# 操作系统概述

1. 操作系统**概念**：

是指控制和管理整个计算机系统的硬件与软件资源，合理地组织、调度计算机工作与资源分配，为用户提供方便接口与环境程序的集合。

1. OS是最基本的系统软件
2. 操作系统的**基本特征**：**并发、共享、虚拟、异步**（没有并发和共享，就没有虚拟和异步）
3. 单核CPU：同一时间只能执行一个程序，各个程序间并发运行
4. **共享**：即资源共享，指系统资源可供内存多个并发执行的进程共同使用
   1. 共享方式：①互斥共享（打印机） ②同时访问（磁盘）
5. **虚拟**：将物理实体变为若干逻辑上的对应物；用于实现虚拟的技术（用户感觉）

虚拟处理器：用多道程序设计技术，让多道程序并发执行，分时使用一个处理器

* 1. 虚拟技术：①时分复用（处理器的分时共享） ②空分复用（虚拟存储器）

1. 操作系统的**目标和功能**：
   1. **管理计算机系统的资源分配**
      1. 进程管理（进程控制、进程同步、进程通信、死锁处理、CPU调度）
      2. 存储器管理（内存分配回收、地址映射、内存保护与共享、内存扩充）
      3. 文件管理（文件存储空间管理、目录管理、文件读写管理、保护）
      4. 设备管理（缓冲管理、设备分配、设备处理、虚拟设备）
   2. 操作系统作为用户与硬件之间的接口
      1. 命令接口（联机命令接口-交互式命令接口；脱机命令接口-批处理命令接口）
      2. 程序接口（由一组系统调用组成，用户通过使用系统调用来请求OS提供服务）
   3. 作为扩充机器（将CPU、内存……协调起来，相互协调配合）
2. 操作系统的发展过程：
   1. 手工操作阶段（无OS）：

用户独占全机；资源利用率低、CPU利用不充分（CPU等待手工操作）

* 1. 批处理阶段（OS开始出现）

为了解决人机矛盾、I/O设备和CPU速度不匹配的矛盾

|  |  |  |  |
| --- | --- | --- | --- |
|  | 特点 | 优点 | 缺点 |
| 单道批处理系统 | 1. 自动性 2. **顺序性** 3. 单道性 |  | 内存中仅有一道程序能够运行，CPU有大量时间在等待IO |
| 多道批处理系统 | 1. 多道 2. 宏观上并行 3. 微观上串行 | 并发执行、  共享资源、  系统吞吐量大、提高了CPU和IO利用率 | **不提供人机交互能力** |

* 1. 分时操作系统

利用分时技术，把CPU运行时间划分成多个短的时间片，轮流给各个作业使用。

**主要目的**：让用户能与自己的作业进行交互

**主要特征**：①同时性（多路性-多个用户使用一台主机）、②交互性（可以与程序交互）、③独立性（多个用户彼此独立）、④及时性（用户请求能够快速响应）

缺点：不能优先处理紧急任务

* 1. 实时操作系统

改善了分时的缺点（不能处理紧急任务）——必须在规定时间内完成对外部事件的处理

**主要特点**：实时性、可靠性

硬实时系统：某个动作必须在规定时刻发生（导弹发射）

软实时系统：偶尔接受违反时间规定（飞机订票、银行）

* 1. 网络操作系统和分布式计算机系统

网络操作系统：实现网络中各种资源的共享及各台计算机之间的通信

分布式计算机系统：任意一台计算机构成一个子系统，作业可以分布在几台计算机上，由它们并行、协同完成。主要特点是分布性和并行性。

① 提高单机资源利用率的关键：多到程序设计技术

**9. 甘特图的绘制**

给出几个不同的程序，每个程序的各个任务时间片给出，使用甘特图求解：

① 横轴表示时间

② 纵轴表示不同的程序

③ 用几种不同的线代表对不同资源的占用

10. 命令接口

联机式命令接口、脱机式命令接口

程序接口（由一组系统调用，也是广义指令构成），编写程序代码间接调用底层接口

用户可以通过系统调用和命令接口两种方式使用计算机

**系统调用：只能通过用户程序间接使用**

## 操作系统的运行环境

1. 操作系统指令两种不同性质的程序

①操作系统内核程序 ②用户自编的应用程序

2. 特权指令：内核程序可以执行而应用程序不可(I/O指令、置中断指令、修改PSW指令)

3. CPU状态划分：①用户态（目态） ②核心态（管态、内核态）——PSW对CPU状态进行标记

4. 操作系统内核的构成（层次结构）：

① 与硬件联系较紧的（最底层）：时钟管理、中断处理、设备驱动等

② 运行频率较高的：进程管理、存储器管理、设备管理等

5. 操作系统内核包含的四个内容（按功能划分）：

① **时钟管理**（时钟中断，由时钟部件发出的信号）

② **中断机制**（需要OS介入的地方都会发出中断信号）

中断是OSkernel夺回CPU使用权的唯一途径

③ **原语**（公用小程序）：OS最底层、运行具有原子性、运行时间较短

④ **系统控制的数据结构及处理**：

进程管理：进程状态管理、进程调度、分派创建撤销PCB

存储器管理：存储器空间分配和回收、内存信息保护、代码对换

设备管理：缓冲区管理、设备分配和回收

6. 核心态指令实际上包括系统调用类指令和一些针对时钟、中断、原语的操作指令

7. 引入中断和异常的原因

① OS引入了核心态和用户态，CPU运行上层程序进行两种状态切换的途径就是通过中断或异常

② 对于某个程序，在不需要某个资源时，需要通过中断将那种资源进行释放

8. **中断（外中断）**定义：来自CPU执行以外的以外时间的发生，如I/O结束中断

9. **异常（内中断）**定义：源自CPU指令内部事件，分为自愿中断（陷入指令）以及强迫中断（硬件故障；软件中断-程序的非法操作码、除零等），**内中断不能被屏蔽**，应该立即处理**（不一定返回原指令继续执行，可能直接转入下一条指令）**

开机第一条指令就是JUMP至中断服务程序进行机器自检

10. **从用户态到核心态的转换由硬件实现**（一个特殊的寄存器来表示CPU所处的工作状态，0表示和心态，1表示用户态）

**核心态到用户态由操作系统执行程序完成后自动转化**

11. 中断处理过程：

**由中断隐指令完成前三步：**

① 关中断（不响应其他中断）

② **保存断点（PC值）**

③ 引出中断服务程序（中断服务程序入口地址送入PC）

**由中断服务程序执行后面几步：**

④ **保存现场和屏蔽字**（寄存器，如PSW和通用寄存器）

⑤ 开中断

⑥ 执行中断服务程序

⑦ 关中断

⑧ 恢复现场和屏蔽字

⑨ 开中断、中断返回

12. 系统调用：

① 设备管理

② 文件管理

③ 进程控制

④ 进程通信

⑤ 内存管理

凡是与资源共享（**系统资源管理、进程管理等**）有关的操作，都必须通过系统调用的方式向OSkernel请求服务请求

13. **大内核和微内核比较**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 特点 | 优点 | 缺点 |
| **大内核** | 将OS主要功能模块作为一个整体在核心态运行 | **性能好** | 接口**复杂**、代码难以维护 |
| **微内核** | 将不需要核心态执行的功能移动到用户态执行，只将最基本的东西保存在内核 | **接口清晰、维护代价低** | **性能差** |

# 进程管理

## 进程与线程

1. 进程定义：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

2. 进程实体：由程序段、相关数据、PCB三部分构成

3. PCB是进程存在的唯一标志，创建进程也即创建进程实体的PCB；销毁进程也即撤销进程PCB;其中进程实体是静态的，进程是动态的

4. **PCB的作用**：让一个在多道环境程序环境下不能独立运行的程序（及其数据）成为一个能独立运行的基本单位，一个能与其他进程并发执行的进程。（书P44）

① **作为独立运行基本单位的标志**（配备了PCB的程序，就是能合法执行的程序）

② 能实现间断运行方式（CPU将**现场信息**保存至PCB中，以供恢复现场）

③ 提供**进程管理的信息**（调度程序通过PCB中记录的起始指针来找到程序、PCB中还有该进程的**全部资源清单**，可以了解其要使用的资源）

④ 提供了进程处于何种**状态信息**（程序是处于就绪态/阻塞态/执行态，以供OS调度）

⑤ **实现进程的同步与通信**（PCB中有信号量、实现进程通信的区域、通信队列指针等）

5. PCB中的信息

① 两种标识符：外部标识符、内部标识符

外部标识符：方便用户访问，由创建者提供。有父进程、子进程、用户等标识指示

内部标识符：OS使用

② 处理机相关信息（处理机的上下文，主要指处理机中各个寄存器的值）

③ 进程控制和管理信息（进程状态、抢占处理机的优先级等；程序、数据在内存或外存中的地址、消息队列指针、信号量、链接指针-本进程所在队列中下一进程地址）

④ 资源分配清单

6. 进程控制的程序段为**原语**，是一个不可分割的基本单位（有进程创建、终止、阻塞和唤醒、进程切换）

7. 唤醒原语和阻塞原语必须成对使用，进程因何事被阻塞就应何事被唤醒

8. 进程切换的过程

① 保存处理机上下文

② 更新PCB信息（状态字、环境）

③ 将该进程移入某个队列（就绪、某事阻塞队列）

④ 选择另一个进程执行，并更新其PCB信息

⑤ 更新内存管理的数据结构

⑥ 恢复新进入处理机的进程的上下文

9. 组织进程的方式：链接方式（队列）、索引方式（索引表，表项指向PCB）

10. 进程状态转化：

① 运行态

② 就绪态

③ 阻塞态

④ 创建态

⑤ 结束态

**就绪态不可能转化为阻塞态**

**阻塞态不可能转化为执行态**

11. 挂起操作：当该操作作用于某个进程时，这个进程就保持静止。如果进程在执行，它将暂停执行变成静止就绪状态，如果它原本处于就绪态，则该进程不接受调度。

与挂机对应的是激活操作

12. 高级进程通信（进程之间的信息交换）方式有：

① **共享存储**：在对共享空间进行读写时，要用同步互斥工具(PV操作)

低级方式：基于数据结构的共享（比如说每次只穿长度固定的数组）

高级方式：基于存储区的共享（灵活、速度快）

② **消息传递**：以格式化的消息,通过系统提供的**发送\接受消息两个原语**进行信息交换

**直接**通信方式：发送进程直接把信息发送至接受**进程的消息队列**中

**间接**通信方式：把信息发送至中间实体——**信箱中**

③ **管道通信**：**半双工通信方式**

管道即用于连接一个读进程和一个写进程的一个共享文件（实质上是固定大小的缓冲区）

特性：**没读完就不能写，没写完就 不能读**

13. 引入线程的目的：减少程序在并发执行时所付出的时空开销

引入进程的目的：更好的使多道程序并发执行

14. 线程的**组成**：**线程ID、程序计数器、寄存器集合、堆栈**

15. 线程：是进程中的一个实体（拥有一定资源），被系统调度和分派的基本单位

16. 线程**属性**：

① 每个线程有个唯一标识符ID，和一个线程控制块TCB（记录当前栈、寄存器状态）

② 不同的线程可以执行相同的程序（同一个服务被不同用户调用，分配成多个线程）

③ 同一进程的线程共享该进程所有资源，但有独立的栈空间

④ **线程是CPU独立调度单位**，多个线程可以并发执行

⑤ 线程在它生命周期会经历不同线程状态的切换

17. 线程实现方式

用户及线程ULT（user-level-thread）：

早期OS不支持线程，程序员通过调用线程库实现；所有的线程管理由应用程序线程库完成，不需要切换至内核态；CPU意识不到线程存在

优点： 线程管理**开销小**（不需要频繁模式切换），效率高

缺点： 并发度低（**一个线程被阻塞，则该进程内所有其他线程也被阻塞**）

不能利用多处理机

内核级线程KLT（kernel-level-thread）

线程管理所有工作由OS完成，为应用程序提供线程的编程接口。

优点： 并发强；线程切换快、开销小

能够调度一个进程中的多个线程并行执行（多处理器中）

一个线程被阻塞，该进程中其他线程可以继续被调度运行

缺点： 线程管理需要，CPU切换到内核态，导致**管理开销大**

**用户级线程是“代码逻辑”的载体**

**内核级线程是“运行机会”的载体（CPU分配的基本单位）**

18. 多线程模型

① 多对一模型（纯用户级线程）

② 一对一模型（纯内核级线程）

③ 多对多模型：

n个用户级线程映射到m个内核级线程上，n≤m

19. C语言编写程序时使用的内存：

正文段：（代码和赋值数据段）二进制代码和常量

数据堆段：动态分配的存储区（malloc()要求分配的存储区）

数据栈段：临时使用的变量（函数调用的实参、未赋值的局部变量）

## 进程调度

1. 进程调度：在就绪队列中以一定的算法选择一个进程，将处理机分配给它

2. 调度的层次：

① **作业调度**（高级调度）：

从外存中挑一个作业到内存，并分配相应资源同时建立线程

② 中级调度（内存调度）：

将暂时不能运行的进程调至外存，此时进程状态为挂起态（PCB不会被调走，而是常驻于内存）;或者有空闲时将该进程从外存调至内存。用于提高内存利用率

③ **进程调度**：从就绪队列中选一个进程分配处理机

3. 三级调度的联系：

① 调度频率：作业调度最低、进程调度最高

② 进程调度是最基本的，不可缺少

③ 作业调度为进程活动做准备，进程调度使进程活动起来，中级调度将暂不能运行的进程挂起

**4. 不能进行调度与切换的进程（进行调度的时机）：**

① **处理中断**的过程

② 进程在OS**内核程序临界区**中（需要独占访问共享数据）（**普通临界区可以进行CPU调度**）

③ 其他需要完全屏蔽中 断的**原子操作**（加锁、解锁、中断现场保护、回复）

5. 应该进行调度与切换的进程：

① 满足引起调度条件且当前进程无法继续运行（非剥夺方式）

② **中断处理结束或自陷处理结束后**（剥夺方式）

6. 两种进程调度方式

非剥夺调度（非抢占方式）：只允许进程主动放弃处理机，而不能强制其放弃

剥夺调度（抢占方式）：立即暂停正在执行的进程，让给更紧急的进程

7. CPU调度算法性能

CPU利用率（CPU忙碌时间占总时间比例）

系统吞吐量（单位时间内CPU完成作业的数量）

**周转时间（作业提交到完成所经历的时间）**

**平均周转时间（n个作业的周转时间取平均值）**

**带权周转时间（作业周转时间除以作业实际运行时间，越小越好）**

平均代权周转时间

等待时间（进程处于等待处理机状态的时间和）

响应时间（从用户提交请求到首次OS响应所用时间）

8. 调度算法

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 思想 | 规则 | 作用对象 | 抢占 | 优缺点 | 饥饿 |
| FCFS | 先来先服务 | | 作业、进程 | 非抢占式 | 有利于CPU繁忙,而I/O不利 | × |
| SJF | 短作业优先 | | 作业、进程 | 可抢占式 | **时间最少**/长作业不利 | √ |
| 优先级调度 | 按优先级选择 | | 作业、进程 | 可抢占式 | 有利于**紧急处理任务** | √ |
| 高响应比优先 | 响应比最高 | | 作业 | 非抢占式 | 克服饥饿、兼顾长作业 | × |
| 时间片轮转 |  |  | 进程（分时） | 抢占式 | **多用户**能及时干预系统 | × |
| 多级反馈队列 | 轮转+优先级 | | 作业（UNIX） | 抢占式 | 所有类型作业用户都满意 | √ |

**9. 响应比**

① 作业等待时间相同时，要求服务时间越短，响应比越高，有利于短作业

② 要求服务时间相同时，等待时间越长，响应比越高，有利于等待时间长的作业

③ 对于长作业，响应比随着等待时间增多而增多，克服了饥饿状态

10. 作业：用户提交，且以用户任务为单位

进程：由操作系统自动生成，以操作系统控制为单位，**进程管理通过原语实现**

## 进程同步

1. 进程同步的引入：进程是并发执行，不同进程之间存在不同的相互制约关系；**为了协调不同进程之间的相互制约关系（如：先乘除后加减）**

2. 临界资源：一次只允许一个进程互斥使用（包括物理设备、变量数据）

临界区：**进程中用于访问临界资源的那段代码（PV、加减锁），并不是临界资源**；

OS只会关心临界区的操作过程，具体对临界资源的操作是进程的事’

3. 临界区代码：

① 进入区：检查并上锁

② 临界区：访问临界资源的代码

③ 退出区：退出并解锁

④ 剩余区：代码中剩余部分

4. **同步：直接制约关系**，为完成某个任务而建立的进程需要协调他们的工作次序

**互斥：间接制约关系**

5. 同步机制应遵循的规则：**空闲让进；忙则等待；有限等待；让权等待**

6. 实现临界区互斥的基本方法：

① 单标志法（软件）：

一个公用整型变量turn，表示哪个编号的进程可以进入该临界区

**违背空闲让进、让权等待**

② 双标志先检查法：

设置一个数据组flag[i]，表示第i个进程是否有意愿进入临界区

**违背忙则等待、让权等待**

③ 双标志后检查法：

先检测对方的进程状态标志，再给自己的标志赋值

**解决了忙则等待，违背了空闲让进、有限等待（会出现饥饿）、让权等待**

④ 皮尔孙算法：

设置谦让变量turn=i表示让给i先进入，每个进程先设置自己的标志flag[i]=true表示自己想进入，后再设置turn=j，表示谦让给j

当j也想进入，也谦让给了j，就会一直等待

**解决了饥饿，违背了让权等待**

⑤ 中断屏蔽方法（硬件）：

当进程正在使用处理机执行临界区代码时，禁止一切中断发生->屏蔽中断/关中断

缺点：不适用多核CPU用户进程（关中断权力太大）只适用OSkernel进程（开关中断只能在内核态使用）

⑥ 硬件指令方法（TS/TSL指令）

TestAndSet指令：该指令原子操作，功能：读出共享变量lock(表示是否上锁)，并且赋值为true

Swap指令：功能为交换两个字的内容

使用TSL和Swap指令实现进程互斥p84

7. 硬件方法优缺点

优点：支持进程中内有多个临界区

缺点：**不能实现让权等待**；选择进程是随机的，可能会导致**饥饿**

8. 信号量机制：只能被两个标准原语P: wait(S), V: signal(S)

9. 整型信号量：定义为资源数目量，

P操作：循环检查S是否大于0，是的话S=S-1**（未遵循让权等待）**

V操作：S=S+1

没有遵循“让权等待”，处于**忙等**状态

**10. 记录型信号量：**除了一个代表资源数的value变量外，还有一个**进程等待链表（所有等**

**待该资源的进程）记录型信号量不会处于忙等**

P操作：semaphore S的value值减一，当value小于0时，表示资源分配不够，调用block原语进行自我阻塞，并插入等待队列**（遵循了让权等待）**

V操作：value值加一，如果value的值还小于等于0,用wake原语唤醒队头进程

11. **利用信号量实现进程同步：**每一个前驱结点都要设置一个信号量，初值为0；前面的进

程代码完成后用V（增加一个资源数），后面进程的用P检查

12. **利用信号量实现进程互斥：**设置一个信号量，初值为1，P代表进入区，V代表退出区： P（S）；临界区代码；V（S）

13. 分析进程同步和互斥问题的方法步骤：

关系分析（分析进程之间的同步、互斥关系）；整理思路（每个进程的PV大致顺序）；设置信号量

14. 管程引入原因：信号量机制中每个访问临界资源都要PV操作，大量、分散且容易出错造成死锁

15. 管程：利用共享数据结构抽象表示系统中的共享资源，并将操作用一组过程（函数）来定义，包括对资源的申请、释放等。整个数据结构及操作形组成的资源管理程序，称为管程

特点：①一个进程只有调用管程内的过程才进入管程访问共享资源

②每次仅允许一个进程进入管程

16. 管程的构成**（类）**：

管程的名字；

局部于管程内部的共享数据结构（对应系统中的资源）

对该数据结构的一系列操作（申请、释放）

对共享数据设置初始值的语句

管程的特点：

① 只有通过管程的过程才能进入管程访问共享资源

② 每次仅允许一个进程进入管程，各个进程只能串行执行管程过程

**17. 管程的条件变量（实现进程同步互斥）：**

condition x,y 条件变量为一抽象数据类型，对每个变量保存一个阻塞链表

x.wait：正在调用的进程因x而被**阻塞**或挂起，该进程调用wait将自己**插入到x的等待队列上**，并释放管程

x.signal：正在调用管程的进程P**重新启动**一个阻塞队列中的进程Q；有人规定signal作为过程体中的最后一步操作

**18. 条件变量和信号量的比较：**

**相似点：条件变量的wait/signal的操作类似于P/V操作，可以实现进程的唤醒和**

**阻塞**

**不同点：条件变量是无“值”的，仅实现了排队等待的功能，而信号量的值反映了资**

**源数**

## 经典同步问题

1. 生产者-消费者问题

问题：一组生产者进程、一组消费者进程共享一个初始为空的缓冲区，缓冲区为临界资源，只允许一个生产者放入消息，或一个消费者取出消息**（互斥）**；

缓冲区没满，生产者才能生产；缓冲区不空，消费者才能消费**（同步）**

信号量设置：mutex 互斥信号量（初值为1）

full 表示缓冲池中“满”的缓冲区

empty 表示缓冲池中“空”的缓冲区

2. 生产者-消费者问题（2）

问题：桌子上有个盘子，每次只能放入一个水果。爸爸专门往盘子放苹果，妈妈专门往

盘子放橘子，儿子专门等盘子里的橘子吃，女儿专门等盘子里的苹果吃

分析：盘子变空（儿女造成）触发放入水果（父母）

实际上可以抽象为两个生产者和两个消费者被连接到大小为1的缓冲区上

3. 读者-写者问题

问题：有读和写两种进程并发进程共享一个文件，要求①允许多个读者同时对文件进行

读、②只允许一个写者往文件中写信息、③任一写者在写操作完成前不允许其他

进程读、④写者执行写操作前，应让其他读写进程全部退出

分析：对写者来说，只要一个互斥信号量rw即可

对读者来说，设置一个计数器count表示当前读者的数量，mutex信号量互斥访

问count；当第一个读进程读的时候，P(rw)；最后一个进程读完后，V(rw)

写进程优先（读写公平法）：

4. 哲学家进餐问题（多临界资源）

分析：五名哲学家与其左右邻居对中间的筷子访问是互斥的

应该避免饥饿、死锁

为五个筷子设置信号量chopsticks[i]，同时每个进程取筷子时再单独设置一个信

号量mutex，实现互斥拿两边的筷子（如果没拿到等待释放即可）

5. 吸烟者问题（可以生产多个产品的单生产者）

问题：抽烟者需要三种材料：烟草、纸、胶水；三个抽烟者第一个自己有烟草、第二个

有纸、第三个有胶水；供应者无限提供三种材料，每次将两种材料放到桌上

分析：供应者与三个抽烟者分别同步；三个抽烟者对抽烟互斥（供应者每次只能满足一

个人的抽烟材料）；

设置信号量offer1, offer2, offer3 分别表示烟草和纸组合的资源、烟草和胶水、

纸和胶水的资源。finish 用于表示互斥的进行抽烟

## 死锁

1. 死锁**定义**：多个进程因为竞争资源，全部处于阻塞状态

2. 死锁产生**原因**：

① 系统资源竞争

② 进程推进顺序非法（如两个进程都占据了对方想要的资源）

③ 信号量使用不当

3. 死锁产生的**必要条件**：

① 需要的资源必须要进程**互斥**访问

② 资源**不可剥夺**

③ 进程**保持**了资源又请求其他资源（已被占用），被阻塞又**不释放**自己的资源

④ 进程**相互循环等待**（一条链）

4. 死锁的处理：

1. 死锁**预防**：破坏死锁的四个条件

2. **避免**死锁：资源动态分配策略

3. 死锁的**检测**和解除：OS检测机构及时检测死锁并解除

5. 死锁预防

① 破坏**互斥条件**：

大部分场景不适用，反而需要保护互斥条件；**SPOOLing技术**，把独占式设备改造

成共享设备（逻辑上）

② 破坏**不剥夺条件**：

抢占式调度方式；实现复杂，反复申请和释放资源会增加系统开销；

常用于易于保存和恢复的资源，如CPU寄存器和内存资源，一般不用于打印机

③ 破坏**请求并保持**：

预先静态分配，进程在运行前就一次性申请完所有资源（资源未满就不投入运行）

缺点：资源严重浪费，会产生饥饿

**④ 破坏循环等待**

顺序资源分配法：首先给系统资源编号，只有占有小编号资源才有资格申请大编号

资源；已经持有了大编号资源，意味小编号资源已经全部申请到

问题：编号必须稳定，限制了新设备的增加；每个作业的使用顺序不同，会造成资

源浪费；给编程带来麻烦

6. 死锁避免

① **系统安全状态**：

系统可以以某种进程的顺序（P1, P2, P3……）为每个进程分配资源，当前分配进程

在结束后会归还占据的资源，后续进程也可以正常推进。该进程执行的顺序为**安全序列**

**不安全状态：无法找到安全序列，并不一定是死锁状态，而是可能进入死锁状态。**

**反之，只要处于安全状态，就不会发生死锁**

**② 银行家算法**

1. 检查此次申请是否超过之前申明的最大需求数

2. 检查系统的剩余资源是否能满足这次请求

3. 试着分配，更改各个数据结构

4. 用安全性算法检测是否会进入不安全状态（如果不安全则作废并恢复数据）

‘’

安全性算法

检查当前剩余可用资源能否满足某个进程的最大需求，如果可以就把该进程加入到

安全序列（初始为空），并把该进程的资源全部回收；

不断重复上述过程，看看能否让所有进程都加入安全序列

7. 死锁检测和解除：

资源分配图：圆代表进程，框代表资源；进程到资源为请求边，资源到进程为分配边

**死锁检测算法**：依次消除与不阻塞进程相连的边，直到无边可消除

**死锁定理：S为死锁的条件是当且仅当资源分配图是不可完全简化的（最终还有边）**

死锁解除：①资源剥夺法：挂起某些死锁进程，并抢占其资源分配给其他进程

②撤销进程法：强者撤销部分或全部死锁进程，并剥夺资源（代价大）

③进程回退法：让进程回退至无死锁状态（需要保持进程历史信息）

《玉米案件——预防、避免、检测和解除》

# 内存管理

## 内存管理的基本概念

1. 内存管理的功能：

① 内存的分配和回收（主存空间）

② 地址转换（逻辑地址与物理地址）

③ 内存空间的扩充（虚拟存储技术/ 自动覆盖技术）

④ 内存保护（各道作业在各自的内存空间运行，互不干扰）

**2. 程序转化为可执行程序：编译——链接——装入**

源代码 -> 编译后 -> 若干目标模块（程序段，独立的逻辑地址）

目标模块 -> 链接后 -> 完整的装入模块（完整的逻辑地址）

装入模块 -> 装入后 -> 装入程序（物理地址）

3. 程序的链接方式

① 静态链接：先将各个目标模块和所需的库链接成一个完整的可执行程序

② 装入时动态链接：在装入时，边装入边链接

③ 运行时动态链接：程序执行时需要哪些模块就链接哪些模块（灵活性高）

4. 内存的**装入方式**

① 绝对装入：程序中的逻辑地址与实际内存地址完全相同（单道程序系统）

② **可重定位装入：**静态重定位，地址变换在装入时一次完成（早期多道批处理系统）

必须分配**连续**的所有要求的内存空间

③ **动态运行时装入：动态重定位**，可将程序分配到**不连续**的空间；装入部分程序代码

即可运行；在运行期间需要动态申请分配内存，便于程序段的共享；

5. 重定位寄存器：存放装入模块的起始位置

6. 逻辑地址空间：程序经过编译后每个目标模块都从0开始编址，称为目标模块的相对地

址；各个模块连接成一个完整的可执行目标程序后，连接程序顺序按各个模块的相对地

址构成统一的从0号开始编址的逻辑地址空间

物理地址空间：地址转换的最终地址

**7. 内存保护：**

**保护用户进程不受其他用户进程影响、OS不受用户进程影响**

① 上、下限寄存器：访问地址空间时进行比较，判断有无越界

② 重定位寄存器（基址寄存器）：存储最小物理地址值——相加（和逻辑地址）

界地址寄存器（限长寄存器）：存储逻辑地址的最大值——比较（和逻辑地址）

8. **扩充内存**的方法：**覆盖和交换（节省主存空间）**

覆盖是在同一进程中的，交换是在不同进程中的

~~9. 覆盖：将用户空间分为一个固定区（经常活跃的部分），多个覆盖区；按调用关系分段~~

~~将即将访问的段放入覆盖区，其他段放在外存中，需要调用时覆盖替换原有的段；~~

~~内存更新的只有覆盖区，固定区不会更新~~

~~10. 交换（中级调度）：~~

~~将处于等待状态的程序从内存移到外存/ 准备竞争CPU的程序从外存移到内存~~

~~暂时换出外存的进程为~~**~~挂起态~~**~~（就绪挂起、阻塞挂起）~~

### 11. 连续分配方式：

为用户提供一个连续的内存空间（包括单一连续、固定分区、动态分区）

12.

|  |  |  |  |
| --- | --- | --- | --- |
| 分配方式 | 特点 | 优点 | 缺点 |
| 单一连续分配 | 内存被分为用户区和系统区（低地址）；内存中仅有一道程序； | 简单，**无外部碎片**；不需要内存保护 | 只能用于单用户、单任务的OS；**有内部碎片**、存储器利用率低；  需要**利用覆盖**来解决空间不足、利用交换来提高作业道数 |
| 固定分区分配 | 将用户空间划分为多个大小固定（可以相等或者不等）的区域，每个分区只装入一道作业；并建立分区说明表以供检索 | 最简单的多道程序存储管理方式，**无外部碎片** | ①有些程序太大无法装入内存，**必须使用覆盖**技术  ②**有内部碎片** |
| 动态分区分配 | 根据进程的大小动态建立分区；系统中分区的大小和数目是可变的；用空闲分区表/链来记录每个分区的起始部分和末尾部分，起始部分还可以记录大小等信息 | 无外部碎片 | 有外部碎片（可用**紧凑**技术解决） |

13. **内部碎片**：发生在以固定长度分割区来进行配置的记忆体中当一个程式载入到固定大小的分割区时，假如程式小于分割区，则剩余的空间将无法被使用，称为内部碎片。

**外部碎片**：因为行程持续地被载入与置换，使得可用的记忆体空间被分割成许多不连续

的区块。虽然记忆体所剩空间总和足够让新行程执行，却因为空间不连续，导致程式无法载入执行。

14. 动态分区的分配策略

① 首次适应：**顺序查找地址按递增**次序的空闲分区（简单、性能最好、保留大分区）

② 最佳适应：顺序查找按容量递增次序的空闲分区（产生最多外部碎片）

③ 最坏适应：顺序查找按容量递减次序的空闲分区（没有大的内存块使用）

④ 邻近适应：循环首次适应算法，从上次查找结束的位置继续查找

### 15. 非连续分配管理方式：

根据索引实现程序分散装入不相邻的内存分区，

根据分区大小是否固定分为：**分页存储管理、分段存储管理方式**

### 16. 分页存储：

**基本思想**：把主存划分成大小相等且固定的块，块相对较小，为主存的基本单位；同时每个进程也以块为单位划分，在执行时以块逐个申请空间

进程中的块为页/页面；内存中的块为页框；外存中的块为块

**地址结构**：页号P+页内偏移量W

页号 = [逻辑地址 / 页面长度]

页内偏移量 = 逻辑地址 % 页面长度

**页表**：系统为每个进程建立一张页表，记录页面在内存中对应的物理块号，一般存在内存（中的PCB）中；由一行行的页表项构成；页表始地址（基地址）存放在存储器中

**基本地址变换机构**：

系统通常设置一个页表寄存器PTR，存放页表在内存中的起始地址和长度

已知页面大小，得到**逻辑地址后映射到物理地址**：

① 计算页号和页内偏移量

② 比较页号和页表长度，如果大于等于页表长度就发出越界中断

③ 从页表中取出对应页号存储的物理块号

④ 计算物理地址，访存

整个地址变换由硬件完成

17. 页面大小 <-> 页内偏移量位数 -> 逻辑地址结构

18. 快表：具有并行查找能力的高速缓冲存储器（也称相联存储器TLB）

慢表：内存中的页表

快表与普通cache的区别：TLB只有页表项的副本，而普通cache有其他数据的副本

19. 具有快表的分页机制：

① CPU给出逻辑地址后，硬件进行转换，将页号送到高速缓存中，并将此页号与快表中的所有页号进行比对

② 如果有匹配的，直接读出，t = 差快表+访存

③ 如果未找到匹配，从主存中的页表访问，读出表项后存入快表（替换）

t = 2\*访存 + 查表

（某些系统快表和慢表是同时查询的）

20. 单级页表的**问题**：

① 所有页表要连续存储，当页表很大时需要占用多个连续页框；（多级页表）

② 没必要让整个页表常驻内存（虚拟存储技术）

21. 二级页表：在原有的页表结构上再加一级页目录表

22. **多级页表的特点**：

① 若采用多级页表机制，各级的页表大小不能超过一个页面

② 对于两级页表的访问次数（无快表）：1. 页目录表 2.二级页表 3.内存单元

导致访存次数变多（n级页表 要访存n+1次）

### 23. 分段存储：

段式管理按照用户进程中的自身逻辑划分逻辑空间。要求段内连续，段间可以不连续

**逻辑结构**：段号+段内偏移量

**段表**：段号（隐含，不占存储空间） + **段长 + 本段在主存中的始址**

**地址变换机构**：

① 从逻辑地址中取出段号、段内偏移量

② 判断段号是否小于段表长度

**③** 按照段号取出表中对应的表项，判断段内偏移量是否小于该段表项的段长

④ 无越界则计算物理地址访存

24. 段的共享和保护：

**通过两个作业的段表中相应表项指向被共享的段的同一个物理副本来实现**

**不能修改的代码**被称为纯代码或**可重入代码（不属于临界资源）**

25. 分段和分页的优缺点比较

|  |  |  |
| --- | --- | --- |
|  | 优点 | 缺点 |
| 分页 | **不会产生外部碎片，只有少量页内碎片** | 不便于共享和保护 |
| 分段 | 按逻辑模块实现信息共享和保护 | 会产生外部碎片（可通过紧凑来解决） |

26. 段页式管理方式

将两种方法结合起来，作业的地址空间**首先被分为若干逻辑段**，每段都有自己的段号；

**再将每段分成若干大小固定的页**。内存空间则和分页一样，分为若干和页面大小相同的

块，内存的分配以块为单位

作业逻辑结构：段号 + 页号 + 页内偏移量

系统为每个进程建立一张段表，每个分段有一张页表；同时有一个段表寄存器，指出作

业的段表始址和段表长度

段表表项至少包含：段号、页表长度、页表始址

页表表项至少包含：页号、块号；

**核心问题：① 逻辑地址结构 ② 表项结构 ③寻址过程**

## 虚拟内存管理

1， 传统存储管理方式特性

① **一次性：**作业需要一次性装入内存->作业太大无法装入/大量作业存在时效率下降

② **驻留性：**被装入后一直驻留在内存，直到结束

2. 局部性原理：

时间局部性：某条指令一旦被执行，不久后可能还会执行（循环大量存在于程序）

空间局部性：一旦程序访问了某个存储单元，其附近的存储单元不久后也会被访问

3. 虚拟存储器：将程序的一部分装入内存，其余部分留在外存；

实质是建立了“内存——外存”两级存储器结构

4. 虚拟存储器**性质**：

① **多次性：**作业无需一次全部装入内存，允许多次装入

② **对换性：**作业在运行的过程中无需常驻内存，可以换入换出

③ **虚拟性：**从逻辑上扩充内存的容量，用户看到的内存远大于实际内存

5. 虚拟内存**实现方式**：（建立在离散分配的内存管理基础上）

**请求分页存储、请求分段存储、请求段页式存储**管理方式

需要的**硬件**支持：

* 一定容量的内存外存
* 页表机制（或者段表机制）作为主要的数据结构
* 中断机构（访问的部分未调入内存时，产生缺页中断）
* 地址变换机构，逻辑->物理

### 6. 请求分页管理方式：

在基本分页的基础上增加了“请求调页”、“页面置换”两个功能

**请求页表项**：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 页号 | 物理块号 | 状态位P | 访问字段A | 修改位M | 外存地址 |
| 基本页表项 | | 指示该页表是否已经调入内存 | 已有多长时间未被访问，用于置换算法 | 标识在调入内存后有无被修改，如果有还需写会外存 | 该页在外存的地址 |

**缺页中断机构**：

若访问的页面不在内存，便产生一个缺页中断，请求OS调入内存（I/O）；将缺页进程

阻塞（调页完成后唤醒）；若内存有空闲块，则分配一个，没有则换页

与普通中断的区别：

① 在指令执行期间产生中断，属于内部中断

② 在一条指令执行期间，可能产生多次缺页中断（一条读取数据的多字节指令，指令

本身跨越两个页面）

**地址变换机构**：

① 地址变换时，先检索快表；如果找到，修改访问位（如果是写指令还要重置修改位）

然后利用页表中的物理块号和页内地址形成物理地址

② 若没有找到，则去内存中查找页表，再比对页表项中的状态位P；看该页是否已经

调入内存，如果没有，则产生缺页中断，请求从外存调入内存

### 7. 页面置换算法：

最佳置换算法（OPT）：置换最长时间不再被访问的页面，无法实现但是可用于评价其

他算法

**先进先出FIFO**：优先置换驻留内存最久的页面

实现方式：用队列将页面链接起来，设置一个指针指向最前的页面

**Belady异常：物理块数增大，缺页次数反增（仅FIFO有）**

**最近最久未使用 LRU**：选择最近最长时间未被访问过的页面淘汰（Least Recently Used）

实现方式：为每个页面设置一个访问字段，记录上次被访问以来所经历的时间，值最大

的优先淘汰

特点：需要寄存器和栈（对所有的页排序），属于堆栈类算法，无belady异常

**时钟置换算法 CLOCK 最近未用NRU算法**：

实现方式：将内存中的页面通过指针链接成循环队列，

**增加一个使用位**（刚加入时为1，扫描后为0），每次扫描一圈

目的：试图以较小的开销接近LRU算法的性能

**改进**的时钟置换算法：**再增加一个修改位，优先替换没有变换的页**

置换优先级：

① 最近未被访问u=0; 未被修改m=0

② 最近未被访问u=0；但被修改 m=1

③ 最近被访问u=1；但未被修改 m=0

④ 最近被访问 u=1；被修改 m=1

流程：先扫描看看有没有①，

没有再扫描看看有没有②，此次扫描，每扫描一个都将使用位置0

①②都没有，此时使用位已全置为0，再重复①②

8. 驻留集：给一个进程分配的物理页框的集合

8. 页面分配：需要决定给特点的进程分配多少个页框

考虑的**因素**：

① 分配给进程的存储量越小，驻留在主存的进程越多

② 若页面过小，会频繁产生缺页中断

③ 若页面过多，由于局部性原理缺页中断并不会显著减少，反而降低并发度

9. 页面分配的三种策略：

① 固定分配**局部**置换

为每个进程分配一定数目物理块，驻留集大小在整个运行期间不变

遇到缺页时，只能从该进程中选页置换

**弊端**：灵活性低，难以确定为每个进程适合分配的物理块数

② 可变分配**全局**置换

为每个进程分配一定数目物理块，同时系统自身保持一个空闲物理块队列

缺页时会从空闲物理块队列中去一个物理块分配

**弊端**：会盲目增加物理块，导致并发度下降

③ 可变分配**局部**置换

为每个进程分配一定数目物理块

缺页时只允许从该进程在内存的页面选一页换出，同时根据缺页率动态调整给该进

程分配的物理块

**弊端**：实现复杂，开销大，但其他表现良好

④ **无固定分配全局置换（**固定了多少块，就不会把别人的页给你置换）

可变全局和可变局部的区别：

可变全局：只要缺页就会分配新物理块，置换的页面可以是别人的（全局）

可变局部：会根据缺页率和动态增加/减少物理块，置换的页面只能是自己的（局部）

10. 调页策略：

预调页策略：程序运行由程序员前指出要调用的页（若干相邻的页）（运行前）

请求调页策略：缺页才调（运行中）

一般情况两种策略同时使用

11. **抖动**：刚刚换出的页面又要换入主存，刚刚换入主存的页面又要换出主存

主要原因：某个进程频繁访问的页面数高于可用物理页帧数目

12. 工作集：某段时间间隔内，进程要访问的页面集合

工作集大小决定了驻留集的大小（一般驻留集大小 > 工作集大小）

# 文件系统管理

## 文件系统基础

1. 文件：一组有意义的信息/ 数据的集合（涉及到文件的属性、内部组织、文件间组织）

2. 文件的属性：名称（唯一）、标识符、类型、位置、大小、保护、时间、日期、用户

3. 文件的基本操作：

① 创建文件：根据所需外存空间大小，结合各种算法，在文件中找到空间；

在目录中为新文件创建目录项（文件名、位置等可能信息）

② 写文件：执行一个write**系统调用**，指明文件名和要写入的内容；OS为每个文件维

护一个写位置指针（用于根据名称，查找文件位置）每次写操作时，更新

指针

③ 读文件：执行一个read**系统调用**，使用open系统调用返回的文件描述符；OS为

每个文件维护一个写位置指针；每当读的是也更新读指针；一个进程通常

只对一个文件读或写，所以读写只使用一个指针。

④ 文件重定位（文件寻址）：按照某个条件搜索目录，

⑤ 删除：找到要删除的目录项，找到外存存放位置，删除目录项同时回收存储空间

⑥ 截断文件：文件所有属性不变，删除文件内容（将长度设为0），并释放空间

4. 文件的打开：

通过**系统调用open**（需要文件存放路径、文件名、打开方式）

open：将指定文件从外存复制到内存**打开文件表的一个表目**中，返回表目的索引

当用户需要文件操作时，通过该索引指定文件（省略了访问目录(I/O)）

打开文件表：**OS维护一个**所有打开文件信息的表；**每个进程各自维护一个**自己的打开

文件表

**5. 文件控制块 FCB**

① 文件**指针**：OS跟踪上次的读写文职作为当前文件位置，这种指针对打开文件的某

个进程来说是唯一的，因此与磁盘文件分开保存

**② 文件打开计数：**OS在删除打开文件条目之前，必须等待最后一个进程关闭文件

③ 文件磁盘位置

④ 文件访问权限：判断是否合法

⑤ 文件索引号：文件在打开文件目录表中的索引号

### 6. 文件的逻辑结构：

无结构文件（流式文件）：以字节为单位，只能通过穷举搜索，txt

有结构文件（记录式文件）：

顺序文件：文件的记录一个一个顺序排序，通常是定长的，可以顺序或链式存储

索引文件：

索引顺序文件：将记录分为若干组，为每组第一个记录建立一张索引表

直接文件或散列文件：通过记录的键值直接决定记录的物理地址

7. **文件目录**：FCB的有序集合

一个FCB就是一个文件目录的目录项（包含了文件名、物理地址等信息）

**8. 目录结构：**

① **单级目录结构**：整个文件系统只建立一张目录表

**实现了“按名存取”；但是查找速度慢、不允许重名、不便于文件共享**

② **两级目录结构**：分成主文件目录、用户文件目录

主文件目录记录用户名及相应用户文件存放位置

用户文件目录由该用户的文件FCB组成

**允许“重名”、可以保证文件安全；缺乏灵活性、不能分类**

③ **多级目录结构**：用文件的路径名标识文件，文件路径名是个字符串；

从根目录出发的路径为绝对路径。

读取下一级目录要从外存读取相应目录表(I/O)->层次太多时需要多次IO

改良：引入“当前目录”

**不便于文件共享**

④ **无环图目录结构**：**便于共享，**但是管理麻烦

**对于删除操作：**

为每个共享节点设置一个共享计数器，增加共享链时加一；

请求删除时共享计数器减一，只有为0时才能删除

9. **索引结点**（FCB的改良）：除了文件名之外的信息都存放在索引结点里

可以节省目录的空间、加快查找速度（减少了磁盘IO次数-原本要检索很多目录项）

由文件名和指向该文件对应结点的指针构成

10. 磁盘索引结点：存放在外存的索引结点（每个文件都有唯一一个）

文件被打开后，索引结点会被复制到内存中：

内存索引结点：存放在内存的索引结点，需要增加信息（是否被修改，几个进程在访问）

10.5. 文件共享：

基于索引节点的共享方式**（硬链接）**：

将共享文件链接到多个用户的目录中

索引结点额外存放一个链接计数count，count>1时，删除仅删除自己相应目录项

利用符号链实现文件共享**（软链接）**：

为共享文件A创建链接文件F，F仅存放A的路径名字（多次IO，访问速度慢）

10.6. 文件保护

口令保护：通过口令访问文件

加密保护：为文件加密

访问控制：根据访问者身份进行限制

### 11. 文件物理结构（文件分配方式）

**文件在存储器中的组织形式（物理结构）：**

① **连续分配**：要求每个文件在磁盘上占用一组连续的块（逻辑上相邻、物理上也相邻）

用户给出逻辑块号后，OS找到该文件对应的FCB

物理块号 = 起始块号 + 逻辑块号

**优点**：

支持随机访问和顺序访问

连续分配的文件在顺序读写的时候速度最快（磁头移动距离短）

**缺点**：

不方便文件扩展；反复增删文件会产生外部碎片；同时很难确定一个文件的大小

**② 链接分配（隐式和显式）**

**隐式连接：**每个盘块后面都有一个指针指向下一个盘块，目录中包括文件的第一

块的指针和最后一块的指针

**显式链接：**将指针显式的存在一张链接表中（每个磁盘一张），为**文件分配表FAT**

-1表示文件的最后一块；-2表示该盘块空闲（**可用于空闲块管理**）

FAT**常驻内存**，不需要读磁盘进行IO

**③ 索引分配**

索引块的组织（做题时需要看清顶级索引是否在内存里）：

* + 链接方案：通过指针将多个索引块链接起来，FCB只需要记录第一个索引块，但是访问后续的索引块需要多次IO操作
  + 多层索引：第一层的索引块指向第二层的索引块，第二层的索引再指向（若用多层索引，各层索引表不能超过一个磁盘块——以计算文件的最大长度）

K层索引需要K+1次IO

* + **混合索引**：多种索引分配结合

**12. 文件存储空间管理（空闲块管理）**

① 文件存储空间的划分与初始化：

一个文件存放在一个**文件卷**（逻辑卷、逻辑盘）中；一个文件卷分为**文件区**

（存放数据信息）和**目录区（文件控制信息FCB、空闲表等）**

② 文件存储器空间管理：

对空闲块的组织和管理，包括组织、分配、回收

* **空闲表法：**

用一张表记录了空闲的盘块地址、盘块数量等信息（适用于连续分配）

需要注意合并问题



* **空闲链表法；**

空闲盘块链：空闲的盘块链成一个链表

空闲盘区链：空闲的盘区链成一个链表（一个盘区可以包含多个空闲盘块）

* **位视图法**（用二级制的一位来表示磁盘中一个盘块的使用情况）

用二进制的一位来表示磁盘中的一个盘块使用情况

0表示空闲，1表示该盘块已分配

字号表示第几行、位号表示第几列（左上角为1,1）

* **成组链接法**：（UNIX）

把空闲扇区地址保存在一个扇区A内，A的最后一个扇区存B扇区的指针，B也存储了一批空闲扇区地址，以此类推CDE…

## 磁盘组织与管理

1. 磁盘的组成：安装在磁盘驱动器中，由磁头币、旋转用主轴、用于数据IO的电子设备

2. 磁盘地址：柱面号 + 盘面号 + 扇区号（或块号）

根据柱面号——移动磁臂至指定柱面

根据盘面号——激活指定磁头

根据扇区号——指定扇区会从磁头下划过

3. **一次磁盘读写操作时间**：由寻道时间、延迟时间、传输时间决定

**① 寻道时间：**磁头移动到指定磁道所需要时间（跨过n条磁道时间+启动磁臂时间）

Ts = m \* n + s

m为每跨一条磁道的耗时，约为0.2ms；磁臂的启动时间s约为2ms

**② 延迟时间：**磁头定位到某一磁道的扇区所需要时间

Tr = 1/2r

r为磁盘的旋转速度，1/r为旋转一周的时间，找到目标平均为半圈

**③ 传输时间：**从磁盘读出或向磁盘写入数据所经历的时间

Tt = b/rN

b为写入/出的字节数，r为每秒转速，N为一个磁道上的字节数

磁头要读b/N 个磁道；读写一个磁道的时间为1/r

**4. 磁盘调度算法：**

① 先来先服务 FCFS：

少进程且访问数据集中时：性能好；大量进程竞争时：性能接近随机调度

② 最短寻找时间优先 SSTF：

贪心算法：选择离当前磁头最近的磁道，**使每次寻找时机最短；但是不能使平均寻**

**找时间最小**，但比FCFS性能好。会出现**饥饿**

**③ 扫描算法 SCAN（电梯调度算法）：**

在当前移动方向上选择与当前磁头所在磁道距离最近的请求作为下一次服务对象

只有磁头到最外侧才能往内，最内侧才能往外；无饥饿

**缺点：有一小段边界可以不扫描；且各个磁道相应频率不平均**

**④ 循环扫描 C-SCAN**

磁头仅单向扫描，到头就快速移至起始端而不服务任何请求

**改进点：磁道相应频率平均**

**⑤ LOOK 和 C-LOOK 调度：**

**磁头移动只需要到达最远端的一个请求即可返回，不需要到达磁盘端点**

5. 磁盘调度比较

|  |  |  |
| --- | --- | --- |
|  | 优点 | 缺点 |
| FCFS | 公平、简单 | 平均寻道距离大、仅应用在磁盘IO较少的场合 |
| SSTF | 性能比FCFS好 | 不能保证平均寻道时间最短，可能会饥饿 |
| SCAN | 寻道性能较好，可避免饥饿 | 不利于远离磁头一端的访问请求 |
| C-SCAN | 消除了对两端磁道请求的不公平 | - |

6. 减少延迟时间（磁头定位到某一磁道的扇区所需要时间）

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**

7. 磁盘的初始化：

① **低级格式化：**将空白盘分成扇区；每个扇区由头、数据区域、尾部组成；

头部和尾部包含了磁盘控制器所使用的信息（链接指针、扇区校验码）

接着OS将自己的数据结构记录在磁盘上：

② **磁盘分区：**将磁盘分为由一个或者多个柱面组成的分区（C盘D盘等形式分区）

③ **逻辑格式化：**创建文件系统（创建文件系统的根目录、初始化管理数据结构，位视

图、空闲分区表等）

8. 引导块：初始化程序（自举程序），自举装入程序一般保存在ROM中，完整程序保存在

磁盘的启动块上；启动块在磁盘固定位上，有启动分区的磁盘称为启动磁盘或系统磁盘

9. 坏块（硬件故障）：扇区损坏，有磁盘坏块链表或者FAT上标明，程序不会使用；同时、有备用块在逻辑上替代坏块（扇区备用）

# IO管理

## IO管理概述

1. IO设备按**使用特性**分类

- 人机交互类：鼠标键盘（速度较慢、通常以字节为单位进行数据交换）

- 存储设备：光盘、磁盘（速度较快、通常以字块为单位进行数据交换）

- 网络通信设备：网络接口、调制解调器

IO设备按**传输速率**分类

* 低速设备：键盘鼠标（几百字节每秒）
* 中速设备：行式打印机、激光打印机（几万字节每秒）
* 高速设备：磁带机、磁盘机、光盘机、移动硬盘

IO设备按**信息交换**分类

* 块设备：磁盘（可寻址、以块为基本单位传输）
* 字符设备：键盘（以字符为基本单位、传输速率低、不可寻址、IO以中断驱动方式）

2. IO控制方式：控制设备和内存或处理机直接的数据传送

3. IO控制方式分为：程序直接控制方式、中断驱动方式、DMA方式、通道控制方式

4. 程序直接控制方式：

计算机从外部每读取一个字，都要轮询检查，直到该字已经在IO控制器的数据寄存器

缺点：CPU一直在等待IO（测试IO设备的状态、未采用中断方式导致IO设备无法高

速CPU它已经完成了一个字符的输入），CPU和IO利用率都低

5. **中断驱动方式**

允许IO设备主动打断CPU的允许并请求服务，可以使CPU发出读命令后干其他的事

在每个指令周期的末尾，CPU检查中断

缺点：数据传送（IO控制器到存储器直接）仍然需要经过CPU

6. **DMA方式**

IO和内存之间开辟直接的数据交换通路，彻底解放CPU

DMA方式的**特点**

* 基本单位是**数据块**
* **传输数据无需经过CPU，与内存直接相连**
* 仅在传输一个或者多个数据块**开始、结束时，才需要CPU干预，**整块数据传输是在DMA控制器的控制下完成的

DMA控制器的**组成**：

* 命令/状态寄存器（CR）：接受CPU发来的IO命令或者控制信息，或设备的状态
* 内存地址寄存器（MAR）：存放数据块在内存中对应的地址
* 数据寄存器（DR）：暂存从设备到内存或者内存到设备的数据
* 数据计数器（DC）：存放本次要传送的字（节）数

DMA方式**工作过程**：

CPU接受到IO设备的DMA请求时，给IO控制器发出一条命令，启动DMA控制器，接着做其他事。CPU把控制操作委托给DMA控制器，由该控制器负责处理。

DMA控制器直接与存储设备交互，传送整个数据块，每次传送一个字，整个过程不需要CPU参与。传送完成后DMA控制器发送一个中断信号给处理器。

DMA方式主要应用于块设备（磁盘）

7. **通道控制方式**

通道：弱鸡版CPU，可以识别并执行一系列的**通道指令**——专门负责输入输出的处理机

IO通道方式是DMA方式的发展，可以进一步减少对CPU的干预，用于完成**内存与外**

**设的信息交换**

设置通道后，CPU只需要向通道发送一条IO指令。通道收到指令后就从内存中取出本

次要执行的通道程序，然后执行该通道程序，仅当完成IO任务后**才发送中断信号**。

IO通道与一般处理机的区别：

通道指令类型单一、没有自己的内存、通道所执行的通道程序是放在主机的内存中

的，也即与CPU共享内存

**IO通道与DMA方式的区别：**

**DMA方式需要CPU来控制传输数据块的大小、传输内存位置**；而通道中这些信息

都是由**通道控制**的。另外每个DMA控制器对应一台设备与内存传输数据，而一个

通道可以控制多台设备与内存进行数据交换

**DMA方式和中断控制方式的区别：**

中断控制方式在每个数据传输完之后中断CPU，而DMA仅在完成全部时才中断

中断控制方式的数据传送在CPU控制下完成；DMA方式在DMA控制器下完成

DMA方式以存储器为核心（可并行），中断控制方式以CPU为核心

DMA方式批量传输数据，中断控制方式以字节为单位

8. **IO子系统的层次结构**：使IO软件具有更清晰的结构、良好的移植性、适应性

**引入目的：对种类繁多差异巨大的IO设备进行设备控制**

分为**四个层次结构**：

① **用户层IO软件：**实现用户交互的接口，用户可以直接调用在用户层提供的与IO操

作有关的库函数对设备进行操作，如printf()

② **设备独立性（设备无关性）软件：处理系统调用**

引入了**逻辑设备名**，可以使：应用程序独立于具体使用的物理设备

* + 增加设备分配的灵活性
  + 易于实现IO的重定向（IO设备可以更换）

设备独立性软件的**功能**：

* + 执行所有设备的共有操作，包括分配回收、逻辑到物理设备名映射；对设备的保护；缓冲管理；提供独立于设备的大小统一的逻辑块，屏蔽传输速率差异
  + 向用户层提供统一接口：read/write等系统调用

③ **设备驱动程序：**与硬件直接相关，负责实现系统对设备发出的操作指令，驱动IO

设备工作。通常每类设备配置一个设备驱动程序，是IO进程与设备控制器之间的

通信程序，以进程形式存在。

如：接受read和write命令，转换为具体要求后发送给设备控制器，控制IO设备

工作，同时也将设备控制器发来的信号传送给上层软件

④ **中断处理程序：**

保存被中断进程的CPU环境，转入相应的中断处理程序进行处理，处理完成后并

恢复被中断的现场后，返回被中断进程；（与硬件紧密相关，对用户应屏蔽）

⑤ **硬件设备：**

包括一个机械部件（设备本身）和一个电子部件（**设备控制器**）

9. IO设备与内存**统一编址：不需要专门的指令，与内存操作相同，被称为内存映像IO**

IO设备**独立编址：需要专门的指令来操作设备**

10. 设备控制器的主要**功能**：

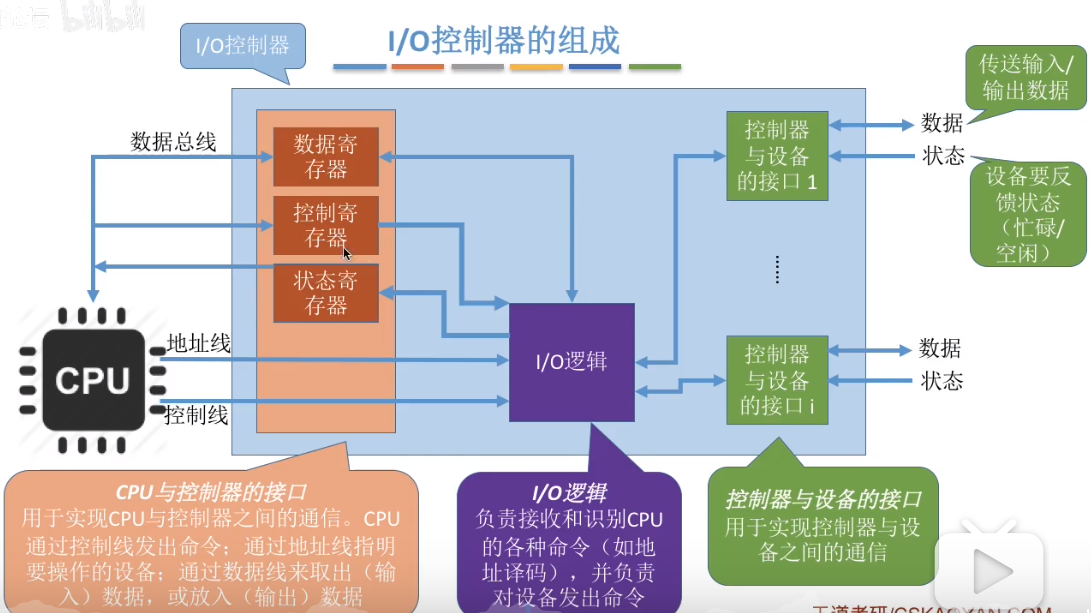
- 接受和识别CPU或者通道发来的指令，如磁盘控制器能接受读、写、查找等命令

- 实现数据交换，包括设备和控制器之间；控制器和主存之间（通过总线或者通道）

- 发现和记录设备及自身状态信息，供CPU使用

- 设备地址的实变

11. 设备控制器的组成部分



① 设备控制器和CPU接口：

数据线、地址线、控制线和两类寄存器相连

数据寄存器：存放从设备来的输入数据或从CPU来的输出数据

控制/状态寄存器：从CPU送来的控制信息或者设备的状态信息

② 设备控制器和设备接口：

实现设备与控制器之间通信，一个接口连接一台设备；每个接口都存在数据、控

制、状态三种类型的信号

③ IO控制逻辑：

实现设备控制的功能；对从CPU收到的IO命令进行译码，然后传给控制器；

13. I/O子系统：将CPU从繁重的I/O设备管理中解放出来

提供的服务有：I/O调度、缓冲与高速缓存、设备分配与回收、假脱机、设备保护、

差错处理等

14. I/O调度：确定一个好的顺序来执行IO请求；OS维护一个请求队列来实现调度，当一个程序执行阻塞IO系统调用时，该请求就加到相应设备的队列上；磁盘调度算法就是IO调度之一

15. 磁盘高速缓存（Disk Cache）：

目的：提高磁盘的IO速度，使高速缓存复制的访问比原始数据访问更为有效

形式：①在内存中开辟一个单独的存储空间作为高速缓存，大小固定；

②把未利用的内存空间作为一个缓冲池，供请求分页系统和磁盘IO时共享

16. 缓冲区（Buffer）：

目的：①缓和CPU与I/O设备速度

②减少对CPU的中断频率、放宽对CPU中断响应时间的限制（信号充满一次

全部取走）

③解决基本数据单元大小（数据粒度）不匹配的问题

④提高CPU和I/O设备之间的并行性

实现方法：硬件缓冲器（成本太高，用的少）；在内存中划分缓冲区（管道）

特点：缓冲区数据非空时，不能往里注入数据；只有充满后才能把数据传出

17. 缓冲技术：

① 单缓冲：在设备和处理机之间设置一个缓冲区

研究缓冲技术对每块数据处理时间时，需要假设一种初始状态（如工作区满、缓

冲区为空），进而计算下一次到达相同状态时所需要的时间，就是处理一块数据的时间（默认缓冲区和工作区的大小相等）

② 双缓冲：I/O设备输入数据时先装填到缓冲区1，等待缓冲区1填满后再开设装填

缓冲区2，此时缓冲区1的数据可以给CPU处理；

初始状态规定为：工作区是空的，缓冲区一个空一个满

③ 循环缓冲：包含多个大小相等的缓冲区，每个缓冲区有个链接指针指向下一个缓

冲区，最后一个缓冲区指向第一个缓冲区；同时还有两个指针：in和out；

in指向第可以输入数据的第一个缓冲区

out指向可以提取数据的第一个缓冲区

④ 缓冲池：由多个系统公用的缓冲区组成；

缓冲区按使用情况可以形成三个队列：空缓冲队列、装满输入数据的缓冲队列

（输入队列）、和装满输出数据的缓冲队列（输出队列）

还有四种缓冲区：（**内存中**）用于收容输入数据的工作缓冲区、用于提取输入数据

的工作缓冲区、用于收容输出数据的工作缓冲区及用于提取输出数据的工作缓冲区

18. 缓冲区和高速缓存的对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | | 高速**缓存** | **缓冲**区 |
| 相同点 | | 都介于高速设备和低速设备之间 | |
| 不同点 | 存放数据 | 存放的是低速设备上的某些数据的复制数据，就是高速缓存上有的，低速设备上必然有 | 存放的是低速设备传递给高速设备的数据（或相反），而这些数据在低速设备上却不一定有备份，这些数据再从缓冲区传到高速设备 |
| 目的 | 存放的是高速设备经常要访问的数据，若高速设备要访问的数据不在高速缓冲中，就需要访问低速设备 | 高速设备和低速设备的通信都要经过缓冲区，高速设备永远不会直接去访问低速设备 |

19. 设备分配

根据用户的I/O请求分配所需要的设备，

将设备按使用特性可分为：独占设备、共享设备、虚拟设备

12. **独享设备、共享设备和虚拟设备**

- 独享设备：在一个用户作业未完成或退出之前，此设备不能分配给其他作业用。**所有字符设备都是独享设备**。如输入机、磁带机、打印机等——很明显：需要装驱动。

- （分时式）共享设备：多个用户作业或多个进程可以“同时”从这些设备上存取信息。**软硬盘、光盘**等块设备都是共享设备——**无需驱动**。

**共享设备必须是可寻址的和可随机访问的设备**，如果是顺序访问或者不能寻址，则一个进程一旦占用设备，就必须到使用完后才能释放。

- 虚拟设备：把一个物理设备变换成多个对应的逻辑设备。例如：通过SPOOLing（假脱机）技术将一台打印机虚拟成多台打印机——实质还是独享设备，需要驱动。——将一台独占式设备改造为共享设备

20. 设备分配的数据结构：设备控制表（DCT）、控制器控制表（COCT）、通道控制表

（CHCT）、系统设备表（SDT）

**系统设备表（SDT）**：整个系统只有一张SDT，记录所有物理设备情况，每个物理设备

占一个表目

**设备控制表（DCT）**：**一个设备控制表表征一个设备**，各个表项就是设备的各个属性；同时表项中需要有一项指出该设备的控制器控制表（COCT）的指针

**控制器控制表（COCT）**：现代OS的IO都用通道控制，而设备控制器需要请求通道为其服务，所以有个表项指向相应通道控制表（CHCT）的指针

**通道控制表（CHCT）：**因为**一个通道可以为多个设备控制器服务**，所以CHCT必有一个指针，指向一个表，表上的信息表达的是CHCT提供服务的那几个设备控制器。

**CHCT与COCT是一对多的关系**

设备分配的过程中访问数据结构按**先后顺序为**：

SDT -> DCT -> COCT -> CHCT

山东（SD）大（D）葱（CO）存货（CH）

21. 设备分配方式：

静态分配：用户作业开始执行前，由系统一次性分配该作业要求的全部设备（独占）

动态分配：现请求先分配、优先级高者优先、短作业优先等等（共享设备）

22. 设备分配的安全性：

**安全分配方式**：每当进程发出I/O请求就进入阻塞态，直到I/O完成才能被唤醒；一

旦进程获得设备后就阻塞、不能请求任何资源。

优点是不会死锁，缺点是对于一个进程来说，CPU和I/O设备是串行工作的

不安全分配方式：仅当进程所请求的设备被其他进程占用才会进入阻塞态

优点是一个进程可以同时操控多个设备；缺点是可能会死锁

23. SPOOLing技术（假脱机技术）

将低速IO设备上的数据传送到高速磁盘上

**虚拟设备技术，将独占性设备虚拟（在磁盘上）成共享设备（处理后为虚拟设备）**

由预输入程序、井管理程序、缓输出程序组成

**输出井和输入井：**

在**磁盘**上开辟的两个存储区域。输入井模拟脱机输入时的磁盘，用于收容I/O设备输

入的数据；输出井模拟脱机输出时的磁盘，用于收容用户程序的输出数据

**输入缓冲区和输出缓冲区：**

**内存**中开辟的两个缓冲区。输入缓冲区用于暂存由输入设备送来的数据，以后再传到

输入井

**输入进程和输出进程：模拟控制机**

# 其他

中断和异常的再总结<https://blog.csdn.net/u011240016/article/details/53117093>