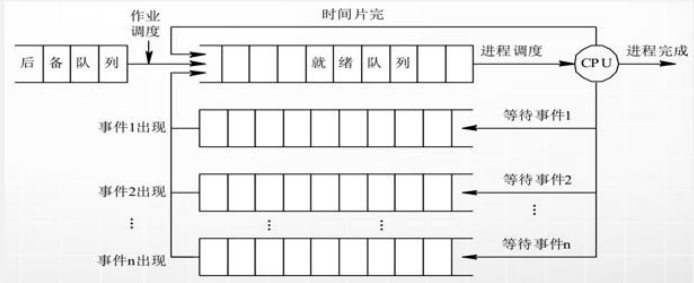
作业和任务的区别

- 作业是任务实体，进程是完成任务的执行实体；没有作业任务，进程无事可干，没有进程，作业任务没法完成。
- 作业概念更多地用在批处理操作系统，而进程则可以用在各种多道程序设计系统。
- 作业调度算法也适用于进程调度，区别是前者是从后备队列中选取，而后者是从就绪队列中选取。进程调度算法多数适用于线程调度。

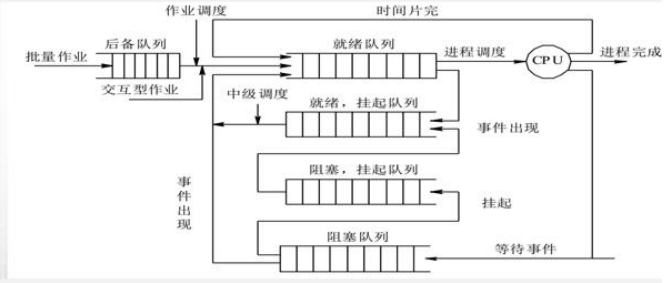
1. 仅有进程调度的调度队列模型



2. 具有高级和低级调度的调度队列模型



3. 同时具有三级调度的调度队列模型



3. 可剥夺调度和不可剥夺调度（抢占&非抢占）的定义和特点 **★简答★**

（1）抢占式：就绪进程可以抢占运行进程的 CPU 资源。应用：最短剩余时间 SRT，多级反馈队列

（2）非抢占式：就绪进程不可抢占运行进程的 CPU，只能等运行进程终止或阻塞让出 CPU

（3）比较：抢占策略可能带来更多开销，但是可以对进程提供更好的服务，避免独占 CPU；高效进程切换机制和更大的主存，可以使大部分程序都存在主存，抢占带来的开销会减少

4. 调度算法的性能评价准则 **★简答★**

追求更高：CPU 利用率、吞吐率（单位时间内完成执行的进程数）

追求更短：周转时间（进程进入就绪队列开始到终止）、等待时间（进程在就绪队列里的时间）、响应时间（进程发出调度请求，到得到 CPU 调度的时间）

公平性：以合理的方式让各个进程共享 CPU

5. 调度程序的功能、时机 **★简答★**

（1）功能：①按某原则决定就绪队列中哪个进程能获得 CPU，并将 CPU 让给它 ②合理分配 CPU、提高利用率、公平调度 ③保存现场、完成上下文切换

（2）时机（只介绍基础四种，实际上有十几种）

①抢占式：running -> ready ; waiting->ready ②非抢占式：running->waiting ; terminated

6. 三个计算 **★计算★**

假设作业 i 提交给系统的时刻是 ts，完成的时刻是 tf，一共 n 个作业那么

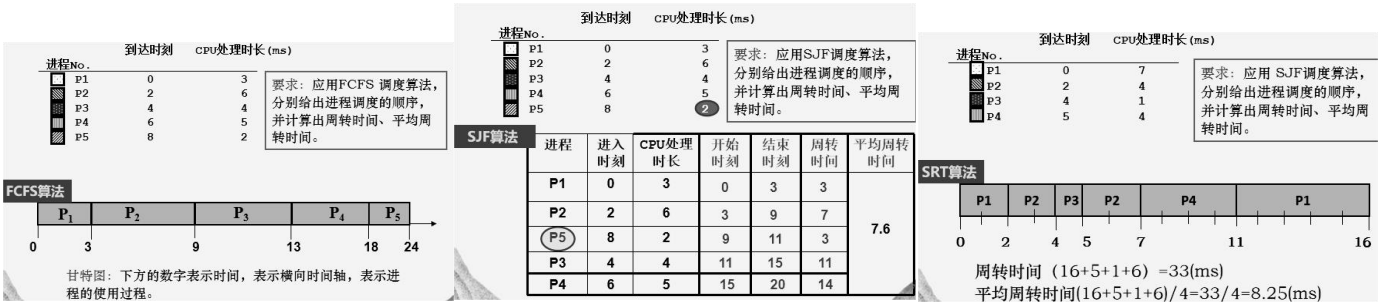
周转时间： 结束时间-提交时间=等待时间+执行时间 $t_i = t_f - t_s$

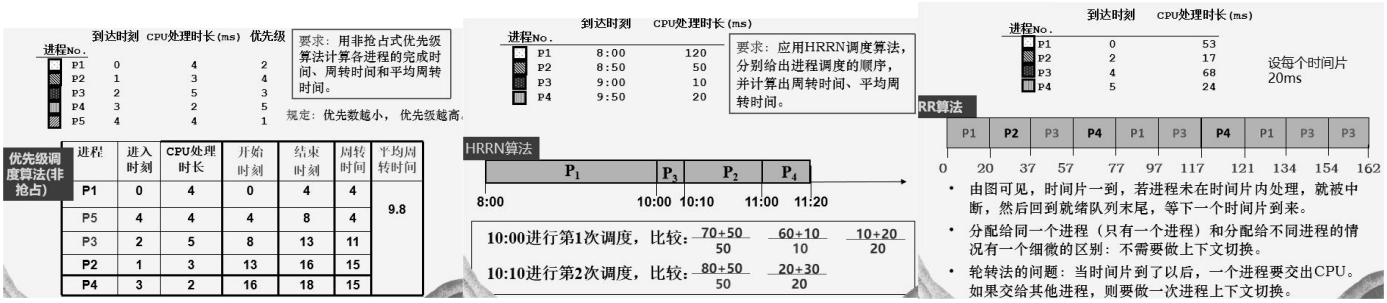
平均周转时间：周转时间总和/作业数（进程数） $T = \left(\sum_{i=1}^{n} t_i\right) * \frac{1}{n}$

7. 调度算法 ★★★★★重点：大题&简答（优缺点）★★★★★

算法名称	算法描述	优缺点
FCFS 先来先服务	就绪了就加入就绪队列，CPU 空了最早进队列的就运行	①优点：公平、实现简单 ②缺点：未考虑任务重要程度；长进程有利，短进程不利；非抢占；不能用于分时系统
SJF 短作业优先	选择当前就绪队列中，处理时间最短的进程；若两进程处理时间相同，遵从 FCFS	①优点：利于短进程 ②缺点：当短进程多时，不利于长进程（饥饿）；**缺少剥夺**机制；预测执行时间必须精准（然而实际情况不可能精准）
SRT 最短剩余时间优先	当一个新的进程到达的时候，如果其所需时间比当前运行进程的剩余时间更少，当前进程就会被挂起，而运行新的进程。	①优点：优化响应时间 ②缺点：不利于长进程；难预测
优先级调度	选择优先级最高的作业或者进程，将 CPU 分配给它。具有相同优先级的进程按 FCFS 顺序调度	①分类：抢占式&非抢占式，静态&动态优先级 ②优点：综合考虑作业等待时间、运行时间、重要程度，进一步优化响应时间 ③缺点：可能导致部分进程饥饿（改善：老化思想）
HRRN 高响应比优先调度	每次选取响应比 R 最高者投入运行，**是 FCFS 和 SJF 的结合；若 n 个进程按顺序在 0 时刻同时进入（所有进程 R 相同情况）那么按 FCFS 原则运行第一顺序的进程** $响应比=\frac{等待时间+服务时间}{服务时间}$	①优点：既照顾短作业，也考虑到达次序，兼顾长作业；公平；吞吐量大 ②缺点：计算响应比开销大
RR 时间片轮转	每个就绪进程获得一个 CPU 时间片。当时间片用完，这个进程的执行被中断，重新挂回到就绪队列。如果进程在时间片用完之前结束也（主动）交出 CPU。假设 n 个就绪进程，时间片 q，每个就绪进程得到 1/n 的 CPU 时间。任何就绪进程最多等待（n-1）q 单位时间。	①优点：在**分时系统 or 事物处理器系统**中非常有效；优化了响应时间 ②缺点：对长作业带来额外的切换开销；时间片长度设置不当会导致响应时间过长；
多级队列调度算法	就绪队列分成多个独立队列，进程所属的队列固定。①实时进程就绪队列（优先级调度） ②分时进程就绪队列 （RR 调度） ③批处理进程就绪队列（优先级调度）	各队列的区别对待，达到一个综合的调度目标。
多级反馈队列调度算法	设置多个就绪队列，进程所属队列可变，即进程可以在不同的就绪队列之间移动①分别赋予各队列不同的优先级，如逐级降低，队列 1 的优先级最高。 ②每个队列执行时间片的长度也不同，规定优先级越低则时间片越长，如可逐级加倍	①优点：是**时间片轮转算法**和**优先级算法**的综合和发展，基于**抢占式**，使用**动态优先级**机制。 ②缺点：可能导致饥饿

算法运算举例：





第八章 内存管理

1. 存储器管理程序的功能

完成逻辑地址到物理地址的映射（重定位）

2. **逻辑地址、物理地址、地址重定位（地址映射）**的概念

（1）物理地址（绝对地址、实地址）：内存中存储单元的地址，可以直接寻址

（2）逻辑地址：用户程序经过汇编 or 编译后的目标代码的地址

①与当前数据在**内存**中的分配情况**无关**

②其**首地址为 0**，其余指令中的地址都**相对于首地址**来编址

③不能用逻辑地址在内存中读取信息，在实现对存储器**访问之前**必须**转换成物理地址**。

④用户程序处理逻辑地址，它不能看到真正的物理地址

（3）地址重定位：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。由硬件实现。

①静态重定位：加载时生成绝对地址　　②动态重定位：虚拟存储

3. 连续内存分配（固定分区、可变分区）

（1）固定分区

①定义：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。划分由 OS 或管理员来决定，一旦划分结束，在整个执行过程中每个分区的长度和内存的总分区个数将保持不变。每块分区需要一个**调度队列**

②优点：适用于**多道程序系统和分时系统**；支持多个程序**并发**执行；有一定灵活性，需要很小 OS 软件和处理开销

③缺点：**难以**进行内存分区的**共享**，可能存在内碎片和外碎片。分区数目和大小固定，限制了活跃进程，降低了效率；

• 内碎片：占用分区之内未被利用的空间；　　外碎片：占用分区之间难以利用的空闲分区（小空闲分区）

（2）动态分区

①定义：内存不是预先划分好的，而是当进程装入时，根据进程的需求和内存空间的使用情况来决定是否分配。

②优点：没有内碎片

③缺点：在开始时是很好的，但最后，导致在存储器中出现很多空洞－外部碎片

• 解决办法是压缩：OS 不时地移动进程，将它们放在一起，并且使所有空闲空间连成一片。十分费时

（3）分配算法

①首次适配法：按分区的先后次序查找（按分区按起始地址递增的顺序排列），分配第一个足够大的分区分配给进程。

• 简单快速，但会产生很多小分区，查找开销大，需要经常压缩

②最佳匹配法：找到其大小与要求相差最小的空闲分区（按空闲区大小从小到大的顺序排列）

• 整体卡莱，外碎片还是会形成很多。但是较大的空闲分区可以被保留

③最差匹配法：找到最大的空闲分区进行分配（按空闲区大小从大到小的顺序排列）

• 基本不留小空闲分区，但较大空闲分区不被保留

4. 页式管理：原理、数据结构、地址、地址映射过程、特点

（1）背景：允许进程的物理地址空间可以是非连续的

（2）一些概念（提炼出原理）

①帧（页帧）frame：**主存**被划分成大小相等的块，且块相对比较小，称为一帧，大小必须是 2 的幂

②页（页面）page：每个**进程**的逻辑内存也被划分成同样大小的块，称为一页，大小必须是 2 的幂

③**一帧可以保存一页数据**，较大的进程需要较多的页（帧）

④当一个进程被装入时，它的**所有页都被装入到可用帧中**，并且可以建立一个**进程页表**。

⑤每个进程浪费的空间仅是由进程最后一页的一小部分形成的内部碎片。

（3）数据结构

①进程页表：**每个进程有一个页表**，描述该进程占用的**物理页面**及**逻辑排列**顺序，并能够实现**地址映射**

②物理页面表：**整个系统有一个物理页面表**，描述**物理内存空间的分配使用状况**。

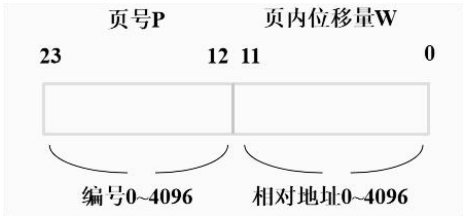
③页的大小

（4）地址变换 **★★★重点：大题！！！！★★★**

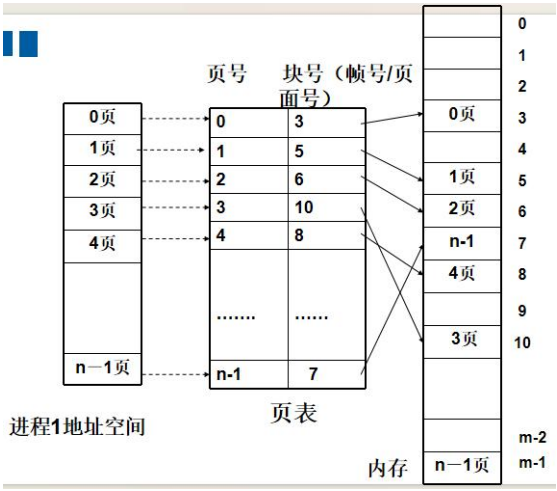
指令所给出地址分为两部分：逻辑页号，页内偏移地址：查进程页表，得帧号—>物理地址

2、考虑一个由8个页面，每页1K字节组成的逻辑空间，把它映射到由32个物理块组成的存储器。问：
（1）有效的逻辑地址有多少位？
（2）有效的物理地址有多少位？

- **物理存储器是32K。**
- **（1）逻辑地址有13位**
  （2）物理地址有15位



- 一个n＋m位的地址：n位页号，m位偏移量。
- 提取页号，即逻辑地址最左边的n位；
- 以这个页号为索引，查找该进程页表中相应的帧号K
- 该帧的起始物理地址为k×2ᵐ，被访问字节的物理地址是这个数加上偏移量。一般物理地址不需要计算，它可以通过把偏移量添加在帧号后面来构造。



（5）高速联想寄存器 TLB（快表）

是快速闪存，由键和值组成。只有当 TLB 失效后，才需要访问页表。TLB 中的条目可以进行替换。

5. 碎片（内碎片，外碎片），怎么产生，如何解决？

（1）内碎片:分区内的（分页、分段），当一个进程不能完全使用分给它的固定内存区域时就产生了内碎片

（2）外碎片:分区之间的。进程频繁地申请&释放内存时,可能会出现大小不一的空闲内存块散布在内存中产生

（3）解决：紧缩

①定义：移动内存内容，以便所有空闲空间合并成一整块。

②条件：允许进行动态重定位可以首先移动程序和数据，然后再根据新基地址的值来改变基地址寄存器。

③实现：一种简单的方法是将所有进程移动到内存的一端，而将所有的空闲区(孔)移动内存的另一端，以生成一个大的空闲块　　④开销大

6. 段式内存管理

（1）原理：将进程的地址空间按内容或函数关系划分为若干个段(segment) ，每个段定义一组逻辑上完整的程序或数据，如一个进程可以被划分为主程序段、子程序段、数据段、堆栈段等。

（2）数据结构：进程段表、系统段表、空闲段表

（3）地址变换

①地址组成：一个 n＋m 位的地址：n 位段号，m 位偏移量。

②地址变换：提取段号，即逻辑地址最左边的 n 位；段号为索引，查找该进程段表中该段的起始物理地址偏移量和段长度进行比较，如果偏移量大于该长度，则地址无效。物理地址为该段的起始物理地址加上偏移量的和。

（4）优点：①没内碎片，外碎片可以紧缩②便于改变进程占用空间的大小③分段对程序员可见，方便模块化程序设计④便于保护和共享

（5）缺点：进程全部装入内存

7. 段页式内存管理

（1）原理：用户的地址空间由程序员划分成许多段，每个段又划分成许多固定大小的页，页的大小等于主存中帧的大小。存储管理的分配单位是：段，页

（2）数据结构：段式+页式的数据结构

（3）地址变换

①逻辑地址的组成：段号、页号、页内偏移地址，程序员可见的仍是段号和段内偏移量。从系统角度看，段偏移量是指定段中的一个页号和页内偏移量。

②地址变换：先查段表，再查该段的页表

（4）特点：是页式和段式存储管理的结合，由于开销比较大，一般只用于大型机系统中。

第九章 虚拟内存

1. 虚拟存储器

（1）概念：具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的一种存储器系统。

- 实质：物理上不存在，利用海量外存进行内存“空间”的扩展。
- 逻辑容量：取决于内存容量和外存容量之和。

（2）原理：局部性原理（时间&空间）：程序在执行过程中的较短时期，指令地址和指令的操作数地址也分别局限在一定区域。指令的执行和数据的访问集中在较短时期，存储在较小区域内

（3）实现方法 ①虚拟页式内存管理 ②虚拟段式内存管理 ③虚拟段页式内存管理

（4）特征：离散性、多次性、对换性、虚拟性

2. 请求页式（虚拟页式）管理

（1）基本原理：基本分页系统+请求调页功能+页面置换功能。把进程的虚拟地址空间、内存空间进行划分，划分为大小相等的小空间。引入页表机制和缺页中断机构

（2）缺页中断

①定义：在地址映射过程中，在页表中发现所要访问的页不在内存（发生时机），则产生缺页中断

②处理过程：操作系统接到此中断信号后，就调出缺页中断处理程序，根据页表中给出的外存地址，将该页调入内存，使作业继续运行下去。

- 此时有空闲块，直接分配；若没有，要按照算法淘汰；若此页在内存期间被修改过，需要写回外存

（3）地址变换：地址变换是由硬件自动完成的。当硬件变换机构发现所要求的页不在内存时，产生缺页中断信号，由中断处理程序做出相应的处理。

3. **页面置换算法 ★★★重点：大题！！！！★★★**

（1）**缺页率**=缺页次数/页面访问的总次数*100%

（2）最佳页面置换算法 OPT

①思想：淘汰以后永不使用的，或下次访问距离当前时间最长的页。

②特点：理论缺页率最低、性能最佳，实际上因为无法预知未来而实现困难，常用语评价其他算法

（3）先进先出置换算法 FIFO

①思想：淘汰最先进入内存的页面，即驻留内存时间最长的页面。

②特点：实现简单；与实际运行规律不符（对常访问的页面不利）

③Belady 异常：缺页率随页帧分配数量增加而增加

（4）最近最久未使用置换算法 LRU

①思想：淘汰在最近一段时间最久未被使用（访问）的页面。

②特点：性能较好；实现复杂

③用栈实现：利用栈保存当前使用的各个页面的页面号。 栈顶是最新被访问页面的页面号，栈底是最近最久未使用页面的页面号。

（5）clock 置换算法

①思想：环形缓冲区，设置访问状态，通过指针访问

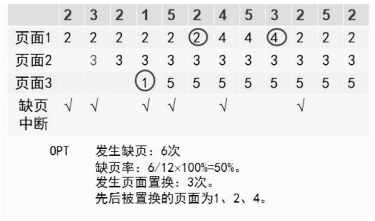
②原版 clock 算法：要置换的时候，指针向下搜，U 为 1 就设置为 0 并推进，否则置换当前页，指针向后。

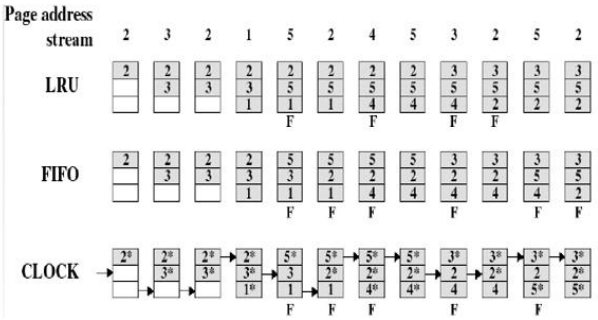
③改进的 clock：U 代表使用情况，M 代表最近修改

先搜并替换第一个 U=M=0 的；否则搜并替换第一个 U=0 且 M=1 的，过程中若遇到 U=1 的就置零；loop

（6）最不经常使用置换算法 LFU

①思想：选择到目前为止被访问次数最少的页面





注意！！！本题访问顺序可能会按逻辑地址给出，需要先进行转换（换成页号），再答题！！！

4. 请求段式（虚拟段式）管理的原理

一个进程只有部分分段放在内存就可以运行。先把当前需要的一个段或者几个段装入内存，其它段仅在调用时才动态装入。

5. 虚拟段页式

（1）原理：是虚拟页式和虚拟段式存储管理的结合，由于开销比较大，一般只用于大型机系统中。存储管理的分配单位是：段，页

（2）地址变换：先查段表，再查该段的页表。产生缺段中断和缺页中断。

6. 快表及其作用

高速联想寄存器 TLB。TLB 只包含少数的页表条目，CPU 产生一个逻辑地址之后，页码发送到 TLB，**TLB 被访问一次**，如果找到这个页码，那它的帧码就可以立刻使用。不在的话，称为未命中，则需要**再访问一次页表**，如果还没命中，**产生缺页中断并再次访问页表**。最后**存取内存数据**。

第十章 文件管理

1. 基本概念

（1）文件：**记录在外存上的相关信息的具有名称的集合**（文件是具有符号名的数据项的集合）

①文件体：文件本身的信息

②文件说明：文件存储和管理相关的信息。如：文件名、文件内部标识、文件存储地址、访问权限&时间

（2）记录：一组**相关数据项的集合**。如：一个雇员记录：名字、职工号、工作单位等

（3）数据项（域）：**记录的基本数据单元**。包括：值和长度两部分，**描述记录的某个属性**。

（4）（文件）目录：为了加快对文件的检索，一般将文件控制块集中在一起进行管理。这种**文件控制块的有序集合称为文件目录**。**文件控制块就是其中的目录项**。类似于图书馆中的图书目录。

（5）文件控制块 FCB：是操作系统用来描述和控制文件的数据结构。每一个文件都有一个 FCB，包含 :基本信息、存取控制信息和使用信息。例如：文件名，文件类型，使用信息，地址信息，……

（6）路径：任何文件可以按照根目录或主目录向下到各个分支最后达到该文件的路径来定位，这一系列目录名和最后达到的文件名自身组成了该文件的路径名

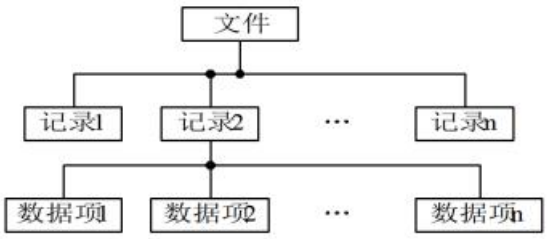
①绝对路径：从根目录开始到指定文件　　②相对路径：从当前工作目录开始

（7）当前目录　　**★简答★**

①定义：对交互用户或进程而言，总有一个当前路径与之相关联，称作**工作目录**。文件通常按照相对于工作目录的方式被访问。当交互式用户登录时，或者当创建一个进程时，默认的工作目录是用户目录。

②访问：系统将当前工作目录中的内容**复制到内存缓冲区**中，在访问文件时先访问内存中的工作目录。当要访问的文件不在当前目录中时，再访问外存中的目录，**提高了查找速度**。

（8）文件管理系统



①一组**系统软件，为用户和应用程序提供服务**

②功能：

- 统一管理文件的存储空间，实施存储空间的分配与回收。
- 实现文件的按名存取。
- 实现文件信息的共享，并提供文件的保护和保密措施。
- 提供与 I/O 的统一接口。
- 负责文件的创建、删除、读写、修改、复制和存取控制等操作。

③目标

- 满足数据管理的要求和用户的需求
- 最大限度地保证文件中的数据有效，减少或消除丢失或破坏数据的可能性。
- 优化性能，从系统的角度提高吞吐量，从用户的角度减少响应时间。
- 为各类存储设备提供 I/O 支持和 I/O 接口例程集。

2. 文件系统的功能和目的

（1）操作系统中的一个重要组成部分

（2）基本功能（目的）：提供高效、快速和方便的**信息**（计算机系统中的重要资源）**组织、存储和访问功能**。

3. 文件逻辑结构 **★★★简答！！！★★★**

（1）定义：从用户观点出发讨论文件内部的逻辑结构或用户访问模式，由用户访问记录的方式确定。

- 独立于外存上的物理存储，是用户可以直接处理的数据及其结构。
- 强调文件各信息项的构成方式和用户的存取方式。

（2）基本要求：①提高检索记录的速度②便于修改记录③降低文件的存储费用

（3）分类

①无结构文件：它们自身不提供任何数据结构，文件内容以字节流的形式存在。节省存储空间

②有结构文件：是由若干记录构成，文件的记录可以有多种排列顺序。

- 按记录长度分类　　- 按组织方式分类（顺序文件、索引文件、索引顺序文件）

（4）**与物理结构的区别：**文件的逻辑结构是从**用户观点**出发，所看到的是**独立于**文件**物理特征**性的文件组织形式，是用户可以**直接处理**的数据与其结构。文件的物理结构是文件在**存储设备上的存储方式，**是从**系统观点**出发，以文件在**物理介质**上的存放方式来研究文件

4. 文件的存取方法；存取方法与存取介质的关系

（1）顺序存取法: 按照文件信息的逻辑顺序(记录的排列顺序)依次存取。如：为了存取记录 Ri　,必须先通过记录 R1 R2… Ri-1 。

（2）随机存取法(直接存取）:可以按任意的次序对文件进行读写操作。如：可根据记录的**编号来直接存取**文件中的任意一个记录。对**流式文件**或者定长的记录式文件**容易**确定存取位置。对不定长的记录式文件的定位比较麻烦（可以建立索引）

（3）索引存取：对文件中的记录按某个域的值进行排列，从而可以根据键值来快速存取。

（4）存取方法与介质的关系

①与怎样使用文件有关：**如源程序文件用顺序存取法，数据库文件用随机存取法**

②与存储介质的特性有关：磁带？磁盘

第十一章 文件系统实现

1. 文件分配方法

方法	原理	优点	缺点
连续分配方式	创建文件时，分配一组连续块，说明起始块和文件长度	简单、对顺序文件有利、适用于一次性写入、存取速度快，寻道次数和时间最少	文件不能动态增长（插入删除）、外部碎片、难找到足够连续快、创建文件时需声明文件大小
链接分配方式	文件名、起始块号&最后块号，指针连接，分显式&隐式	提高磁盘利用率、不存在外部碎片、利于文件动态增长	存取慢、只适合顺序存取、随机存取必须从头遍历、指针出错问题、指针占空间、按簇分配而非按块
索引分配（i 节点）	所有指针放在一起，即索引块	消除外部碎片、充分利用外存、普遍使用	较多寻道次数&时间，开销大

★★★！！！UNIX 的 i 节点 计算！！！★★★

2. 磁盘文件存储空间管理方法

方法	优点	缺点
空闲表法	①连续分配方式，减少 I/O 访问频率②适合文件较小场合或对换取管理③分配与回收容易	浪费存储空间
空闲块链接法	①空间开销小：仅需一个指向链开始的指针及首个分区的长度	①每次分配都要读取并重指向②效率低开销大
位示图法	直观，回收快	扫描开销大，浪费存储空间
成组块链接法		

★★★成组块链接法的分配回收考大题（PPT47 举例）★★★

第十二章 设备管理——I/O 系统

1. 什么是缓冲？

缓冲是在 CPU 和外设之间设置的缓冲区，用于暂存 CPU 与外设之间交换的数据。分为硬缓冲（设备中开辟缓冲区，硬件实现）和软缓冲（内存中开辟空间用于缓冲区）

2. 为什么要引入缓冲？

①缓解 CPU 与 I/O 设备间速度不匹配的矛盾。

②减少对 CPU 的中断频率，放宽对 CPU 中断响应时间的限制。

③提高 CPU 和 I/O 设备、以及 I/O 设备之间的并行性。

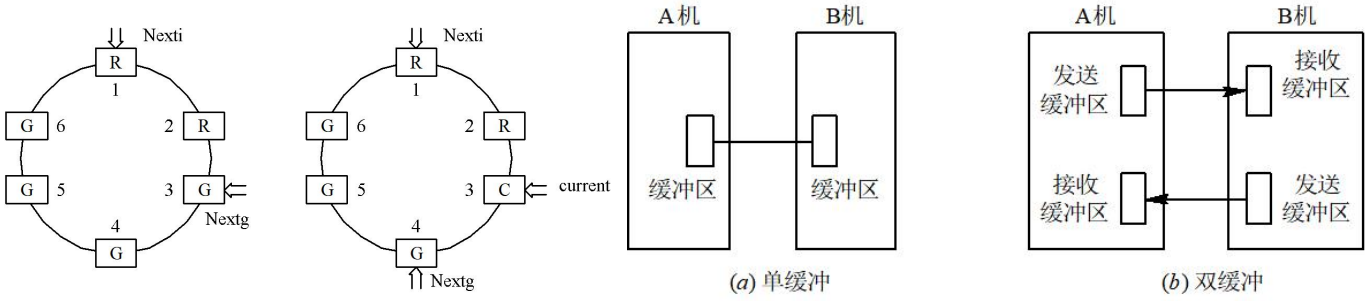
3. 你学过几种缓冲技术？各自适合于什么情形使用？

①单缓冲技术：每当用户进程发出一个 I/O 请求时，OS 便在主存中为之分配一个缓冲区。

磁盘数据输入缓冲区&缓冲区数据传到用户区可并行；CPU 处理和前二者不可并行

②双缓冲技术：在设备输入时，先将数据送入第一个缓冲区，装满后便转向第 2 个缓冲区。这时，OS 可从第一个缓冲区中移出数据，送入用户进程；由 CPU 对数据进行计算。

③循环缓冲：**I/O 设备能跟上进程执行时使用**



(a) 单缓冲　　(b) 双缓冲

4. 区分独占设备、共享设备、虚拟设备

（1）独占设备：一段时间内只允许一个用户(进程)访问的设备，大多数**低速 I/O** 设备（打印机、用户终端），属于**临界资源**，所以多个并发进程须**互斥**进行访问

（2）共享设备：一段时间内允许多个进程同时访问的设备（磁盘），可以获得**良好设备利用率**，是**实现文件系统和数据库系统**的物质基础

（3）虚拟设备：指通过虚拟技术，将**一台独占设备变换为若干台逻辑设备**，供若干进程同时使用（虚拟打印机）一般可以利用假脱机技术(**SPOOLing 技术**）实现虚拟设备。

5. 什么是设备独立性，其实现的条件？

（1）what：在**应用程序中**，使用**逻辑设备**名称来**请求**使用**某类设备**，**物理设备对用户透明**，由**系统实现**逻辑设备到物理设备的**转换**。即通过某种技术使得软件和硬件彼此相对比较独立。

（2）how：为了实现设备独立性而引入了逻辑设备和物理设备这两个概念。在应用程序中，使用**逻辑设备名称来请求**使用**某类设备**；而**系统**在实际**执行时**，**还必须使用物理设备名称**。因此，**系统须具有**将逻辑设备名称**转换**为某物理设备名称的**功能**。

6. 简述实现虚拟设备的基本条件，虚拟设备的实现原理

通过共享设备来模拟独占设备的动作，使独占设备成为共享设备，提高设备利用率和 I/O 速度，实现虚拟设备功能　　条件：需要至少一个可共享的独占物理设备　　原理：spooling 技术

7. 解释 Spooling 系统及其组成

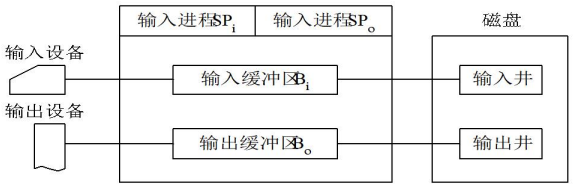
当系统中**引入了多道程序技术**后，可以利用其中的**一道程序**，来模**拟脱机输入**时的**外围控制机**功能，把**低速 I/O 设备**上的**数据**传送到**高速磁盘**上；再用**另一道程序**来**模拟脱机输出**时外围控制机的功能，把**数据**从**磁盘**传送到**低速输出设备**上。基本结构为：输入&输出井，输入&输出缓冲区、输入&输出进程

8. 输入井和输出井的位置及作用

这是在**磁盘上**开辟的两大存储空间。

输入井是模拟脱机输入时的磁盘设备，用于**暂存 I/O 设备输入**的数据。

输出井是模拟脱机输出时的磁盘设备，用于**暂存用户程序的输出**数据。



第十三章 磁盘大容量

1. 磁盘组织结构

（1）存储面：**磁盘**设备可**包括**一到多个物理**磁盘片**，每个**磁盘片分为一**个**或**者**两**个**存储面**。

（2）磁道：每个**存储面**被**组织成**若干个**同心环**，称为**磁道**，各磁道之间**留有必要的间隙**。

- 每条磁道**存储相同数目的二进制位**，磁盘**密度指每英寸中所存储的位数**，**内层**磁道的**密度比外层高**。

（3）扇区：每个**磁道逻辑**上**划分成**若干个**扇区**，通常把一个**扇区称为**一个**盘块**，各**扇区之间**保留一定的**间隙**

（4）柱面：**所有存储面**中处于**同一磁道号**上的**所有磁道**组成一个**柱面**

2. 一次磁盘存取操作的时间组成

寻道时间、旋转延迟时间、数据传输时间

3. 磁盘调度算法

算法	访问顺序	优点	缺点
先来先服务 FCFS	根据进程请求访问磁盘的先后次序，进行调度。	公平、简单	平均寻道距离大，仅适用于 I/O 较少的场合
短查找时间优先 SSTF	该算法优先选择从当前磁头位置出发，移动最少的磁盘 I/O 请求以使每次的寻道时间最短。	性能比“先来先服务”好	不能保证平均寻道时间最短，可能出现“饥饿”现象。
扫描 SCAN	选择在磁头前进方向上从当前位置移动最少的磁盘 I/O 请求执行，到达这个方向上的最后一个磁道时才倒转服务方向，沿相反方向扫描	寻道性能较好，可避免“饥饿”现象	不利于远离磁头一端的访问请求
循环扫描 C-SCAN	在一个方向上使用扫描算法，当沿着某个方向访问到最后一个磁道时，磁头返回到磁盘的另一端的第一个位置，并再次开始扫描。到达边沿时直接移动到另一沿。返回时不为任何的等待访问者服务，返回后可再次进行扫描。	消除了对两端磁道请求的不公平	--
LOOK 和 C-LOOK	磁头只移动到一个方向上的最远的请求为止。接着马上回头，而不是继续到磁盘的尽头。		
磁盘交替编号	对盘面扇区进行交替编号，对磁盘片组中的不同盘面错位命名。		降低旋转时延