Contents

[第一章：操作系统基础 3](#_Toc212502314)

[操作系统基本概念 3](#_Toc212502315)

[操作系统种类 4](#_Toc212502316)

[操作系统结构 6](#_Toc212502317)

[微型计算机使用的操作系统 6](#_Toc212502318)

[“分布式网络” VS “分布式系统” 7](#_Toc212502319)

[CPU状态 7](#_Toc212502320)

[进程切换/CPU调度 8](#_Toc212502321)

[系统调用/防管 9](#_Toc212502322)

[特权指令/特权操作 9](#_Toc212502323)

[第二章：进程与线程管理 11](#_Toc212502324)

[作业简介 11](#_Toc212502325)

[程序 11](#_Toc212502326)

[进程（分配资源的基本单位） 13](#_Toc212502327)

[线程（调度的基本单位） 21](#_Toc212502328)

[系统原语 22](#_Toc212502329)

[死锁 22](#_Toc212502330)

[第三章：内存管理 25](#_Toc212502331)

[作业 25](#_Toc212502332)

[存储管理功能 29](#_Toc212502333)

[存储器的三级结构（计组里面有5层结构） 31](#_Toc212502334)

[分区存储（物理划分） 31](#_Toc212502335)

[分页存储（内存利用率最高，逻辑划分） 33](#_Toc212502336)

[分段存储（逻辑划分） 34](#_Toc212502337)

[段页式存储（逻辑划分） 35](#_Toc212502338)

[虚拟存储器 — 时间换空间 37](#_Toc212502339)

[第四章：文件（外存）管理 43](#_Toc212502340)

[文件 43](#_Toc212502341)

[目录管理 47](#_Toc212502342)

[空闲文件管理 49](#_Toc212502343)

[文件的使用、共享与安全 50](#_Toc212502344)

[第五章：输入输出管理 52](#_Toc212502345)

[IO设备 52](#_Toc212502346)

[IO操作控制方式 54](#_Toc212502347)

[中断 57](#_Toc212502348)

[缓冲技术 59](#_Toc212502349)

[设备分配算法 60](#_Toc212502350)

[独占设备管理 62](#_Toc212502351)

[IO设备独立性与无关性 63](#_Toc212502352)

[Spooling 假脱机技术 64](#_Toc212502353)

[磁盘访问时间与驱动调度 66](#_Toc212502354)

[第六章：操作系统安全与保护 69](#_Toc212502355)

[系统安全 69](#_Toc212502356)

[错题 74](#_Toc212502357)

## 第一章：操作系统基础

### 操作系统基本概念

* 定义：一组系统软件，负责连接用户和计算机硬件，是 裸机上的第一层软件
  + 自下而上分别是**：操作系统，编译系统，支撑软件，应用软件**
  + 提供：命令界面，图形界面，系统调用界面
* 作用：提供人机界面，提高资源利用率，提供功能更强的虚机器
* 发展阶段
  + 人工→监督程序→执行系统→多道程序→操作系统
* **基本特征：并发，共享，虚拟，异步；不包括“并行”**
  + **“多道”** == **并发：**同一时间间隔内交替执行，用于单处理机系统（单CPU或单核），通过“多道程序设计”实现
  + **并行：**同一时刻同时执行，用于多处理机系统（多CPU或多核）；在多道程序下指 **宏观并行，微观串行**。
  + 虚拟：虚拟处理机，虚拟内存，虚拟外部设备，虚拟通信
  + **异步性**：程序执行具有不确定性
* **设计目标：方便性，有效性**，可扩充性，开放性
  + **有效性**：提升资源利用率和系统**吞吐量**
  + **方便性**：提供友好的**用户接口**（命令行、图形界面、系统调用）
  + 可扩充性：支持硬件扩展（如USB）
  + 开放性：兼容不同软硬件，提升可移植性
* 功能：
  + 处理机管理：进程控制，同步，通信，调度，初级中断处理
  + **存储管理（内存）**：分配与回收，存储保护，地址映射，内存扩充
  + 设备管理：设备操作，设备独立性，设备分配和回收，虚拟设备，缓冲区管理
  + **文件管理（外存）**
  + **用户接口**：
    - **命令接口**：键盘，**作业控制语言**（**作业说明书**+**资源申请表**），**图形化用户界面**
    - **编程接口**（也叫**用户程序**）：
      * **系统调用**：低级语言（汇编）
      * **库函数**：高级语言

### 操作系统种类

* 注意事项
  + 批处理不能人机交互
  + **批处理 需要作业控制语言（作业说明书+资源申请表）**
* **单道批处理（提高作业执行效率）（不并发，不并行）**
  + 无需人工干预
  + 优点：缩短作业执行时间，提高吞吐率（因为是以批次处理的）
  + 缺点：设备利用率低，系统性能较差
* **多道批处理 （提高系统资源利用率和吞吐量）（并发）**
  + 程序穿插运行；提高效率和吞吐率；**宏观上并行，微观上串行**
  + 优点：资源利用率高，吞吐量大（单位时间内的工作总量大）
  + 缺点：用户交互性差；作业平均周转时间长（短作业周转时间显著增加，因为一个任务需要多个节点处理，而每个节点可能不是空闲的）
* **分时（提高交互性与响应速度）（并发）**
  + **多路性**：微观上不同时刻轮流使用
  + 独占性：各终端用户感觉自己独占了计算机
  + 及时性：系统能够快速响应用户请求，提供及时的反馈。用户可同时请求服务。
  + 交互性：用户可以直接通过终端与系统进行交互，输入命令并立即得到反馈。
  + 需求：多道程序设计技术，终端命令解释程序（处理键盘命令），中断处理
* **实时（按规定时间完成任务）（并发）**
  + 对外部请求在规定时间内完成；及时处理由过程控制反馈的数据
  + 相同：多路性，独立性
  + 不同：
    - 交互性：不希望太强的交互性
    - 安全性，可靠性，实时性
  + 特点
    - 对外部信号能及时响应
    - **要求高可靠性，安全性；效率放在第二位**
    - 整体性强；不要求强会话能力
    - 支持**自动控制**

### 操作系统结构

* 传统：
  + **无结构 OS**：简单、效率高，但难以维护（如 MS-DOS）
  + **模块化 OS**：结构清晰，但模块耦合高，独立性低，难以修改
  + **分层式 OS**：层次分明，每层基于上一层
* 现代：
  + **微内核结构**
    - 功能：进程管理、内存管理、进程通信、I/O管理
    - 特征：小内核、客户/服务器模式、面向对象技术
    - 运行模式：核心功能运行在**内核态**，其余在**用户态**
  + **混合微内核结构（最常用）**
    - 功能：一样
    - 运行模式：**所有**进程都在**内核**模式，其中**关键**在**微内核**；其他服务使用**客户端/服务器模式**运行
    - 优点：灵活性高、可扩展、可靠

### 微型计算机使用的操作系统

* 大型机操作系统（如 IBM MVS）
* 小型机操作系统（如 VAX/VMS）
* 微型计算机操作系统（如 DOS、Windows、UNIX）
* 优点：便宜，规模小
* 缺点：资源和功能相对有限
* 单用户 单任务：DOS
* 单用户 多任务：**Windows（单CPU并发，多CPU可并行）**
* 多用户 多任务：UNIX**（分时系统）**

### “分布式网络” VS “分布式系统”

|  |  |  |
| --- | --- | --- |
| 特性 | 分布式网络 | 分布式系统 |
| 硬件 | 多台独立计算机通过网络连接 | 多台独立计算机通过网络连接，**协同工作形成整体系统** |
| 同质 | 协议同质，OS 可不同 | OS 同质，紧密耦合 |
| 并行性 | 进程独立运行，各机互不干扰 | 可以将一个进程分布在多机上执行，实现任务级并行 |
| 透明性 | 低：用户需明确指定资源位置和访问方式 | 高：用户无需关心资源位置和系统内部实现 |
| 功能/资源管理 | 主要提供通信功能 | 提供通信、并行计算、资源管理功能。没有物理存储器共享 |
| 存储器共享 | 可共享也可独立 | 无全局共享存储器，每台计算机独立，但资源可统一访问 |
| 健壮性 | 低：单机故障影响局部 | 高：多机协作，系统整体容错能力强 |

### CPU状态

* 3种CPU状态：
  + **（内）核态**：全部资源和指令，包括“特权指令”
  + **管态**：不能修改CPU，除非没有核态
  + **用户/目态**：不能用特权指令，不能修改CPU
    - **特权指令**包含的不能在这里进行
    - 可以进行**系统调用**，**外部中断，缺页**
* 状态转换
  + 目态→管态：唯一途径是通过中断
  + 管态→目态：设置”程序状态字“（PSW），中断返回指令
  + 核态→目态：在运行用户程序的时候，出现了中断，并且处理完后，就会由核态转换成用户态

### 进程切换/CPU调度

* 是操作系统内核在多任务系统中，把CPU从一个进程切换到另一个进程的过程。只能在内核态（或管态）进行，不能在用户态进行。
* 发生
  + 当前进程被阻塞（等待I/O、等待资源）
  + 当前进程时间片用完（分时调度）
  + 高优先级进程需要运行
* 过程
  + 用户程序执行系统调用指令
  + CPU从用户态切换到内核态
  + 内核执行请求的服务
  + 完成后返回用户态，继续用户程序执行

### 系统调用/防管

* 用户程序通过系统调用进入内核态，让操作系统帮它完成**特权操作**。
  + 用户程序请求内核服务，完成进程切换
* **用户程序与操作系统内核交互的桥梁**
  + 由用户态发起，但在内核态中执行。
* 过程
  + 用户程序执行系统调用指令
  + CPU从用户态切换到内核态
  + 内核执行请求的服务
  + 完成后返回用户态，继续用户程序执行

### 特权指令/特权操作

* **只能在CPU处于核态或者管态下执行**
* 包括
  + **中断**
  + **进程切换**
  + **存取 内存保护的 寄存器；即设置”程序状态字“（PSW）**
    - 位于运算器中，PCB包含PSW
    - 控制 CPU 的执行模式（如用户态 / 内核态）
    - 反映程序运行状态（如标志位、条件码）
    - 实现中断与进程切换时的状态保存与恢复
  + **输入和输出**
  + **停止中央处理机**

## 第二章：进程与线程管理

|  |  |  |
| --- | --- | --- |
| 特性 | 单道程序设计 | 多道程序设计 |
| 内存使用 | 内存中只有一道程序 | 内存中同时有多道程序，CPU 可以在这些作业之间切换，使系统始终保持忙碌。既可以在单核CPU运行，也可以在多核CPU运行。 |
| CPU 利用率 | 很低，经常等待 I/O | 高，I/O 与计算可并行 |
| 并发性 | 无（顺序执行） | 有（程序可并发运行） |
| 资源共享 | 无，资源独占 | 有，多个程序共享 CPU、内存、I/O 等资源 |
| 相互制约 | 无，每个程序独占执行 | 有，进程之间可能因为资源竞争而相互制约（同步、互斥、死锁等问题） |
| 系统吞吐量 | 低 | 高（单位时间完成更多作业） |
| 封闭 | 有 | 无 |

### 作业简介

* 定义：用户提交给系统的一个完整任务实体；**一个作业必须由至少一个“程序”组成，且至少包含一个“进程”**
* 特性：
  + 存于外存

### 程序

* 定义：一组指令或语句的集合；**一个“程序”至少对应一个“进程”**
* 特性：
  + **静态、永久**
  + 顺序性
  + 封闭性（执行中不可更改）
  + 可再现性（给定输入→确定输出）
* **程序并发执行**：若干程序在时间上重叠执行
  + 并发程序关系
    - 无交互：互不影响
    - 有交互：可能破坏**封闭性**和**可再现性**
  + 特点
    - 迸发性：执行时间不固定
    - 共享性：可能访问公共资源
    - **制约性/交互性**：进程间互相影响
  + **时间有关的错误**：当多个程序或进程并发执行时使用**公共资源**，则在并发时可能得到不同的结果；错误与占用 CPU 时间总量无关
    - 使用共享资源
    - 进程交替执行的次序
    - 进程被打断的时间：当进程在执行临界操作时被操作系统中断（如发生时钟中断、I/O中断），另一进程可能马上运行并修改共享数据。
* **程序装入**
  + **编译**：将源程序转化为可执行代码（机器指令）。如果程序是机器码，则可以跳过这一步。
  + **链接**：
    - **静态链接**：程序运行前完成库函数连接
    - **装入时动态链接**：程序装入内存时完成链接
    - **运行时动态链接**：执行过程中按需链接
  + **装入**
    - 两个操作：把程序从外存放入内存 + **地址重定位（逻辑地址→物理地址**）
    - **绝对装入**：编译或汇编时确定地址
    - **静态重定位**（可重定位装入）：装入内存时确定地址，装入后程序不可移动
    - **动态重定位**（动态运行时装入）：把程序原封不动存储到内存；地址映射推迟到程序真正执行时，通过**重定位寄存器（装目标模块的相对基地址）**实现；装入后程序能移动（操作系统有条件的移动）

| **特性** | **静态重定位** | **动态重定位** |
| --- | --- | --- |
| **装入时间** | 装入内存时一次性完成 | 执行过程中多次按需完成，可动态映射 |
| **硬件依赖** | 不依赖硬件支持 | 需要硬件支持（如 MMU 或重定位寄存器） |
| **内存连续性** | 必须占用连续内存 | 不必须连续，可在任意空闲内存块 |
| **程序可移动性** | 装入后程序不能移动 | 装入后程序可移动 |
| **虚拟存储** | 无法实现虚拟存储 | 可以实现虚拟存储 |

### 进程（分配资源的基本单位）

* 定义：具有独立功能的程序在某个数据集上的一次执行活动；是**资源分配的基本单位**；**一个“进程”必须至少有一个“线程”**才能执行代码
  + 组成：**进程 = 程序 + 数据 + PCB**
    - 静态部分：程序 + 数据集合
    - 动态部分：进程控制块（PCB）
* 特征：
  + **动态、暂时存在**、**并发性**、异步（特征由PCB体现）
* **进程控制块 PCB**：操作系统用来管理和控制进程的核心数据结构。
  + **标识进程的唯一实体**，随进程创建而创建。
  + **只能由操作系统修改，用户无法直接操作**。
  + 构成
    - **进程标识符 PID**：唯一标识进程
      * 包括内部标识符和外部标识符
    - **处理机状态**：记录与处理器有关的各种**现场信息**， 主要**用于进程切换**
      * 通用寄存器，控制寄存器，程序状态字PSW
      * **现场信息是在“中断处理程序 处理中断 前”保存**
    - **进程调度信息**：用于调度算法的决策
      * 进程状态，优先级，其它信息，以及事件
    - **进程控制信息**：管理进程与资源的关系
      * 程序和数据的地址，进程同步和通信机制，资源清单，链接指针
* **进程状态**

A diagram of a process

AI-generated content may be incorrect.

* + **等待/阻塞/睡眠态 Waiting**：由于发生某个事件，暂时无法执行，不能占用CPU
    - Waiting不能直接到Running
  + **就绪态 Ready**：等待系统分配CPU
    - Ready不能直接到Waiting
  + **运行态 Running**：正在占用CPU
    - IO会导致 Running变Waiting
    - 优先级更高的程序导致 Running变Ready
    - **时间片用完**导致 Running变Ready
  + **新建 New**：刚创建进程，还没加入可执行进程组
  + **消亡 Terminated**：进程已经完成或被迫中止
* **进程队列**
  + 处于相同状态的进程链接在一起，进程队列中的**元素**实际上**是**进程的 **PCB（进程控制块）**。
  + 组织方式
    - **链接方式**：将具有相同状态的 PCB 用**指针字段（链接字）**连接起来形成链表
    - **索引方式**：为不同状态的进程建立多张索引表，每张表记录相应 PCB 的位置。索引表的首地址被操作系统保存，用于快速定位。
* **进程同步**
  + **互斥**：保证同一时刻只有一个进程进入临界区。
    - 互斥进程不知到互相的存在。
    - 如CPU，打印机，共享数据。
  + **同步**：协调进程执行的先后顺序。一个进程的执行会因为要等待另一个而受到影响。本身不能独立完成具体任务。
    - 同步进程知道互相的存在。
    - **同步包含互斥（互斥是同步的特殊情况）**
  + **通信**：进程之间交换数据
  + **临界区**：是进程中访问共享资源（如共享变量、缓冲区、文件、设备等）的那段代码。只要某段代码会读写同一份共享数据，这段代码就是临界区。
    - **进程不能无限制的在临界区，不能无限制的等待进入临界区**
    - **访问准则**
      * **空闲让进**：一次允许一个进程进入空闲的临界区
      * **忙则等等**
      * **有限等待**：进入临界区的进程要在有限时间内退出
      * **让权等待**：如果进程不能进入临界区，则应让出占用的CPU
  + **相关临界区**：并发进程中涉及相同变量的那些临界区
  + **临界/独占资源**：一段时间内只允许一个进程访问的资源
  + **P / V 操作（成对使用）**：对信号量/口令的操作，属于同步原语
    - **P**：在进入临界区或申请资源前执行。语义通常为 S = S - 1（要在占用资源前执行）；
      * 若 S < 0，则当前进程被阻塞并加入等待队列Q（因为没有CPU资源可以进行分配）。
    - **V**：在离开临界区或释放资源后执行。语义通常为 S = S + 1（要在释放资源前执行）；
      * 若 S ≤ 0，表示有|S|进程在等待队列Q，在释放一个资源后，应唤醒等待队列中的一个进程并将其转为就绪。
    - P/V 保证对临界区的互斥访问与必要的唤醒机制，但不能直接用于通信（通信需要其他机制，如消息缓冲、管道、共享内存等）。
  + 用**信号量/口令**实现互斥与同步
    - 信号量表示：一个二元组 (S, Q)
      * **S**：整型变量，初值≥0，表示可用资源数；
        + S>0 表示可用资源数
        + S≤0 表示无可用资源且 |S| 表示等待进程数。
      * **Q**：等待队列，存放因资源不足被阻塞的进程。
      * 若 m 个进程共享 n 个资源（m>n），则 S 的取值范围可视为 n-m … n。
* **进程通信**
  + 不同进程之间交换信息的机制，用于协调或共享数据。
  + **低级通信**：只能传递状态和整数值，如互斥和同步。PV属于低级通信
    - 信息量小，效率低，信息量固定，编程复杂
  + **高级通信**：用于传递大量数据，编程相对简单。
    - **直接通信**：固定某一对进程通信。
      * 使用**消息缓冲机制**：发送方将消息放入缓冲区（队列），接收方从缓冲区取消息。
      * 访问缓冲区时需互斥，且处理时间有限制。
    - **间接通信**：通过**信箱（共享存储器）通信**。
      * 对处理时间没有限制。
      * 信箱结构：信箱头（存信件信息）+ 信箱体（存信件内容）。
    - **管道通信**：将一个进程的输出作为另一个进程的输入，通常用于数据流处理。
  + **通信原语**（用于管理和操作通信机制）：
    - 创建：建立通信通道或信箱。
    - 撤销：删除通信通道或信箱。
    - 发送：发送数据或消息。
    - 接收：接收数据或消息。
* **进程切换**
  + 将 CPU 从一个进程转移到另一个进程的过程
  + **可剥夺方式**
    - 含义：如果有更高优先级的进程到达，或者当前进程的运行时间超过预定时间片，操作系统可以 强制剥夺 当前进程的 CPU 使用权，将其切换出去。
    - 应用场景：实时操作系统、分时操作系统等。
  + **不可剥夺方式**
    - 含义：一旦某个进程获得 CPU，它会一直运行，直到：
      * 进程完成任务，或
      * 主动放弃 CPU（例如等待 I/O 操作）。
* **进程调度**
  + **调度时机**：当 CPU 空闲或者需要分配给其他进程时，操作系统会进行调度。
    - **完成任务**：当前进程执行完它的全部工作，CPU 空闲，需要调度其他就绪进程。
    - **等待资源**：当前进程因为 I/O 操作或其他资源请求被阻塞（进入等待态），CPU 无法继续执行该进程，需要调度其他就绪进程。
    - **运行到时（时间片用完）**：在分时系统中，每个进程有固定的时间片，当时间片耗尽时，CPU 会被剥夺，调度器选择下一个进程运行。
    - **发现标志（Flag）**：指进程状态或外部事件的变化触发调度
      * 一个更高优先级进程到达
      * I/O 完成，等待的进程变为就绪
      * “标志”是操作系统用来记录事件或状态变化的信号，操作系统发现该标志时，会判断是否需要调度新的进程。
  + **调度算法**
    - 衡量标准：CPU利用率，用户程序响应时间，系统吞吐量，公平合理性，设备利用率

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 调度算法 | 可抢占 | 原理 | 适用场景 | 优点 | 缺点 |
| 先来先服务（FCFS） | ❌ | 按进入就绪队列顺序执行 | 批处理系统 | 简单、公平 | 长作业拖慢短作业，队头效应 |
| 短作业优先（SJF） | ❌ | 选择预计运行时间最短的作业 | 批处理系统 | 平均等待时间短 | 长作业可能长期等待（饥饿） |
| 最短剩余时间优先（SRTF） | ✅ | SJF的可抢占版本，每当有新进程到达时，比较其所需的剩余时间，若更短，则抢占当前进程。 | 分时系统 | 灵活、平均等待短 | 调度开销大，频繁比较剩余时间 |
| 优先级调度 | ✅/❌ | 优先级高的先执行，可抢占或非抢占 | 分时/实时系统 | 紧急任务快速响应 | 低优先级可能饥饿，需要老化机制 |
| 时间片轮转（RR） | ✅ | 为每个进程分配固定时间片，轮流执行 | 分时/交互式系统 | 响应快、公平 | 时间片过小开销大，过大退化为 FCFS |
| 多级反馈队列（MLFQ） | ✅ | 综合 **时间片轮转 + 优先级调度** 思想。有多个优先级队列。新任务进入高优先级队列；若时间片用完后未完成则逐步降低优先级，进入更低优先级队列；在不同队列中采用不同的时间片，一般高优先级队列时间片短，低优先级队列时间片长。 | 通用操作系统 | 动态调整优先级，兼顾短长作业 | 算法复杂，实现开销大 |

### 线程（调度的基本单位）

* 线程是进程中的独立执行路径，线程不能独立于进程存在。
  + 目的是提高并发性。
  + 有自己”独立的堆栈“和CPU环境，所以支持独立执行。
    - 线程共享父进程的主存（堆、全局区等），所以线程的资源申请主要依赖于 **进程的内存空间**
    - **同一进程的线程** → 可以共享进程的虚拟地址空间
  + 线程之间的开销比进程小。
* **用户级线程 ULT**
  + **管理方式**：由用户空间的线程库实现，内核不感知。
  + **优点**：
    - 创建、销毁和切换速度快
    - 避免频繁的用户态 ↔ 内核态切换
  + **缺点**：
    - 若某个线程阻塞，整个进程被阻塞
    - 无法充分利用多核 CPU（内核只看到一个进程）
  + **特点**：**线程切换可能触发进程切换，因为内核不知道同一进程中还有其他线程存在**
* **内核级线程 KLT**
  + **管理方式**：由操作系统内核直接管理
  + **优点**：
    - 单个线程阻塞不会影响同一进程中其他线程
    - 能利用多核 CPU，实现真正并行
  + **缺点**：
    - 管理需要系统调用，开销较大
    - 线程切换涉及模式切换（用户态 ↔ 内核态），开销更高

### 系统原语

* 特殊的系统调用，执行过程中不允许被打断（原子操作）。
  + 撤销父进程时需要同时撤销子进程
  + 父进程和子进程的运行顺序与优先级有关，但与父子关系无关。
* 种类
  + **创建原语**：建立一个PCB，分配工作区（即内存空间），进程进入就绪状态
  + **撤销原语**：回收进程工作区占用内存，撤销PCB；只能由父进程或者祖先进程调用
    - **只有Unix操作系统进程可以自杀（自己调用撤销原语），别的都不行**
  + **阻塞原语**：将“运行进程”状态转换为“等待进程”状态
  + **唤醒原语**：将“等待进程”状态转换为“就绪进程”状态

### 死锁

* **死锁的四个必要条件**：四个条件同时成立时死锁**可能**发生；若其中一个条件不成立，则一定不会死锁。
  + **互斥条件**：每个资源在某一时刻只能分配给一个进程使用。
  + **占有且等待条件**：进程已经占有资源的同时，又请求新的资源。
  + **不可抢占条件**：一个进程不能抢夺另一个进程的资源。
  + **循环等待条件**：形成进程环路，每个进程等待下一个进程持有的资源。
* **死锁处理方法**
  + **死锁预防**（破坏必要条件，**互斥条件难以破坏**）：死锁不会发生，但可能浪费资源或降低并发性
    - **预先静态分配法**：只有当资源足够让进程完成时才分配，破坏“占有且等待”和“循环等待”。
    - **有序资源使用法**：要求进程按资源编号顺序申请资源。破坏“循环等待”
    - **可剥夺资源法**：高优先级进程可以抢占低优先级进程的资源（适用于主存和CPU）。破坏“不可抢占”
  + **死锁避免（银行家算法）**：在资源分配时动态检测系统安全状态，若分配后可能导致系统不安全，则拒绝分配，从而在理论上防止死锁发生。
    - **安全状态**：存在某种进程执行顺序，使所有进程都能顺利完成。
      * **安全状态必须先变成不安全状态，才可能变成死锁**
    - **不安全状态**：不存在这样的顺序，可能导致死锁。
  + **死锁检测与解除**：无法避免死锁发生，只能解决已经出现的死锁。需要周期性的检测。
    - 检测方法
      * **定时检测**：系统周期性检测死锁状态。
      * **等待时检测**：进程请求资源时检测死锁。
    - 解除方法：**撤销进程法，资源剥夺法，重启系统法**

## 第三章：内存管理

### 作业

* 定义：计算机系统按指定步骤对原始数据进行处理，并计算结果的一次加工过程，是用户提交给系统的一个完整任务实体。**作业进入内存后才能运行（因为作业存于外存）**。
  + 作业由一个或多个 **作业步**组成，每个作业步完成作业中的一个处理步骤。作业步和进程不是一一对应的。
  + 一个**计算单位**，它代表一次完整的数据处理任务。
* 作业全生命周期
  + **提交状态**：用户把数据放在输入设备上
  + **后备/收容状态**：系统根据需要，将作业及数据存放在后备存储器（磁盘、磁鼓等外存）中。作业仍处于外存，等待调度进入内存执行。
  + **执行状态**：作业开始，即作业进入内存
  + **完成状态**：系统释放作业占用的资源。
* **作业控制块 JCB**
  + **每个作业对应一个JCB**
  + 是作业在系统中存在的唯一标识
  + 包含：作业名，作业类型，资源要求，资源使用情况，优先级，当前状态，作业提交时间，作业运行时间。
* **作业调度**
  + 定义：从后备状态队列中选取合适的作业投入运行
  + **高级调度（作业调度 / 宏观调度）**
    - 对象：作业（Job）
    - 作用：从后备队列选择合适作业，调入内存准备运行；回收结束作业的资源
    - 时间尺度：分钟～小时～天
  + **中级调度（交换调度 / 中程调度 / 平衡负载调度）Unix才有**
    - 对象：已在内存中的进程
    - 作用：
      * 调整内存中进程数量
      * 平衡 CPU 和 I/O 负载
      * 决定哪些进程暂时挂起（换出到外存）或调回内存
    - 时间尺度：几秒到几分钟
    - 特点：，属于“中等时间粒度”的调度
  + **低级调度（短程调度 / CPU 调度）**
    - 对象：就绪队列中的进程
    - 作用：决定哪一个进程获取 CPU 执行，通常按优先级或调度算法执行
    - 时间尺度：毫秒级
    - 特点：微观、频繁、面向 CPU 使用效率
* **周转时间**
  + 定义：作业从提交到完成所经历的总时间，即等待时间和运行时间之和（包含后备状态，执行状态）
  + 公式：
    - 其中 为作业提交时间， 为作业完成时间
* **平均周转时间**
  + **目的**：反映作业调度效率，平均周转时间越小越好
* **带权周转时间**
  + 定义：周转时间与作业服务时间的比值，表示作业等待时间相对于运行时间的比例。这个是≥1的，反映作业等待时间占服务时间的比例。
  + 公式：
    - 其中 为作业实际需要的 CPU 时间（服务时间）
    - 值越大 → 作业等待占比越高，调度效率相对越低
    - 值越接近 1 → 作业等待少，调度效率高
* **平均带权周转时间**
* **作业调度算法**
  + 定义：提高系统吞吐量，尽可能使系统处于忙碌
  + **FCFS（First-Come, First-Served，先来先服务）**
    - **原则**：按作业提交系统的顺序执行
    - **特点**：
      * 简单易实现
      * 公平，每个作业按先后顺序调度
      * **缺点**：可能导致“长作业阻塞短作业”（饥饿问题）
    - **适用场景**：批处理系统
  + **SJF（Shortest Job First，最短作业优先）**
    - **原则**：优先调度服务时间最短的作业
    - **特点**：
      * 可以最小化平均周转时间
      * 可能导致**长作业饥饿**
      * 分为非抢占式 SJF和抢占式 SJF（最短剩余时间优先 SRTF）
    - **适用场景**：CPU 时间可预测的系统
  + **HRN（Highest Response Ratio Next，最高响应比优先）**
    - **原则**：选择响应比最高的作业调度
    - 响应比：周转时间/运行时间 ；这里的运行时间是估计的
    - **特点**：
      * 综合考虑等待时间和服务时间
      * 避免 SJF 的长作业饥饿问题；既照顾短作业，也照顾长作业
      * 但是计算量太大
    - **适用场景**：批处理系统，需要兼顾效率和公平性
  + 优先级调度算法
    - 使用外设的作业优先级更高，因为他们使用CPU的时间少
  + 均衡调度算法
    - **原则**：综合考虑作业等待时间、服务时间、优先级等多种因素进行调度

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 调度算法 | 原则 | 优点 | 缺点 | 适用场景 |
| FCFS（先来先服务） | 按作业提交顺序执行 | 简单、公平 | 长作业可能阻塞短作业，平均周转时间不一定最优 | 批处理系统 |
| SJF（最短作业优先） | 优先执行运行时间最短的作业 | 最小化平均周转时间 | 长作业可能饥饿，需要预知作业时间 | 作业时间可预测系统 |
| SRTF（最短剩余时间优先） | 抢占式 SJF | 更小平均周转时间 | 长作业可能饥饿，频繁切换 | 实时或批处理系统 |
| HRN（最高响应比优先） | 响应比最高的作业先执行 | 兼顾短作业和长作业，避免长作业饥饿 | 计算量大 | 批处理系统，兼顾效率和公平 |
| 优先级调度 | 按作业优先级调度 | 灵活，可控制关键作业 | 低优先级作业可能饥饿 | 系统需要区分重要作业 |
| 均衡调度 | 综合等待时间、服务时间、优先级等因素 | 系统整体效率和公平兼顾 | 实现复杂 | 多任务混合环境 |

### 存储管理功能

* **存储保护**：防止用户程序之间互相干扰，保证各自运行在自己的内存空间内。这项功能通常由 **硬件和软件配合实现**：
  + 软件上通过规定程序的 上界和下界地址 来限制访问范围；
  + 硬件上通过 界限寄存器或重定位寄存器 来检测是否越界。
* **存储扩充**：提高主存的利用率，使得程序的大小和结构不再受主存容量的限制。
  + 主要依靠 **虚拟存储技术** 来实现，也就是利用辅存（如磁盘）作为主存的扩展，让系统表现出比实际更大的“逻辑内存”。
* **地址映射 / 地址重定位 / 地址变换**：这是存储管理中最核心的功能之一。其作用是将程序中使用的 逻辑地址 转换为 主存中的物理地址。**由硬件内存管理单元MMU完成**。
  + 因为 CPU 只能直接访问主存（内存），不能直接访问辅存（磁盘等）。所以，当程序被加载运行时，必须先从辅存调入主存；然后 CPU 才能通过地址映射访问主存中的数据或指令。
* **主存分配**：当系统中有多个作业或进程同时运行时，操作系统需要把空闲的主存空间 分配给不同的作业或进程使用，以保证多道程序能够并行运行。
* **内部碎片**：分配给作业或进程的内存块大于其实际需要的空间，未被使用的部分称为内部碎片。
* **外部碎片**：内存中空闲空间分散在各处，无法形成足够大的连续空间以满足作业的分配请求，这些零散空间称为外部碎片。

|  |  |  |
| --- | --- | --- |
| **存储管理方式** | **内部碎片** | **外部碎片** |
| **固定分区** | 有 | 无 |
| **动态分区** | 无 | 有 |
| **分页** | 有 | 无 |
| **分段** | 无 | 有 |
| **段页式** | 有 | 无 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 概念 | 存储位置 | 功能 | 大小 | 示例 |
| 物理块 | 主存（RAM） | 内存分配 & 数据传输单位 | 512B~4KB（与页大小一致） | 逻辑页 4KB → 物理块 4KB |
| 盘块 | 外存（磁盘、磁带） | 文件存储分配单位 | 若干扇区（通常 4KB） | 文件 10KB → 占 3 个盘块 |

**每次启动磁盘通常读写一个盘块**

### **存储器的三级结构**（计组里面有5层结构）

* **高速缓存（Cache，SRAM）**
  + 速度最快
  + 位于 **CPU 与主存之间**
  + 利用**空间换时间**的思想，加快数据访问速度
* **内部存储器（主存，DRAM）**
  + 程序运行的主要存储空间
  + 通常分为：
    - **系统区**：存放操作系统及其相关程序
    - **用户区**：存放用户的程序和数据
* **外部存储器（辅存）**
  + 提供**大容量的存储空间**
  + 解决主存空间不足问题
  + 利用**时间换空间**的思想（访问速度慢，但容量大）

### 分区存储（物理划分）

* 定义：将内存划分为若干连续（内存）的固定或可变大小的区块（分区），**每个分区只能同时分配给一个作业或进程使用**，用于存放其程序和数据。。每个作业被装入一个或多个分区。
* 分区方式
  + **固定分区**
    - 分区大小在分配时就固定不变
    - 通过**上限寄存器**和**下限寄存器**控制
    - 可能产生内部碎片（分区未被充分利用）
  + **可变分区**
    - 按作业需求划分，分区数量和位置动态变化
    - 可能产生外碎片（分区之间无法利用的空间）
    - 常用数据结构管理空闲区块：
      * 空间表：表格形式
      * 分区链：链表形式
      * 请求表：记录内存资源请求
    - 分区回收与合并规则
      * 回收区有下邻空闲区 → 空闲区数量不变（合并）
      * 回收区有上邻空闲区 → 空闲区数量不变（合并）
      * 回收区上下都有空闲区 → 空闲区数量减1
      * 回收区上下都无空闲区 → 空闲区数量加1
    - 分配算法
      * **首次适应（FF）**：首地址递增排序，从低地址开始找到第一个够用的空闲区；可能产生外部碎片，高地址利用率低
      * **循环首次适应（NF）**：首地址递增排序，从上次分配位置开始扫描，将地址最小的够用的空间分配出去；大作业可能找不到空间
      * **最佳适应（BF）**：按空闲区大小递增排列，分配最小足够空间；会产生外部碎片
      * **最坏适应（WF）**：按空闲区大小递减排列，分配最大空闲区；较大空闲区不会被保留；会产生外部碎片

### 分页存储（内存利用率最高，逻辑划分）

* 定义：分页存储是一种将内存和进程的逻辑地址空间划分为**固定**大小单元的方法，用来实现**非连续内存**分配。**页的大小由系统决定**。
  + 内存被划分为大小相同的“页”，逻辑地址连续，但物理地址不连续。
  + 进程的最后一页可能装不满，形成**内部碎片**。
  + 不能直接实现进程间的共享，因为分页存储只管理单个进程的逻辑页到物理页框的映射，不考虑不同进程间的逻辑共享，必须由操作系统额外安排共享页。
* 分页方式
  + **基本分页**：判断作业是否能完整装入内存，满足则装入，不满足则等待。
  + **请求分页**：作业启动时不必全部加载，运行时按需调入。
* 数据结构：
  + **页表**：记录逻辑页号对应的物理块号，保存于内存。**每个作业对应一张页表**。
  + **快表（TLB）**：CPU 内的高速缓存，存储部分页表项，保存于高速缓存。
    - 命中时：直接获得物理页号 + 页内偏移，仅需一次主存访问。（查快表不访问主存）
    - 未命中时：需两次主存访问。（先查页表，再取数据）
  + **物理页面表**：描述物理内存分配情况，常用**位示图**或空闲页链表。
    - **位示图**
      * 一种用 **二进制位（bit）** 来表示**内存块或磁盘块分配状态**的数据结构
      * 每个字长为 32 位；一个字就能表示 32 个内存块的使用情况；
  + **请求表**：记录各进程页表位置和大小，便于多进程管理。
  + 访问数据时需两次主存访问（查页表 + 取数据）
* 虚拟地址 = 页号（高位） + 页内偏移（低位） **一维**的（因为逻辑上看像一条线被分成了很多份）

### 分段存储（逻辑划分）

* 定义：将程序按逻辑结构划分为若干 **长度可变的段**（如代码段、数据段、栈段等），每个**段**占用 **连续的物理内存空间**，段与段之间的**物理地址不必连续**。段的完整性便于实现共享。**段的大小由程序员决定**。
* **段内地址位数 = 总地址位数 − 段号位数**
  + 例如 32 位地址、8 位段号 → 段内地址 24 位，可表示 0 ~ 224−1，共 16 MB。参考虚拟地址的构成。
* 数据结构：
  + **段表**：存放在**内存**中，记录段号、段长、段基址等信息。**每个作业对应一张段表**。
  + 段号寄存器：类似于快表，不是必需的。
  + 物理内存分配管理
  + 请求表
* 地址变换过程：
  + 从虚拟地址取出段号
  + 查段表，得到段表项
  + 检查段内地址是否越界
    - **越界 → 产生越界中断**
    - 合法 → 物理地址 = 段基址 + 段内地址
* 虚拟地址 = 段号（高位）+ 段内地址（低位）；是**二维**的（一个二维表格，一维是“段”，每个段内部再有一维是“偏移”。）

### 段页式存储（逻辑划分）

* 定义：段页式存储管理是一种结合分段和分页优点的内存管理方式。**段之间不必有连续的物理地址，段内也不必有连续的物理地址**。
  + 逻辑上：程序按段划分（如代码段、数据段、堆栈段）。
  + 物理上：每个段再被划分为若干页，页为物理分配的基本单位。
* 地址变换过程
  + 用段号查段表 → 得到该段页表的起始地址。
  + 用段内页号查页表 → 得到对应的物理块号。
  + 拼接物理块号 + 页内地址 → 得到物理地址。
* 数据结构
  + 段表：记录每个段对应的页表起始地址、段长等。**每个作业对应一张段表**。
    - **段的长度必须是页长的整数倍，段的起点必须是某一页的起点**
  + 页表：记录逻辑页号到物理块号的映射关系。**每个段对应一个页表**。
  + 若无快表，需访问内存 **三次**（段表、页表、取数据）；若有快表可减少一次或多次
* 虚拟地址 = 段号 + 段内页号 + 业内偏移；是**二维**的

**总结表格**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 对比项 | 分区存储 | 分页存储 | 分段存储 | 段页式存储 |
| 划分依据 | 物理划分 | 逻辑划分 | 逻辑划分 | 逻辑划分 |
| 内存分配 | 连续 | 不连续 | 段间不连续、段内连续 | 段间不连续、段内不连续 |
| 地址空间 | 无虚拟地址 | 一维虚拟地址 | 二维虚拟地址 | 二维虚拟地址 |
| 内外碎片 | 内/外碎片均有 | 仅内部碎片 | 仅外部碎片 | 几乎无外碎片 |
| 共享与保护 | 不支持 | 不直接支持 | 支持 | 支持 |
| 地址变换 | 物理=逻辑+基址 | 页表查页 | 段表查段 | 段表→页表 |
| 地址转换次数（无TLB） | 1 | 2 | 2 | 3 |
| 管理单位 | 分区 | 页 | 段 | 段 + 页 |
| 是否支持虚拟存储 | 否 | 是 | 是 | 是 |
| 实现复杂度 | 低 | 中 | 中 | 高 |

|  |  |  |  |
| --- | --- | --- | --- |
| 项目 | 逻辑上连续 | 物理上连续 | 说明 |
| 虚拟存储（虚拟地址空间） | ✅ 是 | ❌ 否 | 对程序员连续；映射后物理可不连续 |
| 物理内存（实存） | ❌ 否 | ❌ 否 | 操作系统任意分配，不要求连续 |

### 虚拟存储器 — 时间换空间

* 只把当前需要的部分装入内存，其他部分留在磁盘中。**“分区”无法划分虚拟内存，所以没有“请求分区虚拟内存”的说法**。
* **理论基础：局部性原理**
  + **时间局部性：** 程序在一段时间内往往反复访问相同的指令或数据。
  + **空间局部性：** 程序执行时倾向于访问与当前地址相邻的区域。
* **物质基础（支撑条件）：**
  + **相当容量的 辅存（例如磁盘），**用于存放暂时未装入的部分**。**
  + **一定容量的 主存，**作为执行时的工作区**。**
  + **必要的 硬件，软件，固件支持：**
    - **页/段表机制（软件） —** 记录逻辑页/段与物理块的映射**。**
    - **缺页/段中断机构（硬件） —** 发现页/段不在内存时发出中断请求**。**
    - **地址变换机构（硬件或固件） —** 在页或段调入内存时完成逻辑地址到物理地址的转换**。**比如**MMU**
* **特征：**
  + **多次性：** 一个作业可分多次调入内存运行。
  + **对换性：** 运行过程中允许在主存与辅存之间来回调入、调出。
  + **虚拟性：** 从逻辑上扩充了内存容量，使得用户感觉内存比实际更大。
* **请求分页虚拟内存技术**
  + 定义：当访问的页不在内存中时，产生缺页中断，由系统选择一个页调出，再将所需页调入。
  + **虚拟地址**
    - **页号**：指明要访问的页
    - **页内偏移**：指明在该页内的具体位置
  + **页表**
    - **作用**：记录每页在内存中的状态及调入调出的信息，用于地址映射与页面管理。
    - **主要字段**：
      * 页号，物理块号
      * **状态位 P**：显示页面是否在内存中，供程序访问参考。
      * **访问位 A**：页面在调出或置换时参考，表示近期是否被访问过。
      * **修改位 M**：页面调入内存后是否被修改，用于决定换出时是否需要写回磁盘。
      * **外存地址**：页面在辅存（磁盘）中的存放位置。**盘块号**
    - **页面淘汰优先级（A/M组合判断）**：
      * **未被访问 + 未被修改 → 最易淘汰**
      * 未被访问 + 已修改
      * 已访问 + 未修改
      * **已访问 + 已修改 → 最难淘汰**
  + **缺页中断**
    - **定义**：进程访问的页面不在内存中时，缺页中断机构会向CPU发送缺页中断。
    - **处理流程**：**修改页表信息；磁盘 I/O，将缺页调入内存；分配主存页框。**和操作系统配合完成
    - **缺页中断率**：
      * F：缺页次数
      * A：总访问次数
    - **影响因素**：**主存分配页框数量，页面大小，程序编制方式，页面置换算法**
  + **页面调度（置换）**
    - **定义**：当内存页框已满且缺页发生时，选择一页调出到磁盘，为新页腾出空间。
    - **问题**：若调出后不久又被访问，可能导致 颠簸/抖动。
    - 常见页面置换算法：**都不考虑空间局部性（因为是按页调取，所以包含了），只有FIFO不考虑时间局部性**
      * **最佳调用算法（OPT）**
        + 理想算法，总是淘汰未来最长时间内不会被访问的页面。用作衡量其他算法效果的标准。
      * **先进先出（FIFO）**
        + 淘汰最早进入内存的页面，不管它是否经常用
        + 特点：简单，但可能出现 **Belady 异常**（随着分配页框增多，某些“旧但频繁使用”的页面可能被保留在内存中，而真正不常用或未来很快就会被访问的页面也可能被淘汰，进而导致缺页中断率上升）。
      * **最近最久未使用（LRU）**
        + 淘汰最长时间未被访问的页面。需要维护一个访问顺序的结构，如窗口、链表等。因此开销大。
      * **最近最不经常使用（LFU）**
        + 淘汰访问次数最少的页面。需要记录每个页面被访问的次数和进入内存的时间。
        + 示例：window=3，若三页访问次数相同，最早加入 window 的页被替换。
      * **Clock 算法**
        + 提供了一种 低开销（因为只用记录1位A）、近似 LRU 的实现。
        + 页面按**环形排列**，每页有**访问位 A**：
        + A = 1 → 最近访问过
        + A = 0 → 较长时间未访问
        + 指针指向上次扫描位置。
        + 置换步骤：

从指针处扫描页面。

A = 0 → 淘汰该页；  
A = 1 → 清 0，指针移至下一页，继续扫描。

找到 A = 0 的页进行置换，调入新页并置 A = 1。

* **请求分段虚拟内存技术**
  + **虚拟地址**
    - **段号**：指明要访问的段
    - **段内偏移**：指明在该段内的具体位置
  + **段表**
    - 段名、段长、段在内存中的起始地址
    - **存取方式**：段的访问属性（只执行、只读或可读写）
    - **访问位 A**：记录段在一段时间内的访问情况
    - **修改位 M**：标记该段调入内存后是否被修改
    - **存在位 P**：指示该段是否在内存
    - **增补位**：表示该段是否在运行过程中动态增长过
    - **外存起始地址**：标记该段在外存（磁盘）中的起始位置。**盘块号**。
  + **缺段中断**
    - 定义：当进程访问的段尚未调入内存时，缺段中断机构会产生缺段中断信号，通知操作系统将所需段调入内存。
    - **缺段中断在一条指令执行期间可以产生和处理多次，但不会出现指令或数据被拆分在两个段的情况**。
* **段页式虚拟内存技术**
  + **虚拟地址**：
    - **段号（高位），段内页号（中位），页内地址（低位）**
  + 把**程序按逻辑结构分段**，**页**为基本**传输单位**
  + **段的长度必须是页长的整数倍，段的起点必须是某一页的起点**
  + 地址变换机构**需要查2次表（段表和页表），系统开销大（访问3次主存）**

## 第四章：文件（外存）管理

**常见存储介质特性**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 存储介质 | 访问方式 | 访问速度 | 文件组织形式 | 适用特点 |
| 磁带 | 顺序访问 | 慢 | 顺序文件 | 顺序读写效率高，随机访问低，适合备份、归档 |
| 光盘 | 顺序/随机访问（主要顺序） | 中 | 顺序文件 | 顺序访问效率高，随机访问效率一般 |
| 磁盘 (HDD) | 随机访问 | 快 | 顺序 / 索引 / 链接 / 散列文件 | 支持随机访问，灵活选择文件组织 |
| 固态硬盘 (SSD) | 随机访问 | 很快 | 顺序 / 索引 / 链接 / 散列文件 | 高速随机访问，适合频繁查找或散列存取 |
| 磁鼓 | 顺序/循环访问 | 中 | 顺序文件 | 早期随机访问有限，顺序存取效率高 |
| 磁芯存储 | 随机访问 | 较快 | 索引 / 链接 / 散列文件 | 小容量，主要用于高速随机访问或缓存 |
| 光磁盘 | 随机访问 | 中 | 顺序/ 链接 / 索引文件 | 可擦写光盘，适合归档和随机访问 |

### 文件

* 定义：逻辑上有完整意义的信息集合；每个文件要用一个名字作标识，称为**文件名**；
  + 文件被划成大小不同的逻辑记录，不是块！
* 文件分类
  + 性质和用途：系统文件，库文件，用户文件
  + 组织和处理方式：普通文件，目录文件，特别文件（把设备看作文件，比如标准输入设备是鼠标和键盘；无需打开和关闭）
  + 使用和管理情况：临时，档案，永久
  + 保护方式：只读，读写，执行，不保护
  + 信息的流向：输入，输出，输入输出
* 文件系统
  + 系统角度的定义：负责文件存储空间管理的机构
  + 用户角度的定义：存储信息和使用信息的接口
  + 目地：**按名存取**，这是为了方便用户使用
  + 组成部分：**目录，组织，存储空间的管理，文件操作，文件安全措施**
* 文件的 **存储介质**
  + **存储介质**：用于**存储数据的物理材料或媒介**，提供信息的存储和保持功能。磁盘、磁带、磁鼓
  + **存储设备**：利用存储介质实现数据**存储、读写和管理**功能的**硬件装置**。
  + 介质单位划分
    - **物理单位**：**卷**
      * 操作系统管理的逻辑存储单位
      * 一个卷可以对应 **单个磁盘** 或 **多个磁盘组成的磁盘组**（如 RAID）。
      * **文件系统建立在卷上**。
    - **交换单位**：**块**或**簇**
      * **主存与外存交换信息的基本单位**。
      * **由存储介质上连续的物理地址组成（通常由若干扇区构成）**。
  + **逻辑文件存放在存储介质上时，采用的 组织形式 与 存储介质特性 有关**。
    - 磁带，光盘 → 顺序文件
    - 磁盘（随机存取介质） → 索引文件，散列文件，顺序文件
* 文件的 **逻辑存储方式**
  + **定义**：**从用户或应用程序角度看**，文件的组织方式，描述数据之间的逻辑关系，而不关心物理存储。
  + **分类**：
    - **流式文件**
      * 相关的有序字符的集合。有固定结构。
      * 适合顺序访问
      * 典型例子：UNIX、DOS、Windows 普通文件，图片
    - **记录式文件**
      * 文件由若干条记录组成，每条记录有固定或可变长度。有固定结构。
      * 适合随机访问
      * 适合数据库表、Excel表等结构化数据
* 文件的 **物理存储方式**
  + **定义**：文件在磁盘或其他**存储介质上实际存放的方式**，由操作系统管理。
  + **选择时需要考虑：文件大小，记录是否定长，访问的频繁程度，存取方法等**
  + **类型**：
    - **顺序结构**
      * 文件占用连续的磁盘块
      * 检索效率高，存储空间利用率低，但扩展困难
      * 典型系统：UNIX
    - **链式结构**
      * 文件由不连续磁盘块组成，通过指针连接
        + 指针占用额外内存，随机存取效率低，破坏了物理块的完整性（文件映照克服后2个缺点）
      * 文件扩充容易，顺序存取效率高，空间利用率高
      * 分两种：
        + **文件映照/显示链式**：指针集中存放在表格中，数据块仅存数据，随机访问快
        + **隐式链式**：指针存放在数据块内部，随机访问慢，破坏文件完整性
      * 典型系统：Windows, DOS
    - **索引结构**
      * 文件有独立索引表存放各数据块地址
      * 支持随机访问，扩展容易，但占用额外空间
    - **连续索引结构**
      * 文件部分连续，辅以索引表
      * 兼顾顺序访问效率和随机访问能力

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 结构类型 | 磁盘块连续性 | 存取方式 | 随机访问效率 | 顺序访问效率 | 文件扩展 | 空间利用率 | 指针开销 | 典型系统 |
| 顺序结构 | 完全连续 | 顺序 | 低（几乎不可随机访问） | 高 | 扩展困难 | 低 | 无 | UNIX |
| 链式结构 | 不连续，通过指针连接 | 顺序/随机 | 低（隐式链式）或高（文件映照链式） | 高 | 扩展容易 | 高 | 指针占用额外空间 | Windows, DOS |
| 索引结构 | 不连续 | 顺序/随机 | 高 | 中等 | 扩展容易 | 中 | 索引表占用额外空间 | 通用（磁盘） |
| 连续索引结构 | 文件部分连续 + 索引表 | 顺序/随机 | 高 | 高 | 扩展容易 | 高 | 索引表占用空间 | 通用（磁盘） |

### 目录管理

* **文件目录/文件说明/文件控制块（File Control Block, FCB）**
  + 相当于文件的“名址录”。 
  + 完成 **文件名 → 文件物理地址** 的转换。**一个文件对应一个FCB**
  + 记录文件的基本信息（名称、属性、全息、物理位置等）
* 每个文件对应一个 **目录项**，**负责文件名和FCB的映射**。
* **文件目录页**：是外存中保存目录项和FCB信息的物理页
  + **系统运行时仅将当前目录对应的目录页加载到主存**
* **目录管理的基本要求**
  + **按名存取**：用户通过文件名访问文件，而无需关心物理地址。
  + **提高检索速度**：通过组织结构优化文件查找效率。
  + **文件共享**：允许多个用户或进程访问同一文件。
  + **文件同名**：不同用户可拥有相同文件名的文件。
* 目录结构
  + **一级目录结构** 
    - 只有一个线性表用于存储；只能实现按名存取，但是后三点管理要求无法达成
    - **目录表**存在 **磁盘上**
  + **二级目录结构** 
    - 分成主文件目录和用户文件目录；UFD内文件名唯一，但是不同UFD内可以重名
    - **主目录 MFD**: 管理整个文件系统中的所有用户文件目录。每个用户对应一个唯一的 UFD，MFD 保存 UFD 的位置和相关信息。
    - **用户文件目录 UFD**：文件名、文件属性（大小、创建时间、权限等）、文件存储位置。只对单个用户可见，用户通过自己的 UFD 管理文件。
    - **目录表**存在 **磁盘上**
  + **树形目录结构** 
    - 由 根目录，目录文件，一般文件形成
    - 有绝对路径和相对路径；调入内存查找时命中率不高（因为路径越深，涉及的目录页越多，而内存通常只缓存当前或最近访问的几个目录页）
    - **目录表**存在 **磁盘上**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 目录结构类型 | 层级关系 | 特点 | 优点 | 缺点 |
| 一级目录结构 | 单层（线性表） | 所有文件目录集中存放 | 结构简单，易于实现 | 无法区分用户，不能支持文件重名或共享 |
| 二级目录结构 | 两层（主目录 + 用户目录） | 分为主文件目录（MFD）与用户文件目录（UFD） | UFD内文件名唯一，但是不同UFD内可以重名，查找速度快 | 结构固定，无法灵活嵌套子目录 |
| 树形目录结构 | 多层（层次化） | 根目录、子目录、文件形成树状层级 | 组织灵活，支持路径访问与子目录管理，有绝对路径和相对路径 | 查找时需逐层访问，命中率较低 |

### 空闲文件管理

* 用于记录和分配磁盘中未被占用的存储空间，以便系统在创建新文件时能够快速找到空闲区。

|  |  |  |
| --- | --- | --- |
| 方法 | 基本思想 | 特点与适用场景 |
| 空闲文件项法 | 将每个空闲区视作一个特殊的“文件”，系统扫描整个目录表，选择标志为空闲的项进行分配。 | 📌 简单直观，但会使目录表增大；适用于**少量大空闲区**的情况。 |
| 空闲文件目录法 | 设立一个**单独的空闲文件目录**来记录所有空闲区信息。 | 📌 适用于**少量空白区或连续文件系统**；若系统中有大量零碎小空闲区，目录会膨胀、效率下降。 |
| 空闲块链法 | 将所有空闲磁盘块用**链表结构**连接起来（非连续结构）。 | 📌 灵活但实现复杂，可按不同策略组织：1，2很简单，效率低；3复杂，效率高 • 按空闲大小顺序链接（便于快速找到合适空闲块） • 按释放先后顺序链接（便于回收） • **成组链接（UNIX采用）**：把多个空闲块地址集中存放在一个块中，从而减少访问指针次数、提高分配效率。 |
| 位视图法（Bit Map） | 使用一个位图（bitmap）结构，每一位对应一个磁盘块： 0 → 空闲；1 → 已分配 | 📌 占用空间极小，支持快速判断与分配；但位运算逻辑相反（释放时置 0，分配时置 1）。适用于**磁盘块数量多、块尺寸小的系统**。 |

### 文件的使用、共享与安全

* **文件删除**
  + 删除文件 不会 对文件占用的存储单元全部清零（只会标记，并加入回收站），只有清空回收站时才会。
* **文件使用**
  + 通过**系统调用或命令调用**实现
  + 文件在被打开后，**仅允许打开者使用**。
  + 打开/建立文件→读/写→关闭文件[→删除文件]
* **文件共享**
  + 自然灾害 / 误操作（“天灾人祸”）→ 使用文件副本（备份）
  + 系统故障 → 定期保存副本 + 定时存储（快照、日志）
  + 多用户同时访问带来的破坏 → 设置访问权限与锁机制，防止非法修改
  + 计算机病毒感染 → 普通用户仅赋予只读权限 + 安装杀毒软件
* **文件保护**
  + **文件保密**
    - **口令保护**：访问文件时需输入正确口令（password）。
    - **加密保护**：文件内容经算法加密，需密钥解密才能访问。
  + **访问控制**：通过控制文件访问的类型与主体（用户/进程）实现。
    - **访问控制矩阵**（Access Control Matrix）：二维表记录每个用户对每个文件的权限。
    - **访问控制表**（Access Control List, ACL）：针对每个文件记录所有用户的权限。
    - **用户权限表**（Capability List）：针对每个用户记录其能访问的文件及权限。
    - **隐藏文件目录证明**（Hidden Directory Authentication）：隐藏敏感文件或使用特殊验证方式。

## 第五章：输入输出管理

### IO设备

* 定义：CPU 和主存（内存）以外，用于与外界交换数据的所有硬件设备。
* 分类
  + **按速度划分**
    - **低速设备**：例如键盘、显示器
    - **中速设备**：例如打印机
    - **高速设备**：例如磁盘、光盘、闪存
  + **按传输方式划分**
    - **块设备**：以“块”为传输单位，典型设备有磁盘、光盘、闪存等高速设备，通常支持随机访问。
    - **字符设备**：以“字节”为传输单位，典型设备有调制解调器、键盘、显示器等低速设备，通常不可寻址，采用中断驱动方式。
  + **按传输方向划分**
    - **存储设备**：输入/输出操作以“块“为单位，例如磁盘、光盘、闪存
    - **输入/输出设备**：输入/输出操作以“字节“为单位，例如键盘、显示器、调制解调器
  + **按共享属性划分**
    - **独占设备**：一次仅能被一个进程使用，例如 CPU、打印机
    - **共享设备**：多个进程可以在同一时间间隔内访问，但同一时刻只能有一个进程进行实际交换信息，例如磁盘。共享设备一般可寻址，但不一定支持完全随机访问（如磁带）。
    - **虚拟设备**：通过脱机操作（**假脱机**）或模拟，将独占设备改造成共享设备。
* IO设备管理功能
  + 对外围设备的分配和回收
  + 外围设备的启动（用户不能启动，只能操作系统启动）
  + 磁盘的驱动调度
  + 外围设备的中断事件
  + 虚拟设备技术：通过软硬件，把独占模拟为共享 或 共享模拟为独占
* **设备控制器**：
  + 定义：**IO设备与计算机之间的数据交换接口**。
    - 在微机中通常称为 **接口卡**。
    - 每个控制器可以控制一台或多台设备（每台设备有一个唯一的地址）。
  + 种类：
    - **字符控制器**（按字符传输）
    - **块控制器**（按块传输）
  + 功能
    - 接收识别CPU的命令
    - CPU和设备直间的数据交换
    - 标识和报告设备的状态
    - 地址识别
    - 数据缓冲：IO设备和CPU之间数据传送的缓冲
    - 差错控制
  + 组成部分
    - **CPU 接口**：连接 CPU，实现指令和数据的交互
    - **设备接口**：连接具体的 IO 设备
    - **I/O 逻辑**：实现设备的控制逻辑、命令执行和数据缓冲

### IO操作控制方式

* 核心目标：减少 CPU 对 I/O 操作的直接干预，提高系统效率。
* **程序控制 I/O**（软件）
  + **过程**：
    - CPU 发出 I/O 测试指令。
    - CPU 不断测试设备是否空闲（忙等待）。
    - 数据传送由 CPU 完成。
  + **特点**：
    - CPU 与 I/O 串行工作。
    - CPU 一直等待，效率低。
  + **适用场景**：低速设备或小量数据传输。
* **中断驱动 I/O**（软件）
  + **定义**：CPU 发出 I/O 请求后，不再等待，设备准备好时发出中断通知 CPU。
  + **特点**：
    - CPU 可在 I/O 期间处理其他任务。
    - 数据传送完成后由中断机制通知 CPU。
  + **优点**：减少 CPU 空等，提高效率。
* **直接存储器访问DMA**（硬件）
  + **定义**：DMA 控制器直接负责内存与 I/O 设备之间的数据传输。
  + **过程**：
    - CPU 发出 I/O 请求并设定参数。
    - DMA 控制器接管数据传输。
    - 传输完成后 DMA 发中断通知 CPU。也就是说本来传输一直需要CPU参与，但是现在可以托管给DMA完成。
  + **特点**：
    - CPU 只在启动和完成时干预。
    - 数据传送完全由 DMA 负责，直接和内存成批交换数据。
    - **会和CPU抢占总线控制权，因此不能和CPU并行工作。**
  + **适用场景**：高速设备、大量数据传输。
* **通道控制 I/O**（硬件）
  + **定义**：专门的 I/O 处理机（通道）代替 CPU 管理 I/O 操作。
  + **特点**：
    - CPU 只发高层指令。
    - 通道负责整个 I/O 控制和数据传输。
    - 主存是**数据的存放位置**，通道是**负责数据搬运和控制的硬件**，CPU只需要下指令告诉通道“去传输这些数据”，然后可以去做别的事情。
  + **通道类型**：
    - **字节多路通道**
      * 功能：把通道的带宽以 **时间片方式分配** 给多个中低速设备。
      * 特点：一次传一个字节；通道快速切换，实现“并行”效果。
      * 应用：打印机、读卡机、终端机等低速外设。
    - **选择通道**
      * 功能：一次只服务一个高速设备。
      * 特点：一次传输一块数据（Block Transfer）；独占式。
      * 应用：磁带机、磁盘机等高速大数据量设备。
    - **成组多路通道**
      * 功能：字节多路通道 + 选择通道的改进。
      * 特点：
        + 一个通道可连接多个高速设备。
        + 一次传输一块数据。
        + 多设备间轮转，实现并行 I/O。
      * 应用：磁盘阵列、大型机并行 I/O。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| I/O 控制方式 | 数据传送由谁完成 | 串行/并行 | 软件/硬件 | CPU 负载 | 特点 | 速度 | 适用场景 |
| 程序控制 I/O | CPU | 串行 | 软件 | 高 | CPU 不断忙等待；CPU 与 I/O 串行工作 | 慢 | 低速设备、小数据量 |
| 中断驱动 I/O | CPU（中断协调） | 串行 | 软件 | 中等 | CPU 可处理其他任务；I/O 完成后由中断通知 | 中 | 中速设备、少量数据 |
| DMA | DMA 控制器 | 并行（CPU与I/O并行） | 硬件 | 低 | CPU 只在启动/完成时干预；数据传送由 DMA 完成 | 高 | 高速设备、大量数据 |
| 字节多路通道 | I/O 通道 | 并行（时间片多设备） | 硬件 | 很低 | 可同时连接多个中低速设备；一次传一个字节 | 中 | 打印机、终端机、读卡机等低速设备 |
| 选择通道 | I/O 通道 | 独占单设备 | 硬件 | 很低 | 一次只服务一个高速设备；一次传一块数据（Block Transfer） | 高 | 磁带机、磁盘机等高速大数据量设备 |
| 成组多路通道 | I/O 通道 | 并行（轮转多设备） | 硬件 | 很低 | 结合字节多路和选择通道；一次传输一块数据，多设备轮转 | 高 | 磁盘阵列、大型机并行 I/O |

### 中断

* **定义**：中断是 CPU 停止当前程序去处理别的事情，完成后再恢复原程序的一种机制，由硬件和软件协作实现。中断由CPU响应，但由操作系统负责具体处理。
  + 提高 CPU 利用率，实现 CPU 与低速设备并行工作
  + 能处理事故和异常情况
  + 支持分时操作，提高系统响应能力
  + **中断不属于资源，而是一种信号**
* 组成与分工
  + **中断响应**：由硬件完成，负责检测中断信号并通知 CPU
  + **中断处理**：由软件完成，执行对应的中断服务程序
* **检测方式**：CPU 在每条指令执行后检测是否有中断事件
* **中断源**：引起中断的事件或硬件部件，包括以下几类，优先级从高到低：
  + **硬件故障中断**
    - 来自硬件故障信号
    - 例子：掉电、存储器校验错
  + **程序自愿访管中断（系统调用）**
    - 程序员主动执行访管指令请求操作系统服务
    - 例子：用户程序请求 I/O、创建进程、文件操作、资源申请/释放
  + **程序性中断（内部异常/陷阱）**
    - 程序运行中因错误或特殊条件触发
    - 例子：浮点溢出、除零、用户态执行特权指令、存储越界、单步跟踪
  + **外部中断**
    - 来自处理器外部的非 I/O 设备
    - 例子：定时器中断、操作员按键中断
  + **输入输出中断**
    - 由 I/O 设备完成或出错引起
    - 例子：打印机缺纸、磁盘读写完成
* **中断处理过程**
  + **中断请求**：事件产生
  + **中断判优**：根据优先级判断是否响应
  + **中断响应**：CPU保存现场并跳转到中断服务程序
  + **中断处理**：执行服务程序完成操作
  + **中断返回**：恢复 CPU 原来状态，继续执行被中断程序

### 缓冲技术

* 定义：缓冲技术是在**主存中**设置一个临时存储区（缓冲区），用于 CPU 与 I/O 设备之间的数据交换。它通过暂存数据来缓解 CPU 与 I/O 设备速度不匹配的问题。
* **作用**
  + **缓和速度不匹配**
    - CPU 速度快，I/O 设备速度慢，缓冲区可以减少 CPU 空等。
  + **放宽中断响应时间**
    - I/O 数据先进入缓冲，不必 CPU 马上响应中断。
  + **解决数据粒度不匹配**
    - CPU 按字/字节处理，I/O 可能按块处理，缓冲区协调两者单位差异。
  + **提高并行性**
    - CPU 处理数据的同时，I/O 可与缓冲交互，实现并行工作。
* **原理**
  + 数据先写入缓冲区，再由 CPU 或 I/O 设备按需读取。
  + 缓冲区相当于 CPU 和设备之间的“过渡区”。
* **限制**
  + 如果 I/O 设备处理时间远短于 CPU，缓冲作用不大。
  + 不能减少对 CPU 的访问次数，只能减少中断次数。
  + 不能直接提高主机与设备间交换信息的速度。
* **分类**
  + **单缓冲**
    - CPU 和外设之间只有一个缓冲区。
  + **双缓冲**
    - 两个缓冲区，CPU 与外设交替使用，实现部分并行。
  + **环形缓冲**
    - 若干缓冲区组成环，进程或设备轮流使用，提高连续数据传输效率。
  + **缓冲池**
    - 系统维护多个缓冲区集合，供多个进程或设备共享，提高系统整体利用率。

### 设备分配算法

* 定义：操作系统如何把有限的 I/O 设备分配给多个进程。
* **排队/调度算法**
  + **FCFS（First-Come, First-Served）**
    - 按 I/O 请求到达的先后顺序排队执行。
    - 简单，但可能导致某些设备长时间等待（饥饿问题）。
  + **优先级调度（Priority）**
    - 根据 I/O 请求的优先级执行，高优先级请求先处理。
    - 优先级低的请求可能被延迟（可能产生饥饿）。
* 分配方式
  + **安全分配方式**
    - **定义**：系统保证每次分配资源后，都存在一种顺序，使得所有进程都能顺利完成。
    - **特点**：
      * 永远不会发生死锁。
      * 进程每次请求资源量有限，系统检查安全性后才分配。
    - **例子**：银行家算法（Banker’s Algorithm）就是典型实现。
  + **不安全分配方式**
    - **定义**：系统不保证资源分配后的状态安全，可能存在某种序列导致进程无法完成。
    - **特点**：
      * 可能出现死锁，但不一定每次都会死锁。
      * 允许进程多次请求资源（一次请求不足以完成任务）。
      * 系统不检查安全状态，分配策略更灵活，但有潜在风险。

|  |  |  |
| --- | --- | --- |
| 特性 | 安全分配 | 不安全分配 |
| 死锁风险 | 永远不会 | 可能发生 |
| 分配前检查 | 安全性检查 | 不检查 |
| 灵活性 | 较低 | 较高 |
| 适用场景 | 关键系统，需要保证进程完成 | 对性能要求高、允许风险的系统 |

### 独占设备管理

* 定义：独占设备是指同一时间只能被一个进程使用的 I/O 设备，如打印机、CPU、调制解调器等。操作系统需要管理这些设备的分配与使用，保证互斥访问。
* 设备标识方式：
  + **设备绝对号**
    - 系统内部使用的唯一标识符，用于区分每台设备的物理位置或硬件编号。
    - 例如：系统中第 1 台打印机的绝对号为 001。
  + **设备相对号**
    - 用户或程序使用的标识符，通常与设备类结合使用。
    - 例如：用户指定“打印机 2”，系统通过相对号映射到绝对号 001。
* 设备分配管理：
  + **设备类表**
    - 按设备类型分类，记录每类设备的数量和对应的设备表指针。
    - 作用：管理同类设备的集合，方便分配。
  + **设备表**
    - 记录具体设备的详细信息，包括：绝对号，相对号，当前占用的进程 ID，状态（空闲/占用/故障）
    - 作用：实现对每台设备的独占分配和状态跟踪。

### IO设备独立性与无关性

* 定义：操作系统提供抽象层，使得用户编程与具体物理设备脱钩，从而实现程序的可移植性和灵活性。
  + 用户在写程序时，不必关心具体哪台物理设备在执行输入输出操作。
  + 系统通过逻辑层和驱动层隐藏底层设备差异。
* **逻辑设备独立性**
  + 含义：程序只需指定设备类型（如打印机、显示器），不必关心哪一台具体设备。
  + 作用：同类设备可互换使用，系统自动匹配可用设备。
  + 例子：程序声明“打印机”，操作系统选择空闲的 HP 或 Canon 打印机执行任务。
* **物理设备独立性（IO设备无关性）**
  + 含义：程序只需提供 I/O 信息（读/写），无需关注设备操作方式、接口或传输方式。
  + 作用：应用程序与硬件操作隔离，底层由设备驱动和操作系统负责。
  + 例子：print("Hello") → 系统自动将信息通过显卡输出到显示器，无需程序关心 HDMI、VGA 或缓冲机制。
* 这两种设备独立性支撑了：**IO重定向**，即**程序不直接依赖具体硬件**，而是依赖操作系统提供的抽象接口。

### Spooling 假脱机技术

* 定义：通过 磁盘或磁鼓 作为高速缓冲，把独占设备（如打印机、磁带机）模拟成可被多个用户“共享”的虚拟设备。**全局并行，局部串行**。
  + **Spooling = Simultaneous Peripheral Operations On-Line**
  + **高速缓冲：**缓冲 I/O 设备与 CPU 之间的数据，协调 CPU 与慢速设备的速度差
* **功能**
  + **提高IO速度和吞吐量**：用户不用等待设备空闲，任务先落到磁盘，设备按序处理。
  + **将独占设备变为共享**：多个用户看起来像同时使用打印机等设备，但实际操作仍有序进行。
  + **实现虚拟设备功能**：用户看到的是一个“可同时使用”的设备，实际底层可能是单个独占设备，操作系统负责调度和管理。
* 组成
  + **CPU先把数据丢到内存“输入缓冲区” → 写入磁盘“输入井” → 程序生成输出 → 写入磁盘“输出井” → “输出缓冲区” → 设备执行**
  + **硬件部分（负责存储与缓冲）**
    - **输入井**：位于磁盘，保存用户提交的输入数据，形成排队等待的队列。
    - **输出井**：位于磁盘，保存程序输出的结果，形成排队等待外设处理的队列。
    - **输入缓冲区**：位于内存，暂存准备进入输入井的数据，加快 CPU 与磁盘的交互。
    - **输出缓冲区**：位于内存，暂存从输出井取出的数据，准备送给外设（如打印机）。
  + **软件部分（负责管理与调度）**
    - **预输入进程**：将内存中的“输入缓冲区“数据转移到磁盘”输入井“，实现全局并行，使用户无需等待设备。
    - **缓输出进程**：将“输出井“的数据送到”输出缓冲区“，再交给外设执行，保证独占设备局部串行执行。
    - **输入井管理进程**：管理输入井中的任务队列，把数据送给需要的程序，协调多个用户的输入数据，避免冲突。
    - **输出井管理进程**：管理输出井中的结果队列，组织不同用户的输出顺序，保证输出正确性与公平性。
* **例子**： 3 个用户 A、B、C 同时想打印作业，只有 1 台打印机，打印机很慢。
  + 没有 Spooling：
    - 用户 A 先打印 → CPU 等待打印机打印完
    - 用户 B 等待 → CPU 也被阻塞
    - 用户 C 等待 → 整个系统效率低
  + 有 Spooling：
    - 每个用户的打印内容先写入 磁盘上的“打印队列”（输入井）
    - CPU 可以继续处理其他任务，不用等打印机
    - 打印机从队列按顺序打印 → 用户无需一直等待
    - 输出完成后，系统通知对应用户
    - 效果：CPU 和打印机 同时工作，打印机虽然慢，但不会阻塞用户程序，独占设备被模拟成“可共享”。
  + 全局并行：CPU、磁盘、其他用户可以同时工作。
  + 局部串行：每个独占设备（打印机）仍按顺序执行作业，但用户不用等待。

### 磁盘访问时间与驱动调度

* **最外侧磁道号最小（一般从 0 开始），**
* **越往内侧，磁道号越大。**
* 磁盘访问时间主要由三个部分组成：
  + **寻道时间（最耗时的部分）**
    - 磁头移动到目标磁道所需时间。
    - 是磁盘访问中最耗时的部分。
    - 举例：磁头从 5 号磁道移动到 120 号磁道，需要若干毫秒。
  + **延迟时间**
    - 磁头在目标磁道上后，等待盘片旋转到目标扇区的时间。
    - 与磁盘转速（RPM）相关，平均延迟时间约为半圈旋转时间。
    - 举例：7200 RPM 硬盘 → 一圈 8.3 ms → 平均延迟约 4.15 ms。
  + **传送时间**
    - 数据从磁盘扇区传送到内存缓冲区所需时间。
    - 取决于数据量、磁盘转速和通道带宽。
    - 举例：一圈 512KB，盘片一圈 8.3 ms → 传输速率约 61 MB/s。
* **磁盘驱动调度**：目的是减少访问时间、尤其是寻道时间。
  + **移臂调度 (控制磁头移动顺序)**
    - **FCFS（先来先服务）**：按请求到达顺序处理；
      * 磁头可能来回“乱跑”，平均寻道时间可能很大
    - **SSTF（最短寻道时间优先）**：优先处理离当前磁头最近的请求；
      * 可能出现 饥饿现象（远处的请求长期得不到服务）。
    - **SCAN（电梯算法）**：磁头像电梯一样往返扫描，沿当前方向处理所有请求，到达最远请求后反向。
      * 平均寻道时间比 FCFS/SSTF 更稳定。
      * 既考虑了减少寻找时间，又不频繁的改变指针方向
    - **CSCAN（循环扫描）**：磁头沿一个方向扫描，到最远后立即回起点继续。
    - **N-SCAN**：将请求分块，每次处理一个块，块内按 SCAN 调度，
      * 同时兼顾时间顺序，因为先提交的请求会先被分成块。
  + **旋转调度**
    - 控制哪个扇区先被访问，减少延迟时间
    - 总是选择当前磁头所在磁道上最靠近读写位置、能最快访问的扇区进行数据传输。

## 第六章：操作系统安全与保护

### 系统安全

* **主要威胁：**
  + 假冒用户身份
  + 数据截取
  + **拒绝服务**：让合法用户无法正常使用系统资源或服务的攻击方式。攻击者的目的不是窃取数据或破坏内容，而是让系统“瘫痪”或“卡死”，导致服务不可用。
  + 非法修改信息
* **三大安全目标CIA**
  + **机密性（Confidentiality）**：确保系统中的信息**不被未授权的用户或实体访问**。
  + **完整性（Integrity）**：确保系统中的信息**不会被未授权地修改、删除或伪造**。
  + **可用性（Availability）**：确保系统资源和服务**在需要时可被合法用户正常访问和使用**。
* **三类典型攻击：**
  + 窃取机密信息
  + 擅自修改系统数据
  + 瘫痪系统运行
* 特征
  + **多面性**：涵盖物理安全、逻辑安全、安全管理
  + **动态性**：威胁和防御手段都在不断变化
  + **层次性**：从硬件、操作系统到应用层逐级防护
  + **适度性**：安全与性能之间的平衡
* 安全等级（美国TCSEC标准）
  + 共 **4类7级**，从低到高为：  
    **D < C1 < C2 < B1 < B2 < B3 < A1**
  + D：最低，几乎无保护
  + A1：最高，具形式化安全验证
* **安全保障体系：整个系统的安全框架与策略指导**
  + **安全策略**：定义系统如何防护
  + **安全模型**：用数学模型表达安全规则
    - **访问矩阵模型**：描述谁能访问什么
    - **信息流控制模型（Bell-La Padula）**：确保信息只能沿安全路径流动（例如从低机密到高机密不允许）
  + **安全机制**：实现安全目标的手段
    - 认证、授权、加密、审计、最小特权
* **访问控制与文件保护：实现安全的具体技术手段**
  + **访问控制策略**：从系统角度规定“谁可以访问哪些资源”（更宏观）。
    - 口令、访问权限、角色认证（不含加密）
  + **文件存储控制**：从文件系统角度规定“哪些文件如何被访问”（更具体）。
    - 口令、存取控制矩阵、存取控制表（不含加密）
* **数据加密与数字签名:** **从根本上不可能被未授权用户解密的加密算法**
  + **加密方式**
    - **对称加密**（如 DES）：加解密密钥相同，速度快，用于签名或局部数据加密。
    - **非对称加密**（如 RSA）：公钥加密，私钥解密，安全性高，用于签名和加密。
  + **加密类型**
    - **数据传输加密**：
      * **链加密**：每一段通信链路（节点之间）都会加密。
      * **端加密**：只在**通信的起点和终点**（发送方和接收方）进行加密/解密。
    - **数据存储加密**：
      * **文件级加密**：**单个文件或文件夹**加密。
      * **驱动器级加密**：**整个磁盘（驱动器）或分区加密**。
  + **数字签名**
    - 用公钥机制验证身份与完整性
    - 满足三点：
      * **接收者能验证发送者身份**
      * **发送者无法抵赖**
      * **接收者无法伪造**
  + **数字证明（数字证书）**
    - 由 CA（认证机构）签发
    - 内容包括：用户名、机构、公钥、有效期、证书编号、CA签名
* **用户认证方法:** **保证网络安全的第一道防线**
  + 口令认证
  + 物理认证（磁卡、IC卡等）
  + 生物识别
* **系统内部攻击**
  + 合法用户滥用权限
  + 木马程序、陷阱、伪造口令、非法系统调用
    - **木马**是一种 **伪装成正常程序、诱骗用户主动运行的恶意程序**。它看似无害，实则在后台执行窃密、远程控制、开后门等恶意操作
* **系统外部攻击**
  + **病毒**：寄生性，依附宿主文件（文件型、宏病毒、引导扇区病毒等）
    - 只有引导程序
  + **蠕虫**：一个**独立的可执行程序（**没有寄生性**）**，不依附其他文件。自我复制传播, 传播性没有病毒强。
    - 引导程序+蠕虫本身
  + **移动代码攻击**：指**可以在网络中迁移、在不同计算机上运行的程序**。
    - **隔离法**：将移动代码限制在一个“沙箱”里运行。
    - **解释法**：先解析代码，再决定是否允许它在沙箱中执行。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 是否依附 | 自我复制 | 传播方式 | 典型危害 |
| 病毒 | 依附宿主 | 是 | 文件、程序、介质 | 文件破坏、系统破坏 |
| 蠕虫 | 独立 | 是 | 网络自动传播 | 网络拥堵、系统资源占用 |
| 移动代码 | 独立 | 否（可复制） | 网络传输 | 未授权操作、信息泄露 |

* 操作系统的安全需求
  + 机密性需求，完整性需求，可记账性需求，可用性需求
* **可信计算机（TCB）**
  + 由 **硬件、软件、固件、协议** 组成，**包含OS功能**
  + 核心组件：**访问监视器**（不包含分层设计机制）
    - **彻底仲裁**：所有访问必须经过审查
    - **隔离性**：不同主体/客体访问互不干扰
    - **可证明性**：系统行为可验证
* **隔离式操作**：通过隔离保障安全
  + **物理隔离**：独立硬件（如专线）
  + **时间隔离**：不同任务轮流使用系统资源
  + **密码隔离**：加密隔离数据或通信
  + **逻辑隔离**：软件级隔离不同用户空间

## 错题

5.一个多道批处理系统中仅有P1和P2两个作业,P2比P1晚5ms到达。它们的计算和I/O操作顺序如下:

P1:计算60ms,I/O 80ms,计算20ms

P2:计算120ms,I/O 40ms,计算40ms;

若不考虑调度和切换时间,则完成两个作业需要的时间最少是。

A.240ms

B.260ms

C.340ms

D.360ms

B

多道批处理特点：

* CPU 与 I/O 可并行执行
* 每个作业有 CPU 阶段和 I/O 阶段
* 在多道批处理系统里，CPU 仍然是独占的资源，不能真正同时被两个作业共享。

P1：

第一段 CPU：0~60

I/O：60~140

第二段 CPU：180~200

P1 完成时间：200 ms

P2

第一段 CPU：60~180

I/O：180~220

第二段 CPU：220~260

P2 完成时间：260 ms

26.死锁与安全状态的关系是(

A.死锁状态有可能是安全状态

安全状态有可能成为死锁状态

C.不安全状态就是死锁状态

D.死锁状态一定是不安全状态

D

28.某系统中有A、B两类资源各6个,t时刻资源分配及需求情况如下表所示:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 进程 | A已分配数量 | B已分配数量 | A需求总量 | B需求总量 |
| P1 | 2 | 3 | 4 | 4 |
| P2 | 2 | 1 | 3 | 1 |
| P3 | 1 | 2 | 3 | 4 |

t 时刻安全检测结果是

A.存在安全序列P1,P2,P3

B存在安全序列P2,P1,P3

C.存在安全序列P2,P3,P1

D.不存在安全序列

B

因为这里已经分配出去了5A6B，所以只有1A能进行分配，因此需要先满足P2才行

42.某计算机主存按字节编址,采用二级分页存储管理,地址结构如下所示。

页目录号(10位) 页号(10位) 页内偏移(12位)

虚拟地址2050 1225H对应的页目录号、页号分别是

A.081H, 101H

B.081H, 401H

C.201H、101H

D.201H, 401H

A

20501225H → 0010 0000 0101 0000 0001 0010 0010 0101

目录号0010 0000 01 → 00 1000 0001 → 081H

页号01 0000 0001 → 101H

45.某硬盘有200个磁道(最外侧磁道号为0),磁道访问请求序列为130,42,180,15,199,当前磁头位于第58号磁道并从外侧向内侧移动。按照SCAN调度方法处理完上述请求后,磁头移过的磁道数是

A.208

B.287

C.325

D.382

C

正从外侧（0）向内侧移动（即磁道号增大方向）

46.虚存管理和实存管理的主要区别是(

A.虚存分逻辑地址和物理地址,实存不分

B.实存要求一程序在内存必须连续,虚存不需要连续的内存

C.实存要求一程序全部装入内存才开始运行,虚存允许程序在执行的过程中逐步装入

D.虚存以逻辑地址执行程序,实存以物理地址执行程序

C

47.下面所列的内容里,()不是DMA方式传输数据的特点。(多选)

A.直接与内存交换数据

B.成批交换数据

C.与CPU并行工作

D.快速传输数据

ABD

55.若用8个字(字长32位)组成的位示图管理内存,假定用户归还块号为100的内存块时,它对应位示图的位置为。

A.字号为3,块号为5

B.字号为4,块号为4

C.字号为3,块号为4

D.字号为4,块号为5

C

100/32 = 3 ... 4 → 字号3块号4

1.操作系统的不确定性是指

A.程序的运行次序不确定

B.程序多次运行的时间不确定

C.程序的运行结果不确定

D.程序周转时阁时间不确定

ABCD

下列选项中, 属于操作系统所管理的资源。(多选)

A.CPU

B.中断

C数据

D.内存

ACD

中断不是一种资源；此外类似于底层硬件，比如风扇，电源也都不归OS管理

多道程序设计具有如下特性: (多选)

A.相互制约

B.运行环境封闭性

C.执行过程间断性

D.资源共享

ACD

9.在下面的叙述中,不正确的是。(多选)

A.线程是比进程更小的能独立运行的基本单位.

B.引入线程可提高程序并发执行的程度,可进一步提高系统效率

C.线程的引入增加了程序执行时时空开销

D.1个进程一定包含多个线程

CD

下面关于线程的叙述中,不正确的是(多选)

A.不论是系统支持线程还是用户级线程,其切换都需要内核的支持

B.线程是资源的分配单位,进程是调度和分配的单位

C.不管系统中是否有线程,进程都是拥有资源的独立单位

D.在引入线程的系统中,进程仍是资源分配和调度分派的基本单位

ABD

没有线程，则进程是资源分配和调度分派的基本单位

11.下列关于父进程与子进程的叙述中,正确的是(多选)。

A.父进程与子进程可以并发执行

B.父进程与子进程共享虚拟地址空间

C.父进程与子进程有不同的进程控制

D.父进程与子进程不能同时使用同一临界资源

ACD

感觉D不是绝对的，为啥要选？同时访问只是容易产生冲突而已，不是不能

12.下列关于死锁的叙述中,正确的是(多选)。

A.可以通过剥夺进程资源解除死锁

B.死锁的预防方法能确保系统不发生死锁

C.银行家算法可以判断系统是否处于死锁状态

D.当系统出现死锁时,必然有两个或两个以上的进程处于阻塞态

ABD

银行家算法是判断处于安全状态

9.进程的调度算法有很多,如果选择不当,就会造成死锁。

错误

调度算法影响的是CPU**利用率、响应时间、吞吐量**。

17.可顺序存取的文件不一定能随机存取,但可随机存取的文件都可以顺序存取

正确

现有3个同时到达的作业J1、J2、J3,它们的执行时间分别为T1、T2和T3,且T1<T2<T3。系统按单道方式运行且采用短作业优先算法,则平均周转时间为

A.T1+T2+T3

B.(T1+T2+T3)/3

C.(3T1+2T2+T3)/3

D.(T1>2T2+3T3)/3

C

因为周转时间包含等待时间所以是 T1 + T1 + T2 + T1 + T2 + T3

8.当用户程序执行时,下面关于主存空间保护措施的叙述中,正确的是(多选)。

A.对自己主存区域中的信息既可读,又可写

B.对系统区域中的信息可读,不可写

C.对共享区域中的信息可读,不可写

D.对非共享区域中的信息不可读,也不可写

ACD

B是为了避免用户程序破坏系统。

【例3-6-6】在虚拟存储系统中,若进程在内存中占3块(开始时为空)采用先进先出页面淘汰算法,当执行访问页号序列为1、2、3、4、1、2、5、1、2、3、4、5、6时,将产生()次缺页中断。

A.7

B.8

C.9

D.10

D

1 1

2 12

3 123

4 234

1 341

2 412

5 125

1

2

3 253

4 534

5

6 346

【例3-4-2】一个分段存储管理系统中,地址长度32位,其中段号占8位,则最大段长是

A.2的8次方字节

B.2的16次方字节

C.2的24次方字节

D.2的32次方字节

C

【例4-4-6】某系统中,一个FCB占用32B,盘块大小为1KB,文件目录中共有3200个FCB,查找该目录中的一个文件,平均启动磁盘次数为

A.50

B.64

C.100

D.200

A

每次启动磁盘通常读写一个盘块

**1️⃣ 计算每个盘块可以存放的 FCB 数**

**2️⃣ 计算文件目录需要的盘块数**

**3️⃣ 平均查找磁盘次数**

* 假设顺序查找：

6.设磁盘的转速为3000转/分,将盘面划分成10个扇区。那么需要( )时间读取一个扇区

1转需要 60/3000 = 0.02秒

那么一个扇区就需要0.02 / 10 = 0.002秒