# 概述

## 操作系统的定义：

## 提供用户的使用方式

## 五大功能；

## 操作系统的分类、各种类别的特点；

## 特权指令、处理器的状态及程序状态字。

## 操作系统结构

## 现代操作系统特点

# 进程与线程

## 多道程序并发执行的特点

## 进程的定义及其与程序的区别、进程的组成、进程的三态及转换、进程控制原语

## 线程的定义、与进程的区别。

## 多核、多线程与超线程

# 互斥与同步

## 并发进程产生与时间有关错误

## 临界区，对临界区的管理要求

## 信号量的定义，其物理意义

## 原语概念

## 用信号量实现进程互斥与同步

### 哲学家餐问题同步与互斥：

#### 方案一（最多允许四个哲学家拿筷子）：

semaphore chopstick[5]={l，1，l，1，1}，s=4；

void philosopher(int i) ／\*哲学家进程\*／

{ while (true) {

P(s) ;

P(chopstick[i])；P(chopstick[(i+1)％5])；

…；

eat； ／\*进餐\*／

…：

V(chopstick[i])； V(chopstick[(i+1)％5])；

V(s) ;

…：

think； ／\*思考\*／

…；

}

}

解释：

这段代码使用信号量的机制来避免死锁，主要体现在两个方面：

1. \*\*有限的资源：\*\* 通过`s`信号量，限制同时进行进餐的哲学家数量。`P(s)`和`V(s)`分别是对`s`信号量的等待和释放操作。初始时`s`为4，表示最多允许4个哲学家同时进餐。当一个哲学家开始进餐时，首先要通过`P(s)`等待信号量，如果信号量的值大于0（有可用资源），则允许进餐，同时`V(s)`会释放信号量，增加可用资源的数量。这样可以确保系统不会因为资源不足而导致死锁。

2. \*\*筷子的获取：\*\* 哲学家获取筷子的过程通过`P(chopstick[i])`和`P(chopstick[(i+1)%5])`操作，其中`P`操作是对信号量的等待操作。在这里，如果哲学家不能同时获取左手边和右手边的筷子（即等待两个`P`操作的过程中某一个失败），他会一直等待，不会一直占有一个筷子而导致其他哲学家无法获取筷子。这避免了资源争夺的情况，从而避免了死锁。

总体来说，通过合理的信号量设计，确保对资源的获取和释放操作都是原子的，能够避免死锁的发生。在这个问题中，哲学家在获取资源（筷子）时会等待，而且通过有限的资源控制，能够防止所有哲学家同时进餐，从而有效地避免了死锁。

#### 方案二（设定拿筷子的规则，避免死锁）：

semaphore chopstick[5] = {1, 1, 1, 1, 1};

void philosopher(int i) {

while (true) {

if (i < (i+1)%5) {

* + - 1. P(chopstick[i]);

P(chopstick[(i+1)%5]);

} else {

P(chopstick[(i+1)%5]);

P(chopstick[i]);

}

// 进餐操作

V(chopstick[i]);

V(chopstick[(i+1)%5]);

// 思考操作

}

}

解释：

初始化： semaphore chopstick[5] = {1, 1, 1, 1, 1}; 初始化了5个信号量，每个信号量对应一个筷子，初始值为1，表示筷子可用。

互斥性： 这段代码确保哲学家在进餐前会同时获取左右两只筷子，以确保互斥性。通过比较i和(i + 1) % 5的大小关系，决定先获取左边的筷子还是右边的筷子。

信号量操作：

P(chopstick[i]); 和 P(chopstick[(i + 1) % 5]); 分别是对左手边和右手边的筷子进行等待操作。

V(chopstick[i]); 和 V(chopstick[(i + 1) % 5]); 分别是对左手边和右手边的筷子进行释放操作。

循环： while (true) 表示哲学家会一直循环进行思考、进餐和思考的过程。

这段代码确保了哲学家在进餐时会先获取左右两只筷子，确保了互斥性，从而避免了死锁。这种策略被称为“左右等待”或“左右规则”，通过一个规定的获取筷子顺序，确保哲学家不会陷入循环等待的死锁情况。

## 直接通信，间接通信；

## 死锁的定义，四个必要条件

## 死锁的防止

## 死锁的避免与银行家算法

## 死锁的检测与恢复

# 处理器调度

## 处理器调度层次

## 调度准则

## 短程调度算法

## 单处理器调度算法

## 多处理器调度算法

## 实时调度

# 存储管理

## 存储管理的功能、存储分配的三种方式

## 重定位的定义、两种重定位的特点与区别、覆盖与交换

## 分区存储管理、页式存储管理(各种方法采用的分配回收算法，数据结构，地址变换过程，共享与保护，优缺点比较）

## 虚拟存储器基本思想，页式虚拟存储工作流程，

## 常用的页面置换算法（FIFO, LRU, OPT,CLOCK, NRU,LFU)

## 段式存储管理的思想，段式虚拟存储管理流程

# 文件管理

## 文件

## 文件目录

## 文件共享与保护

## 文件存储空间的管理

## 文件分配

# 第七章 I/O管理

## I/O管理概述

## I/O控制方式

## I/O缓冲

## 设备驱动程序

## 设备分配

## 磁盘存储器的管理,常用调度算法（FCFS,SSTF,SCAN,C-SCAN)；

## 磁盘阵列(RAID)

## 时钟管理和电源管理

# 操作系统的安全性

## 安全的计算机系统的三个特性：保密性、完整性和可用性

## OS的安全性是计算机系统安全的基础

## 有选择的访问控制、内存管理与对象重用、审计能力、文件和数据的加密以及进程间的安全通信机制。

## 保护机制：保护域、ACL、权能表

## 安全机制:

## 用户认证；

## 恶意代码及时防范、发现和清除

## 使用情况的监控和审计

## 加密技术

## 备份

# 新型操作系统

## 嵌入式操作系统

## 微软嵌入式操作系统

## 嵌入式Linux与安卓

## TinyOS

## VxWorks

## 多媒体操作系统

## 分布式操作系统

## 集群系统

## MapReduce计算模型

## 虚拟化技术