

<<操作系统>>

图书基本信息

书名：<<操作系统>>

13位ISBN编号：9787121185106

10位ISBN编号：7121185105

出版时间：2012-9

出版时间：电子工业出版社

作者：[美] William Stallings

页数：541

字数：850000

译者：陈向群,陈渝

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<操作系统>>

前言

译者序 操作系统在计算机系统中占据重要的地位，是计算机系统的核心和灵魂，是构建在计算机硬件之上的第一层软件，也是基础软件运行平台的主要成分。

“操作系统”是计算机科学与技术专业的一门非常重要的专业基础课，一本好的教材可以使学生通过仔细阅读、深入思考、不断练习，循序渐进地提高自己的专业知识和技能水平。

一本教材，承载着什么？

学生和教师对它的期望又是什么？

William Stallings博士编写的《操作系统——精髓与设计原理》第七版给出了完美的答案。

William Stallings博士在教材的修订过程中，将第六版交给了一批从事相关教学和研究的教授们审阅，使新一版的叙述更加清晰、紧凑，说明文字也进行了修正，使本书在教育学及用户友善性方面有了新的改进。

作为一本经典的教材，在美国，多所大学采用本书的前几个版本作为教材或参考书目；在中国，讲授操作系统课程的教师和学习操作系统课程的学生对“精髓”一书也是非常熟悉。

我们真诚地向广大读者推荐William Stallings博士的这本教材。

本教材提供了操作系统原理的全景图，对操作系统的概念、结构和机制，进行了系统、全面的阐述，并清晰地展示了当代操作系统的本质和特点。

特别是，由于本教材已经出到第七版，每一版本都在多位教师使用、反馈建议的基础上，对不足进行了更新，并结合新技术的发展，增加了新内容以反映操作系统的新进展。

教材第七版主要内容分为八个部分。

第一部分综述了计算机组织与系统结构，重点讲述与操作系统设计相关的主题。

第二部分详细分析了进程、多线程、对称多处理（SMP）和微内核，以及单一系统中的并发机制，重点讲述互斥和死锁。

第三部分全面介绍了内存管理技术，包括虚拟存储器。

第四部分讨论了线程调度、SMP调度和实时调度，分析比较了多种进程调度方法。

第五部分主要分析了操作系统中对I/O功能的控制，特别是磁盘I/O，给出了关于文件管理的综述。

第六部分讨论了嵌入式操作系统的一般性原理，并介绍了两个实例系统：TinyOS和eCos。

第七部分概述了涉及计算机和网络安全的威胁与机制。

第八部分围绕分布式系统，分析了计算机系统网络化技术的主要趋势，同时还介绍了分布式系统开发中的一些主要设计领域。

本书依然选择最流行、应用最广泛的Windows（Windows7）和Linux作为实例操作系统，并增补了多核操作系统、虚拟机、新的调度程序实例、面向服务的架构（SOA）、概率、统计和排队分析、B树等反映操作系统领域进展变化的新内容。

在讲解操作系统每一功能的过程中，引入了实例操作系统，以加深读者对原理的理解。

本书配备有相应的网站，包括一系列相关链接，包括作业题目和解答、编程项目、重要论文、支持文档、在线章节、在线附录等，并补充了大量的“现场测试”型作业练习。

本书在每一章开始给出了学习目标，还按章节给出了学生应该在本章关注的主要概念。

本书给教师提供了几个教学大纲样例，允许教师在有限的时间（如16周或12周）内完成教学任务。

这些样例是在一些教师们在第六版的真实教学经验的基础上形成的。

上述内容为学生和教师提供丰富的辅助材料。

为了配合基于本书的操作系统教学，我们参照本书的重要知识点实现了一个教学用uCore操作系统，并基于uCore操作系统设计了8个增量式的操作系统实验。

有兴趣的学生可在了解基本的操作系统概念和原理的基础上，通过阅读实验指导书，分析uCore操作系统源码，完成实验中的练习，扩展uCore操作系统功能，来逐步体会一个操作系统是如何实现的，以及操作系统的概念原理和实际实现之间的紧密联系与巨大差异。

相关实验指导书文档和实验代码放在https://github.com/chyyuu/ucore_pub上，且会持续更新。

参加本书翻译、审阅和校对的还有刘伟、杜圆圆、阎梵茜、徐淑琪、孔俊俊、闫林、张琳、冯涛

<<操作系统>>

、刘晗、王浩宇、周欣、陈刚、李小奇、刘梦馨、闫丰润、袁鹏飞、陈昕、吕骁博等人，在此对他（她）们的贡献表示诚挚的谢意。

由于译者水平有限，本书的译文中必定会存在一些不足或错误之处，欢迎各位专家和广大读者批评指正。

译者 2012年8月

<<操作系统>>

内容概要

本书是一本关于操作系统的概念、结构和机制的教材，其目的是尽可能清楚和全面地展示现代操作系统的本质和特点；同时，本书也是讲解操作系统的经典教材，不仅系统地讲述了操作系统的基本概念、原理和方法，而且以当代最流行的操作系统——Windows 7、UNIX和Linux为例，全面清楚地展现了当代操作系统的本质和特点。与本书配套的专用网站，为帮助教师和学生理解书中内容，提供了及时、生动的材料。

<<操作系统>>

作者简介

作者: (美) William Stallings (威廉·斯托林斯) 译者: 陈向群

<<操作系统>>

书籍目录

第0章 读者与教师指南

0.1 本书概述

0.2 实例系统

0.3 读者和教师的学习路线图

0.4 互联网和网站资源

第一部分 背景知识

第1章 计算机系统概述

1.1 基本构成

1.2 微处理器的发展演化

1.3 指令的执行

1.4 中断

1.4.1 中断和指令周期

1.4.2 中断处理

1.4.3 多个中断

1.5 存储器的层次结构

1.6 高速缓存

1.6.1 动机

1.6.2 高速缓存原理

1.6.3 高速缓存设计

1.7 直接内存存取

1.8 多处理器和多核计算机组织结构

1.8.1 对称多处理器

1.8.2 多核计算机

1.9 推荐读物和网站

1.10 关键术语、复习题和习题

1.10.1 关键术语

1.10.2 复习题

1.10.3 习题

附录1A 两级存储器的性能特征

第2章 操作系统概述

2.1 操作系统的目标和功能

2.1.1 作为用户/计算机接口的操作系统

2.1.2 作为资源管理器的操作系统

2.1.3 操作系统的易扩展性

2.2 操作系统的发展

2.2.1 串行处理

2.2.2 简单批处理系统

2.2.3 多道批处理系统

2.2.4 分时系统

2.3 主要的成就

2.3.1 进程

2.3.2 内存管理

2.3.3 信息保护和安全

2.3.4 调度和资源管理

2.4 现代操作系统的特征

<<操作系统>>

2.5 虚拟机

2.5.1 虚拟机和虚拟化

2.5.2 虚拟机架构

2.6 针对多处理器和多核的操作系统设计考虑因素

2.6.1 对称多处理器计算机的操作系统设计考虑因素

2.6.2 多核计算机的操作系统设计考虑因素

2.7 微软Windows系统简介

2.7.1 历史

2.7.2 现代操作系统

2.7.3 体系结构

2.7.4 客户/服务器模型

2.7.5 线程和SMP

2.7.6 Windows对象

2.7.7 Windows 7中的新特性

2.8 传统的UNIX系统

2.8.1 历史

2.8.2 描述

2.9 现代UNIX系统

2.9.1 System V版本4(简称SVR4)

2.9.2 BSD

2.9.3 Solaris 10

2.10 Linux操作系统

2.10.1 历史

2.10.2 模块结构

2.10.3 内核组件

2.11 Linux Vserver虚拟机结构

2.12 推荐读物和网站

2.13 关键术语、复习题和习题

2.13.1 关键术语

2.13.2 复习题

2.13.3 习题

第二部分 进程

第3章 进程描述和控制

3.1 什么是进程

3.1.1 背景

3.1.2 进程和进程控制块

3.2 进程状态

3.2.1 两状态进程模型

3.2.2 进程的创建和终止

3.2.3 五状态模型

3.2.4 被挂起的进程

3.3 进程描述

3.3.1 操作系统的控制结构

3.3.2 进程控制结构

3.4 进程控制

3.4.1 执行模式

3.4.2 进程创建

<<操作系统>>

- 3.4.3 进程切换
- 3.5 操作系统的执行
 - 3.5.1 无进程的内核
 - 3.5.2 在用户进程中执行
 - 3.5.3 基于进程的操作系统
- 3.6 安全问题
 - 3.6.1 系统访问威胁
 - 3.6.2 对抗措施
- 3.7 UNIX SVR4进程管理
 - 3.7.1 进程状态
 - 3.7.2 进程描述
 - 3.7.3 进程控制
- 3.8 小结
- 3.9 推荐读物
- 3.10 关键术语、复习题和习题
 - 3.10.1 关键术语
 - 3.10.2 复习题
 - 3.10.3 习题
- 第4章 线程
 - 4.1 进程和线程
 - 4.1.1 多线程
 - 4.1.2 线程功能特性
 - 4.2 线程分类
 - 4.2.1 用户级和内核级线程
 - 4.2.2 其他方案
 - 4.3 多核和多线程
 - 4.3.1 多核系统上的软件性能
 - 4.3.2 应用示例：Valve游戏软件
 - 4.4 Windows 7线程和SMP管理
 - 4.4.1 进程对象和线程对象
 - 4.4.2 多线程
 - 4.4.3 线程状态
 - 4.4.4 对OS子系统的支持
 - 4.4.5 对称多处理的支持
 - 4.5 Solaris的线程和SMP管理
 - 4.5.1 多线程体系结构
 - 4.5.2 动机
 - 4.5.3 进程结构
 - 4.5.4 线程的执行
 - 4.5.5 把中断当做线程
 - 4.6 Linux的进程和线程管理
 - 4.6.1 Linux任务
 - 4.6.2 Linux线程
 - 4.7 Mac OS X的GCD技术
 - 4.8 小结
 - 4.9 推荐读物
 - 4.10 关键术语、复习题和习题

<<操作系统>>

4.10.1 关键术语

4.10.2 复习题

4.10.3 习题

第5章 并发性：互斥和同步

5.1 并发的原理

5.1.1 一个简单的例子

5.1.2 竞争条件

5.1.3 操作系统关注的问题

5.1.4 进程的交互

5.1.5 互斥的要求

5.2 互斥：硬件的支持

5.2.1 中断禁用

5.2.2 专用机器指令

5.3 信号量

5.3.1 互斥

5.3.2 生产者/消费者问题

5.3.3 信号量的实现

5.4 管程

5.4.1 使用信号的管程

5.4.2 使用通知和广播的管程

5.5 消息传递

5.5.1 同步

5.5.2 寻址

5.5.3 消息格式

5.5.4 排队原则

5.5.5 互斥

5.6 读者/写者问题

5.6.1 读者优先

5.6.2 写者优先

5.7 小结

5.8 推荐读物

5.9 关键术语、复习题和习题

5.9.1 关键术语

5.9.2 复习题

5.9.3 习题

第6章 并发：死锁和饥饿

6.1 死锁原理

6.1.1 可重用资源

6.1.2 可消耗资源

6.1.3 资源分配图

6.1.4 死锁的条件

6.2 死锁预防

6.2.1 互斥

6.2.2 占有且等待

6.2.3 不可抢占

6.2.4 循环等待

6.3 死锁避免

<<操作系统>>

- 6.3.1 进程启动拒绝
 - 6.3.2 资源分配拒绝
 - 6.4 死锁检测
 - 6.4.1 死锁检测算法
 - 6.4.2 恢复
 - 6.5 一种综合的死锁策略
 - 6.6 哲学家就餐问题
 - 6.6.1 基于信号量解决方案
 - 6.6.2 基于管程解决方案
 - 6.7 UNIX的并发机制
 - 6.7.1 管道
 - 6.7.2 消息
 - 6.7.3 共享内存
 - 6.7.4 信号量
 - 6.7.5 信号
 - 6.8 Linux内核并发机制
 - 6.8.1 原子操作
 - 6.8.2 自旋锁
 - 6.8.3 信号量
 - 6.8.4 屏障
 - 6.9 Solaris线程同步原语
 - 6.9.1 互斥锁
 - 6.9.2 信号量
 - 6.9.3 多读者/单写者锁
 - 6.9.4 条件变量
 - 6.10 Windows 7并发机制
 - 6.10.1 等待函数
 - 6.10.2 分派器对象
 - 6.10.3 临界区
 - 6.10.4 轻量级读者-写者锁和条件变量
 - 6.10.5 锁无关同步机制
 - 6.11 小结
 - 6.12 推荐读物
 - 6.13 关键术语、复习题和习题
 - 6.13.1 关键术语
 - 6.13.2 复习题
 - 6.13.3 习题
- 第三部分 内存
- 第7章 内存管理
- 7.1 内存管理的需求
 - 7.1.1 重定位
 - 7.1.2 保护
 - 7.1.3 共享
 - 7.1.4 逻辑组织
 - 7.1.5 物理组织
 - 7.2 内存分区
 - 7.2.1 固定分区

<<操作系统>>

7.2.2 动态分区

7.2.3 伙伴系统

7.2.4 重定位

7.3 分页

7.4 分段

7.5 安全问题

7.5.1 缓冲区溢出攻击

.....

第四部分 调度

第五部分 输入/输出和文件

第六部分 嵌入式系统

第七部分 计算机安全

第八部分 分布式系统

附录A 并发主题

附录B 编程和操作系统项目

术语表

参考书目

<<操作系统>>

章节摘录

版权页：插图：在查看包含两个新挂起状态的状态转换图之前，必须提到另一点。到现在为止的论述都假设没有使用虚拟内存，进程或者都在内存中，或者都在内存之外。在虚拟内存方案中，可能会执行到只有一部分内容在内存中的进程，如果访问的进程地址不在内存中，则进程的相应部分可以被调入内存。

虚拟内存的使用看上去会消除显式交换的需要，这是因为通过处理器中的存储管理硬件，任何期望的进程中的任何期望的地址都可以移入或移出内存。

但是，正如在第8章中将会看到的，如果有足够多的活动进程，并且所有进程都有一部分在内存中，则有可能导致虚拟内存系统崩溃。

因此，即使在虚拟存储系统中，操作系统也需要不时地根据执行情况，显式地、完全地换出进程。

现在来看图3.9 (b) 中我们已开发的状态变转换型（图中的虚线表示可能但并不是必需的转换）。

比较重要的新转换如下：阻塞 / 挂起：如果没有就绪进程，则至少一个阻塞进程被换出，为另一个没有阻塞的进程让出空间。

如果操作系统确定当前正在运行的进程，或就绪进程为了维护基本的性能要求而需要更多的内存空间，那么，即使有可用的就绪态进程也可能出现这种转换。

阻塞 / 挂起 / 就绪 / 挂起：如果等待的事件发生了，则处于阻塞 / 挂起状态的进程可转换到就绪 / 挂起态。

注意，这要求操作系统必须能够得到挂起进程的状态信息。

就绪 / 挂起 / 就绪：如果内存中没有就绪态进程，操作系统需要调入一个进程继续执行。

此外，当处于就绪 / 挂起态的进程比处于就绪态的任何进程的优先级都要高时，也可以进行这种转换。

这种情况的产生是由于操作系统设计者规定，调入高优先级的进程比减少交换量更重要。

就绪 / 挂起 / 就绪 / 挂起：通常，操作系统更倾向于挂起阻塞态进程而不是就绪态进程，因为就绪态进程可以立即执行，而阻塞态进程占用了内存空间但不能执行。

但如果释放内存以得到足够空间的唯一方法是挂起一个就绪态进程，那么这种转换也是必需的。

并且，如果操作系统确信高优先级的阻塞态进程很快将会就绪，那么它可能选择挂起一个低优先级的就绪态进程，而不是一个高优先级的阻塞态进程。

还需要考虑的几种其他转换有：新建 / 就绪 / 挂起及新建 / 就绪：当创建一个新进程时，该进程或者加入就绪队列，或者加入就绪 / 挂起队列。

不论哪种情况，操作系统都必须建立一些表以管理进程，并为进程分配地址空间。

操作系统可能更倾向于在初期执行这些辅助工作，这使得它可以维护大量的未阻塞的进程。

通过这一策略，内存中经常会没有足够的空间分配给新进程，因此使用了（新建 / 就绪 / 挂起）转换。

另一方面，我们可以证明创建进程的适时（just-in-time）原理，即尽可能推迟创建进程以减少操作系统的开销，并在系统被阻塞态进程阻塞时允许操作系统执行进程创建任务。

阻塞 / 挂起 / 阻塞：这种转换在设计中比较少见，如果一个进程没有准备好执行，并且不在内存中，调入它又有什么意义？

但是考虑到下面的情况：一个进程终止，释放了一些内存空间，阻塞 / 挂起队列中有一个进程比就绪 / 挂起队列中任何进程的优先级都要高，并且操作系统有理由相信阻塞进程的事件很快就会发生，这时，把阻塞进程而不是就绪进程调入内存是合理的。

运行 / 就绪 / 挂起：通常当分配给一个运行进程的时间期满时，它将转换到就绪态。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>