

# 操作系统

Created By LYP

2025年 12 月 10 日

真正危险的不是计算机开始像人那样去思考  
而是人类开始  
像计算机一样去思考  
——西德尼·哈里德

# Contents

<b>1</b>	<b>第一章 操作系统引论</b>	<b>4</b>
1.1	操作系统的目标: . . . . .	4
1.2	操作系统的作用: . . . . .	4
1.3	操作系统的发展过程: . . . . .	4
1.4	简答题-操作系统的基本特性: . . . . .	4
1.5	操作系统的主要功能: . . . . .	4
<b>2</b>	<b>第二章 进程的描述与控制</b>	<b>5</b>
2.1	进程 . . . . .	5
2.2	PCB(进程控制块) . . . . .	6
2.3	引起进程创建的事件: . . . . .	7
2.4	引起进程终止的事件: . . . . .	7
2.5	引起进程阻塞和唤醒的事件: . . . . .	7
2.6	进程同步 . . . . .	7
2.7	管程 . . . . .	7
2.8	进程通信 . . . . .	8
2.9	线程 . . . . .	8
2.10	理解P、V操作 . . . . .	8
<b>3</b>	<b>第三章 处理机调度与死锁</b>	<b>9</b>
3.1	死锁 . . . . .	9
3.2	处理机调度 . . . . .	9
<b>4</b>	<b>第四章 存储器管理</b>	<b>13</b>
4.1	多级存储器结构 . . . . .	13
4.2	程序的装入和链接 . . . . .	13
4.3	连续分配存储管理方式 . . . . .	13
4.4	对换 . . . . .	14
4.5	离散分配方式 . . . . .	14
4.6	分页存储管理方式 . . . . .	14
4.7	二级页表 . . . . .	14
4.8	分段存储管理 . . . . .	14
4.9	简答题-分页和分段的联系与区别: . . . . .	15
4.10	信息共享 . . . . .	15
4.11	段页式存储管理方式 . . . . .	15
<b>5</b>	<b>第五章 虚拟存储器</b>	<b>16</b>
5.1	虚拟存储器概述 . . . . .	16
5.2	请求分页中的硬件支持 . . . . .	17

5.3	请求分页中的内存分配 . . . . .	17
5.4	页面调入策略 . . . . .	17
5.5	页面置换算法(具体方法见大题) . . . . .	18
5.6	”抖动”和工作集 . . . . .	18
5.7	请求分段存储管理方式 . . . . .	18
<b>6</b>	<b>第六章 输入输出系统</b>	<b>19</b>
6.1	I/O系统 . . . . .	19
6.2	I/O设备 . . . . .	20
6.3	设备控制器 . . . . .	20
6.4	I/O通道 . . . . .	20
6.5	中断机构和中断处理程序 . . . . .	21
6.6	设备驱动程序 . . . . .	21
6.7	设备分配 . . . . .	21
6.8	假脱机(Spooling)系统 . . . . .	22
6.9	缓冲区管理 . . . . .	22
6.10	磁盘存储器的性能和调度 . . . . .	22
<b>7</b>	<b>第七章 文件管理</b>	<b>24</b>
7.1	文件和文件系统 . . . . .	24
7.2	文件的逻辑结构 . . . . .	24
7.3	文件目录 . . . . .	24
7.4	文件共享 . . . . .	24
<b>8</b>	<b>第八章 磁盘存储器的管理</b>	<b>26</b>
8.1	外存的组织方式 . . . . .	26
8.2	FAT技术 . . . . .	26
8.3	文件存储空间管理方法 . . . . .	26
8.4	提高对文件的访问速度的途径 . . . . .	26
8.5	提高磁盘I/O速度的途径 . . . . .	27
8.6	引入检查点的目的: . . . . .	27

# 1 第一章 操作系统引论

## 1.1 操作系统的目标:

方便性、有效性、可扩充性和开放性

## 1.2 操作系统的作用:

- (1) OS作为用户与计算机硬件系统之间的接口。
- (2) OS作为计算机系统资源的管理者。
- (3) OS实现了对计算机资源的抽象。

## 1.3 操作系统的发展过程:

无操作系统的计算机系统 → 单道批处理系统 → 多道批处理系统 → 分时系统 → 实时系统

## 1.4 简答题—操作系统的基本特性:

- (1) 并发性: 多个进程能交替进行
- (2) 共享性: 系统资源可被多个进程共同使用
- (3) 虚拟性: 利用时分与空分技术将物理资源转换为逻辑资源
- (4) 异步性: 进程的执行速度不可预测, 但操作系统保证其在有限时间内向前推进

并发和共享是OS最基本的特征

并发和共享互为存在的条件

虚拟技术以并发和资源共享为前提

异步性是并发和共享的必然结果

## 1.5 操作系统的主要功能:

- (1) 处理机管理功能: 进程控制、进程同步、进程通信、调度
- (2) 存储器管理功能: 内存分配、内存保护、地址映射、内存扩充
- (3) 设备管理功能: 缓冲管理、设备分配、设备处理
- (4) 文件管理功能: 文件存储空间管理、目录管理、文件读/写管理和保护
- (5) 操作系统与用户的接口: 用户接口、程序接口

## 2 第二章 进程的描述与控制

### 2.1 进程

①[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

**2024-进程的定义：**

进程是程序的一次执行。

进程是一个程序及其数据在处理机上的顺序执行时所放生的活动。

进程是具有独立功能的程序在一个数据集合上运行的过程，是系统进行资源分配和调度的一个基本单位。

②**引入进程的目的：**更好描述和控制程序的并发执行，实现OS的并发性和共享性，能极大提高系统资源的利用率，增加系统的吞吐量。

③**进程的组成：程序、数据和PCB**

程序可以为其他进程共享

PCB是进程存在的唯一标志

④**2024-简答-进程的特征/特点：**

- (1) **动态性：**进程是程序的一次执行过程，具有创建、执行、暂停、终止等动态变化，是动态产生、动态消亡的。
- (2) **并发性：**多个进程可以在同一时间段内交替执行，实现系统的并发活动。
- (3) **独立性：**进程是资源分配和调度的基本单位，拥有独立的地址空间，不同进程之间相互独立。
- (4) **异步性：**进程的执行速度不可预知，受调度、资源等影响呈不可再现的异步运行。
- (5) **(拓展)结构性：**进程由程序段、数据段和 PCB 组成，具有一定的结构。

⑤**简答题-进程与线程的比较：**

- (1) **调度：**线程是独立调度的基本单位，进程是拥有资源的基本单位。
- (2) **拥有资源：**进程是拥有资源的基本单位，线程不拥有系统资源。
- (3) **系统开销：**进程在创建或撤销进程时系统需分配或回收资源，开销较大；线程开销小。
- (4) **并发性：**进程可以并发执行，多个线程之间也可以并发执行，提升了系统的并发性和吞吐量。
- (5) **地址空间和资源：**进程间的地址空间独立，线程共享同一进程的资源；一个进程内的线程对其他进程不可见。
- (6) **通信：**进程间通信需要同步和异步手段的支持，而线程间可以直接通过读写全局变量进行通信。

⑥**2024-简答题-进程与程序的联系与差异：**

- (1) 进程是程序、数据和PCB的组合，程序是一组有序的指令集合，是静态的概念。

- (2) 进程是程序的一次执行过程，具有生命周期，而程序是静态存在的，可以长期保存。
- (3) 一个进程可以执行多个程序，一个程序也可以启动多个进程。进程可以创建新进程，而程序不可以创建新程序。

### ⑦分时系统与实时系统的比较

- (1) **及时性**：分时系统没有严格的时间上限要求，实时系统对任务有严格的时间约束。
- (2) **交互性**：分时系统面向多用户交互，响应时间短。实时系统主要执行任务而非交互。
- (3) **可靠性**：分时系统要求系统可靠，实时系统要求系统高度可靠。

Q：可靠性→实时系统更强；交互性→分时系统更强。

### ⑧进程的三种状态：

- (1) **就绪状态**：进程已具备了运行的条件，但在等待CPU
- (2) **执行状态**：进程占用CPU正在执行。
- (3) **阻塞状态**：进程等待事件(如I/O资源)而暂时不能继续执行。

状态转换图：

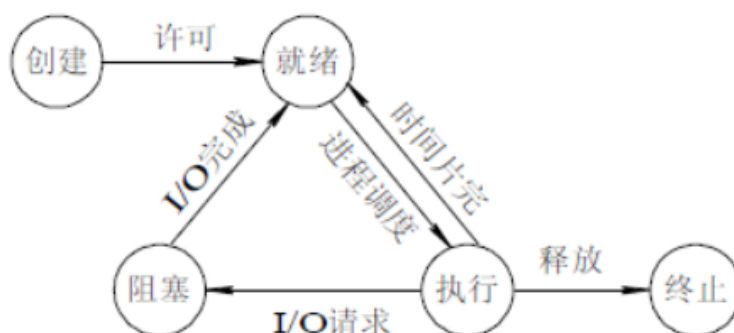


图 2-7 进程的五种基本状态及转换

三个基本状态之间转换的典型原因：

- (1) **就绪→执行**：获得了CPU使用权
- (2) **执行→就绪**：时间片用完、被更高优先级进程抢占、OS进行进程调度需要切换
- (3) **执行→阻塞**：等待I/O完成、等待资源、等待某个同步事件或信号量、执行sleep等系统的调用
- (4) **阻塞→就绪**：I/O完成、所需资源被释放、获得信号量、同步条件满足

## 2.2 PCB(进程控制块)

### ①PCB的作用：

- (1) 作为独立运行基本单位的标志

- (2) 能实现间断性运行方式
- (3) 提供进程管理所需要的信息
- (4) 提供进程调度所需要的信息
- (5) 实现与其他进程的同步与通信

②为什么说PCB是进程存在的唯一标志:

PCB是系统感知和管理进程的唯一实体和数据结构。

进程的创建以分配PCB为标志。

进程的执行表现为操作系统对其PCB状态的不断更新与操作。

进程的终止以回收PCB为标志。

③PCB中的信息: 进程标识符、处理机状态、进程调度信息和进程控制信息。

## 2.3 引起进程创建的事件:

用户登录、作业调度、提供服务和应用请求。

## 2.4 引起进程终止的事件:

正常结束、异常结束和外界干预。

## 2.5 引起进程阻塞和唤醒的事件:

向系统请求共享资源失败、等待某种操作的完成、新数据未到达、等待新任务的到达。

## 2.6 进程同步

①同步机制遵循的规则: 空闲让进、忙则等待、有限等待、让权等待。

②两种形式的制约关系:

- (1) 间接相互制约关系: 相互制约-互斥访问
- (2) 直接相互制约关系: 相互合作-同步访问

③临界资源: 是指在同一时间内只能被一个进程或线程独占使用的资源。

采用互斥方式实现对这种资源的共享。

④临界区: 每个进程中访问临界资源的那段代码为临界区。

## 2.7 管程

代表共享资源的数据结构以及由对该共享数据结构实施操作的一组过程所组成的资源管理程序共同构成了一个操作系统的资源管理模块。

## 2.8 进程通信

①进程通信的类型:

- (1) 共享存储器系统
- (2) 管道通信系统
- (3) 消息传递系统
- (4) 客户机-服务器系统

②消息传递通信的实现方式:

- (1) 直接消息传递系统
- (2) 信箱通信(间接通信方式)

直接消息传递系统实例:

为实现消息缓冲通信,在PCB应该添加 消息队列首指针mq、消息队列互斥信息量mutex;  
消息队列资源信号量Sm

## 2.9 线程

进程-资源分配和调度的独立单位

线程-调度和分派的基本单位

线程的引入: 为了减少程序在并行执行时所付出的时空开销,是OS具有更好的并发性。

## 2.10 理解P、V操作



### 3 第三章 处理机调度与死锁

#### 3.1 死锁

①死锁产生的原因：

- (1) 竞争不可抢占性资源
- (2) 竞争可消耗资源
- (3) 进程推进顺序不当

②死锁的定义：

如果一组进程中的每一个进程都在等待仅由该组进程中的其他进程才能引发的事件—该组进程是死锁的

③死锁产生的必要条件：

- (1) 互斥条件
- (2) 请求和保持条件
- (3) 不可抢占条件
- (4) 循环等待条件

④处理死锁的方法：

- (1) 预防死锁：破坏”请求和保持”条件、破坏”不可抢占”条件、破坏”循环等待”条件
- (2) 避免死锁：银行家算法(大题)
- (3) 检测死锁
- (4) 解除死锁：抢占资源、终止/撤销进程

死锁避免：

一次分配法破坏了死锁中的请求和保持条件；有序分配法破坏了死锁的循环等待条件

死锁定理：当且仅当某状态的资源分配图是不可完全简化的

#### 3.2 处理机调度

①处理机调度层次：

- (1) 高级调度：长程调度、作业调度
- (2) 低级调度：进程调度、短程调度
- (3) 中级调度：内存调度：提高内存利用率和系统吞吐量

调度类型	高级调度	中级调度	低级调度
批处理系统	√	√	√
分时系统		√	√
实时系统			√

②进程调度的任务:

- (1) 保存处理机的现场信息
- (2) 按照某种算法选取进程
- (3) 把处理器分配给进程

③进程调度机制的三个基本部分: 排队器、分排器、上下文切换器

④进程调度的方式:

- (1) 非抢占方式: 只允许进程主动放弃处理机。
- (2) 抢占式方式: 允许调度程序根据某种原则, 去暂停某个正在执行的进程, 将已分配给该进程的处理机重新分配给另一个进程。

抢占原则: 优先权原则、短进程优先原则、时间片原则

⑤引起进程调度的原因:

非抢占:

- (1) 正在执行的进程运行完毕, 或因发生某事件而使其无法再继续运行
- (2) 正在执行中的进程因提出I/O请求而暂停执行
- (3) 在进程通信或同步过程中, 执行了某种原语操作(PV原语、Block原语)

抢占:

- (1) 更高优先级的进程就绪
- (2) 时间片耗尽运行进程被剥夺处理机

⑥调度算法

- (1) 先来先服务调度算法 (FCFS): 按作业或进程到达的先后次序进行处理, 是一种非抢占式调度算法。实现简单, 但对长作业有利, 对短作业不利, 平均等待时间较长。
- (2) 短作业优先调度算法 (SJF): 优先选择运行时间最短的作业或进程。能使平均等待时间最小, 但需要估计运行时间, 对长作业不利。
- (3) 优先级调度算法: 按优先级大小调度作业或进程, 优先级高者优先运行, 可采用抢占式或非抢占式。缺点是可能产生“饥饿”现象。
- (4) 高响应比优先调度算法 (HRRN): 高响应比优先调度算法 (HRRN) 通过计算响应比来确定调度顺序:

$$\text{响应比 } R = \frac{W+S}{S}$$

其中:  $W$ : 等待时间  $S$ : 要求服务时间

响应比最高的作业优先运行。该算法兼顾长作业和短作业, 避免了 SJF 中长作业长期等待的问题。

**Q:** 当各个作业的等待时间相同时, 要求服务时间短的优先调度; 当各个作业要求运行的时间相同时, 等待时间长的作业将优先调度。(According to 公式)

- (5) **时间片轮转调度算法 (RR)**：所有进程按固定大小的时间片轮流运行，适用于分时系统，响应快；时间片过小会增加切换开销。
- (6) **多级反馈队列调度算法 (MFQ)**：设多个优先级不同的就绪队列，时间片逐级增大。新进程先进入高优先级队列，未完成则转入下一队列。无需估计运行时间，是现代操作系统常用算法。

#### 面向用户的调度准则选择：

- (1) 实时系统→截止时间的保证
- (2) 分时系统→响应时间块
- (3) 批处理系统→平均周转时间短

#### 调度算法的选择(理解各个调度算法)：

- (1) 照顾短作业用户→短作业优先
- (2) 使作业的平均周转时间最短→短作业优先
- (3) 照顾紧急作业的用户→基于优先权的剥夺调度算法
- (4) 实现人机交互作用→时间片轮转法
- (5) 兼顾短作业和长时间等待的作业→高响应比优先调度算法
- (6) 使短作业、长作业以及交互作业用户都比较满意→多级反馈队列调度算法

#### 2024—简答—选择调度和算法的准则

- (1) **提高CPU利用率**，尽可能减少CPU空闲时间。
- (2) **增加系统吞吐量**，单位时间内完成更多作业。
- (3) **缩短作业周转时间**，提高用户体验。
- (4) **减少等待时间**，使进程在就绪队列中停留时间更短。
- (5) **缩短响应时间**，特别适用于交互式系统。
- (6) **满足实时性要求**，确保实时任务在截止日期前完成。

#### ⑦2025-期中-简答—多级反馈队列的调度机制：

- (1) 设置多个就绪队列
- (2) 每个队列都采用FCFS算法
- (3) 按照队列优先级调度

#### ⑧为什么说多级反馈队列能满足各种类型用户的需要？

- (1) **终端型用户**：终端型用户多属于较小的交互型作业，能在第一队列的时间片内完成。
- (2) **短批处理作业用户**：短作业可以在前几个队列中完成，周转时间短。

- (3) **长批处理作业用户：**长作业将依次在第1,2...n个队列中运行，然后再按照轮转方式运行，用户不用担心作业得不到处理。

⑧操作系统的调度算法分类：

- (1) **批处理系统：**先来先服务、短作业优先、高响应优先调度算法
- (2) **分时系统：**时间片轮转算法、多级反馈队列调度算法、非抢占/弱抢占优先级调度算法
- (3) **实时系统：**最早截止时间优先算法、最低松弛度优先算法、抢占式优先级调度算法

## 4 第四章 存储器管理

### 4.1 多级存储器结构

- (1) CPU存储器：寄存器
- (2) 主存：高速缓存、主存、磁盘缓存
- (3) 辅存：磁盘、可移动存储介质

### 4.2 程序的装入和链接

用户程序要在系统中运行，必须先装入内存，然后再将其转变为一个可以执行的程序：

编译→链接→装入

程序的装入：

- (1) 绝对装入方式：  
单道程序环境中使用  
绝对地址=物理地址：实际内存地址  
相对地址=逻辑地址：进程空间地址  
程序中所使用的绝对地址，既可以在编译或者汇编时给出，也可以由程序员直接赋予
- (2) 可重定位装入方式：  
多道程序环境中使用
- (3) 动态运行时的装入方式：

程序的链接：静态链接方式、装入时动态链接、运行时动态链接

### 4.3 连续分配存储管理方式

#### ①连续分配方式

- (1) 单一连续分配
- (2) 固定分区分配
- (3) 动态分区分配

#### ②动态分区分配算法

- (1) 基于顺序搜索：首次适应算法、循环首次适应算法、最佳适应算法、最坏适应算法
- (2) 基于索引搜索：快速适应算法、伙伴系统、哈希算法

#### ③动态可重定位分区分配

拼接/紧凑：通过移动内存中作业的位置，把原来多个分散的小分区拼接成一个大分区的方法。

## 4.4 对换

把内存中暂时不能运行的进程或者暂时不用的程序和数据调出到外存上，以便腾出足够的内存空间，再把已具备运行条件的进程或进程所需要的程序和数据调入内存。

对换是提高内存利用率的有效措施

## 4.5 离散分配方式

分类：

- (1) 分页存储管理方式
- (2) 分段存储管理方式
- (3) 段页式存储管理方式

## 4.6 分页存储管理方式

①页面和物理块：

将一个进程的逻辑地址空间分成若干大小相等的片，称为页面或页  
(页面大小一般为2的幂，通常是512B-8KB)

将物理内存空间分成与页面相同大小的若干存储块，称为物理块或页框

②地址结构：

前一部分是页号，后一部分是位移量(页内地址)

③页表：

能在内存中找到每个页面所对应的物理块，系统又为每个进程建立了一张页面映像表，称为页表

作用：实现从页号到物理块号的地址映射

③地址变换机构 分为基本的地址变换机构和具有快表的地址变换机构

由于页表是存放在内存中，这使CPU在每存取一个数据时，都要访问两次内存。

第一次形成物理地址，第二次获取数据

为了提高地址变换速度

在地址变换机构中增设 具有并行查寻能力的特殊高速缓冲寄存器→联想寄存器or快表(TLB转换查找缓冲器)：用于保存当前访问的段表项。

## 4.7 二级页表

利用外层页表和内层页表这两级页表，实现从进程的逻辑地址到内存中物理地址的转换。

## 4.8 分段存储管理

①分段：

分段地址中的地址由段号和段内地址组成。在该地址中，允许一个作业最长有64K个段，每个段最大长度是64KB。

②段表：

为找出每个逻辑段所对应的物理地址，在系统为每个进程建立了一张段映射表→段表  
每个段在表中占有一个表项，其中记录了该段在内存中的**起始地址(基址)**和**段的长度**

#### 4.9 简答题—分页和分段的联系与区别：

联系：

- (1) 二者都采用离散分配方式
- (2) 二者都要通过地址映射机构来实现地址变换

区别：

- (1) **页是信息的物理单位**，分页是为实现离散分配方式，提高内存利用率。分页仅仅是由于系统管理的需要而不是用户的需要。  
**段则是信息的逻辑单位**，分段的目的是为了能更好地满足用户的需要。
- (2) **页的大小固定且由系统决定**。  
**段的长度不固定**，决定于用户所编写的程序，通常由编译程序在对源程序进行编译时，根据信息的性质来划分。
- (3) **分页的用户程序地址空间是唯一的**。即单一的线性地址空间，程序员只需利用一个记忆符，即可表示一个地址。  
**分段的作业地址空间是二维的**。程序员在标识一个地址时，既需要给出段名，也需要给出段内地址。

#### 4.10 信息共享

①对程序和数据共享的方便性：分段系统>分页系统

②可重入代码(纯代码)：

- (1) 允许多个进程同时访问的代码。
- (2) 为使各个进程所执行的代码完全相同，不允许任何进程对它进行修改。

#### 4.11 段页式存储管理方式

①作用：

分页系统以页面作为内存分配的基本单位，能够有效地提高内存的利用率。

分段系统作为内存分配的基本单位，能够更好满足用户多方面的需要

**段页式存储管理方式**能像分页系统很好地解决内存外部碎片化问题，又能像分段系统一样具有便于实现、分段可共享、易于保护、可动态链接等一系列优点。

②**段页式地址**由段号、段内页号、页内地址三部分组成。

③**基本原理**：

先将用户程序分为若干个段，然后把每个段分为若干的页，并为每个段赋予一个段名。

## 5 第五章 虚拟存储器

### 5.1 虚拟存储器概述

①**虚拟存储器**是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。

②**虚拟存储器的特征**：

- (1) **多次性(最重要)**：一个作业中的程序和数据允许被分成多次调入内存运行。
- (2) **对换性**：一个作业中的程序和数据，无需在作业运行时一直常驻内存，而是允许在作业的运行过程中进行换进和换出。
- (3) **虚拟性**：能够从逻辑上扩充内存容量，使用户所看到的内存容量远大于实际内存容量。

虚拟性是以多次性和对换性为基础(仅当系统允许将作业分多次调入内存，并能将内存中暂时不运行的程序和数据换至盘上，才能实现虚拟存储器。)

多次性和对换性必须建立在离散分配的基础上。

③**2024-简答题-虚拟存储器的原理**

- (1) **程序具有局部性原理**：程序在时间和空间上倾向于访问少量集中区域，因此只需将当前需要的部分页面装入内存即可保证程序运行。
- (2) **采用按需调页实现部分装入**：程序无需全部装入内存，系统根据需要动态调入页面，不常用的页面则保留在外存，从而显著减少内存占用。
- (3) **依靠地址映射机制完成虚拟地址到物理地址的转换**：CPU 产生的虚拟地址通过 MMU、页表（或段表）转换为物理地址，使程序看到的地址空间连续而实际物理内存可以离散。
- (4) **通过缺页中断与页面置换机制自动管理页面调入与调出**：当访问的页面不在内存时产生缺页中断，由系统从外存调入；当内存不足时利用页面置换算法换出部分页面，从而支持更大的虚拟空间与更高的系统并发度。

④**虚拟存储器的实现方法**

- (1) **请求分页存储管理**：将程序按固定大小划分为页面，系统只在页面被访问时按需调入内存；通过页表完成地址转换，发生缺页时由操作系统调入页面，内存不足时使用页面置换算法进行换出，是最常用的实现方式。
- (2) **请求分段存储管理**：按照程序的逻辑结构（如代码段、数据段、栈段）划分为大小不等的段，仅在需要时装入段；利用段表记录段的基址、长度和权限，支持逻辑保护与共享，但可能产生外部碎片。
- (3) **段页式存储管理**：先按逻辑将程序分段，再将每个段进一步分页；地址由段号、段内页号和页内偏移构成，通过段表定位页表，再由页表定位物理页框。该方式既保留分段的逻辑结构，又避免外部碎片，是实际系统中最常使用的综合方案。



## 5.2 请求分页中的硬件支持

①**缺页中断机构**：在请求分页系统中，每当所要访问的页面不在内存时，便产生一缺页中断，请求OS将所缺之页调入内存。

②**缺页中断与一般中断的区别**：

- (1) **在指令执行期间产生和处理中断信号**：缺页中断是在指令执行期间，发现要访问的指令或数据不在内存时产生和处理的。
- (2) **一条指令在执行期间可能产生多次缺页中断**。

## 5.3 请求分页中的内存分配

①**最小物理块**：指能保证进程正常运行所需要的最小物理块，当系统为进程分配的物理块小于此值时，进程将无法运行。

②**内存分配策略**：

- (1) **固定分配局部置换**
- (2) **可变分配全局置换**
- (3) **可变分配局部置换**

③**物理块分配策略**：

- (1) **平均分配算法**
- (2) **按比例分配算法**
- (3) **考虑优先权的分配算法**

## 5.4 页面调入策略

①**何时调入页面**：

- (1) **预调页策略**
- (2) **请求调页策略**

②**从何处调入页面**：

- (1) **系统拥有足够的对换区空间**：这时可以全部从对换区调入所需页面，以提高调页速度。
- (2) **系统缺少足够的对换区空间**：这时凡是不会被修改的文件，都直接从文件区调入。
- (3) **UNIX方式**：UNIX 系统采用文件区与对换区结合的策略：程序初次装入时从文件区调页，运行过程中若页面被换出则写入对换区；再次缺页时优先从对换区调入，以兼顾效率与空间利用。

### ③页面调入过程:

页面调入时，首先发生缺页中断，操作系统检查页面是否合法，分配或置换页框；若置换出的页面已修改则写回外存；之后从对换区或文件区读入所需页面，更新页表，最后恢复进程继续执行。

### ④影响缺页率的因素:

- (1) 页面大小：页面划分较大，缺页率较低。
- (2) 进程所分配的物理块的数目：所分配物理块数目越多，缺页率越低。
- (3) 页面置换算法：缺页率是衡量页面置换算法的重要指标。
- (4) 程序固有特性：程序编制的局部性越高，相应执行时的缺页程度越低。

## 5.5 页面置换算法(具体方法见大题)

- (1) 最佳置换算法
- (2) 先进先出置换算法(FIFO)
- (3) 最近最久未使用置换算法(LRU)
- (4) 最少使用置换算法(LFU)
- (5) Clock置换算法
- (6) 页面缓冲算法

## 5.6 ”抖动”和工作集

①工作集：指在某段时间间隔里，进程实际所要访问页面的集合。

### ②产生抖动的原因:

同时在系统中运行的进程太多，分配给每一个进程的物理块太少，使每个进程在运行时，频繁出现缺页，必须请求系统将所缺之页调入内存。导致在系统中排队等待页面调入和调出的进程数目增加。对磁盘的访问时间增加，造成每个进程大部分时间都用于页面的换进和换出，几乎不能进行有效的工作，从而导致发生处理机的利用率急剧下降并趋于0的情况(抖动状态)。

### ③抖动预防的方法:

- (1) 采取局部置换策略
- (2) 把工作集算法融入到处理机调度中
- (3) 利用 $L=S$ 准则调节缺页率
- (4) 选择暂停的进程

## 5.7 请求分段存储管理方式

共享段的分配与回收

## 6 第六章 输入输出系统

### 6.1 I/O系统

①I/O系统管理的主要对象是I/O设备和相应的设备控制器。

②I/O系统最主要的任务：

- (1) 完成用户提出的I/O请求
- (2) 提高I/O速率
- (3) 提高设备的利用率
- (4) 为更高层的进程方便地使用这些设备提供手段

③I/O系统的基本功能：

- (1) 隐藏物理设备的细节
- (2) 与设备的无关性
- (3) 提高处理机和I/O设备的利用率
- (4) 对I/O设备进行控制
- (5) 确保对设备的正确共享
- (6) 错误处理

④I/O软件的层次结构：

- (1) 用户层I/O软件，实现与用户交互的接口。
- (2) 设备独立性软件，用于实现用户程序与设备驱动的统一接口等，同时为设备管理和数据传达提供存储空间。
- (3) 设备驱动程序，与硬件直接相关，用于具体实现OS发出的指令，驱动I/O设备工作的驱动程序。
- (4) 中断处理程序，保护、处理、恢复、返回被中断进程的CPU环境。

⑤I/O系统本身的层次结构：

- (1) 中断处理程序
- (2) 设备驱动程序
- (3) 设备独立软件

⑥I/O系统的接口：

- (1) 块设备接口
- (2) 流设备接口
- (3) 网络通信接口

## 6.2 I/O设备

I/O设备类型:

按照使用特性分类:

- (1) 存储设备(外存、辅存)
- (2) I/O设备: 又分为输入设备、输出设备和交互式设备。

按照传输速率分类:

- (1) 低速设备
- (2) 中速设备
- (3) 高速设备

## 6.3 设备控制器

①设备控制器的基本功能:

- (1) 接收和识别命令
- (2) 数据交换
- (3) 标识和报告设备的状态
- (4) 地址识别
- (5) 数据缓冲区
- (6) 差错控制

②设备控制器的组成:

- (1) 设备控制器与处理机的接口
- (2) 设备控制器与设备的接口
- (3) I/O逻辑

## 6.4 I/O通道

①I/O通道是一种特殊的处理机, 它具有执行I/O指令的能力, 并通过执行通道程序来控制I/O操作。

I/O通道与一般处理机不同的两个方面: 指令类型单一、通道没有自己的内存。

②通道类型

根据信息交换方式的不同

- (1) 字节多路通道
- (2) 数组选择通道
- (3) 数组多路通道

③解决“瓶颈”问题有效方式: 增加设备到主机间的通路而不增加通道

## 6.5 中断机构和中断处理程序

**中断**是指CPU对I/O设备发来的中断的信号的一种响应。中断是由于外部设备引起的，故称为外中断。

**陷入**是由于CPU内部事件所引起的中断。

例如：进程在运算中发生了上溢或者下溢；程序出错(非法指令、地址越界、电源故障)

①**中断处理程序的步骤：**

- (1) 测定是否有未响应的中断信号
- (2) 保护被中断进程的CPU环境
- (3) 转入相应的设备处理程序
- (4) 中断处理
- (5) 恢复CPU的现场并退出中断

②**简答-为什么说中断是OS赖以生存的基础：**

- (1) 中断处理程序是I/O系统中最低的一层，是整个I/O系统的基础。
- (2) 中断是多道程序得以实现的基础，进程之间的切换是通过中断来完成。
- (3) 中断是设备管理的基础，能够提高处理机的利用率和实现CPU与I/O设备并行执行

## 6.6 设备驱动程序

①**对I/O设备的控制方式：**

- (1) 使用轮询的可编程I/O方式
- (2) 使用中断的可编程I/O方式
- (3) 直接存储器访问方式
- (4) I/O通道控制方式

②**直接存储器访问方式→ DMA控制器**

由主机与DMA的接口、DMA控制器与块设备的接口、I/O控制逻辑 三部分组成。

## 6.7 设备分配

①**进行设备分配需要的数据结构：**

- (1) 设备控制表DCT
- (2) 控制器控制表COCT
- (3) 通道控制表CHCT
- (4) 系统设备表SDT

②**设备分配算法：**

- (1) 先来先服务
- (2) 优先级高者优先

## 6.8 假脱机(Spooling)系统

### ①什么是假脱机技术：

Spooling 是利用磁盘作为大型缓冲区，通过输入井、输出井和假脱机进程，实现输入/输出与 CPU 并行，并使外设可被多个用户共享的技术。

### ②假脱机的组成：

#### (1) 输入井和输出井：在磁盘上开辟两个大型存储区：

输入井：模拟脱机输入的磁盘，用于暂存输入设备送来的数据；

输出井：模拟脱机输出的磁盘，用于暂存用户程序的输出数据。

#### (2) 输入缓冲区与输出缓冲区：在主存中建立两个缓冲区：

输入缓冲区：暂存输入设备送来的数据，再写入输入井；

出缓冲区：暂存从输出井读出的数据，再送往输出设备。

#### (3) 输入进程 SPi 和输出进程 SPo：系统设立两个假脱机进程。

#### (4) 井管理程序：井管理程序负责控制作业与磁盘井之间的数据交换。

### ③假脱机的特点：

#### (1) 提高了I/O的速度

#### (2) 将独占设备改造为共享设备

#### (3) 实现了虚拟设备功能

## 6.9 缓冲区管理

引入缓冲区的原因：

#### (1) 缓和CPU和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) 解决数据颗粒不匹配的问题

#### (4) 提高CPU和I/O设备的之间的并行性

## 6.10 磁盘存储器的性能和调度

### ①磁盘访问时间 = 寻道时间 + 旋转延迟时间 + 数据传输时间

#### (1) 寻道时间——移动磁头的时间

#### (2) 旋转延迟时间——等待扇区转到磁头下

#### (3) 数据传输时间——读/写数据的时间

### ②磁盘调度算法：

早期的磁盘调度算法：

- (1) 先来先服务FCFS
- (2) 最短寻道时间优先SSTF

基于扫描的磁盘调度算法:

- (1) 扫描算法SCAN
- (2) 循环扫描算法CSCAN
- (3) NStepSCAN
- (4) FSCAN调度算法

## 7 第七章 文件管理

### 7.1 文件和文件系统

①**数据组成**：数据项、记录、文件

②**文件的打开操作**：

当用户第一次请求对某文件进行操作时，须先利用open系统调用将该文件打开。

”打开”是指系统将文件的属性，从外存拷贝到内存打开文件表的一个目录中，并将该表目的编号返回给用户。

”打开”就是在用户和指定文件之间建立起一个链接。

### 7.2 文件的逻辑结构

分类

- (1) **按是否有结构分类**：有结构文件(定长记录、变长记录)、无结构文件
- (2) **按文件的组织方式分类**：顺序文件、索引文件、索引顺序文件

### 7.3 文件目录

①**对目录管理的要求**：

- (1) 实现”按名存取”
- (2) 提高对目录的检索速度
- (3) 文件共享
- (4) 允许文件重名

②**树形结构目录**：最通用且实用的目录文件。

可以明显提高对目录的检索速度和文件系统的性能  
提高文件系统的灵活性

### 7.4 文件共享

利用符号链实现共享的优点/缺点：

优点：

- (1) **占用空间少**，符号链只保存目标文件的路径名，不需要复制文件内容，开销很小。
- (2) **可跨文件系统共享**，符号链保存的是路径，因此不受文件系统物理结构限制，可跨不同文件系统建立共享。
- (3) **操作灵活**，不改变原文件结构，不影响其他目录，便于组织和管理共享文件。
- (4) **共享效果好**，目标文件发生修改后，通过符号链访问到的内容始终是最新的。

缺点：



- (1) **依赖路径，易产生悬空链接**，当目标文件被删除或路径改变时，符号链将指向无效位置而无法访问。
- (2) **访问效率略低**，访问共享文件需要先解析符号链，再定位目标文件，比直接访问多一次路径转换。
- (3) **可靠性较差**，若链目标发生变化，需手动维护；系统结构调整时容易失效。
- (4) **可能带来安全隐患**，因链目标可被修改，容易引起错误引用或产生危险的链接跳转。

## 8 第八章 磁盘存储器的管理

### 8.1 外存的组织方式

- (1) 连续组织方式
- (2) 链接组织方式
- (3) 索引组织方式

### 8.2 FAT技术

①**FAT技术**：利用文件分配表FAT来记录每个文件中所有盘块之间的链接

②**FAT12**：是以盘块为基本单位分配的

每个分区配有两张相同的文件分配表FAT1和FAT2

FAT 中每个表项存放该盘块的“下一盘块号”，相当于建立盘块之间的链式链接。系统将文件的**起始(第一个)盘块号存入文件的 FCB 中**，由此根据FAT指针逐个找到文件的所有盘块，实现链式文件的组织与管理。

③**分配单位-簇**

为了适应磁盘容量不断增大的需要，文件系统在进行外存分配时，不再以“盘块”为最小分配单位，而改为以“簇”为基本分配单位。

以“簇”为基本分配单位的好处：

- (1) 能适应磁盘容量不断增大的情况
- (2) 能减少FAT表中的项数，使FAT表占用更少的存储空间，并减少访问FAT表的存取开销

④**FAT16**：将FAT表项位数增加至16

### 8.3 文件存储空间管理方法

- (1) 空闲表法和空闲链表法
- (2) 位示图法
- (3) 成组链接法

### 8.4 提高对文件的访问速度的途径

- (1) 改进文件的目录结构以及检索目录的方法来减少对目录的查找时间
- (2) 选取好的文件存储结构，以提高对文件的访问速度
- (3) 提高磁盘I/O速度，能将文件中的数据快速地从磁盘传送到内存

## 8.5 提高磁盘I/O速度的途径

- (1) 磁盘高速缓存
- (2) 提前读
- (3) 延迟写
- (4) 优化物理块的分布
- (5) 虚拟盘

## 8.6 引入检查点的目的:

使对事务记录表中的事务记录清理工作经常化。