目录

[**第1章 绪论** 2](#_Toc9698010)

[**1.1 操作系统概念** 2](#_Toc9698011)

[**1.2 操作系统的历史** 2](#_Toc9698012)

[**1.3 操作系统的基本类型** 3](#_Toc9698013)

[**1.4 操作系统功能** 3](#_Toc9698014)

[**1.5 计算机硬件简介** 4](#_Toc9698015)

[**1.6 算法的描述** 5](#_Toc9698016)

[**1.7 研究操作系统的几种观点** 5](#_Toc9698017)

[**第2章 操作系统用户界面** 5](#_Toc9698018)

[**2.1 简介** 5](#_Toc9698019)

[**2.2 一般用户的输入输出界面** 6](#_Toc9698020)

[**2.3 命令控制界面** 6](#_Toc9698021)

[**2.4 Linux与Windows的命令控制界面** 7](#_Toc9698022)

[**2.5 系统调用** 7](#_Toc9698023)

[**2.6 Linux和Windows的系统调用** 7](#_Toc9698024)

[**第3章 进程管理** 8](#_Toc9698025)

[**3.1 进程的概念** 8](#_Toc9698026)

[**3.2 进程的描述** 8](#_Toc9698027)

[**3.3 进程状态及转换** 9](#_Toc9698028)

[**3.4 进程控制** 9](#_Toc9698029)

[**3.5 进程互斥** 9](#_Toc9698030)

[**3.6 进程同步** 10](#_Toc9698031)

[**3.7 进程通信** 10](#_Toc9698032)

[**3.8 死锁问题** 11](#_Toc9698033)

[**3.9 线程的概念** 11](#_Toc9698034)

[**3.10 线程分类与执行** 12](#_Toc9698035)

[**第4章 处理机调度** 12](#_Toc9698036)

[**4.1 分级调度** 12](#_Toc9698037)

[**4.2 作业调度** 12](#_Toc9698038)

[**4.3 进程调度** 13](#_Toc9698039)

[**4.4 调度算法** 13](#_Toc9698040)

[**4.5 算法评价** 14](#_Toc9698041)

[**4.6 实时系统调度方法** 14](#_Toc9698042)

[**第5章 存储管理** 15](#_Toc9698043)

[**5.1 存储管理的功能** 15](#_Toc9698044)

[**5.2 分区存储管理** 16](#_Toc9698045)

[**5.3 覆盖与交换技术** 17](#_Toc9698046)

[**5.4 页式管理** 17](#_Toc9698047)

[**5.5 段式与段页式管理** 18](#_Toc9698048)

[**5.6 局部性原理和抖动问题** 19](#_Toc9698049)

[**第6章 进程与存储管理示例** 19](#_Toc9698050)

[**6.1 Linux进程和存储管理简介** 19](#_Toc9698051)

[**6.2 Linux进程结构** 20](#_Toc9698052)

[**6.3 进程控制** 21](#_Toc9698053)

[**6.4 Linux进程调度** 21](#_Toc9698054)

[**6.5 进程通信** 22](#_Toc9698055)

[**6.6 Linux存储管理** 22](#_Toc9698056)

[**第7章 Windows的进程与内存管理** 23](#_Toc9698057)

**第1章 绪论**

**1.1 操作系统概念**

定义和概念

**1.2 操作系统的历史**

1.2.1 手工操作阶段

1.2.2 早期批处理

①联机批处理

输入输出设备和主机直接相连或串行，输入输出过程主机不能工作

②脱机批处理

用专门的部件来负责并行处理输入输出，但同一时间宏观微观上处理器都在串行处理任务

1.2.3 多道程序系统multiprogramming

实现宏观上同时处理多个任务的功能，使处理器利用率最大化

1.2.4 分时操作系统

分时间段占用系统

1.2.5 实时操作系统

要求立即反馈和响应的系统

1.2.6 通用操作系统

结合多种功能和场景需求的系统

1.2.7 操作系统的进一步发展

·PC操作系统

·嵌入式操作系统

·网络操作系统

·分布式操作系统

·智能化操作系统

**1.3 操作系统的基本类型**

1.3.1 批处理操作系统batch processing operating system

·特征：

脱机使用

成批处理

多道程序运行

1.3.2 分时系统

时分复用的理念

一个联机的online多用户multi-user交互式操作系统，比如UNIX

1.3.3 实时系统

略

1.3.4 通用操作系统

略

1.3.5 个人计算机上的操作系统

略

1.3.6 网络操作系统

现在的操作系统都需要上网，所以不再有特别的网络操作系统

1.3.7 分布式操作系统

基于网络系统但是更加复杂

**1.4 操作系统功能**

1.4.1 处理机管理

如何调度和处理任务

1.4.2 存储管理

内存分配

存储保护

内存扩充

1.4.3 设备管理

输入输出的分配与管理

独立性

1.4.4 信息管理（文件系统管理）

略

1.4.5 用户接口

操作系统提供两种接口：

程序接口：供程序调用服务

作业接口：供用户控制操作作业命令

**1.5 计算机硬件简介**

1.5.1 计算机的基本硬件元素

处理器、存储器、输入输出控制于总线、外部设备

1.5.2 与操作系统相关的几种主要寄存器

典型的用户可编程寄存器：

①数据寄存器

存储数据用的

②地址寄存器

如名

③条件码寄存器

也称标志寄存器，存储一些标志位

·后面5种为典型的控制与状态寄存器

④程序计数器PC

内有下一周期被执行指令的地址

⑤指令寄存器IR

内有待执行指令

⑥程序状态字PSW

一个每个位代表系统不同状态信息的数

⑦中断现场保护寄存器

用于保存中断处状态的寄存器

⑧过程调用用堆栈

略

1.5.3 存储器的访问速度

由快到慢：

寄存器、高速缓存、内存、硬盘缓存、硬盘、光盘或磁盘

1.5.4 指令的执行与中断

指令的执行周期：

（程序计数器PC含有地址）

指令的读入

（指令寄存器IR存储指令）

指令的执行

中断后执行的仍然是（另一个）程序，所有的寄存器当前取值被备份，执行完中断程序之后再回复。

中断、硬件中断和软中断

（1）中断

中断指当出现需要时，CPU暂时停止当前程序的执行转而执行处理新情况的程序和执行过程。即在程序运行过程中，系统出现了一个必须由CPU立即处理的情况，此时，CPU暂时中止程序的执行转而处理这个新的情况的过程就叫做中断。

（2）硬件中断

·硬件中断是一个异步信号, 表明需要注意, 或需要改变在执行一个同步事件.

·硬件中断是由与系统相连的外设(比如网卡 硬盘 键盘等)自动产生的. 每个设备或设备集都有他自己的IRQ(中断请求), 基于IRQ, CPU可以将相应的请求分发到相应的硬件驱动上(注: 硬件驱动通常是内核中的一个子程序, 而不是一个独立的进程). 比如当网卡受到一个数据包的时候, 就会发出一个中断.

·处理中断的驱动是需要运行在CPU上的, 因此, 当中断产生时, CPU会暂时停止当前程序的程序转而执行中断请求. 一个中断只能中断一颗CPU(也有一种特殊情况, 就是在大型主机上是有硬件通道的, 它可以在没有主CPU的支持下, 同时处理多个中断).

·硬件中断可以直接中断CPU. 它会引起内核中相关代码被触发. 对于那些需要花费时间去处理的进程, 中断代码本身也可以被其他的硬件中断中断.

·对于时钟中断, 内核调度代码会将当前正在运行的代码挂起, 从而让其他代码来运行. 它的存在时为了让调度代码(或称为调度器)可以调度多任务.

（3）软中断

·软中断的处理类似于硬中断. 但是软中断仅仅由当前运行的进程产生.

·通常软中断是对一些I/O的请求.

·软中断仅与内核相联系, 而内核主要负责对需要运行的任何其他进程进行调度.

·软中断不会直接中断CPU, 也只有当前正在运行的代码(或进程)才会产生软中断. 软中断是一种需要内核为正在运行的进程去做一些事情(通常为I/O)的请求.

·有一个特殊的软中断是Yield调用, 它的作用是请求内核调度器去查看是否有一些其他的进程可以运行.

（4）硬件中断和软中断的区别

·硬件中断是由外设引发的, 软中断是执行中断指令产生的.

·硬件中断的中断号是由中断控制器提供的, 软中断的中断号由指令直接指出, 无需使用中断控制器.

·硬件中断是可屏蔽的, 软中断不可屏蔽.

·硬件中断处理程序要确保它能快速地完成任务, 这样程序执行时才不会等待较长时间, 称为上半部.

·软中断处理硬中断未完成的工作, 是一种推后执行的机制, 属于下半部.

1.5.5 操作系统的启动

电源开启---中断电信号---CPU发现操作系统引导区boot block---加载操作系统程序至内存---初始化硬件和数据结构等---显示用户界面

**1.6 算法的描述**

一些伪代码的约定

略

**1.7 研究操作系统的几种观点**

彼此不矛盾，不同角度理解操作系统

1.7.1 操作系统是计算机资源的管理者

略

1.7.2 用户界面的观点

略

1.7.3 进程管理的观点

进程即并发程序的执行

**第2章 操作系统用户界面**

**2.1 简介**

面向使用者（普通用户和管理员用户）：

不同的操作命令组成的集合

面向开发者：

一组系统调用的集合

**2.2 一般用户的输入输出界面**

2.2.1 作业的定义

应用角度的定义：用户要求计算机所做的有关该次业务处理的全部工作内容。而作业也可以细分为作业步，是指一次业务处理中相对独立的一步工作。

系统角度的定义：由程序、数据和作业说明书组成。可以理解为一次针对某个程序某次独立的系统性运行与计算。

·作业job的概念一般用于早期系统或现在的大型机组系统中，其他系统中则称为任务task

2.2.2 作业组织

作业控制块job control block，JCB：系统将作业说明书转换为一个表格来进行解读，该表格即为JCB

作业说明书的内容层次结构：p24/41

·PC系统中，常用批处理文件bat或shell程序方式编写作业说明书

2.2.3 一般用户的输入输出方式

5种

①联机输入输出方式

多用于交互式系统中

②脱机输入输出方式

也称预输入方式，即用户和执行计算的主机并不进行逐字句实时交互，而是先将作业布置好在一独立存储设备上，之后向主机进行统一提交

③直接耦合方式

脱机方式的升级版，在主机和用户（机）之间建立一个公用存储器，用来避免用户手动提交作业以及系统化管理

④SPOOLING系统

译作外围设备同时联机操作。作业输入主机外存的过程由主机中的系统自行控制

⑤网络联机方式

基于以上几种本地联机方式，将用户机和主机布置于网络上即为网络联机方式

**2.3 命令控制界面**

定义：针对普通用户提供的用来组织和控制作业运行的

也分两种形式：

脱机方式——用户将作业的规格以作业控制说明书的方式或命令文件方式一并提交

联机方式——用户一步一步和系统会话，也是最常用方式

联机方式的命令种类：

①环境设置

②执行权限管理

③系统管理

④文件管理

⑤编辑、编译、链接装配和执行编辑命令

⑥通信

⑦资源要求

现代操作系统中，尤其是PC，批处理方式的定义有两种：

·按照作业控制语言编写说明书并提交

·将联机交互命令按照一定格式组合成命令文件，统一提交

**2.4 Linux与Windows的命令控制界面**

2.4.1 Linux的命令控制界面

分2部分：图形化窗口和Shell程序（或Shell脚本）解释器

Linux 的命令分类（9种）：p28/45

Shell程序/脚本命令的组成：p29/46

2.4.2 Windows的命令控制界面

分为两部分：命令行解释器cmd和窗口交互

·cmd支持MS-DOS保留的基本命令和Windows的操作命令，即cmd≠DOS

Windows命令的分类（4类）：p30/47

系统信息命令

系统操作命令

文件系统命令

网络通信命令

Windows命令的基本使用法则：p30/47

使用windows命令的两种方式：

命令行输入

批处理文件统一执行（实际上是多条命令行输入）

**2.5 系统调用**

命令控制界面实际上是在系统调用的基础上开发完成的。

系统调用的分类（6种）：p32/49

设备管理

文件管理

进程控制

进程通信

存储管理

线程管理

系统调用过程实现的基本原理（基于中断机制）：p32/49

陷阱trap指令：也称访管指令，由于系统调用引起处理机中断的指令。指令一般都是指机器指令。陷阱指令必须包括对应系统调用的编号，该编号和系统调用一一对应。

·陷阱实际上是广义上的中断的一种，狭义上陷阱是内中断，中断则指外中断，具体区别见后文。

用户模式和系统模式（或内核模式）：该模式在PSW中由硬件自动设置和体现，用于控制用户程序和系统程序彼此之间独立运行。

系统调用中的参数传递方案：p33/50

**2.6 Linux和Windows的系统调用**

2.6.1 Linux系统调用

每个系统调用分为两部分：

核心函数：操作系统的核心程序

接口函数：用户的API接口，传递参数给核心函数

Linux系统调用的分类（6类）：p33/50

2.6.2 Windows系统调用

Windows系统调用被封装成库函数也称API，分为5类：p34/51

**第3章 进程管理**

**3.1 进程的概念**

现代操作系统的重要特点：

程序并发执行、系统资源共享、用户随机使用系统

3.1.1 程序的并发执行

①程序program

②程序的顺序执行

特点：

顺序、封闭、可再现

③多道程序系统中程序执行环境的变化

程序之间独立性、输入随机性、资源共享

④程序的并发concurrent执行

并发执行的定义和条件：p40/57

3.1.2 进程的定义p42/59

进程的概念引入就是为了解决程序的并发执行问题。

程序（菜单）→进程（炒菜过程）

·一个程序的每次执行实例都可以称为是一个独立的进程，不同的进程之间彼此可以共享程序段或一些逻辑结构。程序的静态特征是一个可执行文件以及相关资源文件，进程的静态特征是机器指令队列、数据和堆栈结构、PCB，这三者统称进程上下文。

**3.2 进程的描述**

进程的静态描述：

进程控制块PCB、有关程序段、被操作的数据结构集

·PCB一般常驻内存（或主要部分常驻），后两者在执行时由外存进入内存

3.2.1 进程控制块PCB

PCB的基本内容：

①描述信息

包括进程名或进程标识号、用户名或标识号、进程之间的家族关系

②控制信息

包括进程当前状态（初始、就绪、执行、等待、终止），优先级，程序开始地址，计时信息，通信信息

③资源管理信息

PCB中的主要内容

④CPU现场保护结构

保存进程被中断时的现场状态

3.2.2 进程上下文

定义：一个抽象的概念，指代进程执行时各种寄存器状态，代码集，数据集和堆栈等全部相关事物。

上文：已执行过的进程指令和数据在相关寄存器与堆栈中的内容

正文：正在执行……

下文：待执行……

Unix System V示例：

·用户级上下文：用户程序段相关数据和栈

·寄存器上下文：各种寄存器状态

·系统级上下文：动态部分是切换上下文时保存的寄存器状态（以栈结构层次化保存），静态部分包括PCB、虚地址映射表格、核心栈（系统调用的序列）

3.2.3 进程上下文切换

切换的原因包括中断或进程调用

包括3部分：

·原进程上下文的存储

·操作系统进程进行调度和资源分配

·新进程上下文的恢复/新建

3.2.4 进程空间与大小

进程空间：也称虚空间，一个进程自己的地址空间

UNIX以及Linux等操作系统中，进程空间被划分为用户空间和系统空间。两个空间分别为用户模式和系统模式使用

**3.3 进程状态及转换**

3.3.1 进程状态

5大状态：初始、就绪、执行、等待、终止

·有的系统存在内存就绪和外存就绪两种，外存需要进入内存才能被执行

·进程虽然是和一个用户程序相关的，但是用户程序运行必定需要系统调用，所以每个进程都实际上至少有两个程序与它相关，即用户程序和系统程序。也就对应了两种执行状态。

·等待状态可以根据等待何种资源进行分类

3.3.2 进程状态转换

不同的控制过程和借助硬件触发器

**3.4 进程控制**

原语：系统态下执行的某些具有特定功能的程序段，分为两类

机器指令级——执行期间不许中断，视为原子指令

功能级——不允许并发执行

·操作系统中，进程控制用程序段一般都是原语，至少都不能并发执行

3.4.1 进程创建与撤销

①进程创建

两种方式：

系统程序模块统一创建

父进程创建

②进程撤销

必须先撤销全部子孙进程才能撤销本进程

3.4.2 进程的阻塞与唤醒

进程阻塞是进程自己调用系统相关进程阻塞自己，唤醒过程则是系统进程执行或由事件发生进程进行系统调用。

**3.5 进程互斥**

3.5.1 资源共享所引起的制约

①临界区critical section/region

定义：不允许多个并发进程交叉执行的一段程序

②间接制约

定义：由于存在共享公有资源的访问限制，导致进程受到访问性的制约，但是进程之间的执行顺序仍然是独立且任意的。

③什么是互斥

纯过程：在执行过程中不改变过程自身代码的过程，可以被并发进程同时访问，如编辑或编译程序。纯过程必须满足一些条件，执行效率上不一定最高。

一组并发进程互斥执行时必须满足的4项准则：p54/71

3.5.2 互斥的加锁实现

一个简单的实现范例：p55/72

3.5.3 信号量和P,V原语

信号量的概念是为了解决互斥锁实现方法所带来的的问题的，如资源分配不均。

①信号量semaphore

一个代表临界区公有资源实体的量，准确的数值（整数）代表以进程为单位的资源数——大于零时表示资源实体数，小于零表示等待获得资源进程数

②P,V原语primitive

一个基于等待队列的资源分配方式，P和V分别表示通过（派发资源）和增加（回收资源）的意思（源自发明人的荷兰语），是两个原语。

实现过程：p57/74（基于加锁法，此处的加锁法虽然也是用于限制对PV原语的并发访问，但是由于原语内有等待队列机制，所以加锁法因循环测试而造成的性能消耗会降低很多）

3.5.4 用P,V原语实现进程互斥

略

**3.6 进程同步**

3.6.1 同步的概念

直接制约：并发进程之间各自的执行结果为对方的执行条件

同步：直接制约进程之间通过彼此发送消息或事件来达到高效合作执行的状态，同步进程之间操作的执行一般互为前提条件

3.6.2 私用信号量private semaphore

即使用信号量的概念实现同步，互斥的实现中信号量称为公用信号量

3.6.3 用P,V原语操作实现同步

见例子：p61/78

·注意同步和互斥并不是矛盾的，二者可以一起实现，比如对于某资源必须进行互斥访问，且多个进程之间的执行彼此为对方的条件

3.6.4 生产者-消费者问题

一般化的进程同步互斥模型

·注意多个信号量P原语的顺序很重要，否则会造成死锁，但是V原语顺序一般任意

**3.7 进程通信**

进程之间的通信按照内容可以分为控制信息传送（也称低级通信）和大批量数据传送（高级通信）

3.7.1 进程的通信方式

4种：

主从式——终端控制进程和终端进程

会话式——使用进程和服务进程

消息或邮箱机制——发送和接收进程

共享存储区方式

3.7.2 消息缓冲机制

发送进程需要申请缓冲区来存放消息内容，然后将该缓冲区挂入消息队列

3.7.3 邮箱通信

不同于缓冲区是多个进程可以通用的数据类型，邮箱可以针对进程而自定义。且不必像缓冲区一样互斥制约，只需同步制约。

3.7.4 进程通信的实例——和控制台的通信

①KCP和DCP的动作

伪代码示例

②CCP和KCP及DCP的接口

略

③CCP与用户进程的接口

略

④CCP的动作

略

3.7.5 进程通信的实例——管道

①管道pipe

略

②示例

略

**3.8 死锁问题**

3.8.1 死锁的概念

①死锁的定义

简单来说就是不正确的同步或互斥产生的所有进程陷入彼此等待的状态

②死锁的起因

资源提供少于并发进程的需求

③产生死锁的必要条件

4个必要条件，只要一个不满足就可以避免死锁：p74/91

3.8.2 死锁的排除方法

分为预防、避免、检测与恢复3种。实用中，前两种难以实现且影响资源利用率，经常使用最后一种

①死锁预防

一些方法例子，但是都有开销或者无法全面解决问题的缺点

②死锁避免

介绍了一个死锁避免的理论，但是检测仍然需要耗费性能

③死锁的检测和恢复

检测技术：有限状态转移图、petriNet等

恢复方法：终止所有相关进程，或按照一定顺序终止；强迫剥夺资源

**3.9 线程的概念**

3.9.1 为什么要引入线程

进程的并行开销仍然很大，比如用于创建进程和切换上下文。实际上线程的引入就是为了替代多个子进程情况的使用，后者功能类似但性能开销大。

3.9.2 线程的基本概念thread

定义：也称轻权进程或轻量级进程light weight process，是进程的一部分

3.9.3 线程与进程的区别

线程也有线程控制表TCB，但是复杂程度远低于PCB，而且线程不像进程一样拥有自己的地址和资源空间，同一进程下的线程使用相同的资源。

进程是系统资源分配时的基本单位，线程则不然

3.9.4 线程的适用范围

线程必定适用在需要多进程调度和切换的环境下才能产生性能优势

3种典型应用：p79/96

**3.10 线程分类与执行**

3.10.1 线程的分类

用户级线程user level threads：管理过程全部由用户程序（或一个软件库）完成，系统只对进程进行管理。用户级线程的管理和使用全部在用户空间完成，且只使用用户堆栈和寄存器。

核心级线程kernel-level threads：由系统内核管理，用户程序需要使用内核提供的接口进行线程管理。这种线程也可以被系统程序使用。

·用户级线程执行效率最高，其次是系统级或核心级线程，最差是进程。不同的操作系统可能只提供用户级线程或系统级线程，也有二者皆支持的。这也说明了降低用户模式和内核模式的切换有助于提高效率。

3.10.2 线程的执行特性

线程只有3种基本状态：执行、就绪、阻塞。因为线程不会像进程一样挂起至外存

·线程也有同步问题，类似于进程的并发问题。

**第4章 处理机调度**

衡量调度策略的指标：

周转时间——即完成一个作业的时间

吞吐率——一定时间内，系统完成的总工作量

响应时间——即对一个命令的周转时间

设备利用率

**4.1 分级调度**

4.1.1 作业的状态及其转换

提交状态、收容状态（后备状态）、执行、完成

4.1.2 调度的层次

4级

作业调度——也称宏观或高级调度，分配资源、建立进程和回收资源的调度

交换调度——也称中级调度，内存和外存之间的调度

进程调度——微观或低级调度，进程占用处理机的调度

线程调度——

·分时系统和实时系统中不存在作业调度，因为要求响应时间很少，作业不会经过创建、收容等过程而是直接在内存建立

4.1.3 作业与进程的关系

作业包括至少一个根进程和若干子进程

**4.2 作业调度**

4.2.1 作业调度功能

作业调度功能的描述p88/105

·作业也有作业控制块JCB

4.2.2 作业调度目标与性能衡量

调度目标或标准：

公平合理

高利用率

吞吐量大

响应时间快

①周转时间

②带权周转时间

**4.3 进程调度**

4.3.1 进程调度的功能

①记录系统中所有进程的执行情况

基于PCB的读取和分析

②选择占有处理机的进程

③进行进程上下文切换

4.3.2 进程调度的时机

7种：

当前进程执行完毕

当前进程自行阻塞进入睡眠或等待

P,V原语的执行

提出I/O请求后

分时系统中时间片用完

系统调用完成后的切换

进程优先级变动（只有在可剥夺方式下工作的CPU才可以）

4.3.3 进程调度性能评价

定性：可靠性、简洁性

定量：CPU利用率，进程在就绪队列中的等待时间和执行时间比例

实用中：测试系统响应时间

**4.4 调度算法**

用于进程和作业调度的算法：

①先来先服务FCFS调度算法（进程或作业）

最基本和简单的算法也最直观，但是不单独使用

②轮转法（进程）

按照一定时间片进程轮流执行，必须是可剥夺资源才能使用此法，因此不适于作业调度

③多级反馈轮转法（进程）

基于优先级分类的轮转法，不同于优先级法的是每个进程可能会轮转多次执行

④优先级法（进程或作业）

静态优先级：脱机情况下设定的不可更改的优先级

动态优先级：也可以和静态优先级一起使用，在运行时根据情况不断更新和计算的优先级

线性优先级调度策略selfish round robin：一种动态优先级的调度算法，介于FCFS和轮转法之间的效果。p96/113

⑤最短作业优先法SJF（作业）

小规模作业获得最高优先权

⑥最高响应比优先法HRN（作业）

响应比：一个综合衡量执行时间和等待时间的量p97/114

·介于SJF和FCFS之间的作业调度算法

**4.5 算法评价**

4.5.1 FCFS方式的调度性能分析

·性能分析的推导非常隐晦和高深（也包含了各种近似假设），只求理解，需要借助运筹学及其分支——排队论的知识。

M/M/1系统：第一个M表示顾客到达时间间隔是指数分布，具有马尔可夫性质，第二个M表示离开的顾客的时间间隔也一样服从类似性质，第三个1表示只有一个服务器

Little’s Result：系统分析中的公式，系统进入稳定状态后系统中的顾客数n和平均响应时间R之间存在一个关系，n=λR，λ为顾客的平均到达率

4.5.2 轮转法调度性能评价

相较于FCFS，短作业的平均响应时间更短，但代价是早来的作业等待时间更长。

4.5.3 线性优先级法的调度性能

·书中公式印刷错误且混乱，不值得深究

结论：若实际服务时间和平均服务时间相同，那么3种调度方法的平均响应时间也相同。只有对于短作业，轮转法响应最快，长作业FCFS响应最快

**4.6 实时系统调度方法**

4.6.1 实时系统的特点

外部事件分类：

硬实时

软实时

外部任务分类：

周期性——无期限，周期性发生，且只是在一个周期内完成或开始处理

非周期性——有一个完成或开始处理的期限，无周期重复性

系统的5个特点：

有限等待时间（决定性）

有限响应时间

用户控制

可靠性高

系统出错处理能力强

实时系统的能力：

①很快的进程或线程切换速度

②快速的外部中断响应能力

③基于优先级的随时抢先式调度策略

基于优先级的调度策略总共有4种：p104/121

4.6.2 实时调度算法的分类

①静态表格驱动类

事先按照固定和设计好的规则生成调度结果，典型例子是最早时限优先法，即先调度时限早的作业

②静态优先级驱动抢先式调度算法类

也按照事先设计的规则进行分析，但是不直接作为结果而是生成优先级，如频率单调调度算法

③动态计划调度算法类

自行动态修改制定调度计划

④尽力而为调度算法类

不考虑时限的限制，只考虑优先级

4.6.3 时限调度算法与频率单调调度算法

（最早）时限调度算法的输入信息6种：

任务就绪时间或事件到达时间

开始时限

完成时限

处理时间

资源需求

优先级

频率单调调度算法：用于多周期性实时处理，周期越长的任务优先级越低。能够保证时限要求的频率单调调度算法的使用条件是——p107/124

**第5章 存储管理**

**5.1 存储管理的功能**

5.1.1 虚拟存储器

virtual store或virtual memory，即虚拟内存。这是简单的介绍概念与背景，实现方法等详细讨论见后文。虚拟内存在现在的pc操作系统中已经广泛使用，但是其导致的额外性能开销在一些嵌入式设备中仍难以应用。

·内存扩充技术有很多种（虽然都不是真正扩充了内存），虚拟内存在其中的特点是自动覆盖、内外存统一管理、进程大小不受内存容量限制。虚拟内存的实现依托多个子技术，首先就是地址重定位技术的一种——姑且称为虚拟空间/虚拟地址的重定位。其主要功能是将不连续的物理内存块映射为连续的虚拟内存地址。这种技术可以用于内存碎片整理（但是代价较高，使用实际内存地址重定位技术也可以实现碎片整理），也因为它使得拆分连续的程序地址成为可能，所以也使得将一部分不经常使用的内存内容置于外存成为可能。虽然支持不连续地址有助于扩充内存，但指令的整体连续存储可以减少指令地址计算的成本，这里需要权衡利弊。

5.1.2 地址变换

也称地址映射或地址重定位，地址重定位的概念不仅仅是虚拟内存中使用的。它泛指一切将地址变换的操作（可能是两个实际地址之间的映射，也可能有一个是虚拟的），这种操作在多种技术中都有使用。

虚拟空间的划分：按照程序模块的不同功能分属不同的区域，还不涉及地址变换p110/127

①静态地址重定位static address relocation

只是等于将程序整体移位，即只能装入连续的内存空间（即程序必须首先链接完毕），且之后直到执行完前不能更改。这种重定位不能实现虚拟内存。

②动态地址重定位dynamic address relocation

要依靠硬件地址变换机构完成，可以面对不连续的内存地址，也可以实现虚拟内存，和程序段共享。这是因为该技术将原本不能拆分的程序指令序列变得可以拆分。

5.1.3 内外存数据传输的控制

两种方式：

（1）用户控制——也称覆盖overlay，是一种早期的主存扩充技术，并不适合实现现代意义的虚拟内存

（2）操作系统控制——

①交换方式swapping：无差别式地替换所有的整个的非就绪进程，也能扩充内存

②请求调入on demand和预调入on prefetch：可以有效实现虚拟内存的特点

5.1.4 内存的分配与回收

包括5个部分（个人理解也可以加入碎片整理）：

（1）分配结构

（2）放置策略

（3）交换策略

（4）调入策略

（5）回收策略

5.1.5 内存信息的共享与保护

常用方法：

上下界保护法——硬件方法，通过界限寄存器设置能够访问的地址区间

保护键法——设置保护键，起到一定的“密钥”作用

界限寄存器和用户态与核心态结合的方式——略

**5.2 分区存储管理**

5.2.1 分区管理基本原理

含义：给内存中的每个进程划分一块存储区，以连续存储程序与数据，便于并发执行

两种方法：

（1）固定分区法

按照分区说明表，对内存区域进行固态划分，执行期间无法更改

（2）动态分区法

即动态分配内存的方法，这是最常用的。需要使用分区说明表、可用分区链表（直接建立在空闲内存的结构上）和资源请求表等数据结构辅助其功能

5.2.2 分区的分配与回收

（1）固定分区时的分配与回收

简单直接的算法p117/134

（2）动态分区时的分配与回收

功能包含3部分：

寻找空闲区回应请求

分配后更新可用链表

回收后更新可用链表

寻找空闲区的3种方法：

最先适应法first fit——要求按照起始地址递增的次序排列可用链表，选第一个合适的

最佳适应法best fit——按照从小到大的尺寸顺序排列可用链表，选第一个合适的

最坏适应法worst fit——正好与最佳适应法相反

（3）动态分区时的回收与拼接

略，只考虑了回收时拼接自然相邻的可用内存分区，且没有讨论最佳适应法中的回收拼接

（4）几种分配算法的比较

最先适应法在搜索速度（且无需排序）、回收速度上最优，且优先利用低地址空间。

最佳适应法名义上找到的空闲区是最优的，但是并不一定导致对内存利用率有利，因为可能会产生非常小的难以利用的内存碎片

最坏适应法正好与最佳法相反，名义上最差，但是缓解了最佳法的隐含问题

5.2.3 有关分区管理其他问题的讨论

（1）关于虚存的实现

分区式管理和虚拟内存没有关系，是两种独立技术。由于分区式管理的标准理念中没有将同一个进程中的程序段或数据拆分至不连续的内存空间（也因此无法交换至外存再任意换回），所以不能实现虚拟内存。

（2）关于内存扩充

略

（3）关于地址变换和内存保护

略

（4）分区存储管理的主要优缺点

P121/138

**5.3 覆盖与交换技术**

两种一个早期一个现代仍使用的内存扩充方法。

5.3.1 覆盖技术

将一个进程中不可能同时执行的程序段赋予共享的内存区域，执行时按照需要彼此进行覆盖处理，来达到缩减内存消耗的目的。

5.3.2 交换技术

类似但是不同于覆盖技术，交换针对于那些在内存中处于等待状态的进程，利用外存交换区将这些进程换出，并换入就绪状态的进程。交换不需只针对同一个进程的不同程序段，也不需要程序员提前定义好交换策略。

**5.4 页式管理**

5.4.1 页式管理的基本原理

页式管理比传统的分区式管理更加高级（虽然严格来说不是完全不同的技术，应该是一种升级），大致的概念就是将进程的虚拟空间和内存空间划分为等长的区域，虚拟空间中的区域称为页，内存空间中的区域称为片或页面page frame。页面一般长度不大，1-4K。不同于分区管理，每个进程不止占有一个区域，即同一个进程的内容可以被分配在不连续的内存空间中。

·页式管理仍然很难实现代码和程序段的进程间共享（也可以理解为动态链接、重用等），真正有效实现此目的的是段式管理。

5.4.2 静态页面管理

作业或进程执行之前，将所有程序段与数据装入内存并通过后述数据结构静态（不能随时更改）管理地址变换，且作业或进程大小仍然受到内存可用页面数的限制，即不能实现虚拟内存。

（1）内存页面分配与回收

3种用于管理的表结构：

①页表page mapping table——管理页号（虚拟地址空间）和页面号（内存空间）的映射的表，每个进程至少拥有一个

②请求表——管理进程的虚拟空间在内存中的对应位置，整个系统只需一张

③存储页面表——即标识内存中的各个页面分配情况的数据结构，可以使用位图法或空闲链表法，后者不占用额外空间

（2）分配算法

流程：p126/143

（3）地址变换

由硬件地址变换机构完成，原理：p126/143

·由于要访问内存两次以上来读取一个数据，页式管理的性能一般不高，只能借助硬件升级来优化

5.4.3 动态页式管理

原理：分为请求页式管理（遇到被访问内容处于外存时发生缺页中断进行请求）和预调入页式管理（提前进行计算和估计出外存中内容的访问情况）

用于动态页式管理的扩展页表：p128/145

动态页式管理流程图：p129/146

抖动thrashing：页面置换算法选用不当时，内存和外存中的内容频繁交替产生的现象

5.4.4 请求页式管理中的置换算法

常用的：

（1）随机淘汰算法——无法确定访问概率高低时，随机淘汰

（2）轮转法RR和先入先出FIFO

轮转法循环换出内存中的页，没有选择标准。

FIFO则总是选择内存驻留时间最长的进行淘汰。

·这两种算法都假设CPU访问顺序遵循线性规律，但是实际情况不然。有些驻留时间久的页面访问率仍然很高，有些近期访问的页遵循一定的动态特征，如果不考虑这些特征使用算法会导致性能不增反降。

（3）最近最久未使用页面置换算法least recently used，LRU

真正实现此算法需要大量的额外系统开销，因此常采用近似算法：p132/149

①最不经常使用页面淘汰算法LFU，least frequently used

②最近没有使用页面淘汰算法NUR

（4）理想型淘汰算法OPT，optional replacement algorithm

无法实现，必须预知进程的访问串，淘汰掉未来最远才会访问到的页

5.4.5 存储保护p132/149

·地址越界保护

·内存存取控制保护

5.4.6 页式管理的优缺点

p132/149

主要优点：减轻碎片问题，实现虚拟内存

主要缺点：依赖硬件支持，额外系统开销和置换算法实现的难度

**5.5 段式与段页式管理**

5.5.1 段式管理的基本思想

段式管理将程序按照内容关系分段（也可以理解为模块或模组），段长视需求而定，不像页式管理不顾内容均等分页。这样做的好处是便于对每一段进行特别的权限管理（共享等），也进一步提升了程序组织的灵活性（不一定非要静态链接成一维线性结构）。其余方面只是页式管理的升级或继承。

5.5.2 段式管理的实现原理

（1）段式虚存空间

将虚地址空间设计为二维结构，虽然看似和页式管理一样，但是页式管理中页号按照顺序排列，在虚地址空间中看来仍属一维。

（2）段式管理的内存分配与释放

使用前文中的相似策略，即分区式和页式管理

（3）段式管理的地址变换

①段表

使用方法类似于页表p135/152

②动态地址变换

类似，略

（4）段的共享与保护

①段的共享

道理很简单，只需要在每个进程的段表中纳入相同的段即可实现共享。共享段也遵循其他的段相似而不相同的规则（因为可能会被多个进程同时使用，对段的修改或不允许修改要有一定的策略和规则）。

②段的保护

地址越界保护和存取方式控制保护

5.5.3 段式管理的优缺点

基本解决前述方法的缺点，但是具备和分区式管理一样的问题——段的大小可变因而内存碎片问题较页式管理严重。

·虚拟内存的所有实现策略均以较大系统开销和需要硬件支持为代价。

5.5.4 段页式管理的基本思想

融合段式和页式管理的思想，以性能消耗为代价实现程序设计的方便和提高内存的利用率（结合CPU和内存的各自发展速度来看这是值得的）。

5.5.5 段页式管理的实现原理

（1）虚地址的构成

虚地址包含段号、页号和页内相对地址。页号和页内相对地址可以连起来解读为段内相对地址（只要每一页的长度选取2的幂）。从此段中的程序和数据逻辑上是连续的但是内存中可以分开存储。

（2）段表和页表

段页式管理中的表和内存的关系：p139/156

（3）动态地址变换过程

实质上，段页式管理其实就是将页式管理中的进程先分为段再分为页。这种关系类似于进程和线程的关系。

必须借助联想寄存器等硬件支持才能保证访问速度

**5.6 局部性原理和抖动问题**

局部性原理principle of locality：CPU总是趋向于集中访问程序的特定部分，而非均等概率访问所有内容。即便该特定部分是改变的，一般改变的速度也很慢

工作集：程序当前存于内存的部分，其他的部分暂置于外存。工作集的比例有一个临界值，超过临界值不会显著降低内外存交换次数，反之暴增

分析工作集与抖动关系的统计模型：p142/159

在系统性能一定的情况下，有两种方法解决抖动问题：

（1）扩大工作集

（2）选用不同的淘汰算法

**第6章 进程与存储管理示例**

**6.1 Linux进程和存储管理简介**

Linux核心主要分为两部分：

（1）文件系统

（2）进程控制系统——4个模块：与文件系统的接口、进程本身控制部分、进程间控制部分、存储管理部分

Linux系统的开机初始化过程：p146/163

系统自举：将系统内核程序装入内存的过程

0#进程：也称idle进程，Linux中唯一一个静态建立的进程，也是每次开机的第一个进程。只在核心态下运行。

init进程：也称1#进程，用于建立与控制终端进程及运行shell进程和初始化。Shell进程进而用于和用户交互以及创建所有用户进程。init及其子进程全部可运行于用户态和核心态。

核心线程kernel thread：只在核心态下执行的进程/线程，包括0#、kblockd、ksoftirqd、kswapd。可运行于不同状态或模式的进程要有两套上下文（堆栈等），分别用于不同的模式，因此这类进程的地址空间一般都有用户空间和核心空间的划分。然而核心线程只需一套上下文。

IPC，inter process communication：进程间通信模块，System V系统中的术语

**6.2 Linux进程结构**

6.2.1 进程的概念

·Linux没有作业概念，其PCB是一个叫做task\_struct的数据结构，这样的数据结构也称进程描述符。

·UNIX中的PCB分为常驻内存的proc和可交换外存的user两个结构

task\_struct的主要部分：p149/166

……

PID、UID、GID

fs\_struct

mm\_struct

……

6.2.2 进程的虚拟地址结构

Linux的进程由逻辑段组成，虚拟地址空间由“区”组成（对应于进程逻辑段，也类似于前文的“段”式管理，唯一的区别可能只在于地址变换的技术上）。虚拟区域名为vm area，核心代码中简写为VMA

每个区的描述符vm\_area\_struct的内容：p150/167

mmap()：请求创建一个虚拟区域的系统调用

munmap()：释放区域的系统调用

6.2.3 进程上下文

（1）进程上下文的基本结构

进程空间结构：p151/168

系统调用时用户栈和核心栈变化示例：p151/168

（2）进程上下文的组成部分

①task\_struct结构

②用户栈与核心栈内容

③用户地址空间的正文段

④用户地址空间的数据段

⑤硬件寄存器的内容

⑥页表

6.2.4 进程的状态和状态转换

Linux进程有5个状态：

①TASK\_RUNNING执行或就绪状态

②TASK\_INTERRUPTIBLE正在睡眠，可唤醒

③TASK\_UNINTERRUPTIBLE不可唤醒

④TASK\_STOPPED暂停状态，软中断

⑤TASK\_ZOMBIE系统调用exit后

6.2.5 小结

·进程上下文和状态转换刻画了Linux进程的静、动两种特性

**6.3 进程控制**

6.3.1 Linux启动及进程树的形成

Linux的启动过程：p154/171

6.3.2 进程控制

（1）进程的创建

·进程的创建使用系统调用fork

系统调用fork()过程的功能：p156/173

·在代码序列的层面，子进程在父进程的fork()调用刚结束后应该具有和父进程一致的进程上下文，因此二者应均处于执行完fork()指令的位置，父进程中fork()调用返回子进程的标识号，而子进程中返回0

写时复制copy on write：子进程刚创建时共享父进程的空间和数据结构，只有在其试图修改这些内容时才单独复制出一个拷贝，这样以提升效率。然而多数时候子进程将运行完全不同的程序。

（2）执行一个文件的调用

系统调用exec()：一般由父进程创建的子进程默认只能是父进程的部分拷贝，而如果需要让该进程运行其他程序，则需要exec()

·可见，虽然进程中包含程序代码，但是一个进程是可以替换其运行的程序的。当然也可以认为exec()重组了该进程的内容与结构。

exec()包含6种不同的调用格式，如execvp(filename, argp)，execlp(filename, arg0, arg1, …, (char \*) 0)等，功能一致，只是参数处理方法不一样。

（3）进程的终止

exit(rv)：调用进程自我终止，进入ZOMBIE状态，由父进程进行善后处理。exit调用释放除子进程的task\_struct结构之外的所有资源。父进程接收子进程的消息rv和时间信息，再将子进程的task\_struct结构释放，最后将时间信息插入到自己的task\_struct中的有关项里

**6.4 Linux进程调度**

Linux没有高级和中级调度，由核心过程schedule()实现调度，亦即调度过程实现于核心态

（1）调度原理

对于普通进程，采用基于时间片的动态优先数调度法

（2）调度的时机

①从核心态向用户态转换之前的瞬间，且系统在进程描述符中设置了need\_resched标识

②进程状态发生变化时，如sleep()和wait()和exit()

（3）调度标识的设置

①运行中的进程时间片耗尽时，need\_resched被系统设置1

②进程被唤醒，且优先级高于当前进程

（4）调度策略与优先数的计算

Linux进程分为实时进程和普通进程，前者优先级更高。调度策略分3种，2种用于实时，1种用于普通

调度策略选择使用sched\_setscheduler()系统调用

①动态优先数调度策略SCHED\_OTHER

简单的从内存就绪队列中选取最大优先级进程，每个新的子进程建立、时间片耗尽中断时更新优先级

通过goodness()函数计算优先级，计算方法书中内容似乎有错

②先来先服务调度SCHED\_FIFO

先来的进程一直运行直到结束或阻塞、或被高优先级抢占

③轮转法调度SCHED\_RR

略

（5）调度的实现

①进程选择——计算优先级，排列进程

②切换进程——用户进程的地址空间切换由switch\_mm()负责，进程的堆栈切换由switch\_to()

**6.5 进程通信**

分为3部分：低级通信、管道通信和IPC。额外的还有TCP/IP

6.5.1 Linux的低级通信

用来传递进程间的控制信号，如文件锁和软中断。Linux中有30个软中断信号（p162/179）和31个实时软中断信号

·软中断是对硬件中断的模拟，硬件中断在受到信号后立刻执行，但是软中断必须要在接收进程执行时才能生效。有些系统中，大部分陷阱（如系统调用）的实现就是当前进程给自己发送软中断信号，然后立即处理。

通过使用系统调用，用户进程可以使用低级通信，如p162/179

6.5.2 进程间通信IPC，inter\_process communication

Linux完整继承System V IPC，包含3个组成部分或子机制：

·消息message

·共享存储器shared memory

·信号量semaphore

3种机制的共同性质（6点）：p163/180

（1）消息机制

4个系统调用和用法示例p164/181

（2）共享存储区机制

4个系统调用和用法示例p167/184

（3）信号量机制

3个系统调用的简单解释p169/186

**6.6 Linux存储管理**

6.6.1 虚存空间和管理

虚存空间和管理依靠硬件支持，同时也是操作系统软件的任务一部分。

80x86与Linux的寻址和页表项内容：p171/188

6.6.2 请求调页技术

原理：p173/190

（1）交换缓冲

Linux不会将一个共享页面立刻交换出去，而是先存于缓冲中

（2）页面换出过程

Linux使用未占用的物理页作为系统的缓冲区和块设备的缓冲区。因此换出过程包括了先考虑缓冲区换出和最后一般的页换出两部分。

**第7章 Windows的进程与内存管理**

**7.1 Windows NT的特点及相关的概念**

7.1.1 Windows NT体系结构的特点

p177/194

7.1.2 Windows的管理机制

（1）核心态kernel mode和用户态user mode

核心态：操作系统的内核代码和设备驱动程序，所有程序共享同一系统地址空间

用户态：用户应用程序，不可以访问系统地址空间和其他用户进程地址空间

（2）Windows操作系统的体系结构

Windows的用户进程种类：p178/195

·用户态进程不仅仅包括用户应用程序，还有一些操作系统必要的支持程序与应用服务器程序

Windows的核心系统服务：p178/195

·不尽包括内核和驱动程序，还有窗口和图形系统

（3）系统调用、中断和陷阱

设备驱动程序——硬件中断

系统调用——软中断

（4）利用对象来共享系统资源

迫使用户进程使用对象和对象句柄来间接使用系统数据结构与进行系统调用等

（5）本地过程调用

含义：Windows为系统服务进程间进行高速消息传递设计的通信机制，没有用户态的调用接口

**7.2 Windows进程和线程**

7.2.1 Windows的进程和线程的定义

Windows中处理器调度的对象是线程，而进程至少包含一个线程以执行。线程可以创建新线程甚至进程，但是其本身一定是在所属进程的虚拟地址空间中运行。

Windows进程和线程的基本信息内容：p180/197

7.2.2 进程和线程的关联p181/198

进程和线程均分为“进程/线程环境块”和“执行进程/线程块”，两种块的从属关系遵循进程和线程的从属关系，环境块位于进程地址空间，执行块位于系统地址空间。

·执行块有时在概念上包含环境块而成为进程/线程的唯一代表。

·注意两种块都不是程序的内容，如指令和堆栈等，而是共同扮演PCB的角色。

7.2.3 Windows进程的结构p181/198

实则表征进程的数据结构，不包括程序的堆栈和指令序列，亦即执行块和环境块的内容。

7.2.4 Windows线程的结构p182/199

类似进程部分

7.2.5 Windows进程和线程的创建

（1）进程的创建过程p183/200

最常用的进程创建函数是CreateProcess，这是一个系统调用的封装

（2）Windows线程的创建过程p184/201

最常用的线程创建函数是CreateThread，也是一个系统调用

**7.3 Windows处理器调度机制**

基于线程调度和时间配额quantum的抢占式策略

7.3.1 调度优先级

Windows优先级的设置：p184/201

·优先级分为进程的基准优先级和所属线程的优先级两部分，调度时共同参考

7.3.2 线程状态

Windows线程状态转换图和说明：p185/202

7.3.3 线程调度机制

（1）调度数据库p186/203

每个优先级维护一个就绪线程队列

（2）时间配额

每个线程被分配一个固定的时间配额，用完后必须切换到其他最高优先级的线程

·时间配额记录在核心进程块

（3）调度算法p187/204

略，实为简述

·目测，Windows中低优先级的线程完全无法在高优先级线程就绪的状态下得到运行

（4）上下文切换p187/204

略

**7.4 Windows的内存管理**

不同版本的Windows操作系统支持的最大物理内存是不一样的，XP原版最大支持4GB物理内存，但Windows Server 2003就可以支持64GB物理内存。32位Windows定义4GB的虚拟内存空间，这个参数和物理内存大小没有关系

7.4.1 内存管理器

是Windows执行体的一部分，由一组运行于核心态的系统服务组成

7.4.2 内存管理的机制

（1）页

页的大小视计算机体系结构而定，x86体系结构的处理器一般4KB

预留机制：内存管理器为了应对连续地址页的分配请求所采用的的方式p188/205

（2）共享内存

拷贝后写入：共享资源在被修改时首先复制一个副本，并成为试图进行修改操作的进程的私有资源

（3）堆管理

除了内存管理器（最小粒度一页），还使用堆管理器进行更小粒度的分配

（4）系统内存池

用于对系统服务进行内存分配的机制，内存池分为可分页（可置换于磁盘，如设备驱动程序）和不可分页（常驻内存，如中断处理程序）两种。所有的核心态系统服务共享同一个系统地址空间，两种内存池都驻留在系统的虚拟地址空间的高端部分。

**7.5 虚拟地址空间**

7.5.1 虚拟地址空间布局

略，p190/207

·假设采用4GB虚拟地址空间，2GB给进程，余下给系统，以此为例。则每个进程均有2GB虚拟地址空间（地址从0开始到2GB处，低2GB位），而系统只有唯一的2GB地址空间（高2GB位）。但是同一时间只有一个进程在工作，因此实际上的工作空间就是4GB。切换进程上下文时，系统会自动更改进程空间以及在系统空间存储的和当前进程相关的信息。

7.5.2 虚拟地址转换p191/208

虚拟地址的页索引结构：p191/208

页表入口entry的结构（即页表项的结构）：p192/209

页表项中的状态位含义：p192/209

**7.6 页面调度**

7.6.1 缺页处理

有多种缺页的情况和分别的处理：p193/210

7.6.2 工作集及页面调度策略

Windows采用请求式簇调度策略，即每次缺页请求会导致局部若干相邻页一起被调入

换出策略Windows采用“最久未使用”

7.6.3 页框号和物理内存管理

页框page frame：即物理内存中和“页”同等大小的对等体

物理页状态种类和说明：p194/211

**第8章 文件系统**

**8.1 文件系统的概念**

（1）文件系统的引入

描述和文件系统任务

（2）文件与文件系统的概念

①文件

定义：一组赋名的相关联字符流的集合（也称无结构文件或流式文件，如源程序和目标代码），或相关联记录的集合（如数据库）

②文件系统

文件系统的特点：p199/216

（3）文件的分类

按性质和用途划分：

①系统文件

②库文件

③用户文件

按组织形式划分：

①普通文件

一般格式文件

②目录文件

检索普通文件的目录

③特殊文件

输入输出设备可看作是特殊文件

**8.2 文件的逻辑结构与存取方法**

8.2.1 逻辑结构

分为两类：字符流或无结构，记录式或有结构

选取文件逻辑结构应遵循的原则：p200/217

记录的含义：p200/217

注意这里的记录的概念非常宽泛，甚至可以包含多个关键字，每个关键字含有多个属性项

记录式结构文件的种类：

①连续结构

字符流式文件实际上是记录长度为1个字符的连续结构文件

②多重结构multi-list

多个关键字和多个记录之间的多对多关系

③转置结构inverted file

便于从关键字查找对应的记录

④顺序结构

将关键字按一定顺序排列的记录文件

8.2.2 存取方法

常用的存取方法：

（1）顺序存取法——按照文件的逻辑地址依次存取

（2）随机存取法（直接存取法）——按照编号和指针移动命令直接针对性存取

（3）按关键字存取法

常用的搜索方法：

①线性搜索法linear search

②散列法hash coding

解决散列冲突的内容详见《数据结构》

③二分搜索法binary search

**8.3 文件的物理结构与存储设备**

8.3.1 文件的物理结构

常用的文件物理结构：

（1）连续文件

最简单直接的存储方式，不支持动态修改

（2）串联文件

类似链表一样的存储方式，不适合随机存取

（3）索引文件

使用额外的索引表数据结构来记录物理结构分布，可以采用多级索引或部分多级索引来扩展其适用性。

8.3.2 文件存储设备

（1）顺序存取设备

如磁带p207/224

（2）直接存取设备

如磁盘p208/225

**8.4 文件存储空间管理**

文件系统由每块512字节或其倍数大小的逻辑块序列组成，同一个文件系统中逻辑块的大小是全局一致的，但是可以统一更改。块长越长有利于数据传输，但是会降低内存的存储管理性能。

3种空闲块管理方法：（类似内存管理策略）

（1）空闲文件目录

空闲区域编制成目录

（2）空闲块链

成组链法p209/226

（3）位示图

即使用位图表示文件存储设备的分配情况

**8.5 文件目录管理**

文件目录：文件名和对该文件实施控制管理的控制管理信息即为文件说明，也称文件的目录或文件控制块FCB

8.5.1 文件的组成

文件说明和文件体

·文件说明组成目录文件，实际上还包含多级目录之间的地址链接。文件说明信息的内容很复杂而且平台依赖性强，可以说除了文件体以外的所有文件相关信息都在其中，包括文件体的地址等。

8.5.2 文件目录

可分为单级、二级和多级目录

·单级目录不适于解决文件名冲突和搜索速度低

·多级目录即现今的文件夹体系结构

主目录MFD：包括所有用户和对应UFD相关信息的目录

用户文件目录UFD：每个用户所有的文件目录

8.5.3 便于共享的文件目录

3种方法实现文件共享：

（1）绕道法p214/231

（2）链接法p214/231

（3）基本文件目录表BFD方法p214/231

此方法也是一个文件目录管理办法，简单来说仍然是充分利用索引（和存储空间）来包装地址的技术。

BFD：除文件名之外的所有说明信息和对应标识符的表格

符号文件目录表SFD：文件名和对应标识符索引的表格

8.5.4 目录管理

打开文件和关闭文件实际上是一种文件系统的系统调用，具体的实际过程是将对应文件的所有目录表移入内存以便加速处理的过程，其他未打开文件和文件说明均在外存中。

活动名字表/活动文件目录表：处于内存中的所有文件说明信息和文件目录组成的表格

**8.6 文件存取控制**

文件存取控制包括文件共享、保护和保密3个问题。

存取控制部分/验证模块的任务：p216/233

存取控制验证模块用以验证用户存取操作的步骤：

·审定权限

·验证存取要求的一致性

·保密性验证

4种验证用户的存取操作的方法：

（1）存取控制矩阵

最简单但是低效的一一定义用户和文件的对应权限

（2）存取控制表

存取控制矩阵的升级版，将用户分组，每组组内权限一致

（3）口令方式

简单但是低效且安全性差的保密方式

（4）密码方式

加密解密技术

**8.7 文件的使用**

4类服务和相应的系统调用：p218/235

·系统调用的名称和参数因系统不同而异

**8.8 文件系统的层次模型**

文件系统层次模型架构介绍（分为8个功能层次）：p219/236

**第9章 设备管理**

**9.1 引言**

9.1.1 设备的类别

按使用特性分类：p224/241

按从属关系分类：

按信息组织方式分类：

字符设备——键盘、打印机

块设备——磁盘、磁带

9.1.2 设备管理的功能和任务

p224/241

**9.2 数据传送控制方式**

设备和内存之间的常用数据传送控制方式（4种）：

9.2.1 程序直接控制方式programmed direct control

由用户进程直接通过CPU和设备取得控制联系。

缺点：

·CPU只能和外设串行工作，不能并行

·容易出错，依赖硬件控制

9.2.2 中断方式

改进了直接控制方式，在外设准备期间CPU不必闲置

缺点：

·使用条件限制在外设较少，速度较低的情况下，反之CPU仍然效率不高，因为数据缓冲寄存器不大而需要CPU频繁切换处理中断程序

中断方式在这里只是代表一个原始的中断方式方法，更高级的方式仍然需要基于中断方式和其所依赖的硬件支持（如传输中断信号的中断请求线，和控制状态寄存器中的中断允许位）

9.2.3 DMA方式

也称直接存取direct memory access。使用一个新的独立的控制器来分担CPU的数据传送控制工作，从而只有在一开始和所有数据完全传送完毕至内存后才中断CPU

缺点：

·每个设备都至少需要一个DMA控制器（DMA控制器本身作为一个硬件，其本身的独立性、并行性和功能性仍然不高）

·多个DMA控制器的使用会产生冲突和并发处理问题

·仍然需要CPU的控制

9.2.4 通道控制方式channel control

简单来说，通道就是DMA的完全升级。通道本身就是一个独立于CPU管理的处理机硬件（有自己的指令集），它接管了更多CPU原本的数据传送控制任务，当然在开始和结束数据传送时仍然需要和CPU平等交互（由CPU发送通道指令，完毕后通道发送中断信号）。

通道的3种类型（信息交换方式不同）：

·字节多路通道——应对低速设备，如终端、打印机

·数组多路通道——应对中速设备，如磁带机

·选择单路通道——应对高速设备，如磁盘机

**9.3 中断技术**

9.3.1 中断的基本概念

中断源：引起中断发送的事件

中断请求：请求中断处理的信号

中断响应：CPU接收中断请求后的转事件处理程序过程

禁止中断：也称关中断，即PSW处理机状态字的中断允许位被清除，反之称为开中断。这是在CPU角度和层面上控制CPU是否可以接受中断的方式，依赖硬件实现

中断屏蔽：使用软件方式来达到部分禁止某些中断的效果

·有些中断请求不能被屏蔽或禁止，例如电源掉电

9.3.2 中断的分类与优先级

外中断：一般狭义上就简单称为中断。来自处理机和内存外部的中断，如设备、外部信号、定时器和调试中断等

内中断：处理机和内存内部产生的中断，也称陷阱trap。包括程序运算引起的各种错误，和用户核心模式切换等

·中断和陷阱有优先级，PSW也有，二者的比较结果决定了中断是否被允许。中断源和中断的优先级一般在系统设计时就已经固定，但是处理机和陷阱的优先级是系统程序动态设定的

中断和陷阱的其他区别：p233/250

9.3.3 软中断

定义：实质上就是一种进程间的信号通信方式，完全基于软件和程序。不像硬件中断依靠硬件产生中断信号。

·软硬中断和内外中断（陷阱和狭义中断）是独立分离的概念。比如陷阱可以实现为硬中断，也可以在部分系统中以软中断实现。外中断则一般都是硬中断。

9.3.4 中断处理过程

中断向量：中断源对应的处理机状态字PSW与中断处理子程序入口地址一起构成

**9.4 缓冲技术**

9.4.1 缓冲的引入

缓冲分为硬件缓冲，即使用专用的硬件如数据缓冲寄存器，或采用内存分配缓冲区，后者即为软件缓冲。

9.4.2 缓冲的种类

按缓冲器的个数划分：

单缓冲——简单直接，但因属于临界区且只有一个，则无法并行

双缓冲——两个缓冲区，但并非图形系统中的使用方式，仍然无法真正实现并行

多缓冲——多个缓冲区，分两组分别负责输入输出

缓冲池——多个缓冲区，统一管理既可输入又可输出

9.4.3 缓冲池的管理

（1）缓冲池的结构

缓冲池

缓冲区1

缓冲首部——存放缓冲区相关信息

缓冲体——存放缓冲数据

缓冲区2

系统维护3种缓冲队列：

空白缓冲队列em

输入缓冲队列in——装满输入数据

输出缓冲队列out——装满输出数据

4种工作缓冲区类型：p237/254

用于收容设备输入的收容输入缓冲区hin

用于提取设备输入的提取输入缓冲区sin

用于收容输出的缓冲区hout

用于提取输出的缓冲区sout

·工作缓冲区可以理解为一种缓冲区状态而非缓冲区的属性

（2）缓冲池管理

具体原理p237/254

**9.5 设备分配**

9.5.1 设备分配用数据结构

（1）设备控制表device control table，DCT

储存设备信息的表，每个设备一张，包括设备特性、相连I/O控制器的信息等

具体内容：p239/256

（2）系统设备表system device table，SDT

系统相关设备信息总表，每个系统一张，记录一些全局信息和DCT的指针

（3）控制器表controller control table，COCT

每个控制器一张，DMA方式下缺省

（4）通道控制表Channel Control table，CHCT

只用于通道控制方式，每个通道一张的信息表

9.5.2 设备分配的原则

（1）设备分配原则

·I/O操作的物理条件是：通道或DMA，控制器或DMA条件下缺省，所需设备。

设备分配有静态和动态两种，即先分配后不改和后分配看需要的常见区别。动态分配有更多灵活性，但是对于分配策略要求也高，否则会出现死锁。

（2）设备分配策略

①先请求先分配

具体过程：p242/259

②优先级高者先分配

具体过程：p242/259

9.5.3 设备分配算法

（奇怪没有很多内容）

**9.6 I/O进程控制**

9.6.1 I/O控制的引入

I/O控制：从响应用户进程的输入输出请求到分配资源和设备，最后在I/O完成后响应中断并善后处理的整个系统控制过程

9.6.2 I/O控制的功能

p243/260

9.6.3 I/O控制的实现

I/O控制在系统中的实现方式：

①作为请求I/O操作的进程的一部分

②作为当前进程的一部分

③由专门的系统进程单独完全负责

I/O独立进程的3种实现方式：

①每个或每类设备一个I/O进程，只在核心态执行

②只分一个输入进程和一个输出进程

③每个或每类设备一个I/O进程，可在用户或核心态执行

**9.7 设备驱动程序**

顾名思义就是驱使设备等硬件完成工作的程序集合。

设备开关表device switch table，DST：给出对应某一设备的各种操作子程序的入口地址

**第10章 Linux文件系统**

**10.1 Linux文件系统的特点与文件类别**

10.1.1 特点

·树型目录结构

·无结构字符流式文件

·文件可以动态消长

·文件可以被设置权限

·外部设备被视为文件

10.1.2 文件类型

·普通文件

·目录文件——文件系统中目录所形成的文件，和普通文件本质上类似，只是系统解释不同

·设备文件——同普通文件的形式类似，只是没有实际的物理存储块

·有名管道FIFO

·软链接

·Unix域套接字

**10.2 Linux的虚拟文件系统**

10.2.1 虚拟文件系统VFS框架

Virtual File System或Virtual File System Switch是Linux内核中的一个软件层，该层提供给用户单一的文件系统接口，但是却隐藏支持多种不同的文件系统。

VFS支持的文件系统种类：p249/266

10.2.2 Linux虚拟文件系统的数据结构

VFS使用一个通用文件模型，包含4个主要对象：

（1）VFS的超级块super\_block

存放已挂装文件系统的有关信息的主体，和相关的文件系统操作接口p250/267

·超级块一般保存在磁盘中，挂装时可读取至内存进行操作，但是两处所在的数据结构可能是不一样的。因此读取至内存可能实为重新构造对象，这是两处存储介质的不同需求导致的必然结果。

（2）索引节点inode

存放关于一个具体文件的一般信息p252/269

·索引节点是很多文件系统中常用的文件管理模式，它可以理解为就是采用占用模式（不占用时该空间仍存在为空闲，且只能被占用为固定用途）管理的文件控制块FCB，当然实用中的情况和普遍原理之间总有差别，尤其是概念和定义上的差别。也可以说inode和dentry共同组成了FCB，因为二者都包含了FCB的部分定义。

·索引节点也和超级块一样有内存和磁盘两个版本，也有二者之间同步更新的需求。

（3）文件file

存放打开文件与进程之间进行交互的有关信息p254/271

·文件对象里包含文件位置指针，而不是存放于索引节点对象中。一个系统能够同时打开的文件数目是固定的。文件对象和文件一一对应，且也可被多个进程同时访问。

·文件和目录项对象都是在内存中于运行时创建的，没有磁盘副本。

（4）目录项dentry

保存目录项与相应文件进行链接的信息p255/272

·为了提高处理目录项的效率，Linux 采用目录项高速缓存dentry cache和对应的索引节点高速缓存inode cache

10.2.3 VFS的系统调用

经过VFS隐藏封装后，Linux 的统一文件系统调用：p256/273

**10.3 文件系统的注册和挂装**

10.3.1 文件系统注册

已注册文件系统的描述符是file\_system\_type类型数据结构

10.3.2 已挂装文件系统描述符链表

根文件系统：某个文件系统的根目录是系统文件树的根目录

已挂装文件系统描述符是vfsmount类型的数据结构

10.3.3 挂装根文件系统

过程：p259/276

·Linux的名义根文件系统是rootfs，它仅提供一个作为初始安装点的空目录。这个设计便于Linux在某些情况下更改系统的实用根文件系统。

10.3.4 挂装一般文件系统

过程：p260/277

10.3.5 卸载文件系统

p260/277

**10.4 进程与文件系统的联系**

10.4.1 系统打开文件表

即file类型的总链表，记录所有打开的文件

文件描述符：一个小整数，作为文件id和句柄，关联到打开文件对象的指针和其他相关参数

10.4.2 用户打开文件表

files\_struct数据结构用以存储进程打开文件表，位于进程描述符，记录一个进程所有的打开文件

操作过程：p261/278

10.4.3 进程的当前目录和根目录

一个进程描述符中会记录进程的绝对路径和相对路径以及相关的对象指针。

·域：本书中多次提到“域”这个概念，感觉类似于命名空间，但是书中源代码并未使用命名空间而是直接采用规范化的命名方式，将域名置前并用下划线和后续名称分开。

**10.5 ext2文件系统**

10.5.1 ext2文件系统的存储结构

详见p262/280

·ext2文件系统相对于传统UNIX文件系统做出了很多改进

块组block group：多个连续的块或从磁盘角度看多个连续的柱面组成的一个存储空间块，将原来的磁盘分区进一步划分以及各自存储各组的相关信息（索引节点表等）可以提高访问效率

·每个块组首部均备份全局的文件系统的综合超级块，随后是该组自身的相关信息，如索引节点和数据信息等。

引导块boot block：引导和初启操作系统的代码所在处，是每个磁盘分区的首部或第0号块

10.5.2 ext2文件系统主要的磁盘数据结构

（1）ext2文件系统的磁盘超级块ext2\_super\_block

ext2文件系统中每个块组都保存一个文件系统的综合信息，即超级块在首部

数据结构：p264/280

（2）ext2的块组描述符

数据结构：p264/280

（3）块位图和索引节点位图

描述组内块和索引节点的占用情况

（4）ext2文件系统的磁盘索引节点ext2\_inode

数据结构：p265/282

10.5.3 ext2文件系统的内存数据结构

·文件系统的磁盘组织和内存组织并不一定相同，如ext2

（1）ext2的内存超级块ext2\_sb\_info

略

（2）ext2的内存索引节点ext2\_inode\_info

略

10.5.4 数据块寻址p267/285

简而言之，还是使用多级数组表达较大数目元素的方式，而且是每一级都使用以达到从最小情况到最大情况均最优的适用性。

·ext2文件系统的文件大小上限为2TB-4KB，块大小为4096bit情况下。

**10.6 块设备驱动**

Linux系统有两类设备，块设备与字符设备，本节以块设备为例。VFS会隐藏所有设备细节，将设备文件的访问做成和普通文件一致。

10.6.1 设备设置

设备文件的手动加载和命令：p269/286

10.6.2 设备驱动程序的接口

驱动程序将一个逻辑设备号和块号组成的文件系统地址转换为物理设备上特定的物理块号。驱动程序有两个接口，和文件系统的接口以及与硬件的接口。

文件系统或内核将每个针对设备文件的系统调用转换为相应设备驱动程序的对应函数调用。这显然需要“注册”设备驱动程序模块，如果该模块不是内核自带的一部分。设备文件的类型一般可以在索引节点中查找到。

·设备文件即使使用方法上尽可能做到了与普通文件一致，但是仍然有显著差别，比如有些设备文件不能被多个进程同时“打开”，即占用。

·设备驱动程序本身也包含有中断处理程序用以处理设备中断。

**10.7 字符设备驱动**

略，基本类似于块设备驱动

**第11章 Windows的设备管理和文件系统**

**11.1 Windows I/O系统的结构**

11.1.1 设计目标

略，p272/289

11.1.2 设备管理服务

Windows设备管理体系结构：p273/290

I/O管理器I/O manager

即插即用管理器PnP manager

电源管理器power manager

设备驱动程序device driver

注册表registry和INF文件

硬件抽象层hardware abstraction layer，HAL

**11.2 设备驱动程序和I/O处理**

11.2.1 设备驱动类型和结构

3类驱动：

文件系统驱动

即插即用设备驱动

非即插即用设备驱动——网络接口和协议等

典型驱动程序包含的例程（也叫函数和方法或子程序）：

初始化、设备加入、调度、I/O起始、中断服务、中断服务延迟过程调用、终止、调出

11.2.2 Windows的I/O处理

（1）I/O处理的类型

同步I/O和异步I/O

快速I/O

映射文件I/O

集中式I/O

（2）单层驱动的I/O处理

p275/292

（3）多层驱动的I/O处理

p276/293

**11.3 Windows的文件系统**

Windows文件系统由文件系统驱动和磁盘驱动构成，即文件系统由设备驱动形式实现。某些意义上来说和Linux相反。

11.3.1 Windows磁盘管理

扇区（硬盘扇区一般为512B，CD-ROM为2048B）→（相邻组成）→分区→（若干个组成）→卷

11.3.2 Windows文件系统格式

（1）CDFS和UDF

CDFS：即CD-ROM文件系统，支持CD-ROM文件的只读文件系统，最大文件限定4GB，最多65535个目录

UDF：即通用磁盘格式文件系统，提供对DVD文件的支持

（2）FAT

FAT：文件分配表文件系统，以簇为单位进行分配，是早期的DOS操作系统的简单文件系统，包括FAT12，FAT16，FAT32等版本。

（3）NTFS

NTFS：NT文件系统，WindowsNT体系结构的主流文件系统

11.3.3 Windows文件系统驱动

p278/295

**11.4 NTFS文件系统**

11.4.1 NTFS的特点

（1）可恢复性

（2）安全性

（3）冗余与容错

（4）动态坏簇重映像

（5）基于统一字符编码UNICODE的命名机制

（6）文件压缩

（7）加密

11.4.2 NTFS的磁盘结构

建立文件系统时需要对磁盘结构进行规划，是为格式化

（1）卷

磁盘逻辑分区的基本单位

（2）簇cluster

卷上固定大小的一部分磁盘空间，一般由偶数个物理扇区组成。簇的大小在格式化时确定，用于代替扇区的概念，为了能够将文件系统独立于不同机器的物理扇区。

逻辑簇号logical cluster number：从卷的开始到结束的顺序编号

虚拟簇号virtual cluster number：从文件的开始到结束的顺序编号

（3）文件

NTFS文件系统 中的所有磁盘数据均保存在文件中，或以文件形式存放，包括文件系统的数据结构。文件系统的相关信息称为系统结构文件，名称一般由$开头，其他的一般文件称为数据文件。

主控文件表MFT：NTFS的核心，由文件记录构成，每个文件记录对应一个文件。MFT本身也是文件。

（4）目录

文件目录也是主控文件表中的文件记录，区别是文件目录的数据包含的是该目录下所有文件名和子目录名的索引记录。p280/298

11.4.3 NTFS的文件系统恢复

类似于数据库的恢复