第 1 章 操作系统概论

1. 早期操作系统设计的主要目标是什么?

   简化硬件操作：为用户和程序员屏蔽复杂的硬件操作。

   提高资源利用率：通过批处理和调度技术，提高计算机硬件的使用效率。

   提供基本服务：为程序执行提供必要的输入/输出、存储管理等服务。

2. 操作系统是资源管理程序，它管理系统中的什么资源?

   处理器资源：CPU 调度与多任务管理。

   内存资源：内存分配与回收。

   存储资源：磁盘、文件系统管理。

   输入/输出设备：管理外围设备的访问。

   网络资源：网络连接和通信。

3. 为什么要引入多道程序系统?它有什么特点?

   引入原因:

   - 提高 CPU 利用率，避免 CPU 空闲时间过多。
   - 加速任务处理，提高系统吞吐量。

   特点:

   - **并发性**：多个程序同时驻留内存，交替执行。
   - **资源共享**：多个程序共享 CPU、内存和 I/O 设备等资源。
   - **独立性**：各程序之间互不干扰

4. 叙述操作系统的基本功能。

   进程管理：创建、撤销、调度和同步进程。

   内存管理：分配、回收和保护内存。

   文件管理：提供文件存储、检索和保护功能。

   设备管理：管理外围设备的访问和操作。

   用户接口：提供命令行界面和图形用户界面。

   安全与保护：保护系统资源和数据的完整性。

5. 批处理系统、分时系统和实时系统各有什么特点？各适合应用于哪些方面?

类型	**特点**	**适用场景**
批处理系统	任务自动批量处理，无需人工干预，强调吞吐量。	数据处理、大型计算任务
分时系统	支持多个用户同时交互，响应时间短，强调交互性。	多用户系统、桌面操作系统
实时系统	实时响应外部事件，保证时间约束，强调及时性和可靠性。	工业控制、航空航天系统

6. 操作系统的主要特性有哪些?

   并发性：多个任务同时执行。

   共享性：多个任务共享系统资源。

虚拟性：通过抽象技术呈现虚拟资源。
异步性：任务执行时间不确定。

7. 衡量 OS 的性能指标有哪些？什么是吞吐量、响应时间和周转时间？

性能指标：吞吐量、响应时间、周转时间、CPU 利用率、系统可靠性。

- **吞吐量**：单位时间内完成的任务数量。
- **响应时间**：从提交任务到开始响应的时间。
- **周转时间**：从提交任务到完成任务所需的总时间。

8. 什么是嵌入式系统?

嵌入式系统是一种专用计算机系统，嵌入到设备中完成特定任务，通常具有低功耗、小体积和高可靠性的特点，例如家电、汽车控制系统。

9. 什么是对称多处理？它有什么好处？

对称多处理（SMP）：多个 CPU 共享内存和外部设备，平等地处理任务。
好处：

- 提高系统性能和并行处理能力。
- 负载均衡，避免资源浪费。

10. 为了实现系统保护，CPU 通常有哪两种工作状态？各种状态下分别执行什么程 序？什么时候发生状态转换？状态转换由谁实现？

两种状态：

- **用户态**：执行普通用户程序，访问受限。
- **内核态**：执行操作系统程序，权限高。

状态转换：

- **用户态 → 内核态**：系统调用、硬件中断。
- **内核态 → 用户态**：完成系统服务后返回用户程序。

实现者：通过硬件（如中断机制）和操作系统调度实现。

11. 什么是系统调用？什么是特权指令？特权指令执行时，CPU 处于哪种工作状态？

系统调用：用户程序向操作系统请求服务的接口。
特权指令：仅允许在内核态执行的指令（如 I/O 操作、内存管理）。
执行状态：特权指令只能在内核态执行

12. 操作系统通常向用户提供哪几种类型的接口？其主要作用是什么？

接口类型：

1. **命令行接口（CLI）**：提供文本命令输入，适合高级用户。
2. **图形用户界面（GUI）**：通过图形化方式操作，用户体验友好。
3. **系统调用接口**：为程序员提供访问 OS 功能的编程接口。

作用：简化用户与系统的交互，为程序开发提供抽象的硬件访问。

第 2-3 章 进程管理

1. 程序顺序执行的特点是什么？

顺序性：程序按编写的顺序一条一条指令执行。

封闭性：程序的执行与外部无关，不受其他程序影响。

确定性：在相同输入条件下，程序执行结果是确定的。

2. 何谓进程？进程由哪些部分组成？试述进程的四大特性（动态性、独立性、并 发性、结构性）及进程和程序的区别。

进程定义：进程是一个正在运行的程序，是系统资源分配和调度的基本单位。

组成部分：

- **程序代码**：描述进程要执行的操作。
- **数据**：程序执行中需要的数据。
- **进程控制块（PCB）**：存储进程状态和管理信息。

四大特性：

- **动态性**：进程是程序的动态执行过程。
- **独立性**：进程有独立的地址空间和资源。
- **并发性**：多个进程可以在时间上交替执行。
- **结构性**：进程包含代码、数据、PCB 三部分。

区别：

- **程序**是静态的指令集合，而**进程**是动态的执行实体。
- 一个程序可以对应多个进程实例。

3. 进程控制块的作用是什么？它主要包括哪几部分内容？

作用：PCB 是操作系统管理和调度进程的核心数据结构，记录进程的所有信息。

主要内容：

1. **标识信息**：进程 ID、父进程 ID。
2. **状态信息**：当前状态、优先级、寄存器值。
3. **资源信息**：打开的文件、设备分配。
4. **控制信息**：进程队列指针、调度信息。

4. 进程的基本状态有哪些？试举出使进程状态发生变化的事件并描绘它的状态转换图。

基本状态：

1. **就绪**：等待 CPU 分配。
2. **运行**：正在占用 CPU 执行。
3. **阻塞**：等待某事件完成（如 I/O）。

状态转换事件：

- **就绪 → 运行**：被 CPU 调度。
- **运行 → 阻塞**：等待 I/O 完成或资源不可用。
- **运行 → 就绪**：时间片用完，被抢占。
- **阻塞 → 就绪**：等待事件完成。

状态转换图：

就绪 → 运行 → 阻塞 → 就绪

运行 ← 就绪

5. 什么是原语？什么是进程控制？

原语：由操作系统提供的用于实现进程控制的基本操作，它不可分割且不被中断（例如创建进程、终止进程）。

进程控制：对进程的创建、撤销、挂起、唤醒和状态切换等操作。

6. 试述进程调度的功能、方式、时机、算法；作业调度，交换调度；作业的周转 时间和作业的带权周转时间。

进程调度：

- **功能**：分配 CPU 资源，提高系统效率。
- **方式**：抢占式、非抢占式。
- **时机**：进程创建、结束、阻塞、时间片到期时。
- **算法**：先来先服务（FCFS）、短作业优先（SJF）、优先级调度、时间片轮转（RR）。

作业调度：决定作业何时进入内存。

交换调度：将不活跃的进程换出到外存。

周转时间：作业完成时间 - 提交时间。

7. 试述线程的定义，线程与进程的比较，系统对线程的支持（用户级线程、核心 级线程、两级组合）。

定义：线程是进程中的一个执行流，是 CPU 调度的基本单位。

比较：

方面	线程	进程
调度单位	线程	进程
资源分配	共享父进程资源	独立分配资源
开销	低（线程间切换）	高（进程间切换）

支持方式：

- **用户级线程**：线程管理在用户空间完成。
- **核心级线程**：线程管理由操作系统内核完成。
- **两级组合**：用户级和核心级线程结合。

8. 并发执行的进程在系统中通常表现为几种关系？各是在什么情况下发生的？

竞争关系：共享临界资源时（如文件）。

协作关系：进程间需要通信或同步（如管道通信）。

9. 什么叫临界资源？什么叫临界区？对临界区的使用应符合的四个准则（互斥使 用、让权等待、有空让进、有限等待）。

临界资源：在同一时间只允许一个进程访问的资源。

临界区：访问临界资源的代码片段。

四个准则：

1. **互斥使用**：一个时刻只有一个进程进入。
2. **让权等待**：无资源时主动放弃 CPU。
3. **有空让进**：空闲时允许进入。
4. **有限等待**：进入等待的进程不会无限期等待。

10. 试述解决进程之间互斥的办法（开、关中断，加锁、开锁（又叫测试与设置， 通常由一条机器指令完成），软件方法，信号量与 P、V 操作）。

- **硬件方法**：开/关中断，测试与设置指令（TSL）。
- **软件方法**：Dekker 算法、Peterson 算法。
- **信号量方法**：通过 P（wait）、V（signal）操作管理资源访问。

11. 若信号量 S 表示某一类资源，则对 S 执行 P、V 操作的直观含意是什么？当进 程对信号量 S 执行 P、V 操作时，S 的值发生变化，当 S>0、S=0、和 S<0 时， 其物理意义是什么?

- **P 操作**：请求资源（S--）。
- **V 操作**：释放资源（S++）。

物理意义：

- **S > 0**：资源可用数为 S。
- **S = 0**：无可用资源。
- **S < 0**：等待队列长度为|S|。

12. 在用 P/V 操作实现进程通信时，应根据什么原则对信号量赋初值？

资源数：信号量表示资源数量时，初值为资源个数。

同步需求：信号量表示同步需求时，初值根据通信顺序设置。

13. 掌握经典的 IPC 问题。

生产者-消费者问题。

读者-写者问题。

哲学家进餐问题。

14. 进程高级通信有哪些实现机制？

共享内存。

消息队列。

管道通信。

套接字通信。

15. 死锁产生的必要条件及解决死锁的方法。

必要条件：

1. **互斥**：资源不可共享。
2. **占有并等待**：持有资源时申请新资源。
3. **不可剥夺**：资源不可强制剥夺。
4. **循环等待**：形成资源依赖环。

解决方法：

- **预防**：破坏死锁条件。
- **避免**：银行家算法。
- **检测与恢复**：周期性检测，释放资源恢复系统。

16. 理解银行家算法的实质。能够利用银行家算法避免死锁。
实质：通过模拟资源分配，判断是否会进入不安全状态，避免死锁。
步骤：

1. 判断请求是否满足安全条件。
2. 分配资源并更新状态。
3. 若资源不足，则等待。

第 4 章 存储器管理

1. 存储器管理的功能与基本概念

功能：

1. **内存分配与回收**：为用户进程分配主存，释放已完成进程的内存空间。
2. **地址映射**：将逻辑地址转换为物理地址。
3. **存储保护**：防止进程非法访问内存。
4. **虚拟存储管理**：实现主存和辅存的联合使用。

基本概念：

- **名字空间**：系统内标识符的集合，用于描述程序中的变量和数据。
- **地址空间**：程序可用的逻辑地址范围。
- **存储空间**：物理内存的实际存储容量。
- **逻辑地址**：程序产生的地址，独立于物理内存。
- **物理地址**：内存单元的实际地址。

---

2. 地址重定位及其分类

定义：地址重定位是将逻辑地址转换为物理地址的过程。
分类：

1. **静态重定位**：在程序加载时完成，根据装载地址调整逻辑地址。
    - **优点**：无需运行时支持，开销低。
    - **缺点**：内存利用率低，不能适应动态需求。
2. **动态重定位**：在程序运行时完成，通过硬件地址转换实现。
    - **优点**：灵活，可支持动态内存分配。
    - **缺点**：需硬件支持，转换开销较高。

---

3. 内存划分与管理

- **用户空间**：为用户程序提供运行环境。
- **操作系统空间**：存储操作系统核心和关键数据。
- **管理目标**：提高用户空间的利用率。

---

4. 存储保护的目的及实现

目的：防止非法访问，保证系统和用户进程的稳定运行。
实现方式：

- **硬件支持**：基址寄存器和界限寄存器、分页和分段保护。
- **软件支持**：操作系统提供进程访问权限管理。

---

5. 可变式分区管理与存储保护

空闲区管理方法：

1. **首次适配**：从头扫描，找到第一个足够大的空闲区。
2. **最佳适配**：找到最接近需求大小的空闲区。
3. **最坏适配**：选择最大的空闲区。

保护方式：使用基址和界限寄存器。

---

6. 页式存储器的内零头与分区管理的碎片

- **内零头**：分配的内存单元未被完全利用的部分，与页大小成正比。
- **分区碎片**：可变分区管理中的内存碎片包括**内部碎片**和**外部碎片**。

---

7. 覆盖与交换的特点

- **覆盖**：将不常用的程序部分移出内存，根据需求加载。
- **交换**：将整个进程移入或移出内存，用于释放主存资源。

---

8. 页表的作用及地址转换过程

作用：记录每一页的物理地址映射关系。

地址转换过程：

1. 逻辑地址分为页号和页内偏移。
2. 查页表获取物理页号。
3. 结合页内偏移形成物理地址。

常用数据结构：页表、帧表、自由帧链表。

---

9. 段式与页式存储器管理的主要区别

方面	段式管理	页式管理
单位	段	页
大小	不固定	固定大小
目的	符合逻辑需求	提高内存利用率
碎片类型	外部碎片	内部碎片

---

10. 虚拟存储器及其容量问题

定义：虚拟存储器通过辅存与主存联合提供大于主存的存储空间。

容量：不能大于主存容量加辅存容量之和，但可通过页面置换提高效率。

---

11. 请求页式管理中的状态位、修改位、访问位

- **状态位**：表示页面是否在内存中。
- **修改位**：页面是否被修改过，决定是否需要回写到辅存。
- **访问位**：页面是否被访问，用于页面置换算法。

---

12. 缺页中断的处理流程

1. 判断地址是否非法。
2. 分配空闲帧或置换页面。
3. 将所需页面调入内存。

4. 更新页表和硬件状态。
5. 重新执行中断指令。

---

13. 页面置换算法及 Belady 异常
常见算法：
1. **FIFO**：先进先出，易产生 Belady 异常。
2. **LRU**：最近最少使用，性能较优。
3. **OPT**：最优置换，不现实但为理论参考。
4. **Clock**：时钟算法，利用访问位优化。

Belady 异常：页框数增加导致缺页率反而上升的现象。

---

14. 程序局部性原理与系统抖动
局部性原理：程序访问局部区域内的内存。
- **时间局部性**：重复访问最近使用的数据。
- **空间局部性**：访问邻近的内存地址。

系统抖动：频繁缺页导致 CPU 等待，性能下降。
工作集模型：通过动态调整工作集大小避免抖动。

---

15. 多级页表与写时复制技术
- **多级页表**：分层存储页表，减少内存开销，页表在需要时动态创建。
- **写时复制**：进程间共享页面，只有在写操作时才创建副本。

---

16. 页的共享与管理
共享机制：
- 共享代码或数据页。
- 专门数据结构记录共享关系（如页表和引用计数）。

第 5 章 文件系统

1. 文件与文件系统
文件：文件是操作系统用于存储数据的逻辑单位，具有命名、存储和访问功能。
文件系统：文件系统是操作系统中用于管理和存储文件的数据结构和相关功能模块。
主要功能：
1. 提供文件的**存储**、**检索**和**管理**功能。
2. 提供文件的**命名机制**和**目录结构**。
3. 实现文件的**存取控制**和**共享管理**。

UNIX 文件分类：
1. 普通文件
2. 目录文件
3. 设备文件（字符设备和块设备）
   好处：分类清晰，便于管理和访问，支持统一的文件接口。

---

2. 文件目录与文件控制块（FCB）
文件目录的作用：
1. 提供文件的**命名与索引**功能。

2. 支持快速定位文件的存储位置。

目录项内容:

1. 文件名
2. 文件属性（如类型、大小、创建时间等）
3. 文件的存储位置指针

文件控制块（FCB）:

文件系统中保存文件信息的数据结构，通常包括:

- 文件标识符
- 文件属性
- 存储位置指针
- 文件状态信息

---

3. 文件的逻辑结构与存取方法

逻辑结构:

1. **顺序文件**：数据按顺序组织，适合顺序访问。
2. **索引文件**：提供索引，支持快速查找。
3. **链接文件**：使用指针串联，节省存储空间。

存取方法:

1. **顺序存取**：按顺序访问文件内容。
2. **直接存取**：通过逻辑地址直接访问。
3. **索引存取**：通过索引表定位文件数据。

---

4. 文件的物理结构与管理

物理结构:

1. **连续结构**：文件存储在连续的物理块中，易管理但会产生外部碎片。
2. **链接结构**：通过指针连接文件块，节省空间但效率低。
3. **索引结构**：使用索引表记录块号，适合大文件存储。

管理方法:

- **连续结构**：需要分区或空闲表管理。
- **链接结构**：通过链表维护文件块。
- **索引结构**：维护索引表并进行快速查找。

---

5. DOS 文件卷与物理结构

文件卷结构:

- 引导扇区
- 文件分配表（FAT）
- 根目录区
- 文件数据区

文件物理结构:

- 使用 FAT 链表记录文件块的分配情况。

---

6. 记录的组块与分解

组块：将记录分组存储以提高磁盘访问效率。

分解：将大的存储块分解为记录，便于数据读取和处理。

---

7. 文件存储空间的管理方法

1. **连续分配**：文件占用连续的磁盘块。
    - **优点**：高访问效率。
    - **缺点**：容易产生碎片。
2. **链接分配**：通过指针连接文件块。
    - **优点**：节约空间。
    - **缺点**：访问速度较慢。
3. **索引分配**：使用索引表存储块号。
    - **优点**：适合随机存取。
    - **缺点**：需要额外存储索引表。

---

8. 多级目录与文件共享

多级目录的好处：

1. 提高文件组织性和查找效率。
2. 支持文件分层管理，减少冲突。

解决重名和共享：

1. 使用目录路径区分文件名。
2. 共享文件使用硬链接或符号链接。

---

9. 文件操作命令

常见命令及功能：

- **创建文件**：建立文件。
- **删除文件**：释放文件空间。
- **读写文件**：操作文件内容。
- **重命名文件**：更改文件名称。
- **打开/关闭文件**：打开文件加载 FCB，关闭释放资源。

---

10. 存取控制表（ACL）

概念：

ACL 是一种存取控制机制，为每个文件定义哪些用户或进程有特定权限（如读、写、执行）。

---

11. 内存映射文件（Memory Mapped File）

过程：

1. 将文件内容映射到内存地址空间。
2. 应用程序通过内存访问操作文件内容，无需 I/O 系统调用。
3. 优点：访问速度快，便于文件共享。

第 6 章 设备管理

1. I/O 设备分类

两大类:

1. **字符设备**：按字节或字符进行数据传输，如键盘、鼠标、打印机。
    - 特点：传输单位为字符，速度较慢，通常需要即时响应。
2. **块设备**：以数据块为单位传输，如磁盘、光盘。
    - 特点：传输单位为块，支持随机访问，速度较快。

---

2. 数据传输方式

四种常用传输方式:

1. **程序直接控制方式**：由 CPU 直接控制数据传输，适用于简单设备。
    - 优点：实现简单。
    - 缺点：占用 CPU 时间，效率低。
2. **中断驱动方式**：设备完成操作后发出中断信号，通知 CPU 处理。
    - 优点：减少 CPU 等待时间。
    - 缺点：适合较低速设备，复杂性增加。
3. **DMA 方式（直接存储器访问）**：设备直接与内存传输数据，无需 CPU 干预。
    - 优点：效率高，适合高速设备。
    - 缺点：需要额外的 DMA 控制器支持。
4. **通道控制方式**：利用专用 I/O 通道处理设备与内存的数据传输。
    - 优点：并行处理能力强，适用于高性能系统。
    - 缺点：硬件成本较高。

---

3. 设备的使用方式与虚拟设备

设备类型:

1. **独占设备**：一次只能供一个用户使用，如打印机。
2. **共享设备**：多个用户可同时使用，如磁盘。
3. **虚拟设备**：通过软件技术将独占设备转换为可供多个用户同时使用的设备。

虚拟设备的实现:

通过 **SPOOLING 技术**（同时外围设备操作与联机操作）。

---

4. I/O 软件的层次结构

I/O 软件分层:

1. **设备驱动程序**：直接与硬件交互，完成具体操作。
2. **设备独立性软件**：提供设备无关的接口和功能。
3. **用户级 I/O 库**：为用户程序提供系统调用接口，简化 I/O 操作。
4. **缓冲管理模块**：对数据进行缓冲，提高传输效率。

---

5. 设备独立性

定义:

设备独立性指用户程序不需要关心具体设备的类型或编号，只需操作逻辑设备，具体映射由操作系统完成。

实现:

通过设备驱动和设备独立性软件隔离设备的物理特性与用户逻辑操作。

---

6. SPOOLING 技术

概念:

SPOOLING（Simultaneous Peripheral Operation On-Line）是一种以空间换时间的技术，用于解决独占设备共享的问题。

实现原理（以输出为例）:

1. 将用户输出请求存入磁盘的**缓冲区**中。
2. 一个专用输出进程负责将缓冲区数据逐个输出到打印机。
3. 用户请求与设备操作分离，实现虚拟设备。

---

7. 磁盘信息编址方式

磁盘上的信息通过**盘面号**、**磁道号**、**扇区号**进行唯一标识:

- **盘面号**：指定磁盘的哪一面（双面磁盘）。
- **磁道号**：指定盘面上的哪一圈磁道。
- **扇区号**：指定磁道上的具体扇区。
  （等价于柱面号、磁头号和扇区号）

---

8. 磁盘数据传输时间

将磁盘上的一个数据块传输到主存涉及以下时间:

1. **寻道时间（Seek Time）**：磁头移动到目标磁道所需时间。
2. **旋转延迟时间（Rotational Latency）**：目标扇区旋转到磁头下所需时间。
3. **传输时间（Transfer Time）**：数据从磁盘传输到主存的时间。

---

9. 常用磁盘调度算法

1. **先来先服务（FCFS）**：按照请求到达的顺序服务。
   - 优点：公平，简单。
   - 缺点：可能引起长时间寻道。
2. **最短寻道时间优先（SSTF）**：优先处理与当前磁头位置最近的请求。
   - 优点：减少平均寻道时间。
   - 缺点：可能导致饥饿现象。
3. **扫描法（SCAN）**：磁头在磁盘上来回移动，按方向处理请求。
   - **C_SCAN**：只在一个方向处理请求，到边界后直接返回起点。
   - 优点：减少饥饿现象，提高响应一致性。
4. **LOOK 法**：只扫描有请求的磁道，不到达磁盘边界。
   - **C_LOOK**：只在一个方向服务请求后返回起点。

Belady 异常：在某些情况下增加磁盘块反而导致更多缺页。

EXTRA 牛逼系列：

第 7 章 Linux 进程管理

1. 进程控制块（PCB）

PCB（Process Control Block）是操作系统管理进程的核心数据结构。

与进程管理相关的字段：

- **进程标识符（PID）**：唯一标识进程。
- **父进程标识符（PPID）**：标识该进程的父进程。
- **进程状态**：如运行、就绪、阻塞等。
- **优先级**：用于调度进程。

与存储器管理相关的字段：

- **页目录表指针**：指向进程的页表，用于地址转换。
- **段表信息**：若使用段式管理，包含段表指针等信息。
- **虚拟地址空间信息**：记录进程的代码段、数据段和堆栈的地址范围。

与文件管理相关的字段：

- **文件描述符表**：记录进程打开的文件及其对应的文件描述符。
- **工作目录**：进程当前的工作目录路径。

线程组标识符（TGID）：

- Linux 中，线程共享同一个 TGID（通常等于主线程的 PID）。
- 用于区分线程组和独立进程。

---

2. 与进程创建有关的函数

1. **fork()**：
   - 创建一个子进程，子进程复制父进程的地址空间，但两者互相独立。
   - 返回值：
     - 父进程中返回子进程的 PID。
     - 子进程中返回 0。
2. **vfork()**：
   - 类似于 fork()，但在子进程执行 exec() 或 exit() 之前，父进程被挂起，子进程与父进程共享地址空间。
   - 适用于需要快速执行的进程创建操作，但需小心避免地址冲突。
3. **clone()**：
   - 用于创建线程或轻量级进程，可以指定共享的资源（如内存、文件描述符等）。
   - 常用标志：
     - CLONE_VM：共享地址空间。
     - CLONE_FILES：共享文件描述符表。
     - CLONE_THREAD：将子进程加入线程组。

---

3. 进程切换的过程

进程切换是指从一个进程切换到另一个进程的过程。涉及以下步骤：

1. **保存当前进程的上下文**：
   - 保存寄存器内容、程序计数器、状态寄存器等硬件上下文。

    - 保存进程的内核栈指针。
2. **切换页目录表**：
    - 更新 CR3 寄存器，指向新进程的页目录表。
3. **恢复新进程的上下文**：
    - 恢复寄存器内容、程序计数器。
    - 设置内核栈指针，切换到新进程的内核栈。

---

4. 进程调度

调度方式：

- **实时调度**：优先服务实时进程，调度算法如 FIFO、RR。
- **非实时调度**：基于动态优先级，调度普通进程。

调度时机：

- 发生中断（如定时器中断）。
- 进程主动放弃 CPU（如调用 sleep()）。
- 进程阻塞（如等待 I/O）。
- 新进程创建或进程退出时。

---

5. Linux 内核线程

0 号进程：

- **名称**：swapper 或 idle 进程。
- **作用**：
    - 系统启动时，作为内核的第一个进程。
    - 后续用于 CPU 空闲时执行，节省功耗。

1 号进程：

- **名称**：init 进程。
- **作用**：
    - 用户态的第一个进程，由 0 号进程通过 fork() 创建。
    - 负责初始化系统并启动所有用户进程（如服务进程和登录进程）。
    - 在现代 Linux 系统中，可能被 systemd 替代。

第 8 章 Linux 存储器管理

1. 进程地址空间的划分及管理

进程地址空间的划分：

- **用户空间**（User space）：存放用户程序及其数据。
    - 包括：代码段、数据段、堆、栈。
- **内核空间**（Kernel space）：存放操作系统内核及其数据。
    - 包括：内核代码、内核数据结构、内核栈等。

管理进程私有地址空间的数据结构：

- **虚拟内存区域（VMA, Virtual Memory Area）**：
    - VMA 代表了进程的连续虚拟内存区域，如堆区、栈区等。
    - Linux 中通过单链表或红黑树来管理这些虚拟内存区域。
    - 每个 VMA 结构包含了该区域的起始地址、结束地址、权限等信息。

指向映射文件对象的指针字段：

- **映射文件**：通过内存映射（如 mmap()）将文件内容加载到进程的虚拟地址空

间中。
    - **映射文件对象指针**：指向包含映射文件信息的数据结构，如文件描述符、文件偏移量等。

指向进程页目录表的指针字段：

- 进程的页目录表存储了虚拟地址到物理地址的映射信息，指向该页目录表的指针通常存储在进程的 PCB（进程控制块）中。

---

2. Linux 堆的管理：malloc() 和 free()

- **malloc()**：为进程的堆分配内存，返回内存块的指针。
- **free()**：释放由 malloc() 分配的内存。

在 Linux 中，堆的管理通常由 glibc 提供的内存分配器（如 ptmalloc）实现。它通过管理堆内存块链表来有效地分配和回收内存。

---

3. 管理物理内存页框的数据结构

内存管理区（Zone）结构：

- **Zone**：Linux 将物理内存划分为不同的管理区域，主要有以下几种：
    - **ZONE_DMA**：用于存储可直接访问的内存区域。
    - **ZONE_NORMAL**：正常的内存区域，常用于应用程序的内存分配。
    - **ZONE_HIGHMEM**：高端内存区域，供内核使用，通常是低地址区域之外的内存。

伙伴系统：

- Linux 使用**伙伴系统**来管理内存页框的分配与回收。
    - 内存页被按 2 的幂大小划分为多个块（大小为 2^k 页），每个块与一个“伙伴”合并形成更大的块。
    - **伙伴分配过程**：当需要分配内存时，系统检查是否有合适的空闲块；如果没有，合并小块形成大块，直到满足需求。

---

4. Slab 分配器的原理

- **Slab 分配器**：用于内核对象的内存分配，提供高效的内存管理。
    - 通过划分内存池为多个“slab”来优化内存的分配和回收，减少内存碎片。
    - 每个 slab 维护一组同类型的对象，内存分配时直接从 slab 中取出对象。
    - **作用**：提高分配速度，减少内存碎片，尤其适用于频繁分配和释放的内核对象。

---

5. 进程页表的建立及页目录表项

进程页表建立的时机：

- 在进程首次访问某个虚拟内存页时，操作系统会通过页面错误（Page Fault）来触发页表的创建和更新。

页目录表项和页表项所包含的字段：

- **页目录表项（Page Directory Entry）**：
    - 含有指向页表的物理地址，标志位（如存在位、读写权限等）。
- **页表项（Page Table Entry）**：

    - 含有实际物理页框的地址，标志位（如存在位、访问权限等）。

逻辑地址的划分与地址转换过程：

- **逻辑地址划分**：
    - 在两级页表系统中，逻辑地址通常由两部分组成：
        - **页目录索引**：用于索引页目录表项。
        - **页表索引**：用于索引页表项。
        - **页内偏移**：访问页内的具体位置。
- **地址转换过程**：
    - 通过页目录表（获取页表的物理地址）和页表项（获取物理页框地址）来实现从逻辑地址到物理地址的转换。

---

6. 请求调页

- **请求调页**：当进程访问未映射的虚拟内存页时，会发生页面缺失，操作系统会触发页面错误，加载所需的页到内存中。

缺页可能存放的地方：

- 页可能存放在**磁盘交换区**（Swap Space）或**文件系统的映射区域**，如果页未在物理内存中，操作系统会从这些地方加载。

---

7. 盘交换区空间的管理方法

- **交换空间**：用于存储被换出的内存页。通常由一个单独的交换分区或文件组成。
- **管理方法**：
    - 当内存不足时，操作系统会将不常用的内存页换出到磁盘的交换区。
    - 通过交换（Swap）管理虚拟内存，当进程需要访问交换出的页时，会触发缺页异常，从交换区将页重新加载到内存中。

第 9-10 章 Linux 文件系统

1. Ext2 文件卷的布局及各部分作用

Ext2 文件卷布局：

1. **引导块（Boot Block）**：
    - 位于卷的第一个块，用于存放引导程序。
2. **超级块（Superblock）**：
    - 描述文件系统的全局信息，如块大小、文件总数、空闲块数等。
3. **组描述符表（Group Descriptor Table）**：
    - 描述每个块组的元信息，如空闲块位图、空闲索引节点位图等。
4. **块组（Block Group）**：
    - 每个块组包括以下部分：
        - **块位图（Block Bitmap）**：记录块的使用情况。
        - **索引节点位图（Inode Bitmap）**：记录索引节点的使用情况。
        - **索引节点表（Inode Table）**：存储文件的索引节点信息。
        - **数据块（Data Blocks）**：存储文件内容。

---

2. 文件目录项的分割及好处

分为两部分：

1. **目录项（Directory Entry）**：存储文件名和指向索引节点的指针。
2. **索引节点（Inode）**：存储文件的元数据（如大小、权限）和文件的物理地址信息。

好处：

- 提高文件系统的灵活性，使得文件名和文件数据分离，可以轻松实现硬链接和符号链接。

---

3. 文件系统的索引节点及地址转换

索引节点（Inode）中的索引表划分：

1. **直接块地址**：直接指向文件数据块。
2. **一级间接块地址**：指向一个块，该块存储多个数据块的地址。
3. **二级间接块地址**：指向一个块，该块存储多个一级间接块的地址。
4. **三级间接块地址**：指向一个块，该块存储多个二级间接块的地址。

文件索引表的增长：

- 文件大小增加时，优先使用直接块地址，当直接块地址不足时，依次使用一级、二级、三级间接块地址。

字节地址到物理块的转换：

- 根据字节地址计算偏移量，确定该字节所在的块号，然后通过索引节点的索引表查找该块的物理地址。

---

4. 硬链接和符号链接的区别

1. **硬链接（Hard Link）**：
    - 多个文件名指向同一个索引节点。
    - 删除任意一个文件名，文件内容不受影响，只有所有硬链接被删除后，文件才被回收。
2. **符号链接（Symbolic Link）**：
    - 是一个特殊文件，存储目标文件的路径。
    - 删除目标文件后，符号链接会失效（悬空链接）。

---

5. 管理空闲存储空间的方法

- **位图法（Bitmap）**：
    - 通过位图标记每个块或索引节点的使用状态，0 表示空闲，1 表示已分配。
- **空闲块链表法**：
    - 使用链表链接所有空闲块，便于快速分配和回收。

---

6. VFS 通用文件模型中的四个主要对象

1. **超级块对象（Superblock）**：描述文件系统的全局信息。
2. **索引节点对象（Inode）**：描述文件的元数据和位置。
3. **目录项对象（Dentry）**：用于缓存目录项，提高文件路径解析效率。
4. **文件对象（File）**：用于描述打开的文件，包含文件偏移量、权限等。

---

7. 打开文件涉及的数据结构及关键字段

涉及的数据结构：

1. **文件对象（File）**：
    - 描述文件的打开状态，包含文件偏移量、访问模式等。
2. **目录项对象（Dentry）**：
    - 用于路径解析，指向文件的索引节点对象。
3. **索引节点对象（Inode）**：
    - 描述文件的元数据和数据块地址。
4. **超级块对象（Superblock）**：
    - 提供文件系统的信息和操作方法。

---

8. 文件使用与不使用时占用的系统资源

使用时：

- 文件对象（File）。
- 索引节点（Inode）。
- 缓冲区或缓存（用于数据读写）。

不使用时：

- 仅占用存储设备上的空间。

---

9. 安装表的作用

- **安装表（Mount Table）**：记录已挂载文件系统的信息，包括文件系统类型、挂载点、设备路径等。
- **作用**：
    - 管理挂载的文件系统，支持不同类型的文件系统同时使用。
    - 提供文件路径到设备的映射，提高系统操作的灵活性。

第 14 章 Windows 2000/XP 模型

1. Windows 的体系结构

- **分层结构**：
    - **用户模式（User Mode）**：包括用户应用程序和系统支持库（如 Win32 API）。
    - **内核模式（Kernel Mode）**：包括核心操作系统组件（如内核、硬件抽象层 HAL、驱动程序）。
- **模块化设计**：
    - 核心分为多个执行体组件（如进程管理器、内存管理器、I/O 管理器、对象管理器）和内核组件。

---

2. 硬件抽象层（HAL）的作用

- **功能**：
    - 抽象硬件细节，提供统一接口。
    - 隐藏硬件差异，使操作系统可移植。
- **作用**：
    - 提供对中断、DMA、计时器等硬件资源的管理。
    - 支持多处理器的均衡负载和中断分配。

---

3. Windows 系统组件的基本机制

1. **陷阱调度（Trap Dispatching）**：
   - 用于处理用户态和内核态之间的切换，例如系统调用或异常处理。
2. **对象管理器（Object Manager）**：
   - 管理系统中的所有对象，提供统一的接口。
   - 支持对象命名空间、引用计数和权限控制。
3. **同步机制**：
   - **自旋锁（Spinlock）**：用于短时间的自旋等待，避免进程切换开销。
   - **内核调度程序对象（Scheduler Objects）**：包括信号量、互斥体、事件等。
4. **本地过程调用（LPC）**：
   - 支持进程间的高效通信。
   - 用户模式和内核模式均支持。

---

4. 延迟过程调用（DPC）与异步过程调用（APC）

1. **延迟过程调用（DPC）**：
   - 在中断处理完成后执行低优先级任务。
   - 由内核安排，用于延迟执行非时间关键的任务。
2. **异步过程调用（APC）**：
   - 将用户定义的例程插入目标线程的执行队列。
   - 用于用户模式线程或内核模式线程的异步操作。

---

5. Windows 中的对象

- **两种类型的对象**：
  1. **执行体对象**：
     - 由执行体组件管理。
     - 例如：进程、线程、文件、内存段等。
  2. **内核对象**：
     - 由内核实现，用户态不可见。
     - 用于支持执行体对象的运行。
- **常见对象及作用**：
  - **进程和线程对象**：管理进程和线程的执行。
  - **文件对象**：表示文件或设备。
  - **信号量对象**：实现资源访问的同步。
  - **事件对象**：用于线程间的信号通信。
  - **互斥体对象**：提供对共享资源的互斥访问。
  - **可等待的定时器对象**：实现定时操作。

---

6. 同步和互斥机制

多处理机系统的机制：

1. **自旋锁（Spinlock）**：
   - 用于保护短时间的共享资源访问。
   - 多处理器环境中常用。
2. **调度程序对象**：

    - 包括：进程、线程、事件、信号量、互斥体、定时器等。
    - 每个同步对象有两种状态：有信号、无信号。

机制的优点：

- 提供细粒度锁，减少多处理器间的资源争用。
- 通过定时器和信号量实现资源的高效分配。

---

7. 线程等待同步对象的操作

1. **线程等待过程**：
    - 调用函数如 WaitForSingleObject 或 WaitForMultipleObjects。
    - 如果同步对象处于无信号状态，线程被挂起并放入等待队列。
2. **实现步骤**：
    - 线程尝试获取同步对象锁。
    - 如果失败：
        - 挂起线程。
        - 将线程加入对象的等待队列。
    - 如果成功：
        - 继续执行。
3. **相关 API**：
    - **WaitForSingleObject**：等待单个对象。
    - **WaitForMultipleObjects**：等待多个对象。

第 15 章 Windows 进程和线程管理

1. 管理进程和线程的数据结构

关键数据结构：

1. **执行体进程块（EPROCESS）**：
    - 用于管理进程的所有信息。
    - 包括：进程的基本信息、虚拟地址空间、句柄表、线程列表等。
2. **执行体线程块（ETHREAD）**：
    - 用于描述线程的基本信息。
    - 包括：线程上下文、线程队列信息、调度状态等。
3. **内核进程块（KPROCESS）**：
    - 提供进程的内核模式信息。
    - 包括：调度相关的数据、内核栈地址等。
4. **内核线程块（KTHREAD）**：
    - 描述线程的内核模式状态。
    - 包括：内核栈信息、当前线程上下文、线程优先级等。

---

2. 创建进程和线程

1. 创建进程：CreateProcess()

- **作用**：创建一个新进程及其主线程。
- **函数签名**：

c

复制代码

```
BOOL CreateProcess(
    LPCSTR lpApplicationName,
    LPSTR lpCommandLine,
    LPSECURITY_ATTRIBUTES lpProcessAttributes,
    LPSECURITY_ATTRIBUTES lpThreadAttributes,
    BOOL bInheritHandles,
    DWORD dwCreationFlags,
    LPVOID lpEnvironment,
    LPCSTR lpCurrentDirectory,
    LPSTARTUPINFO lpStartupInfo,
    LPPROCESS_INFORMATION lpProcessInformation
);
```

- **关键参数**:
  - lpApplicationName：指定可执行文件名。
  - lpCommandLine：命令行参数。
  - lpProcessInformation：返回新进程的句柄和线程信息。

2. 创建线程：CreateThread()

- **作用**：创建一个新线程并分配资源。
- **函数签名**:

c

复制代码

```
HANDLE CreateThread(
    LPSECURITY_ATTRIBUTES lpThreadAttributes,
    SIZE_T dwStackSize,
    LPTHREAD_START_ROUTINE lpStartAddress,
    LPVOID lpParameter,
    DWORD dwCreationFlags,
    LPDWORD lpThreadId
);
```

- **关键参数**:
  - lpStartAddress：线程的入口函数。
  - lpParameter：传递给线程函数的参数。
  - dwCreationFlags：控制线程的创建方式（如是否立即运行）。

---

3. 线程的 7 种状态及解释

1. **初始化（Initialized）**:
   - 线程刚创建但未运行。
2. **就绪（Ready）**:
   - 线程准备运行，等待调度器分配 CPU。
3. **运行（Running）**:
   - 线程正在使用 CPU 执行。
4. **等待（Waiting）**:
   - 线程因等待某事件（如 I/O 操作完成）而挂起。

5. **过渡（Transition）**：
    - 线程被调度运行，但其上下文尚未加载到处理器。
6. **终止（Terminated）**：
    - 线程已完成运行，处于终止状态。
7. **等待终止（Deferred Ready）**：
    - 等待操作完成后，重新进入就绪状态。

---

4. 线程调度

1. 调度算法：

- **基于优先级的抢先式多处理机调度**。
- 高优先级线程可以抢占低优先级线程。

2. 数据结构：

1. **32 个就绪线程队列**：
    - 每个优先级一个队列，范围为 0~31。
2. **线程就绪队列位图**：
    - 用 32 位的位图表示哪些优先级队列非空。
3. **处理机空闲位图**：
    - 用 32 位的位图表示哪些处理器处于空闲状态。

3. 调度过程：

1. 查找线程就绪队列位图中最高优先级的非空队列。
2. 从该优先级队列中选择下一个线程运行。
3. 如果有多个处理器空闲，使用处理机空闲位图分配线程。

---

5. 线程优先级的提升时机

线程优先级范围：

- 用户模式：1~15。
- 内核模式：16~31。

优先级提升时机：

1. **线程从等待状态转为就绪状态时**：
    - 如等待的 I/O 完成时。
    - 提升程度取决于 I/O 操作的类型。
2. **线程运行时间不足时**：
    - 系统会短暂提升优先级，防止线程因低优先级而饥饿。
3. **某些内核事件触发时**：
    - 如交互式任务处理或响应用户输入。

第 16 章 Windows 存储器管理

1. 数据结构：虚拟地址描述符 (VAD) 和区域对象

1. 虚拟地址描述符 (VAD)：

- 用于描述进程虚拟地址空间的一个连续区域。
- 每个 VAD 节点表示一个虚拟地址区间。
- **作用**：
    - 记录进程地址空间的分配状态（如保留的、提交的区域）。

    - 存储与虚拟地址相关的元数据（如访问权限、共享标记等）。

2. 区域对象：

- 描述内存映射文件的区域或共享内存。
- **作用**：
    - 用于支持文件映射和共享内存。
    - 关联文件对象或分页文件，以实现数据与物理内存的映射。

---

2. 虚拟内存区域

虚拟内存区域的三种状态：

1. **空闲（Free）**：
    - 地址空间未被分配，未绑定任何资源。
2. **保留（Reserved）**：
    - 地址空间已被预留，但尚未分配物理存储。
    - 通过 VirtualAlloc 函数可以保留地址空间。
3. **提交（Committed）**：
    - 地址空间已分配，并映射到物理存储（物理页或页面文件）。

---

3. 32 位逻辑地址与二级页表

1. 逻辑地址的划分：

- 32 位逻辑地址分为三个部分：
    1. **页目录索引**（10 位）：
        - 用于选择页目录中的一项。
    2. **页表索引**（10 位）：
        - 用于选择页表中的一项。
    3. **页内偏移**（12 位）：
        - 表示页内具体字节偏移。

2. 页目录表项和页表项的结构：

- 页目录表项和页表项具有相同的数据结构，包含以下字段：
    - **物理页框号（PFN）**：指向物理内存页。
    - **权限位**：表示读、写、执行权限。
    - **存在位**：表示该页是否驻留在物理内存中。
    - **修改位**：表示页是否被写入过。
    - **访问位**：表示页是否被访问过。

3. 页表的建立时机：

- 页表是在页面第一次被访问时动态创建的。
- 通过缺页中断机制完成。

4. 地址转换过程：

1. 使用逻辑地址的 **页目录索引** 查找页目录项。
2. 通过页目录项的内容找到对应页表。
3. 使用逻辑地址的 **页表索引** 查找页表项。
4. 通过页表项的内容找到物理页框，并加上 **页内偏移** 计算出物理地址。

---

4. 管理物理内存的数据结构：页框数据库

1. 页框数据库：

- Windows 使用页框数据库（PFN Database）来管理物理内存中的每个页框。
- 每个页框都有一个对应的记录，包含以下信息：
    - 页框状态。
    - 所属进程或文件。
    - 页框与虚拟页的映射关系。

2. 页框的 8 种状态：

1. **活动（Active）**：
    - 被进程或内核直接使用。
2. **转换（Transition）**：
    - 等待从磁盘加载或写回磁盘。
3. **备用（Standby）**：
    - 可立即分配给其他进程。
4. **更改（Modified）**：
    - 被修改但尚未写入磁盘。
5. **更改不写入（Modified No-Write）**：
    - 被修改但无需写入磁盘（如文件映射的非持久页）。
6. **空闲（Free）**：
    - 未使用，可立即分配。
7. **零初始化（Zeroed）**：
    - 已清零，可分配给需要干净页面的进程。
8. **坏（Bad）**：
    - 不可用的页框（如硬件故障）。

---

5. 原型页表项与共享页

1. 原型页表项：

- 描述文件映射页或共享页。
- **作用**：
    - 用于在多个进程间共享物理内存。
    - 通过映射同一个原型页表，实现数据的共享。

2. 区域对象的页表：

- 每个区域对象都有自己的页表，记录文件数据的物理内存映射。

3. 多进程共享页：

- 多个进程通过原型页表指向同一个物理页，实现共享。
- 常用于共享库或文件映射。

---

6. 页替换策略

Windows 使用 **改进的时钟算法**（Enhanced Clock Algorithm）作为其页替换策略。

策略特点：

1. **基于访问位和修改位**：
    - 优先淘汰长时间未被访问的页。
    - 避免频繁访问的页被替换。
2. **结合 LRU 思想**：
    - 页面使用越久越不可能被替换。
3. **双循环扫描**：

- 第一遍扫描：标记访问位为 0 的页面。
- 第二遍扫描：替换标记为 0 的页面。

第 17 章 Windows 文件系统

1. Windows 所支持的文件系统类型

Windows 操作系统支持以下文件系统类型：

1. **FAT (File Allocation Table)**：
   - **子类型**：FAT12、FAT16、FAT32。
   - 适用于小型存储设备（如 U 盘）。
2. **NTFS (New Technology File System)**：
   - 默认文件系统，支持大文件和复杂权限管理。
3. **exFAT (Extended FAT)**：
   - 为闪存设备优化的文件系统。
4. **ReFS (Resilient File System)**：
   - 专为高可用性和数据完整性设计，适用于服务器。
5. **其他**：
   - 支持通过第三方驱动访问的文件系统，如 ext3/ext4（Linux）。

---

2. 虚拟簇号和逻辑簇号的概念

1. **虚拟簇号 (VCN)**：
   - 虚拟簇号表示文件中的逻辑簇序号。
   - **作用**：标识文件数据在文件内的逻辑位置。
2. **逻辑簇号 (LCN)**：
   - 逻辑簇号表示文件在磁盘上的物理位置。
   - **作用**：标识文件存储的磁盘簇地址。

关系：

文件系统通过文件分配表（如 MFT 中的映射信息）将虚拟簇号 (VCN) 转换为逻辑簇号 (LCN)，完成文件逻辑地址到物理地址的映射。

---

3. NTFS 卷的结构

NTFS 文件系统的卷由以下部分组成：

1. **引导扇区 (Boot Sector)**：
   - 包含引导代码和文件系统信息。
   - 位于卷的第一个扇区。
2. **主控文件表 (Master File Table, MFT)**：
   - **作用**：
     - 存储文件和目录的元数据。
     - 每个文件或目录占用一个或多个记录。
   - **结构**：
     - MFT 中的每个条目记录文件的基本信息（如文件名、大小、权限）和数据存储位置。
3. **日志文件 (Log File)**：
   - 用于记录文件系统的事务操作，确保系统崩溃后的数据一致性。

4. **位图 (Bitmap)**:
    - 标识磁盘簇的使用情况。
5. **卷描述符文件 (Volume Descriptor)**:
    - 存储卷的基本信息（如卷名、大小）。
6. **根目录 (Root Directory)**:
    - NTFS 文件系统的目录结构从根目录开始。

---

4. NTFS 文件的物理结构：索引顺序结构

NTFS 文件的物理存储特点:

1. **数据存储为属性**:
    - 文件的所有信息（包括数据）存储为文件属性。
    - 常见属性:
        - 标准信息（大小、时间戳）。
        - 文件名。
        - 数据流。
2. **小文件的存储**:
    - 如果文件小于 1 KB，其数据存储在 MFT 条目中（称为“驻留文件”）。
3. **大文件的存储**:
    - 如果文件数据较大，MFT 条目中存储文件数据的指针，指向数据实际存储的位置（非驻留文件）。
4. **索引顺序结构**:
    - 数据按照索引顺序分布在磁盘上，便于快速访问和检索。

---

5. NTFS 目录结构：B+树管理

NTFS 使用 **B+树** 数据结构管理目录:

1. **特点**:
    - 每个节点存储多个索引，减少磁盘 I/O。
    - 叶节点存储文件名与 MFT 记录号的映射关系。
    - 支持高效的插入、删除和检索操作。
2. **优点**:
    - **快速查找**：文件名查找时间复杂度为 O(log⁡n)O(\log n)O(logn)。
    - **动态扩展**：能够高效地管理大量文件。