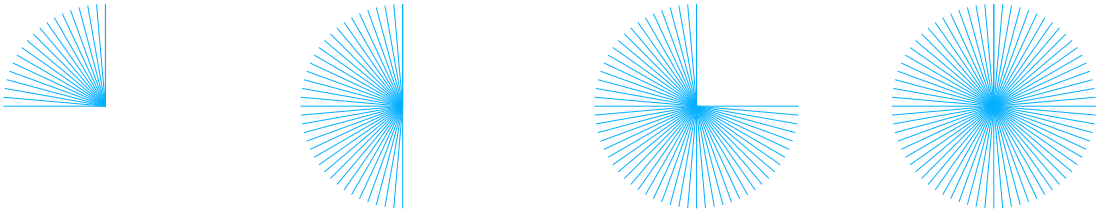
开课吧

计算机硕士

「高频考点」

操作系统



考研

目　录

第一章 计算机系统概论

操作系统的基本概念.....1
操作系统的发展与分类.....1
操作系统的运行环境.....2
操作系统的结构设计（22 考研新加）.....3
操作系统结构（外核）（22 考研新加）.....4
操作系统引导（22 考研新加）.....4
虚拟机（22 考研新加）.....5

第二章 进程管理

进程与线程.....6
处理器调度.....10
进程同步.....15
死锁.....18
线程的状态与转换(22 考研新加).....21
调度程序(22 考研新加).....21
闲逛进程(22 考研新加).....22
内核级线程和用户级线程调度(22 考研新加).....22
上下文及其切换机制(22 考研新加).....22

第三章 内存管理

内存管理概念.....23
连续分配管理方式.....23
非连续分配管理方式.....24
虚拟内存管理.....30
内存映射文件(22 考研新加).....33
虚拟存储器性能的影响因素及改进方法(22 考研新加).....33

第四章 文件管理

文件系统基础.....35
文件系统实现.....37

磁盘组织与管理.....40

第五章 输入输出管理

I/O 管理概述.....43
I/O 核心子系统.....45
输入输出应用程序接口（22 考研新加）.....48
阻塞/非阻塞 I/O（22 考研新加）.....49
缓冲区管理（22 考研新加）.....49
设备驱动程序接口（22 考研新加）.....50
固态硬盘（22 考研新加）.....50

编写人：宿叶露　审核人：于方泽　封面设计：乔宇、刘锦浩、郑悦

勘误信箱:

亲爱的开课吧学员你好，感谢您选择开课吧开启学习之旅。在使用这份学习资料时，希望你能感受到开课吧全体教职人员的用心。经过软件人工三审三校的资料难免有万分之三的概率出现小差错，我们设立勘误激励邮箱，感谢大家和小开共同成长。在使用本资料的同时，如您火眼金睛发现了小错误。可将资料的名称-页码带图反馈到本邮箱:

kaikeba0104@126.com。一经采纳小开感激不尽。

第一章 计算机系统概论

操作系统的基本概念

操作系统的概念

操作系统是计算机系统中最基本的系统软件。

从操作系统的角度出发，计算机系统自下而上可分为四个部分：硬件、操作系统、应用软件和用户。

操作系统的特征

1.并发

并发是指两个或多个时间在同一时间间隔发生。

计算机具备处理和调度多个程序同时执行的能力。

与并发相关的另一个概念是并行。并行是指在同一时刻有两个或多个程序执行，这和并发的同一时间间隔是有明确不同的。

并发的通过分时实现的，而并行依赖于多个处理器。

2.共享

共享即资源共享，是指操作系统中多个并发执行的过程可以使用共同的资源的这一特征，共享有两种方式：

a）互斥共享方式

b）同时访问方式

3.虚拟

在操作系统中，虚拟是把一个物理的实体变为若干逻辑上的对应物。前者是实的，是实际存在的；后者是虚的，是用户感觉到的。

使用多道程序设计技术实现的虚拟处理器让用户（任务）感觉自己在独占一个 CPU；使用虚拟存储器技术实现的虚拟内存让用户（任务）觉得自己独占整个内存。

虚拟的本质可归纳为：时分复用技术，空分复用技术。

4.异步

在多道程序环境中，由于资源的限制，系统中的每道程序何时运行，多道程序的执行顺序，及完成的顺序都是未知的，这个叫异步。

操作系统的发展与分类

1.手工操作系统

特点：所有操作都需要人工干预，人机矛盾（速度和资源利用）越来越大

缺点：用户独占全机；CPU 需要等待用户，利用率低

迫切需要：告诉的机器代替人工操作的任务排序。

2.批处理系统

a)单道批处理

特点：自动性、顺序性、单道性

缺点：高速 CPU 会经常等待低速 I/O 来完成相应操作

迫切需要：提高系统的资源利用率和系统吞吐量，利用好所有时间

b)多道批处理

特点：多道、宏观上并行、微观上串行

优点：资源利用率高，系统吞吐率高

缺点：用户响应时间长；不提供人机交互能力，用户不知程序的运行情况，也不能控制计算机

迫切解决：减少用户响应时间，增加人机交互能力

3.分时操作系统

特点：同时性、交互性、独立性、及时性

优点：人机交互能力强，用户响应较为即使，系统资源利用率高，吞吐率高

缺点：有些特殊场景，无法接受分时操作系统的时间片，需要更快做出响应

迫切解决：对紧急，重要事件的实时响应

4.实时操作系统

特点：及时性、可靠性

分类：硬实时系统、软实时系统

5.网络操作系统和分布式操作系统

a)网络操作系统

特点：网络中各种资源共享及各台计算机之间通信

b)分布式操作系统

特点：分布式、并行性

两者之间的本质不同是：分布式操作系统中的若干计算机互相协同完成同一任务

6.个人操作系统

个人操作系统是目前适用最广泛的操作系统。

操作系统的运行环境

操作系统的运行机制

特权指令是只能由操作系统内核部分使用，不允许用户直接执行的指令。

特权指令包括：I/O 指令、置中断指令、存取用于内存保护的寄存器、送程序状态字到程序状态字寄存器的指令等。

CPU 状态划分为：核心态（管态、内核态）、用户态

操作系统内核包含的 4 方面内容是:

（1）时钟管理

（2）中断机制

（3）原语

原语是一些关闭中断的公用小程序，主要有以下特点。

a)处于操作系统最底层，是最接近硬件的部分。

b)程序运行具有原子性，操作只能一气呵成。

c)这些程序的运行时间较短，调用频繁。

（4）系统控制的数据结构及处理

中断和异常的概念（区别）

中断，也称外中断，是系统正常功能的一部分，例如，因进程调度使系统停止当前运行的进程转而执行其他进程，或者因缺少所需资源而中断当前操作等待资源到达，在系统处理完其他事情之后，会继续执行中断前的进程。

异常，也称内中断，是由错误引起的，如文件损坏、进程越界等。通常异常会引起中断，而中断未必是由异常引起的。

系统调用

用户需要执行系统调用时，首先准备并传递系统调用所需的参数，通过陷入（trap）指令进入操作系统的系统内核，此时将从用户态进入内核态；之后执行相应的系统调用函数，使用特定的系统内核功能；最后将处理结果返回给用户进程，此时将从内核态返回用户态。

操作系统的结构设计（22 考研新加）

1. 无结构操作系统

OS 是由众多的过程直接构成，各过程之间可相互调用，但 OS 内部不存在任何结构，又称为整体系结构。

缺点:既庞大又杂乱，缺乏清晰的程序结构;程序错误多，调试难、阅读难、理解难、维护难。

2. 模块化 OS 结构

OS 是采用“模块化程序设计”技术，按其功能划分为若干个独立的模块，管理相应的功能，同时有:规定好各模块之间的接口，以实现其交互，对较大模块又可按子功能进一步细分下去。福开侧三

优点:提高了 OS 设计的正确性、可理解性和可维护性;增强了 OS 的可适用性;加速了 OS 的开发过程。

缺点:模块及接口划分较困难;未区别共享资源和独占资源;由于管理的差异，使 OS 结构变得不够清晰。

3.分层式 0S 结构

分层式 0S 结构是对模块化结构的种改进， 它按分层式结构设计的基本原则，将 OS 划分为若干个层次，每一层都只能使用其底层所提供的功能和服务，从硬件 开始，在其上面一层一层地自底向上增添相

应功能的软件，这种 OS 结构称为分层式 OS 结构。

特点:每一步设计都建立在可靠的基础上，结构更清晰;调试和验证更容易，正确性更高。

4.微内核 OS 结构

在 0S 内核中只留下一一些最基本的功能，而将其他服务分离出去，由工作在用户态下的进程来实现。

特点:微内核便于扩充操作系统，拥有很好的可移植性。

5.宏内核 OS 结构

在宏内核架构中，用户服务和内核服务在同一空间中实现。内核可以代表内核进程运行代码，就是通常的内核进程;当用户进程经过系统调用或者中断进人内核态后，内核也可以代表它运行代码。因此，宏内核需要管理的资源多于微内核，其大小相对大一些。

宏内核与微内核的区别:微内核主要提供进程间通信(IPC)，自身完成很少功能，主要是传递一个模块对另一个模块的功能请求;而宏内核则是把内存管理、文件管理等全部接管。从理论上来看，微内核的思想更好些，微内核把系统分为各个小的功能块，降低了设计难度，系统的维护与修改也容易，但通信带来的效率损失是个问题;宏内核的功能块之间的耦合度太高，造成修改与维护的代价太高，但宏内核因为是直接调用，所以效率是比较高的。

操作系统结构（外核）（22 考研新加）

外核架构可为不同的应用提供定制化的高效资源管理，按照不同应用领域的要求，将对硬件资源的抽象模块化为一.系列的库(即 LibOS)。这样的设计带来两个主要好处:可按照应用领域的特点与需求，动态组装成最适合该应用领域的 LibOS,最小化非必要的代码，从而获得更高的性能:处于硬件特权级的操作系统内核可以做到非常小，并且由于多个 LibOS 之间的强隔离性，从而可以提升整个计算机系统的安全性与可靠性。

外核架构的劣势在于:LibOS 通常是为某种应用定制的，缺乏跨场景的通用性，应用生态差，较难用于功能要求复杂、生态与接口丰富的场景

操作系统引导（22 考研新加）

1.内核运作环境进行预先检测。

2.它首先按照系统 CMOS 设置中保存的启动顺序搜索磁盘驱动器、CD-ROM、网络服务器等有效的驱动器，读人操作系统引导程序，接着将系统控制权交给引导程序，并由引导程序装人内核代码，以便完成系统的顺序启动。

3.引导程序将 CPU 控制权交给内核，此时内核才可以开始运行。内核将首先完成初始化功能，包括对硬件、电路逻辑等的初始化，以及对内核数据结构的初始化，如页表(段表)等。

4.最后要做的就是启动用户接口，使系统处于等待命令输人状态即可。

虚拟机（22 考研新加）

虚拟机(Virtual Machine)指通过软件模拟的具有完整硬件系统功能的运行在 一个完全隔离环境中的完整计算机系统。每个虚拟机都有独立的 CMOS、硬盘和操作系统。

Java 语言的一个非常重要的特点就是与平台的无关性。而引人 Java 虚拟机后，Java 语言在不同平台上运行时不需要重新编译。Java 语言使用 Java 虚拟机屏蔽了与具体平台相关的信息，使得 Java 语言编译程序只需生成在 Java 虚拟机上运行的目标代码(字节码)。

第二章 进程管理

进程与线程

进程的概念和特征

1.进程的概念

进程（Process） 以便更好地描述和控制程序的并发执行，实现操作系统的并发性和共享性（最基本的两个特性）。

注意： PCB 是进程存在的唯一标志!

从不同的角度，进程可以有不同的定义，比较典型的定义有：

1）进程是程序的一次执行过程。

2）进程是一个程序及其数据在处理机上顺序执行时所发生的活动。

3）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，它是系统进行资源分配和调度的一个独立单位。

2.进程的特点

动态性 ②并发性 ③独立性 ④异步性 ⑤结构性。

进程的状态与转换

通常进程有以下五种状态，前三种是进程的基本状态。

① 运行态。②就绪态。③阻塞态。④创建态。⑤结束态。

图 2.1 1 种进程状态的转换

进程的控制

1.进程的创建

允许一个进程创建另一个进程。此时创建者称为父进程，被创建的进程称为子进程。子进程可以继承父进程所拥有的资源。当子进程被撤销时，应将其从父进程那里获得的资源归还给父进程。此外，在撤销父进程时，必须同时撤销其所有的子进程。

在操作系统中，终端用户登录系统、作业调度、系统提供服务、用户程序的应用请求等都会引起进程的创建。操作系统创建一个新进程的过程如下（创建原语）：

1）为新进程分配一个唯一的进程标识号， 并申请一个空白的 PCB （PCB 是有限的）。若 PCB 申请失败，则创建失败。

2）为进程分配资源，为新进程的程序和数据及用户栈分配必要的内存空间（在 PCB 中体现）。注意，若资源不足（如内存空间），则并不是创建失败，而是处于阻塞态，等待内存资源。

3）初始化 PCB，主要包括初始化标志信息、初始化处理机状态信息和初始化处理机控制信息，以及设置进程的优先级等。

4）若进程就绪队列能够接纳新进程，则将新进程插入就绪队列，等待被调度运行。

2.进程的终止

引起进程终止的事件主要有：①正常结束，表示进程的任务已完成并准备退出运行。②异常结束，表示进程在运行时，发生了某种异常事件，使程序无法继续运行，如存储区越界、保护错、非法指令、特权指令错、运行超时、算术运算错、I/O 故障等。③外界干预，指进程应外界的请求而终止运行，如操作员或操作系统干预、父进程请求和父进程终止。

操作系统终止进程的过程如下（撤销原语）：

1）根据被终止进程的标识符，检索 PCB，从中读出该进程的状态。

2）若被终止进程处于执行状态，立即终止该进程的执行，将处理机资源分配给其他进程。

3）若该进程还有子孙进程，则应将其所有子孙进程终止。

4）将该进程所拥有的全部资源，或归还给其父进程，或归还给操作系统。

5）将该 PCB 从所在队列（链表）中删除。

3.进程的阻塞和唤醒

正在执行的进程，由于期待的某些事件未发生，如请求系统资源失败、等待某种操作的完成、新数据尚未到达或无新工作可做等，由系统自动执行阻塞原语（Block），使自己由运行态变为阻塞态。可见，进程的阻塞是进程自身的一种主动行为，也因此只有处于运行态的进程（获得 CPU），才可能将其转为阻塞态。阻塞原语的执行过程如下：

1）找到将要被阻塞进程的标识号对应的 PCB。

2）若该进程为运行态，则保护其现场，将其状态转为阻塞态，停止运行。

3）把该 PCB 插入相应事件的等待队列，将处理机资源调度给其他就绪进程。

当被阻塞进程所期待的事件出现时，如它所启动的 I/O 操作已完成或其所期待的数据已到达，由有关进程（比如，释放该 I/O 设备的进程，或提供数据的进程）调用唤醒原语（Wakeup），将等待该事件的进程唤醒。唤醒原语的执行过程如下：

1）在该事件的等待队列中找到相应进程的 PCB。

2）将其从等待队列中移出，并置其状态为就绪态。

3）把该 PCB 插入就绪队列，等待调度程序调度。

4.进程切换

进程切换是指处理机从一个进程的运行转到另一个进程上运行，在这个过程中，进程的运行环境产生了实质性的变化。进程切换的过程如下：

1）保存处理机上下文，包括程序计数器和其他寄存器。

2）更新 PCB 信息。

3）把进程的 PCB 移入相应的队列，如就绪、在某事件阻塞等队列。

4）选择另一个进程执行，并更新其 PCB。

5）更新内存管理的数据结构。

6）恢复处理机上下文。

进程的组织

进程是一个独立的运行单位，也是操作系统进行资源分配和调度的基本单位。它由以下三部分组成，其中最核心的是进程控制块（PCB）。

1.进程控制块

进程创建时，操作系统为它新建一个 PCB，该结构之后常驻内存，任意时刻都可以存取，并在进程结束时删除。PCB 是进程实体的一部分， 是进程存在的唯一标志。

进程执行时，系统通过其 PCB 了解进程的现行状态信息，以便对其进行控制和管理；进程结束时，系统收回其 PCB，该进程随之消亡。操作系统通过 PCB 表来管理和控制进程。

下表是一个 PCB 的实例。PCB 主要包括进程描述信息、进程控制和管理信息、资源分配清单和处理机相关信息等。各部分的主要说明如下：

表 2.1 PCB 通常包含的内容

进程描述信息	**进程控制和管理信息**	**资源分配清单**	**处理机相关信息**
进程标识符（PID）	进程当前信息	代码段指针	通用寄存器值
用户标识符（UID）	进程优先级	数据段指针	地址寄存器值
	代码运行入口地址	堆栈段指针	控制寄存器值
	程序的外存地址	文件描述符	标志寄存器值
	进入内存的时间	键盘	状态字
	处理机占用时间	鼠标	
	信号量使用		

2.程序段

程序段就是能被进程调度程序调度到 CPU 执行的程序代码段。注意，程序可被多个进程共享，即多个进程可以运行同一个程序。

3.数据段

一个进程的数据段，可以是进程对应的程序加工处理的原始数据，也可以是程序执行时产生的中间或最终结果。

进程的通信

进程通信是指进程之间的信息交换。进程的互斥与同步就是一种进程间的通信方式。相应地，也可以将 P、V 原语称为两条低级进程通信原语。

目前，高级进程通信方式可以分为 3 大类：共享存储器系统、消息传递系统和管道通信系统。

1.共享存储

为了传输大量数据，在存储器中划出一块共享存储区，多个进程可以通过对共享存储区进行读写来实现通信。

2.消息传递

在消息传递系统中，进程间以消息为单位交换数据，用户直接利用系统提供的一组通信命令（原语）来实现通信。

消息传递系统可以分为以下两类。

1）直接通信方式。

2）间接通信方式。

3.管道通信

管道是用于连接读进程和写进程以实现它们之间通信的共享文件。

线程概念和多线程模型

1.线程的基本概念

引入线程的目的则是为了减小程序在并发执行时所付出的时空开销，提高操作系统的并发性能。

线程也有就绪、阻塞和运行三种基本状态。

引入线程后，进程的内涵发生了改变，进程只作为除 CPU 外的系统资源的分配单元，而线程则作为处理机的分配单元。

2.线程与进程的比较

（1）调度。在传统的操作系统中，拥有资源和独立调度的基本单位都是进程。在引入线程的操作系统中，线程是独立调度的基本单位，进程是拥有资源的基本单位。在同一进程中，线程的切换不会引起进程切换。在不同进程中进行线程切换，如从一个进程内的线程切换到另一个进程中的线程时，会引起进程切换。

（2）拥有资源。不论是传统操作系统还是设有线程的操作系统，进程都是拥有资源的基本单位，而线程不拥有系统资源（也有一点儿必不可少的资源），但线程可以访问其隶属进程的系统资源。

（3）并发性。在引入线程的操作系统中，不仅进程之间可以并发执行，而且多个线程之间也可以并发执行，从而使操作系统具有更好的并发性，提高了系统的吞吐量。

(4) 系统开销。由于创建或撤销进程时，系统都要为之分配或回收资源，如内存空间、I/O 设备等，因此操作系统所付出的开销远大于创建或撤销线程时的开销。

(5) 地址空间和其他资源（如打开的文件）。进程的地址空间之间互相独立，同一进程的各线程间共享进程的资源，某进程内的线程对于其他进程不可见。

(6) 通信方面。进程间通信（IPC）需要进程同步和互斥手段的辅助，以保证数据的一致性，而线程间可以直接读/写进程数据段（如全局变量）来进行通信。

3.线程属性

多线程操作系统把线程作为独立运行（或调度）的基本单位，此时的进程已不再是一个基本的可执行实体，但它仍具有与执行相关的状态。所谓进程处于“执行”状态，实际上是指该进程中的某线程正在执行。线程的主要属性如下：

(1) 线程是一个轻型实体，它不拥有系统资源，但每个线程都应有一个唯一的标识符和一个线程控制块，线程控制块记录了线程执行的寄存器和栈等现场状态。

(2) 不同的线程可以执行相同的程序，即同一个服务程序被不同的用户调用时，操作系统把它们创建成不同的线程。

(3) 同一进程中的各个线程共享该进程所拥有的资源。

(4) 线程是处理机的独立调度单位，多个线程是可以并发执行的。在单 CPU 的计算机系统中，各线程可交替地占用 CPU；在多 CPU 的计算机系统中，各线程可同时占用不同的 CPU，若各个 CPU 同时为一个进程内的各线程服务，则可缩短进程的处理时间。

(5) 一个线程被创建后，便开始了它的生命周期，直至终止。线程在生命周期内会经历阻塞态、就绪态和运行态等各种状态变化。

4.线程的实现方式

线程的实现可以分为两类：用户级线程 (User-Level Thread， ULT) 和内核级线程 (Kernel-Level Thread，KLT) 。内核级线程又称内核支持的线程。

5.多线程模型

3 种不同的多线程模型。

1) 多对一模型。

2) 一对一模型。

3) 多对多模型。

处理器调度

调度的概念

1.调度的基本概念

在多道程序系统中，进程的数量往往多于处理机的个数，因此进程争用处理机的情况在所难免。是对处理机进行分配，即从就绪队列中按照一定的算法（公平、高效）选择一个进程并将处理机分配给它运行，以实现进程并发地执行。

处理机调度是多道程序操作系统的基础，是操作系统设计的核心问题。

2.调度的层次

1) 作业调度。

2) 中级调度。

3) 进程调度。

3.三级调度的联系

作业调度从外存的后备队列中选择一批作业进入内存，为它们建立进程，这些进程被送入就绪队列，进程调度从就绪队列中选出一个进程，并把其状态改为运行态，把 CPU 分配给它。中级调度是为了提高内存的利用率，系统将那些暂时不能运行的进程挂起来。当内存空间宽松时，通过中级调度选择具备运行条件的进程，将其唤醒。

(1) 作业调度为进程活动做准备，进程调度使进程正常活动起来，中级调度将暂时不能运行的进程挂起，中级调度处于作业调度和进程调度之间。

(2) 作业调度次数少，中级调度次数略多，进程调度频率最高。

(3) 进程调度是最基本的，不可或缺。

调度的时机、切换与过程

现代操作系统中，不能进行进程的调度与切换的情况有以下几种：

1.在处理中断的过程中。

2.进程在操作系统内核程序临界区中。

3.其他需要完全屏蔽中断的原子操作过程中。

应该进行进程调度与切换的情况如下：

(1) 发生引起调度条件且当前进程无法继续运行下去时，可以马上进行调度与切换。若操作系统只在这种情况下进行进程调度，则是非剥夺调度。

(2) 中断处理结束或自陷处理结束后，返回被中断进程的用户态程序执行现场前，若置上请求调度标志，即可马上进行进程调度与切换。若操作系统支持这种情况下的运行调度程序，则实现了剥夺方式的调度。

进程调度的方式

所谓进程调度方式，是指当某个进程正在处理机上执行时，若有某个更为重要或紧迫的进程需要处理，即有优先权更高的进程进入就绪队列，此时应如何分配处理机。

通常有以下两种进程调度方式：

1）非剥夺调度方式，又称非抢占方式。在非剥夺调度方式下，一旦把 CPU 分配给一个进程，该进程就会保持 CPU 直到终止或转换到等待态。这种方式的优点是实现简单、系统开销小，适用于大多数的批处理系统，但它不能用于分时系统和大多数的实时系统。

2）剥夺调度方式，又称抢占方式。

采用剥夺式的调度，对提高系统吞吐率和响应效率都有明显的好处。但“剥夺”不是一种任意性行为，必须遵循一定的原则，主要有优先权、短进程有限和时间片原则等。

调度的基本准则

为了衡量调度算法的性能，人们提出了一些评价标准。

1.CPU 利用率。

2.系统吞吐量

系统吞吐量表示单位时间内 CPU 完成作业的数量。

3.响应时间

4.周转时间

从每个作业的角度来看，完成该作业的时间是至关重要的，通常用周转时间或者带权周转时间来衡量。

a)周转时间

周转时间是指作业从提交至完成的时间间隔，包括等待时间和执行时间。周转时间 T；用公式表示为

作业 i 的周转时间 $T=$ 作业 i 的完成时间 − 作业 i 的提交时间

b)平均周转时间

平均周转时间是指多个作业（例如 n 个作业）周转时间的平均值。平均周转时间 T 用公式表示为

$$T=\ (T_1+T_2+..+T_n)\ /n$$

c）带权周转时间

带权周转时间是指作业周转时间与运行时间的比。作业 i 的带权周转时间 W；用公式表示为：

$W=$ 作业 i 的周转时间/作业 i 的运行时间

d）平均带权周转时间

与平均周转时间类似，平均带权周转时间是多个作业的带权周转时间的平均值。公式略。

典型的调度算法

操作系统中存在多种调度算法，有的调度算法适用于作业调度，有的调度算法适用于进程调度，有的调度算法两者都适用。下面介绍几种常用的调度算法。

1.先来先服务（FCFS）调度算法

FCFS 调度算法是一种最简单的调度算法，它既可用于作业调度，又可用于进程调度。在作业调度中，算法每次从后备作业队列中选择最先进入该队列的一个或几个作业，将它们调入内存，分配必要的资源，创建进程并放入就绪队列。

FCFS 调度算法属于不可剥夺算法。。

FCFS 调度算法的特点是算法简单，但效率低；对长作业比较有利，但对短作业不利（相对 SJF 和高响应比）；有利于 CPU 繁忙型作业，而不利于 I/O 繁忙型作业。

2.短作业优先（SJF）调度算法

短作业（进程）优先调度算法是指对短作业（进程）优先调度的算法。短作业优先（SJF）调度算法从后备队列中选择一个或若干估计运行时间最短的作业，将它们调入内存运行；短进程优先（SPF） 调度算法从就绪队列中选择一个估计运行时间最短的进程，将处理机分配给它，使之立即执行，直到完成或发生某事件而阻塞时，才释放处理机。

SJF 调度算法也存在不容忽视的缺点：

1.该算法对长作业不利；同时由于调度程序总是优先调度那些（即使是后进来的）短作业，将导致长作业长期不被调度而产生饥饿。

2.该算法完全未考虑作业的紧迫程度，因而不能保证紧迫性作业会被及时处理。

3.由于作业的长短只是根据用户所提供的估计执行时间而定的，而用户又可能会有意或无意地缩短其作业的估计运行时间，致使该算法不一定能真正做到短作业优先调度。

注意，SJF 调度算法的平均等待时间、平均周转时间最少。

3.优先级调度算法

优先级调度算法又称优先权调度算法，它既可用于作业调度，又可用于进程调度。该算法中的优先级用于描述作业运行的紧迫程度。

根据新的更高优先级进程能否抢占正在执行的进程，可将该调度算法分为如下两种：

1.非剥夺式优先级调度算法。当一个进程正在处理机上运行时，即使有某个更为重要或紧迫的进程进入就绪队列，仍然让正在运行的进程继续运行，直到由于其自身的原因而主动让出处理机时（任务完成或等待事件），才把处理机分配给更为重要或紧迫的进程。

2.剥夺式优先级调度算法。当一个进程正在处理机上运行时，若有某个更为重要或紧迫的进程进入就绪队列，则立即暂停正在运行的进程，将处理机分配给更重要或紧迫的进程。

而根据进程创建后其优先级是否可以改变，可以将进程优先级分为以下两种：

1.静态优先级。

2.动态优先级。

一般来说，进程优先级的设置可以参照以下原则：

1.系统进程>用户进程。

2.交互型进程>非交互型进程（或前台进程>后台进程）。

3.IO 型进程>计算型进程。

4.高响应比优先调度算法

高响应比优先调度算法主要用于作业调度，是对 FCFS 调度算法和 SJF 调度算法的一种综合平衡，同时考虑了每个作业的等待时间和估计的运行时间。在每次进行作业调度时，先计算后备作业队列中每个作业的响应比，从中选出响应比最高的作业投入运行。

响应比的变化规律可描述为：

$$\text{响应比}R_p = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}}$$

根据公式可知：

1）作业的等待时间相同时，要求服务时间越短，响应比越高，有利于短作业。

2）要求服务时间相同时，作业的响应比由其等待时间决定，等待时间越长，其响应比越高，因而它实现的是先来先服务。

3）对于长作业，作业的响应比可以随等待时间的增加而提高，等待时间足够长时，其响应比便可升到很高，从而也可获得处理机。因此，克服了饥饿状态，兼顾了长作业。

5.时间片轮转调度算法

时间片轮转调度算法主要适用于分时系统。在这种算法中，系统将所有就绪进程按到达时间的先后次序排成一个队列，进程调度程序总是选择就绪队列中的第一个进程执行，即先来先服务的原则，但仅能运行一个时间片， 如 100ms。在使用完一个时间片后， 即使进程并未完成其运行，它也必须释放出（被剥夺）处理机给下一个就绪的进程，而被剥夺的进程返回到就绪队列的末尾重新排队，等候再次运行。

在时间片轮转调度算法中，时间片的大小对系统性能的影响很大。若时间片足够大，以至于所有进程都能在一个时间片内执行完毕，则时间片轮转调度算法就退化为先来先服务调度算法。若时间片很小，则处理机将在进程间过于频繁地切换，使处理机的开销增大，而真正用于运行用户进程的时间将减少。因此，时间片的大小应选择适当。

时间片的长短通常由以下因素确定：系统的响应时间、就绪队列中的进程数目和系统的处理能力。

6.多级反馈队列调度算法（融合了前几种算法的优点）

多级反馈队列调度算法是时间片轮转调度算法和优先级调度算法的综合与发展，如下图所示。通过动态调整进程优先级和时间片大小，多级反馈队列调度算法可以兼顾多方面的系统目标。例如，为提高系统吞吐量和缩短平均周转时间而照顾短进程；为获得较好的 I/O 设备利用率和缩短响应时间而照顾 I/O 型进程；同时，也不必事先估计进程的执行时间。

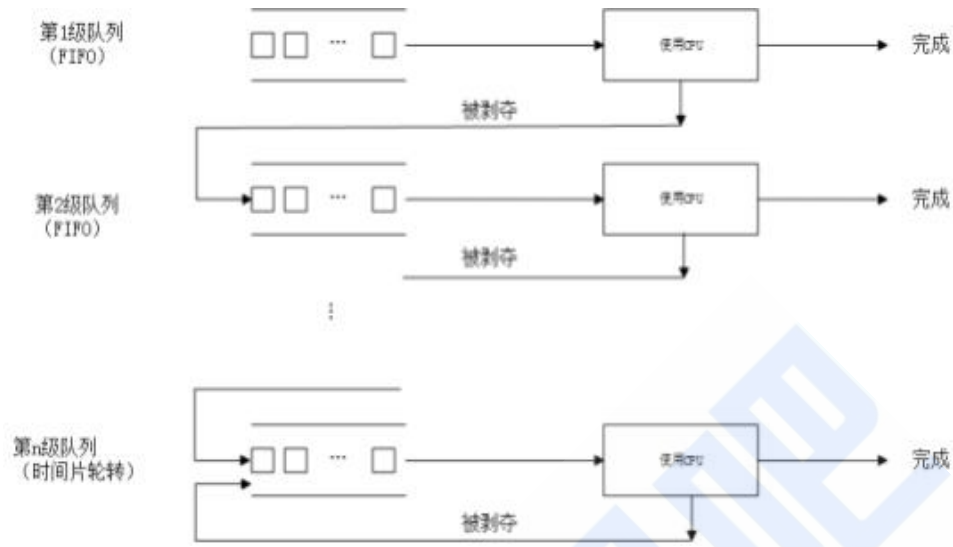


图 2.2.1 多级队列反馈调度算法

多级反馈队列调度算法的实现思想如下：

1.设置多个就绪队列，并为各个队列赋予不同的优先级，第 1 级队列的优先级最高，第 2 级队列次之，其余队列的优先级逐次降低。

2.赋予各个队列中进程执行时间片的大小各不相同。在优先级越高的队列中，每个进程的运行时间片越小。例如，第 2 级队列的时间片要比第 1 级队列的时间片长 1 倍…第 i+1 级队列的时间片要比第 i 级队列的时间片长 1 倍。

3.一个新进程进入内存后，首先将它放入第 1 级队列的末尾，按 FCFS 原则排队等待调度。当轮到该进程执行时，如它能在该时间片内完成，便可准备撤离系统；若它在一个时间片结束时尚未完成，调度程序便将该进程转入第 2 级队列的末尾，再同样按 FCFS 原则等 待调度执行；若它在第 2 级队列中运行一个时间片后仍未完成，再以同样的方法放入第 3 级队列....如此下去，当一个长进程从第 1 级队列依次降到第 n 级队列后，在第 n 级队列中便采用时间片轮转的方式运行。

4.仅当第 1 级队列为空时，调度程序才调度第 2 级队列中的进程运行；仅当第 1~（i-1）级队列均为空时，才会调度第 i 级队列中的进程运行。若处理机正在执行第 i 级队列中的某进程，这时又有新进程进入优先级较高的队列[第 1~（i-1）中的任何一个队列]，则此时新进程将抢占正在运行进程的处理机，即由调度程序把正在运行的进程放回第 i 级队列的末尾，把处理机分配给新到的更高优先级的进程。

多级反馈队列的优势有以下几点：

1）终端型作业用户：短作业优先。

2）短批处理作业用户：周转时间较短。

3）长批处理作业用户：经过前面几个队列得到部分执行，不会长期得不到处理。

进程同步

进程同步的基本概念

在多道程序环境下，进程是并发执行的，不同进程之间存在着不同的相互制约关系。为了协调进程之间的相互制约关系，引入了进程同步的概念。

1.临界资源

虽然多个进程可以共享系统中的各种资源，但其中许多资源一次只能为一个进程所用，我们将一次仅允许一个进程使用的资源称为临界资源。

对临界资源的访问，必须互斥地进行，在每个进程中，访问临界资源的那段代码称为临界区。

为了保证临界资源的正确使用，可把临界资源的访问过程分成 4 个部分：

1）进入区。

2）临界区。

3）退出区。

4）剩余区。

2.同步

同步亦称直接制约关系，是指为完成某种任务而建立的两个或多个进程，这些进程因为需要在某些位置上协调它们的工作次序而等待、传递信息所产生的制约关系。进程间的直接制约关系源于它们之间的相互合作。

3.互斥

互斥也称间接制约关系。当一个进程进入临界区使用临界资源时，另一个进程必须等待，当占用临界资源的进程退出临界区后，另一进程才允许去访问此临界资源。

为禁止两个进程同时进入临界区，同步机制应遵循以下准则：

1）空闲让进。临界区空闲时，可以允许一个请求进入临界区的进程立即进入临界区。

2）忙则等待。当已有进程进入临界区时，其他试图进入临界区的进程必须等待。

3）有限等待。对请求访问的进程，应保证能在有限时间内进入临界区。

4）让权等待。当进程不能进入临界区时，应立即释放处理器，防止进程忙等待。

实现临界区互斥的基本方法

1.软件实现方式

在进入区设置并检查一些标志来标明是否有进程在临界区中，若已有进程在临界区，则在进 入区通过循环检查进行等待，进程离开临界区后则在退出区修改标志。

1）算法一：单标志法。

2）算法二：双标志法先检查。

3）算法三：双标志法后检查。

4）算法四： Peterson's Algorithm。

2.硬件实现方式

1）中断屏蔽方法

2）硬件指令方式

TestAndSet 指令：这条指令是原子操作，即执行该代码时不允许被中断。

信号量

信号量机制是一种功能较强的机制，可用来解决互斥与同步问题，它只能被两个标准的原语 wait(S)和 signal(S)访问，也可记为“P 操作”和“V 操作”。

原语是指完成某种功能且不被分割、不被中断执行的操作序列，通常可由硬件来实现。

1.整型信号量

2.记录型信号量

3.信号量的应用

1）实现进程同步

2）实现进程互斥

管程

管程定义了一个数据结构和能为并发进程所执行的一组操作，这组操作能同步进程和改变管程中的数据。由管程的定义可知，管程由局部于管程的共享数据结构说明、操作这些数据结构的一组过程以及对局部于管程的数据结构设置初值的语句组成，管程把分散在各个进程中互斥访问公共变量的临界区集中起来，提供对它们的保护。

管程有以下基本特征：

1）局部于管程的数据只能被局部于管程内的过程所访问。

2）一个进程只有通过调用管程内的过程才能进入管程访问共享数据。

3）每次仅允许一个进程在管程内执行某个内部过程，即进程互斥地通过调用内部过程进入管程。其他想进入管程的过程必须等待，并阻塞在等待队列。

由于管程是一个语言成分，因此管程的互斥访问完全由编译程序在编译时自动添加，而且保证正确。

经典同步问题

1.生产者-消费者问题

问题描述：一组生产者进程和一组消费者进程共享一个初始为空、大小为 n 的缓冲区，只有缓冲区没满时，生产者才能把消息放入缓冲区，否则必须等待;只有缓冲区不空时，消费者才能从中取出消息，否则必须等待。由于缓冲区是临界资源，它只允许一个生产者放入消息，或一个消费者从中取出消息。

问题分析：

1.关系分析。生产者和消费者对缓冲区互斥访问是互斥关系，同时生产者和消费者又是一个相互协作的关系， 只有生产者生产之后，消费者才能消费，它们也是同步关系。

2.整理思路。这里比较简单，只有生产者和消费者两个进程，正好是这两个进程存在着互斥关系和同步关系。那么需要解决的是互斥和同步 PV 操作的位置。

3.信号量设置。信号量 mutex 作为互斥信号量，用于控制互斥访问缓冲池，互斥信号量初值为 1；信号量 full 用于记录当前缓冲池中的“满”缓冲区数，初值为 0。信号量 empty 用于记录当前缓冲池中的“空”缓冲区数，初值为 n。

2.读者-写者问题

问题描述：有读者和写者两组并发进程，共享一个文件，当两个或以上的读进程同时访问共享数据时不会产生副作用，但若某个写进程和其他进程（读进程或写进程）同时访问共享数据时则可能导致数据不一致的错误。因此要求：①允许多个读者可以同时对文件执行读操作;②只允许一个写者往文件中写信息;③任一写者在完成写操作之前不允许其他读者或写者工作;④写者执行写操作前，应让已有的读者和写者全部退出。

读者-写者问题有一个关键的特征，即有一个互斥访问的计数器 count，因此遇到一个不太好解决的同步互斥问题时，要想一想用互斥访问的计数器 count 能否解决问题。

3.哲学家就餐问题

1）关系分析。5 名哲学家与左右邻居对其中间筷子的访问是互斥关系。

2）整理思路。显然，这里有 5 个进程。本题的关键是如何让一名哲学家拿到左右两根筷子而不造成死锁或饥饿现象。解决方法有两个：一是让他们同时拿两根筷子；二是对每名哲学家的动作制定规则，避免饥饿或死锁现象的发生。

3）信号量设置。定义互斥信号量数组 chopstick[5]= {1,1,1,1,1}，用于对 5 个筷子的互斥访问。哲学家按顺序编号为 0~4，哲学家 i 左边筷子的编号为 i，哲学家右边筷子的编号为(i + 1)%5。

该算法存在死锁。

死锁

死锁的概念

1.死锁的定义

死锁，是指多个进程因竞争资源而造成的一种僵局（互相等待），若无外力作用，这些进程都将无法向前推进。

2.死锁产生的原因

1）系统资源的竞争

2）进程推进顺序非法

3.死锁产生的必要条件

产生死锁必须同时满足以下 4 个条件，只要其中任意一个 条件不成立，死锁就不会发生。

1）互斥条件。

2）不剥夺条件。

3）请求与保持条件。

4）环路等待条件。

死锁的处理策略

目前用于处理死锁的方法主要有以下 4 种：

1）鸵鸟算法。

2）预防死锁。

3）避免死锁。

4）检测及解除死锁。

死锁的预防

防止死锁的发生只需破坏死锁产生的 4 个必要条件之一即可。

1.破坏互斥条件

2.破坏不剥夺条件

3.破坏请求并保持条件

4.破坏循环等待条件

死锁的避免

1.安全状态与不安全状态

1）不安全状态是指系统可能发生死锁的状态，并不意味着系统已经发生死锁。

2）处于不安全状态的系统不会必然导致死锁。

2.银行家算法

具有代表性的避免死锁算法是 Dijkstra 给出的银行家算法。为实现银行家算法，系统中必须设置若干数据结构。假定系统中有 n 个进程（P1，P2，..，Pn）、m 类资源（R1，R2，…，Rm），银行家算法中使用的数据结构如下：

1）可利用资源向量 Available。

2）最大需求矩阵 Max。

3）分配矩阵 Allocation。

4）需求矩阵 Need。

银行家算法的描述如下：

定义 Requesti 向量：Requesti 表示第 i 个进程向系统提出一次申请，申请的各类资源的数量就是该向量的各个分量。当进程 Pi 向系统发出资源请求后，系统进行如下操作：

1）若 $Request_i \le Need_i$，则跳至 2）。否则报错，因为进程 Pi 申请的资源数不应该超过它的需求数。

2）若 $Request_i \le Available_i$，则跳至 3），否则 Pi 进程需要等待，因为可用资源不够。

3）对 Pi 进程所请求的资源进行预分配，修改以下向量：

$$Available = Available - Request_i$$

$$Allocation_i = Allocation_i + Request_i$$

$$Need_i = Need_i - Request_i$$

4）对于修改后的向量调用安全性算法。若安全性算法返回系统处于安全状态，则按 Requesti 表示的资源数量给 Pi 进程分配资源;若安全性算法返回系统处于不安全状态，则不分配给 Pi 进程任何资源，让 Pi 等待，并恢复 3）中所改变的向量。

安全性算法描述如下：

1.建立长度为 m 的向量 Work 和长度为 n 的向量 Finish，并对它们进行如下初始化：. Work = Available; Finish[i] = false; （i = 1，2，…，n）

2.查找满足如下两个条件的 i： Finish[i] = false; $Need_i \le Work$; 若没有这样的 i 存在，则跳到 4）执行。

3.进行如下操作： Work = Work + Allocation; Finish[i] = true 返回到 2）。

如果对于所有 i，都有 Finsih[i] = true，那么系统处于安全状态，否则为不安全状态。安全性算法的时间复杂度达到了 O（$m * n^2$）。

死锁的检测和解除

1.资源分配图

系统死锁可利用资源分配图来描述。如下图所示，用圆圈代表一个进程，用框代表一类资源。由于一种类型的资源可能有多个，因此用框中的一个圆代表一类资源中的一个资源。从进程到资源的有向边称为请求边，表示该进程申请一个单位的该类资源；从资源到进程的边称为分配边，表示该类资源已有一个资源分配给了该进程。

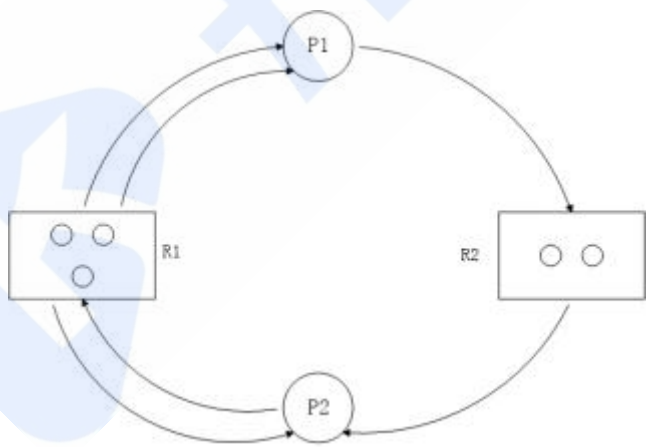


图 2.4.1 资源分配图

在上图所示的资源分配图中，进程 P1 已经分得了两个 Ri 资源，并又请求一个 R2 资源；进程 P2 分得了一个 R1 资源和一个 R2 资源，并又请求一个 R1 资源。

2.死锁定理

简化资源分配图可检测系统状态 S 是否为死锁状态。简化方法如下：

1）在资源分配图中，找出既不阻塞又不孤点的进程 P 消去它所有的请求边和分配边，使之成为孤立的结点。在下图（a）中，PI 是满足这一条件的进程结点，将 P1 的所有边消去，便得到下图（b）所示的情况。

2）进程 Pi 所释放的资源，可以唤醒某些因等待这些资源而阻塞的进程，原来的阻塞进程可能变为非阻塞进程。在 s 上图中，进程 Pi 就满足这样的条件。根据 1）中的方法进行一系列简化后，若能消去图中所有的边，则称该图是可完全简化的，如下图（c）所示。

S 为死锁的条件是当且仅当 S 状态的资源分配图是不可完全简化的，该条件为死锁定理。

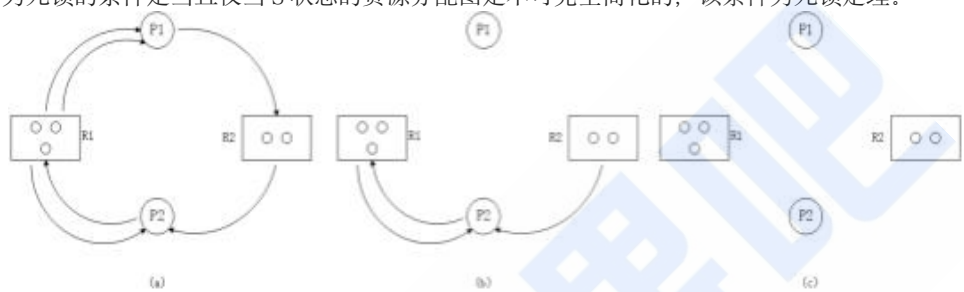


图 2.4.2 资源分配图的简化

3.死锁解除

一旦检测出死锁，就应立即采取相应的措施来解除死锁。死锁解除的主要方法有：

1.资源剥夺法。

2.撤销进程法。

3.进程回退法。

线程的状态与转换(22 考研新加)

与进程一样，线程也有执行就绪阻塞三个状态。

线程同步机制

多线程操作系统大多提供了多种同步机制，如互斥锁，条件变量，计数信号标量等。

条件变量(22 考研新加)

进人管程时的互斥由编译器负责。另外，还需要一种办法使得进程在无法继续运行时被阻塞，解决的方法是引人条件变量(condition variables)以及相关的两个操作: wait 和 signal。

调度程序(22 考研新加)

当计算机系统是多道程序设计系统时，通常就会有多个进程或线程同时竞争 CPU。只要有两个或更多的进程处于就绪状态，就会发生 CPU 竞争。如果只有一个 CPU 可用，那么就必须选择下一个要运行的进程。在操作系统中，完成选择工作的这一部分称为调度程序(scheduler)，该程序使用的算法称为调度算法。

闲逛进程(22 考研新加)

在一个进程退出时必须进行调度。必须从就绪进程集中选择某个进程，如果没有就绪的进程，通常会运行一个系统提供的空闲进程(闲逛进程)。

内核级线程和用户级线程调度(22 考研新加)

许多适用于进程调度的处理方法也同样适用于线程调度。

用户级线程调度:由于内核并不知道有线程存在，所以内核还是和以前一样，选择一个进程，并给予时间片加以控制，进程中的线程调度程序决定哪个线程运行。

内核级线程调度:经常是线程级别，与线程所属的进程基本或根本没有关联。内核选择一个特定的线程运行，不用考虑该线程属于哪个进程。对被选择的线程赋予一个时间片，如果超过了时间片，就会强制挂起该线程。

用户级线程和内核级线程之间的差别在于性能。用户级线程的线程切换需要少量的机器指令，而内核级线程需要完整的上下文切换、修改内存映像、使高速缓存失效，这导致了若干数量级的延迟。另一方面内核级线程一旦阻塞在 1/0 上，不需要像用户级线程那样将整个进程挂起。

上下文及其切换机制(22 考研新加)

上下文指的是一种环境，对于进程而言，上下文实际上是进程执行活动全过程的静态描述。

进程上下文是可以按照层次规则组合起来的。例如在 UNIX System V 中，进程上下文由用户级上下文、寄存器上下文以及系统级上下文组成。

上下文切换机制:当对处理机进行切换时，将发生两对上下文切换操作。对于第一对，将保存当前进程的上下文，而装入分派程序的上下文，以便分派程序运行;对于第二对，将移出分派程序，而把新选进程的 CPU 现场信息装入到处理机的相应寄存器中。

第三章 内存管理

内存管理概念

操作系统对内存的划分和动态分配，就是内存管理的概念。

内存管理的功能有：

a)内存空间的分配与回收。

b)地址转换。

C)内存空间的扩充。。

d)存储保护。

进程运行的基本原理和要求。

1.程序的装入和链接

a)编译。

b)链接。

c)装入。

程序的链接有以下三种方式：

a)静态链接。

b)装入时动态链接。

c)运行时动态链接。

内存的装入模块在装入内存时，同样有以下三种方式：

1)绝对装入。

2)可重定位装入。

3)动态运行时装入，也称动态重定位。逻辑地址空间与物理地址空间

2.内存保护

1)在 CPU 中设置一对上、下限寄存器。

2)采用重定位寄存器（ 或基址寄存器）和界地址寄存器（ 又称限长寄存器）来实现这种保护。

连续分配管理方式

1.单一连续分配

这种方式的优点是简单、无外部碎片，可以采用覆盖技术，不需要额外的技术支持。缺点是只能用于单用户、单任务的操作系统中，有内部碎片，存储器的利用率极低。

2.固定分区分配

a) 分区大小相等。用于利用一台计算机去控制多个相同对象的场合，缺乏灵活性。

b) 分区大小不等。划分为多个较小的分区、适量的中等分区和少量大分区。

固定分区是可用于多道程序设计的最简单的存储分配，无外部碎片，但不能实现多进程共享一个主存区，所以存储空间利用率低

3.动态分区分配

动态分区在开始分配时是很好的，但之后会导致内存中出现许多小的内存块。

动态分区的分配策略。考虑以下几种算法：

1)首次适应（First Fit） 算法。空闲分区以地址递增的次序链接。分配内存时顺序查找，找到大小能满足要求的第一个空闲分区。

2)最佳适应（Best Fit）算法。空闲分区按容量递增的方式形成分区链，找到第一个能满足要求的空闲分区。

3)最坏适应（Worst Fit） 算法。又称最大适应（Largest Fit） 算法，空闲分区以容量递减的次序链接，找到第一个能满足要求的空闲分区，即挑选出最大的分区。

4)邻近适应（NextFit） 算法。又称循环首次适应算法，由首次适应算法演变而成。不同之处是，分配内存时从上次查找结束的位置开始继续查找。

非连续分配管理方式

非连续分配管理方式根据分区的大小是否固定，分为分页存储管理方式和分段存储管理方式。

在分页存储管理方式中，分为基本分页存储管理方式和请求分页存储管理方式。

1.基本分页存储管理方式

固定分区会产生内部碎片，动态分区会产生外部碎片，这两种技术对内存的利用率都比较低。

（1）分页存储的几个基本概念

①页面和页面大小。

②地址结构。分页存储管理的逻辑地址结构如下图所示。

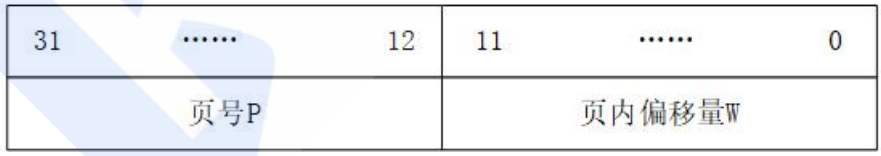


图 3.3.1 分页存储管理的逻辑地址结构

③页表。

页表的作用是实现从页号到物理块号的地址映射，如下图所示。

（2）基本地址变换机构

地址变换机构的任务是将逻辑地址转换为内存中的物理地址。地址变换是借助于页表实现的。下图给出了分页存储管理系统中的地址变换机构。

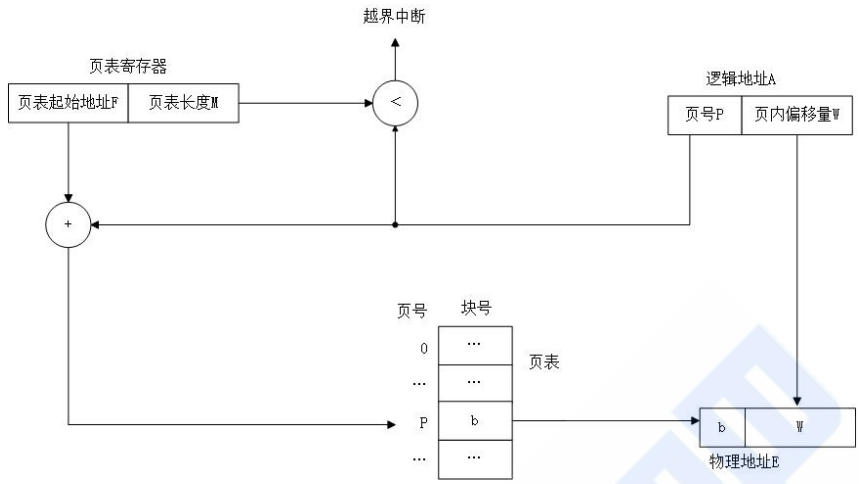


图 3.3.2 分页存储管理系统中的地址变换机构

逻辑地址 A 到物理地址 E 的变换过程如下（逻辑地址、页号、每页的长度都是十进制数）：

①计算页号 P （P=A/L）和页内偏移量 W （W= A%L）。

②比较页号 P 和页表长度 M，若 P≥M，则产生越界中断，否则继续执行。

③页表中页号 P 对应的页表项地址 = 页表始址 F + 页号 P × 页表项长度，取出该页表项内容 b，即为物理块号。要注意区分页表长度和页表项长度。页表长度的值是指一共有多少页，页表项长度是指页地址占多大的存储空间。

④计算 E = b × L + W，用得到的物理地址 E 去访问内存。

分页管理方式存在的两个主要问题：①每次访存操作都需要进行逻辑地址到物理地址的转换，地址转换过程必须足够快，否则访存速度会降低;②每个进程引入页表，用于存储映射机制，页表不能太大，否则内存利用率会降低。

(3) 具有快表的地址变换机制

具有快表的地址变换机构如下图所示。

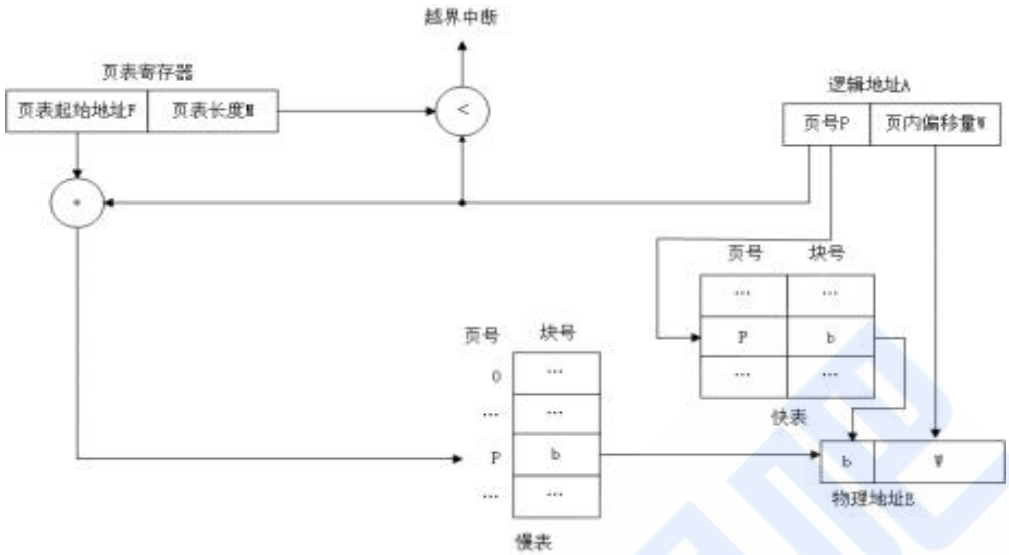


图 3.3.3 具有快表的的地址变换机构

在具有快表的分页机制中，地址的变换过程如下：

①CPU 给出逻辑地址后，由硬件进行地址转换，将页号送入高速缓存寄存器，并将此页号与快表中的所有页号进行比较。

②若找到匹配的页号，说明所要访问的页表项在快表中，则直接从中取出该页对应的页框号，与页内偏移量拼接形成物理地址。这样，存取数据仅一次访存便可实现。

③若未找到匹配的负号，则需要访问主存中的页表，在读出页表项后，应同时将其存入快表，以便后面可能的再次访问。但若快表已满，则必须按照一定的算法对旧的页表项进行替换。

注意：有些处理机设计为快表和慢表同时查找，若在快表中查找成功则终止慢表的查找。

一般快表的命中率可达 90%以上，这样分页带来的速度损失就可降低至 10%以下。快表的有效性基于著名的局部性原理，后面讲解虚拟内存时将会具体讨论它。

(4) 两级页表

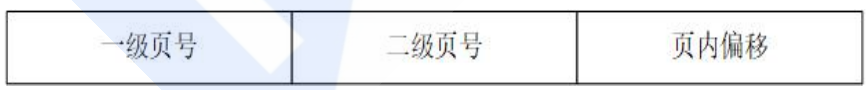


图 3.3.4 逻辑地址空间的格式

二级页表实际上是在原有页表结构上再加上一层页表。

2.基本分段存储管理方式

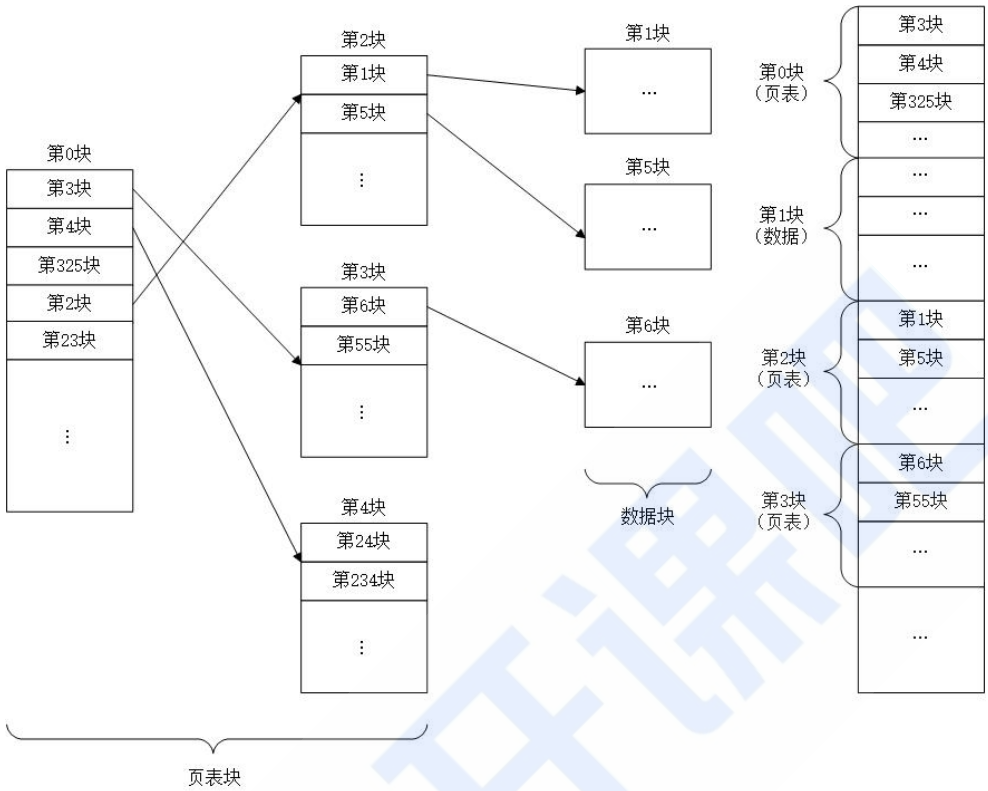
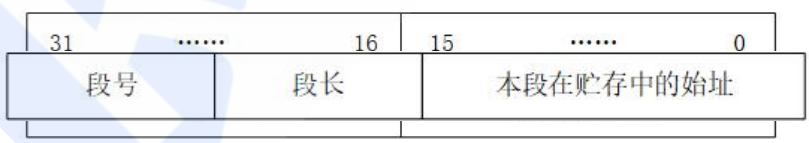


图 3.3.5 二级页表结构示意图

1）分段。段式管理方式按照用户进程中的自然段划分逻辑空间。

在下图中，段号为 16 位，段内偏移量为 16 位，因此一个作业最多有 2^{16}= 65536 段，最大段长为 64KB。

图 3.3.6 分段系统中的逻辑地址结构



2）段表。

图 3.3.7 段表的内容

配置段表后，执行中的进程可通过查找段表，找到每段所对应的内存区。可见，段表用于实现从逻辑段到物理内存区的映射，如下所示。

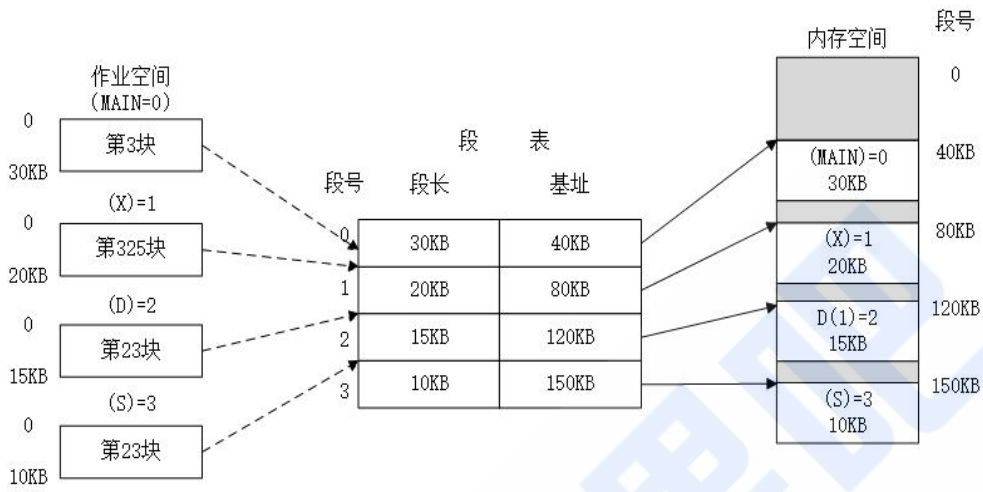


图 3.3.8 利用段表实现物理内存区映射

3）地址变换机构。为了实现进程从逻辑地址到物理地址的变换功能，在系统中设置了段表寄存器，用于存放段表始址 F 和段表长度 M。

从逻辑地址 A 到物理地址 E 之间的地址变换过程如下：

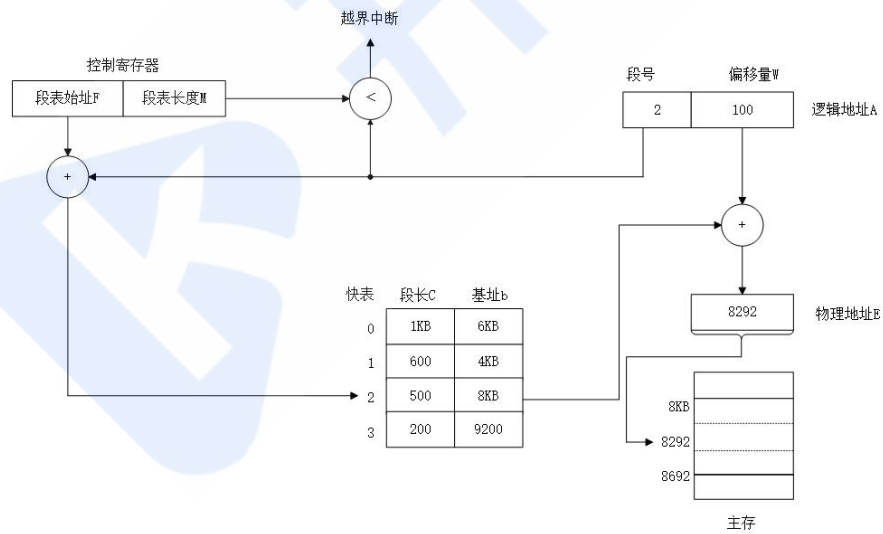


图 3.3.9 分段系统的地址变换过程

① 从逻辑地址 A 中取出前几位为段号 S，后几位为段内偏移量 W，注意在段式存储管理的题目中，逻辑地址一般以二进制数给出，而在页式存储管理中，逻辑地址一般以十进制数给出，读者要具体问题具体分析。

② 比较段号 S 和段表长度 M，若 S≥M，则产生越界中断，否则继续执行。

③ 段表中段号 s 对应的段表项地址=段表始址 F+段号 Sx 段表项长度，取出该段表项的前几位得到段长 C。若段内偏移量≥C，则产生越界中断，否则继续执行。从这句话我们可以看出，段表项实际上只有两部分，前几位是段长，后几位是始址。

④ 取出段表项中该段的始址 b，计算 E=b+W，用得到的物理地址 E 去访问内存。

4）段的共享与保护。

与分页管理类似，分段管理的保护方法主要有两种：一种 是存取控制保护，另一种是地址越界保护。

3. 段页式管理方式

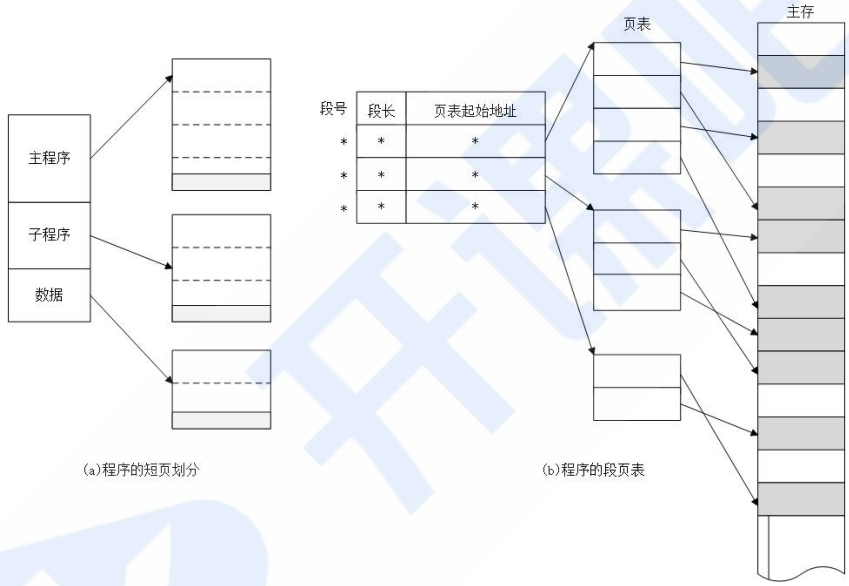


图 3.3.10 段页式管理方式

在段页式系统中，作业的逻辑地址分为三部分：段号、页号和页内偏移量，如下所示。

注意：在一个进程中，段表只有一个，而页表可能有多个。

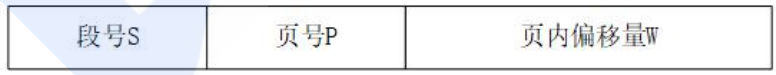


图 3.3.11 段页式系统的逻辑地址结构

在进行地址变换时，首先通过段表查到页表始址，然后通过页表找到页帧号，最后形成物理地址。如下图所示，进行一次访问实际需要三次访问主存，这里同样可以使用快表来加快查找速度，其关键字由段号、页号组成，值是对应的页帧号和保护码。

结合上面对段式管理和页式管理的地址空间的分析，可以得出结论：段页式管理的地址空间是二维的。

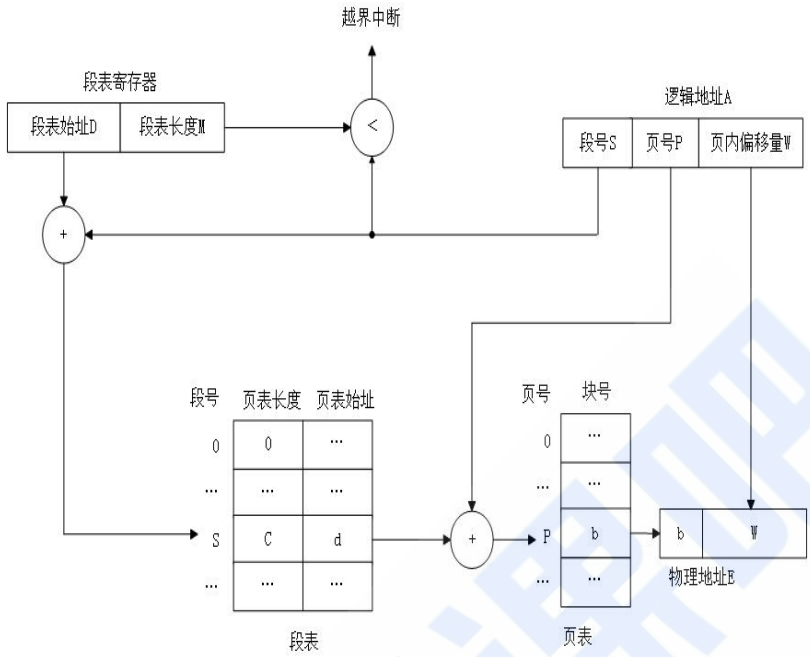


图 3.3.12 段页式系统的地址变换机构

虚拟内存管理

虚拟内存的基本概念

局部性原理表现在以下两个方面：

1）时间局部性。

2）空间局部性。

虚拟存储器有以下三个主要特征：

1）多次性。多次性是指无须在作业运行时一次性地全部装入内存，而允许被分成多次调入内存运行。

2）对换性。对换性是指无须在作业运行时一直常驻内存，而允许在作业的运行过程中，进行换进和换出。

3）虚拟性。虚拟性是指从逻辑上扩充内存的容量，使用户所看到的内存容量远大于实际的内存容量

虚拟内存的实现有以下三种方式：

a)请求分页存储管理。

b)请求分段存储管理。

c)请求段页式存储管理。

不管哪种方式，都需要有一定的硬件支持。一般需要的支持有以下几个方面：

a)一定容量的内存和外存。

b)页表机制（或段表机制），作为主要的数据结构。

c)中断机构，当用户程序要访问的部分尚未调入内存时，则产生中断。

d)地址变换机构，逻辑地址到物理地址的变换。

请求分页管理方式

为了实现请求分页，系统必须提供一定的硬件支持。除了需要一定容量的内存及外存的计算机系统，还需要有页表机制、缺页中断机构和地址变换机构。

1.页表机制

页号	物理块号	状态位P	访问字段A	修改位M	外存地址

图 3.4.1 请求分页系统中的页表项

增加的 4 个字段说明如下：

a)状态位 P。用于指示该页是否已调入内存，供程序访问时参考。

b)访问字段 A。用于记录本页在一段时间内被访问的次数，或记录本页最近已有多长时间未被访问，供置换算法换出页面时参考。

c)修改位 M。标识该页在调入内存后是否被修改过。

d)外存地址。 用于指出该页在外存上的地址，通常是物理块号，供调入该页时参考。

2.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中时，便产生一个缺页中断，请求操作系统将所缺的页调入内存。此时应将缺页的进程阻塞（调页完成唤醒），若内存中有空闲块，则分配一个块，将要调入的页装入该块，并修改页表中的相应页表项，若此时内存中没有空闲块，则要淘汰某页（若被淘汰页在内存期间被修改过，则要将其写回外存）。

缺页中断作为中断，同样要经历诸如保护 CPU 环境、分析中断原因、转入缺页中断处理程序、恢复 CPU 环境等几个步骤。但与一般的中断相比，它有以下两个明显的区别：

a)在指令执行期间而非一条指令执行完后产生和处理中断信号，属于内部中断。

b)一条指令在执行期间，可能产生多次缺页中断。

3.地址变换机构

请求分页系统中的地址变换机构，是在分页系统地址变换机构的基础上，为实现虚拟内存，又增加了某些功能而形成的。

如下图所示，在进行地址变换时，先检索快表：

a)若找到要访问的页，则修改页表项中的访问位（写指令还需要重置修改位），然后利用页表项中给出的物理块号和页内地址形成物理地址。

b)若未找到该页的页表项，则应到内存中去查找页表，再对比页表项中的状态位 P，看该页是否已调入内存，未调入则产生缺页中断，请求从外存把该页调入内存。

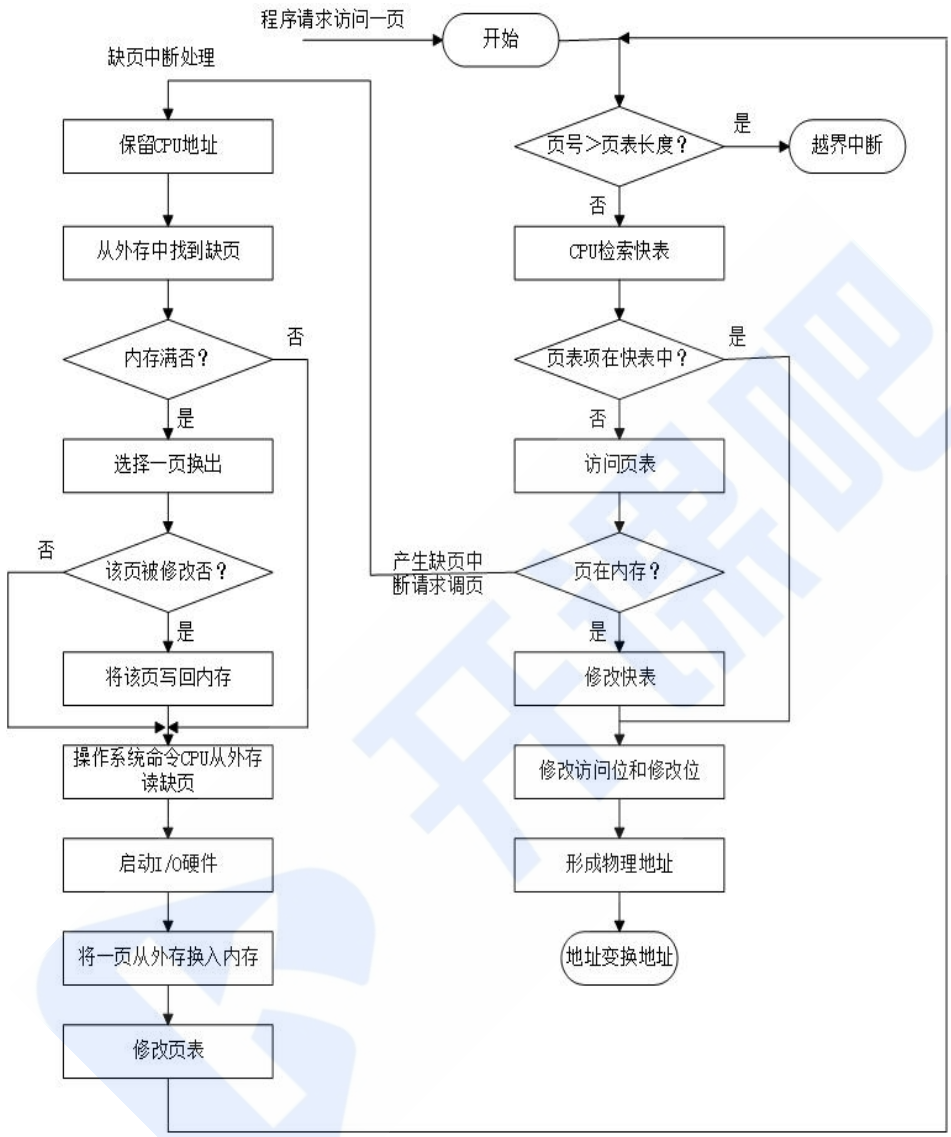


图 3.4.2 请求分页中的地址变换过程

页面置换算法

常见的置换算法有以下 4 种:

1.最佳（OPT）置换算法

2.先进先出（FIFO） 页面置换算法

FIFO 算法还会产生所分配的物理块数增大而页故障数不减反增的异常现象，称为 Belady 异常。

3.最近最久未使用（LRU）置换算法

LRU 算法的性能较好，但需要寄存器和栈的硬件支持。

4.时钟（CLOCK）置换算法

因为算法要循环扫描缓冲区，像时钟的指针一样转动，所以称为 CLOCK 算法。

页面分配策略

1.驻留集大小

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。

2.调入页面的时机

1)预调页策略。

2)请求调页策略。

3.从何处调入页面

抖动

在页面置换过程中，一种最糟糕的情形是，刚刚换出的页面马上又要换入主存，刚刚换入的页面马上又要换出主存，这种频繁的页面调度行为称为抖动或颠簸。

频繁发生缺页中断（抖动）的主要原因是，某个进程频繁访问的页面数目高于可用的物理页帧数目。

内存映射文件(22 考研新加)

”映射”就是建立一种对应关系，将硬盘上文件对象的某- 部分与进程逻辑地址空间中块相同区城之间建立起映射关系。在内存映射过程中，并没有实际的数据拷贝，文件没有被载入内存，只是逻辑上放入了内存。具体到代码，就是建立并初始化了相关的数据结构。当进程代码第一次引用这段虚拟地址时，会产生缺页中断，此时 OS 根据映射关系直接将文件的相关部分数据直接拷贝到进程的私有空间中。 内存映射文件是由一个文件到一块内存的映射，使应用程序可以通过内存指针对磁盘上的文件进行访问，其过程就如同对加妆文件的内存的访问，因此内存文件映射非常适合于用来管理大文件.

虚拟存储器性能的影响因素及改进方法(22 考研新加)

缺页次数是影响性能的一个主要因素，而缺页次数又受到页面本身大小、分配给进程的物理页面数、页面置换算法以及程序的编制方法的影响。

页面大小是由系统的地址结构确定的，无法修改。页面较小，一方面可使内存碎片小，并减少了内存碎片的总空间，有利于提高内存利用率;另一方面，也会使每个进程要求较多的页面，从而导致页表过长，占用大量内存;此外，还会降低页面换进换出的效率。若页面较大，虽然可减少页表长度，提高换进换出效率，但会使页内碎片增大。

分配给进程的物理页面数越接近进程逻辑页面数量，缺页率会越低，但是分配过多也会浪费内存空间。只要保证活跃页面在内存就可以降低缺页率。

页面置换算法选择 LRU、Clock、页面缓冲等置换算法，将未来有可能访问的页面尽量保留在内存中，从而提高页面访问的命中率和访问速度。

编写程序时，如果存储采用的是按行存储，访问时尽量采用相同的访问方式，避免按列访问造成的缺页率高的现象。

第四章 文件管理

文件系统基础

文件的概念

1.文件的定义

1)数据项。

a) 基本数据项。

b) 组合数据项。

2)记录。

3)文件。

2.文件的属性

1) 名称。文件名称唯一。

2) 标识符。标识文件系统内文件的唯一标签，通常为数字，是对人不可读的一种内部名称。

3) 类型。被支持不同类型的文件系统所使用。

4) 位置。指向设备和设备上文件的指针。

5) 大小。文件当前大小（用字节、字或块表示），也可包含文件允许的最大值。

6) 保护。对文件进行保护的访问控制信息。

7) 时间、日期和用户标识。文件创建、上次修改和上次访问的相关信息，用于保护和跟踪文件的使用。

3.文件的基本操作

1) 创建文件

2) 写文件

3) 读文件

4) 文件重定位

5) 删除文件

6) 截断文件

这 6 个基本操作可以组合起来执行其他文件操作。例如，一个文件的复制，可以创建新文件、从旧文件读出并写入新文件。

4.文件的打开与关闭

每个打开文件都有如下关联信息：

1) 文件指针

2) 文件打开计数

3) 文件磁盘位置

4）访问权限

文件的逻辑结构

按逻辑结构，文件可划分为以下两种：

1.无结构文件（流式文件）

2.有结构文件（记录式文件）

有结构文件按记录的组织形式可分为以下四种：

1）顺序文件。

2）索引文件。

3）索引顺序文件。

4）直接文件或散列文件（Hash File）。

目录结构

1.文件控制块和索引结点

1）文件控制块。文件控制块（FCB）是用来存放控制文件需要的各种信息的数据结构，以实现“按名存取”。FCB 的有序集合称为文件目录，一个 FCB 就是一个文件目录项。为了创建一个新文件，系统将分配一个 FCB 并存放在文件目录中，成为目录项。 FCB 主要包含以下信息：

a) 基本信息。如文件名、文件的物理位置、文件的逻辑结构、文件的物理结构等。

b) 存取控制信息。如文件存取权限等。

c) 使用信息。如文件建立时间、修改时间等。

2)索引结点。在检索目录文件的过程中，只用到了文件名，仅当找到一个目录项（查找文件名与目录项中文件名匹配）时，才需要从该目录项中读出该文件的物理地址。也就是说，在检索目录时，文件的其他描述信息不会用到，也不需要调入内存。

表 4.1 UNIX 的文件目录结构

文件名	索引结点编号
文件名 1	
文件名 2	
…	
…	

2.目录结构

在理解一个文件系统的需求前，我们首先考虑在目录这个层次上所需要执行的操作，这有助于后面文件系统的整体理解。

操作时，考虑以下几种目录结构：

1)单级目录结构。

2)两级目录结构。

3)多级目录结构（树形目录结构）。

4)无环图目录结构。

文件共享

文件共享使多个用户（进程）共享同一个文件，系统中只需保留该文件的一个副本。若系统不能提供共享功能，则每个需要该文件的用户都要有各自的副本，会造成对存储空间的极大浪费。

现代常用的两种文件共享方法如下：

1.基于索引结点的共享方式（硬链接）

2.利用符号链实现文件共享（软链接）

文件保护

1.访问类型

1）读。从文件中读。

2）写。向文件中写。

3）执行。将文件装入内存并执行。

4）添加。将新信息添加到文件结尾部分。

5）删除。删除文件，释放空间。

6）列表清单。列出文件名和文件属性。

2. 访问控制

精简的访问列表采用拥有者、组和其他三种用户类型：

1）拥有者。创建文件的用户。

2）组。一组需要共享文件且具有类似访问的用户。

3）其他。系统内的所有其他用户。

口令和密码是另外两种访问控制方法。

1）口令。用户在建立一个文件时提供一个口令，系统为其建立 FCB 时附上相应口令，同时告诉允许共享该文件的其他用户。用户请求访问时必须提供相应的口令。这种方法时间和空间的开销不多，缺点是口令直接存在系统内部，不够安全。

2）密码。用户对文件进行加密，文件被访问时需要使用密钥。这种方法保密性强，节省了存储空间，不过编码和译码要花费一定的时间。

文件系统实现

文件系统的层级结构

1.用户调用接口

文件系统为用户提供与文件及目录有关的调用，如新建、打开、读写、关闭、删除文件，建立、删除目录等。此层由若干程序模块组成，每个模块对应一条系统调用，用户发出系统调用时，控制即转入相应的模块。

2.文件目录系统

文件目录系统的主要功能是管理文件目录，其任务有管理活跃文件目录表、管理读写状态信息表、管理用户进程的打开文件表、管理与组织存储设备上的文件目录结构、调用下一级存取控制模块。

3.存取控制验证模块

实现文件保护主要由该级软件完成，它把用户的访问要求与 FCB 中指示的访问控制权限进行比较，以确认访问的合法性。

4.逻辑文件系统与文件信息缓冲区

逻辑文件系统与文件信息缓冲区的主要功能是，根据文件的逻辑结构将用户要读写的逻辑记录转换成文件逻辑结构内的相应块号。

5.物理文件系统

物理文件系统的主要功能是把逻辑记录所在的相对块号转换成实际的物理地址。

6.辅助分配模块

分配模块的主要功能是管理辅存空间，即负责分配辅存空闲空间和回收辅存空间。

7.设备管理程序模块

设备管理程序模块的主要功能是分配设备、分配设备读写用缓冲区、磁盘调度、启动设备、处理设备中断、释放设备读写缓冲区、释放设备等。

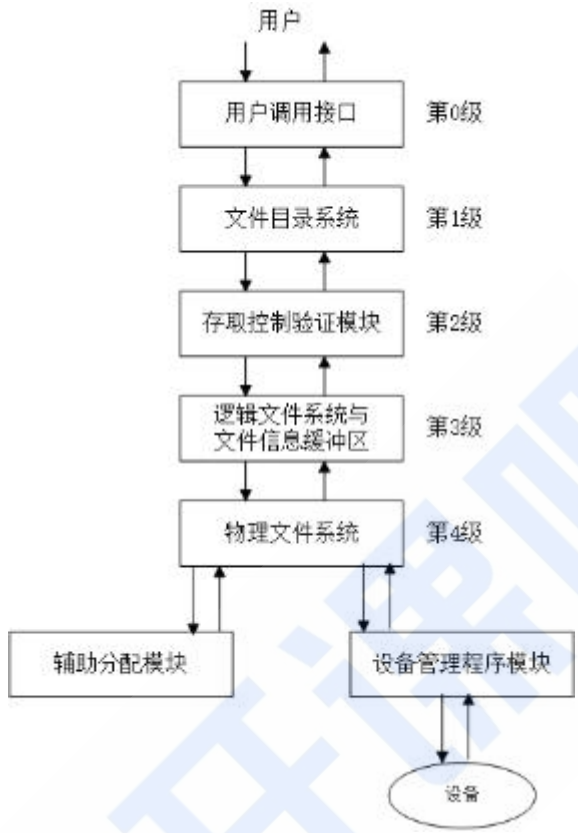


图 4.1.1 文件系统层次结构

目录实现

1.线性列表

2.哈希表

文件实现-文件分配方式。

常用的磁盘空间分配方法有三种:

1) 连续分配

2) 链接分配。

a)隐式链接。

b)显式链接。

3) 索引分配。

a)链接方案。

b)多层索引。

c)混合索引。

表 4.2 文件三种分配方式

	访问第 n 条记录	优点	缺点
索引分配	需访问磁盘 1 次	顺序存取时速度快，文件定长时可根据文件起始地址及记录长度进行随机访问	文件存储要求连续的存储空间，会产生碎片，不利于文件的动态扩充
链接分配	需访问磁盘 n 次	可解决外存的碎片问题，提高外存空间的利用率，动态增长较方便	只能按照文件的指针链顺序访问，查找效率低，指针信息存放消耗外存空间
索引分配	m 级需访问磁盘 m+1 次	可以随机访问，文件已于增删	索引表增加存储空间的开销，索引表的查找策略对文件系统效率影响较大

文件实现-文件存储空间结构

1）文件存储器空间的划分与初始化。

图 4.1.2 逻辑卷与物理盘的关系

2）文件存储器空间管理。

a)空闲表法。

b)空闲链表法。

c)位示图法。

d)成组链接法。

磁盘组织与管理

磁盘的结构

磁盘调度算法

一次磁盘读写操作的时间由寻找（寻道）时间、延迟时间和传输时间决定。

1）寻找时间T_s。活动头磁盘在读写信息前，将磁头移动到指定磁道所需要的时间。这个时间除跨越多条磁道的时间外，还包括启动磁臂的时间。

2）延迟时间T_r。磁头定位到某一磁道的扇区（块号）所需要的时间，设磁盘的旋转速度为 r，则 $T_r = \frac{1}{2r}$。

3）传输时间 T_t。从磁盘读出或向磁盘写入数据所经历的时间，这个时间取决于每次所读/写的字节数 b 和磁盘的旋转速度：$T_t = \frac{b}{rN}$。其中，r 为磁盘每秒的转速，N 为一个磁道上的字节数。

总平均存取时间 T_a 可以表示为 $T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$。

目前常用的磁盘调度算法有以下几种：

1）先来先服务（ First Come First Served，FCFS）算法

2）最短寻找时间优先（ Shortest Seek Time First，SSTF）算法　现象。

3）扫描（ SCAN）算法。

4）循环扫描（ Circular SCAN，C-SCAN）算法。

1. 磁盘初始化

2. 引导块

3. 坏块

磁目录管理（22 考研新加）

一文件目录是种数据结构，用于标识系统中的文件及其物理地址，供检索时使用。

1.对目录管理的要求

实现“按名存取”、提高对目录的检索速度、文件共享和允许文件重名。

2.文件控制块和索引结点

1)文件控制块(FCB)

文件控制块是用于描述 和控制文件的数据结构，文件管理程序可借助于文件控制块中的信息对文件施以各种操作。

文件与文件控制块对应，文件控制块的有序集合称为文件目录，即个文件控制块是 一个文件目录项。通常一个文件目录也被看作是一个文件，称为目录文件。

FCB 通常含有以下三类信息:

基本信息类:包括文件名、文件物理位置、文件逻辑结构、文件物理结构。

存取控制信息类:包括文件主的存取权限，核准用户的存取权限和一般用户的存取权限。

使用信息类:包括文件的建立日期和时间、文件上次修改的日期和时间及当前使用信息。

2)索引结点

文件目录通常存放在磁盘上，当文件很多时，文件目录要占用大量的盘块。在检索目录文件的时候，需要将目录调人内存后比较文件名。但是只用到文件名，面不需要其他的文件描述信息。所以便把文件名与文件信息分开，使文件描述信息单独形成一个索引结点。

磁盘索引结点:存放在磁盘上的索引结点。每个文件有唯一的一个，主要包括文件主标识符、文件类型、文件存取权限、文件物理地址、文件长度、文件链接计数、文件存取时间。

内存索引结点:放在内存中的索引结点。当文件被打开后，将磁盘索引结点复制到内存索引结点中以便使用。与磁盘索引结点比，又增加了索引结点编号、状态、访问计数、文件所属文件系统的逻辑设备号、链接指针。

3.目录结构

目录结构的组织不仅关系到文件系统的存取速度，还关系到文件的共享性和安全性。常用的目录结构形式有单级目录、两级目录和多级目录。

虚拟文件系统（22 考研新加）

虚拟文件系统(VPS)是一种用于网络环境的分布式文件系统，它允许在同-个目录结构中可以挂载者若干种不同的文件系统。

虚拟文件系统是物理文件系统与服务之间的一个接口层，它对每个文件系统的所有细节进行抽象，隐藏了它们的实现细节，为使用者提供统一的接口。严格说来，VFS 并不是种实际的文件系统，它只存在于内存中，不存在于任何外存空间。VFS 在系统启动时建立，在系统关闭时消亡。

对于用户来说，不需要关心不同文件系统的具体操作过程，而只是对一个虚拟的文件操作界面来进行操作。每一个文件系统之间互不干扰，而只是调用相应的程序来实现其功能。VFS 作为内核中的一个软件层，用于给用户空间的程序提供文件系统接口，同时也提供了内核中的个抽象功能，允许不同的文件系统很好地共存。

文件系统挂载（22 考研新加）

文件系统挂载，是指将新的文件系统关联至当前根文件系统。可以使用 mount 命令在目录树中的指定自录(挂载点)附加文件系统。挂载点作为要挂载文件系统的访问人口，必须已经存在，一般是不会被进程使用到的目录。挂载后，挂载点下原有文件将会被临时隐藏。卸载，是指将某文件系统与当前根文件系统的关联关系移除。

第五章 输入输出管理

I/O 管理概述

I/O 设备

计算机系统中的 I/O 设备按使用特性可分为以下类型：

1）人机交互类外部设备。

2）存储设备。用于存储程序和数据的设备，如磁盘、磁带、光盘等。这类设备用于数据交换，速度较快，通常以多字节组成的块为单位进行数据交换。

3）网络通信设备。用于与远程设备通信的设备，如各种网络接口、调制解调器等。其速度介于前两类设备之间。网络通信设备在使用和管理上与前两类设备也有很大不同。

计算机系统中的 I/O 设备可按传输速率可分为以下类型：

1）低速设备。如键盘、鼠标等。

2）中速设备。如行式打印机、激光打印机等。

3）高速设备。如磁带机、磁盘机、光盘机等。

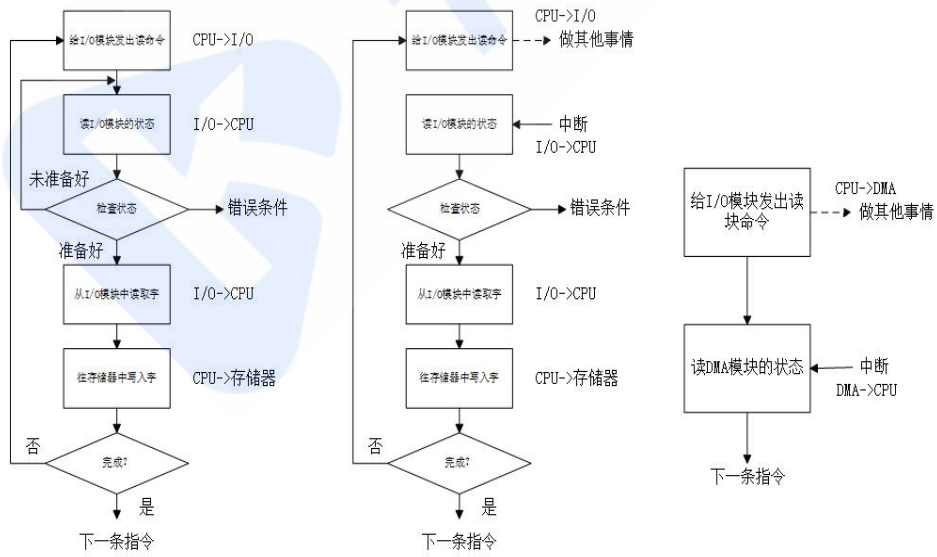
计算机系统中的 I/O 设备可按信息交换的单位分类：

1）块设备。

2）字符设备。

I/O 控制方法

图 5.1.1 I/O 控制方式



（a）程序直接控制方式　　（b）中断驱动方式　　（c）方式

1. 程序直接控制方式

2. 中断驱动方式

3. DMA（直接存储器存取）方式

DMA 方式的特点如下：

1)基本单位是数据块。

2)所传送的数据，是从设备直接送入内存的，或者相反。

3)仅在传送一个或多个数据块的开始和结束时，才需 CPU 干预，整块数据的传送是在 DMA 控制器的控制下完成的。

DMA 控制方式与中断驱动方式的主要区别是，中断驱动方式在每个数据需要传输时中断 CPU，而 DMA 控制方式则是在所要求传送的一批数据全部传送结束时才中断 CPU;此外，中断驱动方式数据传送是在中断处理时由 CPU 控制完成的，而 DMA 控制方式则是在 DMA 控制器的控制下完成的。

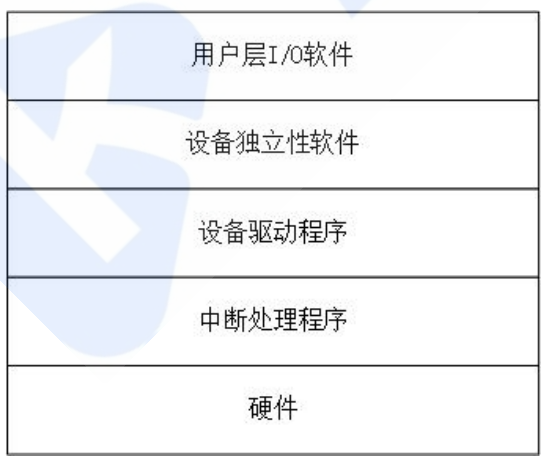
4. 通道控制方式

I/O 通道是指专门负责输入/输出的处理机。I/O 通道方式是 DMA 方式的发展，它可以进一步减少 CPU 的干预。同时，又可以实现 CPU、通道和 I/O 设备三者的并行操作，从而更有效地提高整个系统的资源利用率。

I/O 通道与 DMA 方式的区别是：DMA 方式需要 CPU 来控制传输的数据块大小、传输的内存位置，而通道方式中这些信息是由通道控制的。另外，每个 DMA 控制器对应一台设备与内存传递数据，而一个通道可以控制多台设备与内存的数据交换。

I/O 子系统的层级结构

图 5.1.2 I/O 层次方式



一个比较合理的层次划分如图所示。整个 I/O 系统可以视为具有 5 个层次的系统结构，各层次及其功能如下：

1) 用户层 IO 软件。

2) 设备独立性软件。

3）设备驱动程序。

4）中断处理程序。

5）硬件设备。

设备控制器的主要功能如下：

1）接收和识别 CPU 或通道发来的命令，如磁盘控制器能接收读、写、查找等命令。

2）实现数据交换，包括设备和控制器之间的数据传输;通过数据总线或通道，控制器和主存之间的数据传输。

3）发现和记录设备及自身的状态信息，供 CPU 处理使用。

4）设备地址识别。

为实现上述功能，设备控制器必须包含以下组成部分：

①设备控制器与 CPU 的接口。

②设备控制器与设备的接口。

③I/O 控制逻辑。

I/O 核心子系统

I/O 子系统概述

I/O 调度概念

I/O 调度就是确定一个好的顺序来执行这些 I/O 请求。

磁盘调度算法其实就是 I/O 调度的一种。

高速缓存与缓冲区

1.磁盘高速缓存（Disk Cache）

2.缓冲区（Buffer)

在设备管理子系统中，引入缓冲区的目的主要如下：

1）缓和 CPU 与 IO 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）解决基本数据单元大小（即数据粒度）不匹配的问题。

4）提高 CPU 和 IO 设备之间的并行性。

其实现方法如下：

1）采用硬件缓冲器，但由于成本太高，除一些关键部位外，一般不采用硬件缓冲器。

2）采用缓冲区（位于内存区域）。

根据系统设置缓冲器的个数，缓冲技术可以分为如下几种：

1）单缓冲

2）双缓冲。

3）循环缓冲。

4）缓冲池。

3. 高速缓存与缓冲区的对比

表 5.1 高速缓存和缓冲区的对比

		高速缓存	缓冲区
相同点		都介于高速设备和低速设备之间	
区别	存放数据	存放的是低速设备上的某些数据的复制数据，即高速缓存上有的，低速设备上面必然有	存放的是低速设备传递给高速设备的数据（或相反），而这些数据在低速设备（或高速设备）上却不一定有备份，这些数据再从缓冲区传送到高速设备（或低速设备）
	目的	高速缓存存放的是高速设备经常要访问的数据，若高速设备要访问的数据不在高速缓存中，则高速设备就需要访问低速设备	高速设备和低速设备的通信都要经过缓冲区，高速设备永远不会直接去访问低速设备

设备分配与回收

1.设备分配概述

从设备的特性来看，采用下述三种使用方式的设备分别称为独占设备、共享设备和虚拟设备。

1）独占式使用设备。

2）分时式共享使用设备。

3）以 SPOOLing 方式使用外部设备。

2.设备分配的数据结构

设备分配依据的主要数据结构有设备控制表（DCT）、控制器控制表（COCT）、通道控制表（CHCT）和系统设备表（SDT），各数据结构功能如下。

设备控制表（DCT）：我们可以认为，一个设备控制表就表征一个设备，而这个控制表中的表项就是设备的各个属性。

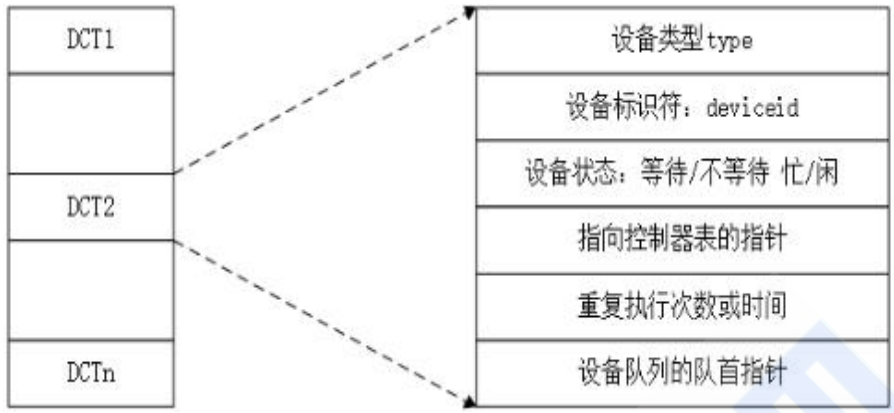


图 5.2.1 设备控制表

每个设备都分为机械部件和电子部件两部分，其中负责解析上层传达的命令并控制机械部件运作的是电子部件 (控制器) , 所以每个 DCT 都需要一个表项来表示控制器, 即需要一个指向控制器控制表 (COCT) 的指针，因此 DCT 与 COCT 有一一对应的关系。 前面我们学过 4 种 I/O 控制方式，通道方式显然要比其他几种方式更加优越，因此现代操作系统的 I/O 控制采用的都是通道控制。设备控制器控制设备与内存交换数据, 而设备控制器又需要请求通道为它服务, 因此每个 COCT 必定有一个表项存放指向相应通道控制表 (CHCT) 的指针，而一个通道可为多个设备控制器服务，因此 CHCT 中必定有一个指针，指向一个表，这个表上的信息表达的是 CHCT 提供服务的那几个设备控制器。CHCT 与 COCT 的关系是一对多的关系。 系统设备表 (SDT) 整个系统只有一张 SDT，它记录已连接到系统中的所有物理设备的情况，每个物理设备占一个表目。

图 5.2.2 COCT、CHCT 和 SDT

由于在多道程序系统中，进程数多于资源数，会引起资源的竞争，因此要有一套合理的分配原则，主要考虑的因素有：I/O 设备的固有属性、I/O 设备的分配算法、I/O 设备分配的安全性以及 I/O 设备的独立性。

3. 设备分配的策略

1) 设备分配原则。

2) 设备分配方式。

3) 设备分配算法。常用的动态设备分配算法有先请求先分配、优先级高者优先等。

4. 设备分配的安全性

设备分配的安全性是指设备分配中应防止发生进程死锁。

1）安全分配方式。

2）不安全分配方式。

5. 逻辑设备名到物理设备名的映射

在系统中可采取两种方式建立逻辑设备表：

1）在整个系统中只设置一张 LUT。这样，所有进程的设备分配情况都记录在这张表中，因此不允许有相同的逻辑设备名，主要适用于单用户系统。

2）为每个用户设置一张 LUT。当用户登录时，系统便为该用户建立一个进程，同时也为之建立一张 LUT，并把该表放入进程的 PCB。

SPOOLing 技术

为了缓和 CPU 的高速性与 I/O 设备低速性之间的矛盾，引入了脱机输入/输出技术。SPOOLing 的意思是外部设备同时联机操作，又称假脱机输入/输出操作，是操作系统中采用的一项将独占设备改造成共享设备的技术。

SPOOLing 系统的组成如图所示。

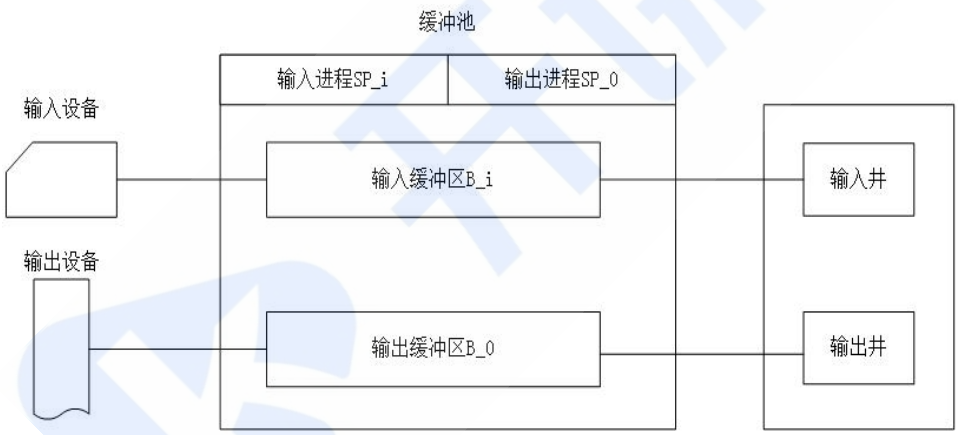


图 5.2.3 SPOOLing 系统的组成

1. 输入井和输出井

2. 输入缓冲区和输出缓冲区

3. 输入进程和输出进程。

SPOOLing 系统的主要特点有：提高了 IO 的速度;将独占设备改造为共享设备：实现了虚拟设备功能。

输入输出应用程序接口（22 考研新加）

中断处理程序的处理过程

在设备控制器控制下，I/0 设备完成了 I/0 操作后，控制器(或通道)便向 CPU 发出一中断请求，CPU 响应后便转向中断处理程序，中断处理程序大致包含以下几步：

在设置 1/0 进程时，当中断处理程序开始执行时，都必须去唤醒阻塞的驱动(程序)进程。在采用信号量机制时，可通过执行 V 操作，将处于阻塞状态的驱动(程序)进程唤醒。

保护被中断进程的 CPU 现场。

分析中断原因，转人相应的设备中断处理程序。

进程中断处理，判别此次 I/0 完成是正常结束中断还是异常结束中断，分别做相应处理。恢复被中断进程或由调度程序选中的进程的 CPU 现场。

返回被中断的进程，或进入新选中的进程继续运行。

阻塞/非阻塞 I/O（22 考研新加）

阻塞 I/O，指应用进程调用 I/0 操作时阻塞，只有等待要 I/O 的数据准备好，并复制到应用进程的缓冲区中或写人外存，才被唤醒继续执行。

非阻塞 I/O,指应用进程调用 I/0 操作时不阻塞该进程，该 I/0 调用返回一个错误返回值。一般情况下，应用进程需要利用轮询的方式不断查询，直到所需的数据复制到进程的缓冲区。

缓冲区管理（22 考研新加）

为了缓和 CPU 和 I/O 设备速度不匹配的矛盾，提高 CPU 和 I/O 设备的并行性，在现代 0S 中，几乎所有的 I/O 设备与处理机交换数据时都使用级冲区。

1.缓冲的引人

引人缓冲区的目的:缓和 CPU 与 I/O 设备问速度不匹配的矛盾；减少 CPU 的中断频率，放宽对 CPU 中断响应时间的限制；提高 CPU 和 I/O 设备之间的并行性。

2.单缓冲与多缓冲

1)单缓冲(single buffer)

在单缓冲情况下，每当用户进程发出一 I/O 请求时，os 便在内存中为之分配一缓冲区。CPU 和 I/O 设备需要争夺缓冲区。

2) 双缓冲(double buffer)

为了加快输入/输出速度，提高设备利用率，人们又引入了双缓冲区机制，也称为级冲对换(bufferswapping)。在设备输人时，先将数据送人第一缓冲区，装后便转向第二缓冲区。此时 OS 可以从第一缓冲区中移出数据，并送人用户进程。接着由 CPU 对数据进行计算。

3.循环缓冲

当输人与输出或生产者与消费者的速度基本相匹配时，采用双缓冲能获得较好的效果，可使生产者和消费者基本上能并行操作。若两者的速度相差甚远，双缓冲的效果不够理想，但随着缓冲区数量的增加，

情况有所改善，因此引入了多缓冲机制，可以将缓冲区组织成循环缓冲形式。

1)循环缓冲的组成

多个缓冲区:循环缓冲有多个大小相同的缓冲区。作为输人的缓冲区有三种类型:用于装输人数据的空缓冲区 R、已装满数据的缓冲区 G 以及计算进程正在使用的现行工作级冲区 C。

多个指针:作为输人的缓冲区可设置三个指针，用于指示计算进程下一个可用缓冲区 G 的指针 Next，指示输人进程下次可用的缓冲区 R 的指针 Next,以及用于指示计算进程正在使用的缓冲区 C 的指针 Current。

设备驱动程序接口（22 考研新加）

接收上层软件发来的抽象要求，再把它转换成具体要求。

检查用户 1/0 请求的合法性，了解 1/0 设备的状态，传送必要的参数，设置工作方式。

由驱动程序向设备控制器发出 I/0 命令，启动分配到的 1/0 设备，完成指定的 1/0 操作。

及时响应由控制器或通道发来的中断请求，并根据其中断调用相应的中断处理程序进行处理。

对于设置有通道的计算机系统，驱动程序还应能够根据用户的 I/O 请求，自动地构成通道程序。

固态硬盘（22 考研新加）

固态硬盘是用固态电子存储芯片阵列制成的硬盘，由控制单元和存储单元(Flash 芯片、DRAM 芯片)组成。固态硬盘在接口的规范和定义、功能及使用方法上与传统硬盘完全相同，但 I/O 性能相对于传统硬盘大大提升，被广泛应用于军事、车载、工控、视频监控、网络监控、网络终端、电力、医疗、航空、导航设备等领域。固态硬盘具有如下优点:

读写速度快。采用闪存作为存储介质，读取速度相对机械硬盘更快。固态硬盘不用磁头，寻道时间几乎为 0。持续写人的速度非常惊人，最常见的 7200 转机械硬盘的寻道时间一般为 12~14ms,而固态硬盘可以轻易达到 0.1ms 甚至更低。

磨损均衡。也是一种基于固态硬盘主控芯片的内置平衡机制，用于均衡固态硬盘内部各个区块闪存颗粒的使用程度，从而延长整体颗粒的使用寿命。在主控制器的固件中添人新的控制命令，让主控制 器在固态硬盘的读写过程中，尽可能均衡地使用各个 block，防止部分闪存区块因过度频繁地擦除和写人命令，而导致整块闪存颗粒提前报废。

开课吧

使 命

让每个人都能公平和便利地获取优质教育服务，并实现可持续职业成长

愿 景

打造全球顶尖的人才科技公司，创办一所国际知名大学，成为令人尊敬的企业

价值观

关心并热爱、只为赋能人才、极致敢为、坦诚开放、拥抱变化、始终创业