

第一章 绪论

1.1 操作系统概述

一、操作系统的地位

计算机系统结构：如图所示。

计算机系统：按用户的要求接收和存储信息、自动进行数据处理并输出结果信息的系统。

计算机系统由硬件子系统（计算机系统赖以工作的实体）和软件子系统（保证计算机系统按用户指定的要求协调工作）组成。

操作系统在硬件基础上的第一层软件，是硬件与其它软件的接口。

➤ 软件的分类：

系统软件：实现资源的管理和控制程序的执行。与硬件共同构成其它软件的运行基础（要求：高效率）

支撑软件：与系统软件和硬件构成一个开发平台，可支持其它软件的开发和维护

应用软件：领域知识与计算机技术结合，按特定需要专门编写的程序，完成特定功能

二、操作系统的定义

操作系统是计算机系统中的一种系统软件，它是这样一些程序模块的集合——它们能以尽量有效、合理的方式组织和管理计算机的软硬件资源，合理地组织计算机的工作流程，控制程序的执行并向用户提供各种服务功能，使用户能够灵活、方便、有效的使用计算机；并使整个计算机系统高效率的运行。

尽量：折中权衡（中庸之道）

有效：系统效率（如 CPU 用的充足与否）

资源利用率（如主存，外部设备是否忙碌）

提高资源利用率

合理：公平性，如果不公平则会产生“死锁”或“饥饿”

方便：用户界面，使用手段上清晰简单



三、操作系统的特征

1. 并发

在计算机系统中同时存在多个程序，从宏观上看这些程序是同时向前推进的；从微观上讲，任何时刻只有一个程序在执行，即单 CPU 条件下，这些程序在 CPU 上轮流执行。

双重含义：用户与用户程序并发

用户与操作系统程序并发

➤ 并行：是从硬件意义上考虑的，是不同硬件部件（例如：CPU 与 I/O）的并行。即使是在微观上，多个程序也是同时执行的。必须要一定的硬件基础。

(而并发是指程序在单 CPU 上轮流执行。)

2. 共享

操作系统与多个用户程序共同使用计算机系统资源。

3. 随机性：操作系统必须随时对以不可预测的次序发生的事件进行响应。

四、操作系统的两个设计目标

提供一个良好的供其它程序执行的运行环境，具体追求：

1、方便使计算机系统用

- 操作系统提供方便使用的接口

2、使计算机系统能高效率工作

- 操作系统扩充硬件的功能：操作系统应使硬件的功能发挥的更好
- 操作系统使用户合理共享资源，防止各用户间相互干扰
- 操作系统以文件形式管理软件资源，保证信息的安全和快速存取

两个目标是一对矛盾，需要折中权衡

1.2 操作系统的类型

一、操作系统的形成与发展

操作系统从无到有，其发展是随着计算机硬件技术的发展而发展的。

- 手工操作阶段（程序员即使操作员，汇编，读卡机，设备驱动程序）
- 管理程序（20世纪50年代末至60年代初，编译程序）
- 多道程序系统：批处理系统、分时系统：

(1) 多道程序设计技术：通道技术，中断技术的出现使多道程序设计技术成为可能

(2) SPOOLing技术 (Simultaneous Peripheral Operation-On Line), 预输入和缓输出功能

(3) 分时系统：把CPU时间划成时间片，轮流为用户服务

- 通用操作系统

二、操作系统的分类

1、多道批处理操作系统

❖ 工作方式

作业 (JOB)：用户要求计算机系统进行处理的一个计算问题。

用户程序及其所需的数据和控制命令一起形成作业。

作业：程序 + 数据 + 作业说明书

用户（准备好自己的作业）将作业交给系统操作员，系统操作员将许多用户的作业组成一批作业，通过输入设备输入到计算机系统中（磁盘上），然后，启动操作系统，执行每个作业，最后由操作员将作业运行结果交给用户。

单道批处理 与 多道批处理

(注：多道指某个作业占用 CPU，若由于某种原因暂时不用 CPU，第二个作业占用 CPU)

❖ 特点：

1) 多道，主存中同时存在多个正在运行的程序，它们并发工作，减少 CPU

的空闲时间，提高了 CPU 的利用率。

2) 成批处理，作业流程自动化，减少了人工操作和作业交接的时间，提高了系统吞吐率。

带来的不足之处：无交互手段，调试程序困难；用户不能直接干预自己作业的运行，一旦发现作业错误，不能及时修改，延长了开发软件时间，所以一般适用于成熟的程序。

3) 采用作业调度策略，按一定的合理搭配选择作业进入主存，可以充分利用计算机系统的各种资源。

CPU 密集型、I/O 密集型、均衡型

4) 采用 SP00L 技术使作业在执行过程中，可以直接从高速磁盘上存取信息，缩短了作业执行时间，提高了运行效率。

❖ SP00Ling 系统特点

在一个计算问题开始之前，把计算所需要的程序和数据从输入设备上预先输入到磁盘上，这样，当进行计算时可以从磁盘上读取程序和数据（速度快），无须再访问输入设备（慢速）。同样，计算过程中，将结果先放到磁盘上保存，全部计算完成后再把全部计算结果输出到打印机上。

Spooling 系统（技术），1961 年，英国曼彻斯特大学，Atlas 机

（Simultaneous Peripheral Operation On-Line，同时的外围设备联机操作--假脱机技术）

利用磁盘作缓冲，将输入、计算、输出分别组织成独立的任务流，使 I/O 和计算真正并行

作业进入到磁盘上的输入井，系统按某种调度策略选择几个搭配得当的作业，调入主存。作业运行的结果输出到磁盘上的输出井，之后再从磁盘上的输出井将结果送到打印机。

❖ 追求目标

系统资源利用率和系统效率

吞吐率（量）：单位时间内处理作业的个数

2、分时操作系统

❖ 工作方式

一台主机连接了若干个终端，每个终端有一个用户在使用，交互式地向系统发出命令请求；系统接受每个用户的命令，并采用时间片轮转的方式处理用户的服务请求，并在终端上向用户显示处理结果；用户根据上一步运行结果发出下一道命令。

- ◆ 分时系统为用户提供交互命令
- ◆ 分时系统中采用分时方法对多个终端用户服务
- ◆ 分时方法：将 CPU 时间划为时间片
- ◆ 分时系统以时间片为单位，轮流为各个用户服务

❖ 时间片

操作系统将 CPU 的时间划分成若干个片段，称为时间片。操作系统以时间

片为单位，轮流为每个终端用户服务，每次服务一个时间片。

（其特点是利用人的错觉，使人感觉不到。）

❖ 特点

1) 同时性（多路性）

同时有多个用户使用一台计算机。

（宏观上看是多个用户同时使用一个 CPU，微观上则是多个用户在不同时刻轮流使用 CPU。）

2) “独占”性（独立性）

各个用户彼此独立，互不干扰地使用计算机，感觉不到计算机同时还在为其他用户服务，有一种独占计算机的错觉。

3) 及时性

系统对用户提出的请求能在较短时间内作出响应，让用户满意。

4) 交互性

采用人-机对话方式，用户在终端上输入、调试、运行自己的程序，能及时修改程序中的错误且直接获得结果。用户根据系统响应结果进一步提出新的请求。

（用户能直接干预作业运行的每一步）

❖ 追求目标

及时响应（衡量指标是响应时间）

响应时间：从用户发出命令到系统给予应答。

❖ 分时系统目标与多道程序目标的对比

分时系统目标：对用户请求快速反应

多道批处理目标：提高计算机系统效率

➤ **通用操作系统**

分时系统与批处理系统结合

原则：分时优先，批处理在后

“前台”：需频繁交互的作业

“后台”：时间性要求不强的作业

3、实时操作系统

❖ 实时（real time）系统

系统能对外部请求作出及时响应，并且要求在规定的严格时间范围内完成处理工作，同时给出反馈结果。

❖ 分类

第一类：实时控制系统

第二类：实时信息处理系统

❖ 主要追求目标

1) 及时性（实时性）

对外部请求在严格时间范围内作出反应和处理（强制性的）

2) 高可靠性和安全性

系统保证不出错

（因此，不追求系统资源利用率）

4、个人计算机操作系统（单用户单任务，单用户多任务）

为个人计算机使用，计算机在同一时间内为单用户服务，其追求目标是界

面友好，操作使用方便，丰富的应用软件。

5、网络操作系统

它是基于计算机网络的、实现网络通信和网络资源管理功能的操作系统。

它实际上就是在各种计算机操作系统上按网络体系结构协议标准开发的软件。应包括除原来操作系统应具有的功能外，网络管理、通信、安全、资源共享和各种网络应用。

目的：相互通信、资源共享

6、分布式操作系统

❖ 分布式系统

基于两种环境：

☞ 一种是多处理机（多 CPU）系统：紧密耦合

建立在多个 CPU 上 物理上相邻 总线或开关网连接处理器，共享主存进行通信

☞ 另一种是基于计算机网络的多计算机系统：松散耦合

建立在网络上 地理上分开，通过网络用报文（Message）连接

分布式计算机系统结构：环形，星形，树形

分布式操作系统是网络操作系统的更高级的形式，它保持了网络操作系统的全部功能，同时具备如下特征：

❖ 特征

- 1) 是一个统一的操作系统。
- 2) 资源进一步共享（最大限度的共享）。
- 3) 透明性：资源共享，分布。用户并不知道，对用户来讲是透明的。
- 4) 自治性：处于分布式系统中的多个主机处于平等地位。
- 5) 处理能力增强，速度更快，可靠性增强。

❖ 网络和分布式的区别

- 1) 分布式具有各个计算机间相互通信，无主从关系；网络操作系统的计算机有主从关系。
- 2) 分布式系统资源为所有用户共享；而网络系统为有限的共享。
- 3) 分布式系统中若干个计算机可相互协作共同完成一项任务。

1.3 研究操作系统的几种观点

一、软件的观点

作为软件来看软件的特性

☞ 外在特性：软件是一种语言，是界面，完全确定软件的使用方式。
（如：命令，系统调用等）

☞ 内在特性：软件的结构

- a. 由有那些部分组成
- b. 每个部分的功能作用
- c 各部分之间的关系，即算法

二、资源管理的观点

操作系统——资源管理者

把 OS 看作是各类资源的管理者，为用户提供一种简便、有效地使用资源的手段，在共享资源使用发生冲突时进行协调，并充分发挥各种资源的利用率。

1、资源

🔗 硬件资源：CPU，主存，辅存，输入输出设备（键盘、显示器、打印机）

🔗 软件资源：程序、数据（信息资源）

任何程序的执行都要占用计算机系统的资源。

问题：请求和使用会产生冲突，控制和协调。

不同的用户要求不同，权限不同，针对性

2、管理资源

🔗 记录资源的使用状况

（如：哪些资源空闲，好坏与否，被谁使用，使用多长时间等）

🔗 合理分配资源：

1) 分配策略

静态分配策略（在程序运行前分配，但效率不高）

动态分配策略（在程序运行过程中何时用资源，何时分配。其缺点时会出现死锁，即谁也不能用，无限等待）

2) 具体执行分配

🔗 回收资源

🔗 实现资源共享（提高资源利用率）

3、目的

1) 实现资源共享

2) 提高资源利用率

4、操作系统的功能

五大功能（四大功能再加一个作业管理），五个部分相互配合、协调工作，完成计算机软硬件资源的组织管理工作，并控制程序的执行。

CPU（进程）管理

◆ 合理分配 CPU 时间，尽可能使 CPU 忙碌，提高 CPU 使用效率

存储管理

◆ 管理主存，分配回收主存，保护主存中的信息不被破坏，提高主存利用率

文件管理

◆ 实现按名存取文件，存储、检索、共享和保护信息，管理文件存储空间

设备管理

◆ 管理各种输入/输出设备，有关技术的实现

作业管理

◆ 作业调度及作业的执行控制

用户与操作系统的接口：教材 P23-24

操作级：用户提供如何控制作业的执行。手段：作业控制语言、操作控制命令

程序级：系统功能调用（系统调用，操作系统功能子程序）

三、进程的观点

是从操作系统动态运行的角度观察操作系统，研究计算机的行为。

从这个观点来看，操作系统是由多个可同时独立运行的进程和一个对这些进程进行协调的核心组成。

进程：某一特定功能的程序的一次执行过程，它是有生命的，当它执行时进程存在，否则消亡。是操作系统中最基本的单位（资源分配的单位，运行的单位？），进程之间相互独立，相互作用，进程之间竞争使用资源。

核心：是一个管理机构（模块），负责创建和调度进程，处理进程间通信等。

四、虚拟机观点

从操作系统内部结构来看操作系统，把操作系统分成若干层，每一层完成其特定功能从而构成一个虚拟机，并对上一层提供支持；通过逐层功能扩充，最终完成整个操作系统虚拟机。而操作系统虚拟机向用户提供完全功能，完成用户请求。

（从操作系统结构出发，把操作系统分成若干个层次，每一层次都对上层次扩充形成一个虚拟机；高层次屏蔽低层次的功能细节，提供高层服务，整个操作系统由若干个虚拟机叠加而成。）

五、服务提供者的观点

在操作系统之外，从用户角度来看：

操作系统为用户提供了一组功能强大的、方便易用的命令或系统调用。

不同的操作系统提供的系统调用不全相同，大致可分为几类：教材 P24
文件操作类、资源申请类、控制类、信息维护类

第二章 操作系统的运行环境

工作环境：硬件环境、软件环境

2.1 计算机系统的结构

一、计算机系统的层次结构

1、软件系统：为使用计算机提供方便。

➤ 软件的分类：

系统软件（不可缺少）：操作系统、编译系统

操作系统：实现资源的管理和控制程序的执行。与硬件共同构成其它软件的运行基础。（要求：高效率）

支撑软件：与系统软件和硬件构成一个开发平台，可支持其它软件的开发和维护。

应用软件：领域知识与计算机技术结合，按特定需要专门编写的程序，完成特定功能。

操作系统在硬件之上，其他软件之下，直接与硬件打交道。

2、硬件系统

硬件系统由 CPU、主存、输入输出控制系统以及各种外围设备组成。

CPU：是对信息进行高速运算和控制处理的部件

主存：用于存放各种程序和数据，可被中央处理器直接访问

I/O 控制系统控制：管理外围设备与主存之间的信息传送

二、系统引导

每个计算机系统都有一个“引导程序”(或称初启程序)。

当接通计算机电源或重新启动系统时，计算机系统立即自动执行“引导程序”。

“引导程序”的工作：进行系统初始化工作，然后把操作系统的核心程序装入主存，等待某个事件发生。

当有某个事件出现，操作系统的服务程序就要处理它，处理结束后，又等待下一个事件的发生。计算机系统中会有许多不同类型的事件发生，对所发生的事件均有硬件识别且触发一个中断，把控制转移给操作系统的某个服务程序。

三、中断系统的作用

中断是实现多道程序操作系统的前提，没有中断，操作系统无法改变 CPU 的状态（即目态 - 管态之间的互换）

中断是现代计算机系统的基本设施之一，它起着通讯联络作用，协调系统对各种外部事件的响应和处理。中断是实现多道程序的必要条件。

操作系统是由中断驱动的。

1、中断系统的重要性

中断是计算机系统结构的一个重要部分，每个计算机系统都有自己的中断机制，中断机制包括硬件的中断装置和操作系统的中断处理服务程序。中断装置由一些特写的寄存器和控制线路组成，中央处理器和外围设备等识别到的事件保存在特写寄存器中，中央处理器每执行完一条指令后，均由中断装置判别是否有事件发生。若无事件发生，中央处理器继续执行指令；若有事件发生，中断装置中断原占用中央处理器的程序执行，而让操作系统的处理事件的服务程序占用中央处理器对出现的事件进行处理，待操作系统对事件处理完成后，再让原来的程序继续占用中央处理器执行。

2、例子 1：用户命令的处理过程

操作系统提供许多服务功能，例如，处理来自用户的命令、读/写某个设备上的文件、分配/收回计算机系统的资源、处理硬件或软件发现的错误以及其他控制功能。这些功能程序的执行都是由相应的事件触发引起的，当用户通过键盘或鼠标输入一条命令后，就由操作系统的命令处理器执行；当这些程序占用处理器执行时，又可请求调用操作系统的功能，这时操作系统的功能处理程序占用处理器，或启动指定的设备为调用者读/写文件，或分配/收回资源，或完成其他的控制任务，调用的功能完成后，操作系统又主动让出处理器供其他程序占用处理器，当硬件或其他软件出现错误时也可由操作系统进行适当处理，然后再继续其他程序的执行。

3、例子 2：多道程序设计技术与中断系统

一个计算机系统，尤其是采用多道程序设计技术的计算机系统，不仅有操作系统和其他的系统软件，而且还有若干应用程序。这些程序只有占用中央处理器执行时才能履行自己的职责。但是，一个中央处理器在任何时刻只能被一个程序占用。那么，应该由谁来决定哪个程序在什么时候可以占用中央处理器？我们已经知道中断装置在判别到有某个事件发生时，就会触发一个中断而让操作系统去占用处理器，操作系统对事件处理结束后，又主动让出处理器，让出的处理器应被哪个程序占用？这与所发生的事件的性质、对事件的处理情况、系统中各个程序的状态有关。因而，操作系统在让出处理器时应根据对事件处理情况从那些具备占用处理器条件的程序选择一个程序，被选中的程序就可占用处理器，直到中再一次发生事件而被中断。被中断的程序什么时候能继续占用处理器？这仍取决于事件的性质和对事件的处理情况，若它具备占用处理器的条件，则与其他程序一起等待操作系统的选择。操作系统总是按预定的策略去选择可占用处理器的程序，因此，刚被中断的程序不一定立即被选中。所以，系统中的若干程序可能交替地占用处理器。于是又出现一个新问题：怎样保证各个被中断的程序能在再一次占用处理器时继承以前执行的情况，继续执行？

4、中断响应及处理过程简介

中断装置在发现中断事件后，首先把被中断程序的断点（当前的指令地址）等保存起来，然后让操作系统的处理程序占用处理器。操作系统在处理事件之前，把被中断程序在处理器的各寄存器中设置的状态保存起来，在事件处理结束后，选中某个程序占用处理器时再把被保存的该程序的状态恢复到各寄存器中，同时把该程序的返回地址（原断点或亲折启动点）装入指令地址计数器中。于是，被选中的程序就可占用处理器，根据被中断前的情况继续执行。操作系统采用的选择策略总能使各个程序在适当的时候占用处理器，从而完成各自所担负的工作。

2.2 硬件环境

一、CPU

在多道程序系统中，为保证安全，把指令系统分为两类。

1、特权指令

只允许操作系统使用，不允许一般用户使用的指令。

（若一般用户可以使用，就会影响系统安全，影响其它用户使用。如：修改程序状态字；设置中断屏蔽；启动 I/O 设备；清内存；设置时钟；停机等）


如果一般用户程序要做与特权指令有关的工作，则需要请求操作系统代劳，即由操作系统统一管理。

2、非特权指令


除特权指令以外的指令，即一般用户可使用的指令。

3、CPU 状态

在程序状态字（PSW）中专门设置一位，它是根据运行程序使用指令的不同权限而设置。

 管态（特态）：能执行指令全集（包括特权、非特权指令），具有改变 CPU 状态的能力。操作系统在管态下运行。

（特权态，核心态，系统态）

 目态（普态）：只能执行非特权指令，用户程序在目态下运行。（如果在

目态下用户执行了特权指令，CPU 拒绝执行该指令（非法操作），产生中断，由操作系统取得控制权并处理，通知用户“程序执行非法指令”。）

（用户态）

4、CPU 状态的转换

在系统运行过程中，CPU 状态是动态改变的，时而运行于管态，时而目态。这两种状态可以相互转换。

🔗 目态到管态 其转换的唯一途径是通过中断。

🔗 管态到目态 可通过设置 PSW 指令（修改程序状态字），实现从操作系统向用户程序的转换。

我们要求用户程序中不使用特权指令，但万一用户程序中出现了特权指令怎么办？为了保证正确的操作，中央处理器有两种工作状态：管态和目态。当中央处理器处于管态时可执行包括特权指令在内的一切机器指令；当中央处理器处于目态时不允许执行特权指令。所以，操作系统程序占用中央处理器时，应让中央处理器在管态下工作，但却取到了一条特权指令，此时中央处理器将拒绝执行该指令，并形成“非法操作”事件。中断装置识别到该事件后，转交给操作系统去处理，由操作系统通知用户：“程序中有非法指令”，必须修改。

当系统启动时，硬件置中央处理器的初始状态为管态，然后装入操作系统程序。如果操作系统选择了用户程序占用处理器，则把管态转换成目态。如果中断装置发现了一个事件，则又将其置为管态，让操作系统去处理出现的事件。所以，总能保证操作系统在管态工作，操作系统退出执行时，让用户程序在目态执行。

二、I/O 结构

1、I/O 结构

在一台通用的计算机系统中，通过输入输出控制系统完成外围设备与主存之间的信息传送。各种外围设备连接在相应的设备控制器上，这些设备控制器又通过通道连接在公共的系统总线上（参见第六章图 6 - 4 - 1）。这种结构允许中央处理器和各种外围设备同时并行工作，它们都能访问共享的主存。当中央处理器和各种外围设备同时访问主存时就要竞争存储周期，主存的控制线路能保证这些访问同步有序地进行。

中央处理器按程序规定的顺序执行指令，当执行到一条“启动外设”（“启动 I/O”）的指令时，就按指令中给定的参数启动指定的设备并把控制移交给输入输出控制系统。由输入输出控制系统控制外围设备与主存之间的信息传送，而中央处理器可继续执行程序。这时，中央处理器与外围设备是并行工作的，外围设备独立工作，不需要中央处理器的干预。但由于外围设备是由中央处理器根据程序的要求而启动的，故当外围设备工作结束后，形成一个由操作系统的处理程序处理这个“输入输出操作结束”事件（通常也称为 I/O 中断事件），操作系统进行分析后就可以知道外围设备的工作情况。

如果程序 A 启动了外围设备，但它必须等待外围设备完成信息传送后才能继续执行，那么程序 A 就应处于等待状态，暂停执行指令。这时操作系统可利用中央处理器与外围设备并行工作的能力，让另一个程序 B 占用中央处理器。程序 B 执行时也可能要启动外围设备，若程序 B 启动了另一台外围设备而等待信息传送，则程序 C 可以占用中央处理器。从这里可以看到：利用硬件的中央处理器与外围设备的并行工作能力，各外围设备之间的并行工作能力，操作系统可以让多

个程序同时执行,在同一时刻各个程序各自使用计算机系统的不同资源。被启动的外围设备工作结束后就形成“ I/O 中断 ”事件,由操作系统进行分析,并让等待外围设备传送信息的程序结束等待状态,在适当的时机操作系统又会选中该程序,让它占用中央处理器继续执行。图 2 - 2 - 1 指出了中央处理器与外围设备之间的并行工作关系。

2、I/O 保护

一个程序可以在其他程度等待外围设备传送信息时占用处理器执行,在执行中如果它也用“启动者”指令去启动一台正在工作的外围设备,那么就会造成冲突。为了保护输入输出的完整性,硬件把“启动 I/O”等一类可能影响系统安全的指令定义为特权指令。特权指令只允许操作系统程序使用,用户程序不能使用特权指令。这是因为用户程序直接使用特权指令可能会造成错误,例如,用户程序使用“启动 I/O”指令请求启动磁带机读取磁带上的信息,但可能操作员错拿了另一用户的磁带,把它装在磁带机上,这时用户程序启动磁带机后,从磁带上得到的信息实际上不是自己需要的信息,就可能造成程序执行后得到的结果不正确。更糟糕的是,若用户程序启动磁带机的目的是要把一些信息记录到磁带上,于是可能覆盖了已在磁带上的信息,即把另一用户磁带上的信息破坏了。为了防止类似于这样的错误,不允许用户程序直接使用“启动 I/O”指令,只能请求操作系统代为启动。也就是说,外围设备的启动工作由操作系统统一管理。当用户请求启动一台外围设备时,首先核对该用户是否有权使用这台外围设备,若无权使用指定的外围设备,操作系统则拒绝用户的请求。若用户有权使用指定的外围设备,则操作系统检查该外围设备是否正在工作。如果外围设备正在工作,则待该外围设备一次工作结束后再为用户启动。如果该外围设备当前是空闲的,则可为用户启动它。还是以磁带机为例,当操作系统启动磁带机后,由操作系统先读出磁带上的标识信息,如果发现该磁带不是当前用户的,则请操作员更换。只有在确认了磁带机上的磁带是当前用户的,才允许把磁带上的信息传送给用户,或把用户需保存的信息记录到磁带上。

由操作系统启动外围设备不仅可保证安全地使用外围设备,正确地传送信息,而且可减少用户为启动外围设备而必须了解外围设备特性及组织启动等工作,大大方便了用户。

3、通道

为了使 CPU 从 I/O 事务中解脱出来,同时为了提高 CPU 与设备、设备与设备之间的并行度,设置了通道。

通道定义:是独立于 CPU 的专门负责数据输入/输出传输工作的处理机,对外部设备实现统一管理,代替 CPU 对输入/输出操作进行控制,从而使输入/输出操作可以与 CPU 并行操作。

引入目的:实现外设与 CPU 的并行操作,实现设备与设备的并行,从而提高整个系统的效率。

三、存储结构

在多道程序系统中同时有多个程序在内存,每个程序在内存的位置不是固定的而是随机的。

1、存储结构

主存是中央处理器能直接访问的惟一的存储空间,因而任何程序和数据必须被装入主存之后中央处理器才能对它们进行操作。主存以“字节”为单位进行编址,若干字节组成一个“字”。中央处理器可以按地址读出主存中一个字节或一个字的内容,读出的内容可以存放在中央处理器设置的内部寄存器(例如,指令寄存器、通用寄存器、各种控制寄存器)中;也可把内部寄存器的内容存储到指定地址的主存空间中。

中央处理器执行程序时,每次从主存中读出一条指令,把读出的指令存入“指令寄存器”中。然后分析指令,根据指令中指定的地址从主存读出操作数存入“通用寄存器”;根据指令中的操作码对操作数进行运算,所得到的结果可暂存在通用寄存器中,也可存储到主存中。利用控制寄存器来保证各程序(交替占用处理器时)能正确执行和系统的安全。例如,“程序状态字寄存器”记录了与当前正在执行的程序有关的系统状态和控制信息,“中断字寄存器”用来记录出现的中断事件,“基址寄存器”和“限长寄存器”用来限定程序执行时可访问的主存空间的范围……这些控制寄存器的作用将在以后的章节中详细介绍。

如果能把各种程序和数据都驻留在主存中,那么中央处理器就可以直接存取各种信息,既方便又快速。然而,一方面受主存容量的限制,不足以存储所有需要的程序和数据;另一方面主存不是一种永久性的存储设备,当电源被切断时主存中的信息就会消失。所以,大多数的计算机系统都配置了辅助存储器。辅助存储器的优点是容量大且能就久地保存信息,但是它们不能被中央处理器直接访问。如果要使用辅助存储器中的信息,应该先把信息传送到主存中,然后,中央处理器才能对信息进行操作。

磁盘和磁带是最常用的辅助存储器,辅助存储器只能与主存相互传送信息。要把主存中的信息存储到磁盘或磁带上,必须启动相应的磁盘机(也称磁盘驱动器)或磁带机;同样,要把磁盘或磁带上的信息读到主存,也必须启动相应的设备。所以,若中央处理器要用的信息在辅助存储器中,则要先启动设备把信息传送到主存,再把信息从主存读到寄存器才能处理,需花费较长的时间。因而,中央处理器存取寄存器中的信息速度最快;存取主存中的信息要通过系统总线,速度其次;要使用辅助存储器的信息,速度最慢。由于寄存器造价高,系统中不能用大量寄存器存放信息,故寄存器只用来存放临时的工作信息或系统必须的控制信息。系统被启动后把操作系统的核心程序装入主存,核心程序在计算机系统工作期间常驻在主存中。主存的其余存储空间用来存放当前需执行的程序和数据,这是可以覆盖的存储区域。大部分程序(编译程序、应用程序和操作系统的非核心程序等)都存放在辅助存储器中,需要执行时才被装入到主存中。

磁带可以存放大量信息且可永久保存,但是,由于磁带机的工作特性使得访问磁带的速度太慢,并被限制于只适合顺序存取,不提供随机存取的功能。所以,磁带主要用于备份,存放不经常使用的信息或存放不同计算机系统间进行交流的信息。

磁盘被装到高速旋转的磁盘驱动器上后,读写磁头就可随机地存取磁盘上的信息,磁盘与存储器之间可以高速地传送信息。因此,磁盘上通常存放经常要用的程序、数据、等待处理的作业信息和作业的执行结果等。因而对一个计算机系统来说,怎样对磁盘进行恰当地管理是非常重要的。磁盘有软盘、硬盘和光盘之分。软盘是一张正反两面可记录信息的盘片,它的容量较小,转速也相对较慢,但能方便地装到驱动器上和从驱动器上卸下来。硬盘是由若干盘片组成的盘组,它有一定的容量(从10兆字节到7500兆字节不等),信息传送速度可达每秒1兆字

节到 5 兆字节。光盘的信息传送速度比硬盘慢，但它的容量相当大且造价低，由于光盘比硬盘更耐用，又能像软盘那样方便装卸，因而正在被广泛采用。我们不去考虑软盘、硬盘和光盘的驱动器有什么区别，从操作系统存储信息的观点来看，它们都可以被笼统地视为普通的硬盘，仅是容量和速度上的差别。

2、存储保护

主存中往往同时装入了操作系统程序和若干用户程序，为了保证正确操作，我们希望操作系统程序不被破坏，也希望用户程序相互间不干扰。这就必须限定用户程序只能在规定的主存区域内执行，以保护各程序的安全。主存有我种管理方式，对不同的管理方式有不同的实现保护的方法，这里概要地介绍一种存储保护的方法。

每个程序在主存中占一个连续的存储空间，硬件设置两个寄存器：一个称为“基址寄存器”，另一个称为“限长寄存器”，用来限定用户程序执行时可以访问的主存空间范围。当操作系统选中某个用户程序占用中央处理器时，把该用户程序占用的主存空间的起始地址存入基址寄存器，把占用的主存空间的长度存入限长寄存器。中央处理器在目态下执行程序时，对每一个访问主存的地址都要进行核对，若关系式

基址寄存器值 ≤ 访问地址 ≤ 基址寄存器值 + 限长寄存器值

成立，则允许访问。这样就保护了该区域之外的存储信息不受到破坏，一旦程序执行中出错也不会涉及其他程序。

中央处理器在管态下执行程序时，对访问主存的地址不进行核对，以允许操作系统对各个用户程序进行管理和控制。

不允许用户随意修改基址寄存器和限长寄存器的值，只有操作系统才有权改变它们的值。硬件把修改这两个寄存器内容的指令定义为特权指令，只能在管态时执行，以防止用户程序改变寄存器的内容。

四、时钟

定时装置，是操作系统进行控制和调度的重要工具。

时钟一般分成硬件时钟和软件时钟。

硬件时钟：某个寄存器来实现。（晶体振荡器，每隔一定间隔产生脉冲频率，根据该频率产生中断，定时加 1）

绝对时钟：绝对时间，一般不会产生中断。

相对时钟：间隔时钟。如闹钟，每隔固定时间发一次中断。

（在时钟寄存器中数值减 1，设一预定值，减到 0，就产生一个间隔时钟中断，起闹钟的作用）

软件时钟：用作相对时钟，用内存单元来模拟时钟。

2.3 操作系统结构设计

一、结构设计目标

1、正确性

影响操作系统正确性的因素：

随机性：任务类型和到达系统的时间是随机的；系统中发生的各种事件是随机的，资源使用情况是随机的。

操作系统必须充分估计和把握各种不确定因素。

结构良好 保证正确性，易于验证其正确性

2、高效性

减少操作系统自身的开销（时间开销、空间开销）

特别是：核心程序的设计（关键，原则：少而精，有效灵活）

3、易维护性

增、删、改、调整

4、移植性

移植性：能否方便地把操作系统移植到一个新的硬件环境中。

原则：减少与硬件直接相关的程序量，将其独立封装。采用标准语言书写操作系统。

二、层次结构

操作系统设计方法：无序模块法、内核扩充法、层次结构法、管程设计法等。

各种设计方法的总目标：保证操作系统的可靠性。

◆ 层次结构法

把一个大型复杂的操作系统分解成若干单向依赖的层次，由各层的正确性保证整个操作系统的正确性。

从操作系统结构出发，把操作系统分成若干个层次，每一层次都对上层次扩充形成一个虚拟机；高层次屏蔽低层次的功能细节，提供高层服务，整个操作系统由若干个虚拟机叠加而成。

◆ 最大特点：整体问题局部化（分解）

◆ 优点：结构清晰、便于设计和调试、利于功能的增删改

◆ 困难：层次的划分和安排，不出现双向依赖关系

◆ 一种层次结构的例子：教材 P22 页（本）

2.4 用户与操作系统的接口

用户如何与操作系统建立联系：两类接口

一、操作员级接口

告诉操作系统控制作业执行的步骤。

提供的手段：作业控制语言 和 操作控制命令

二、程序员级接口

1、系统调用

“系统调用”是操作系统为用户程序提供了一种服务界面。

提供的手段：操作系统功能程序

操作系统编制了许多不同功能的子程序，用户程序在执行中可以调用这些子程序。由操作系统提供的这些子程序称“系统功能调用”程序，或简称“系统调用”。

“系统调用”是在管态下执行的程序。

2、系统调用使用和执行过程

- ◆ 操作系统提供若干功能子程序(系统调用),这些系统调用在管态下执行。
- ◆ 现代计算机系统硬件提供一条“访管指令”,该指令可以在目态下执行。
- ◆ 用户编制程序需要请求操作系统服务时,使用系统调用。编译程序将其转换成目标程序中的“访管指令”及一些参数。
- ◆ 目标程序执行时,当 CPU 执行到“访管指令”,产生自愿性中断,操作系统接过控制权(管态)。
- ◆ 操作系统分析相关参数,让对应的“系统调用”子程序为用户服务。
- ◆ 完成系统调用后,操作系统将 CPU 状态改变为目态,返回到用户程序继续执行。

用户程序与系统调用的转换示意,教材 P23(本)P8(专)。

3、系统调用分类

- 文件操作类

如:打开文件、建立文件、读文件、写文件、关闭文件及删除文件等。

- 资源申请类

如:请求分配主存空间、归还主存空间、分配外围设备及归还外围设备等。

- 控制类

执行中的程序可以请求操作系统中止其执行或返回到程序的某一点再继续执行。操作系统要根据程序中止的原因和用户的要求作出处理,因而这类系统调用有:正常结束、异常结束及返回断点/指定点等。

- 信息维护类

如:设置日期时间、获取日期时间、设置文件属性及获取文件属性等。

第三章 进程管理

3.1 多道程序设计

程序:具有特定功能的一组指令或语句的集合,体现了某种算法,给出了处理机的执行步骤。

一、多道程序设计

1、定义:指允许多个程序同时进入内存并运行。

2、技术支持

1) 存储保护

2) 程序浮动

3) 如何管理其他资源:当各用户在资源的使用上发生冲突时,如何处理竞争。

注意:对 CPU 只能通过调度来解决竞争问题,而对于其他资源通过“申请—分配—使用—回收”的办法进行管理,当且仅当占有 CPU 的时候才可以申请,否则要排队等待。

二、为什么引入多道程序设计

提高系统资源利用率,进而提高系统效率。

- 提高了 CPU 的利用率
- 充分利用外围设备资源
- 发挥了 CPU 与外围设备、外围设备与外围设备之间的并行工作能力

程序的顺序执行：

典型的程序：



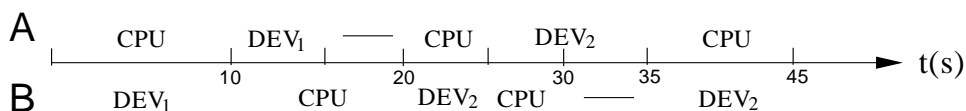
程序的并行执行：

例子：

A 程序：CPU (10 秒) DEV₁ (5 秒) CPU (5 秒) DEV₂ (10 秒) CPU (10 秒)

B 程序：DEV₁ (10 秒) CPU (10 秒) DEV₂ (5 秒) CPU (5 秒) DEV₂ (10 秒)

在顺序环境下 CPU 利用率= 40/80 = 50%
 DEV₁ 利用率=18.75%
 DEV₂ 利用率= 31.25%



在并发环境下 CPU 利用率=89%
 DEV₁ 利用率=33%
 DEV₂ 利用率=66%

多道程序设计充分利用 CPU 与设备并行工作的能力，使各种设备同时工作，增加了单位时间内的算题量。

三、采用多道程序设计应注意的问题

1、可能延长程序的执行时间

采用多道程序设计可增加单位时间的算题量，但对某一道题目来说，从开始计算到全部完成所需要的时间可能比单道时该道题目执行所需要的时间要长。

例子：P29 (本)

2、并发工作的道数与系统效率不成正比

资源量有限，程序数目越多，不一定效率越高。

在确定并行工作道数时应考虑：系统的资源配置和用户对资源的要求。

3.2 进程概念

一、进程的概念

1、定义

进程：一个程序在一个数据集合上的一次执行过程。

(进程是具有独立功能的程序关于某个数据集合上的一次运行活动，是系统进行资源分配和调度的独立单位。)

2、程序与进程之间的区别（例如：放电影的过程）

- 1) 进程是由程序和数据组成的，进程离开程序是没有意义的。
- 2) 程序是静态的，进程是动态的。
- 3) 进程有生命周期，有诞生有消亡。进程是短暂的，而程序是相对长久的。
- 4) 一个程序可以对应多个进程。

可再入程序（两个及两个以上进程共用一个程序）：可被多个进程同时调用的程序。具有下列特征：它是纯代码的，即在执行过程中自身不会改变，调用它的进程应该提供数据区（工作区）。

5) 进程有创建其它进程的功能，而程序没有。
进程比程序更能真实地描述并发执行。

3、为什么要引入进程

- (1) 提高资源利用率

把一个计算任务分解成可独立执行的子任务。

- (2) 正确描述程序的执行情况

程序的执行是走走停停的，从程序的角度已无法正确描述程序执行时的状态。（例如：编译程序）

4、进程的分类

- 1) 系统进程：执行操作系统的进程。
 - 2) 用户进程：完成用户功能的进程。
- 系统进程优先于用户进程。

二、进程的状态及其转换

1、进程的三种基本状态

进程在生命周期内处于且仅处于三种基本状态之一。

- 1) 运行态（Running）：

进程占有 CPU，在 CPU 上运行。

- 2) 就绪态（Ready）：

一个进程已经具备运行条件，但由于无 CPU 暂时不能运行的状态。（进程已得到除 CPU 以外的所有资源，一旦获得 CPU 时，立即可以运行。）

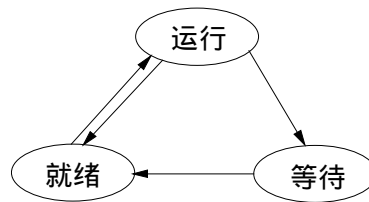
- 3) 等待态（Blocked）：

进程因等待某种事件的发生而暂时不能运行的状态。如等待 I/O 结束，即使 CPU 空闲，该进程也不能运行。

（封锁态、阻塞态、挂起态）

2、状态转换

在进程运行过程中，由于进程自身进展情况及外界环境的变化，这三种状态（可依据一定的条件）可以相互转换。



运行—等待：等待某事件发生

等待—就绪：等待的事件发生了

运行—就绪：时间片到，或有更高优先级进程出现

就绪—运行：被调度程序选中

一个实际的操作系统中，进程的状态可能更细分，状态转换关系也更复杂。

三、进程控制块（Process Control Block -- PCB）

1、概念

系统为了管理进程设置了一个专门的数据结构，用于记录进程的外部特征，描述进程的运动变化过程。

系统利用 PCB 来控制和管理进程，所以，PCB 是系统感知进程存在的唯一标志，进程与 PCB 是一一对应的。

2、PCB 的内容：记录了管理进程所必需的信息

1) 标识信息：进程标识（进程名字，进程的内部标识）；用户名

2) 说明信息：进程状态；等待原因；进程程序和数据存储信息（起始地址，长度）。

3) 现场信息：记录保存了重要寄存器（通用寄存器、控制寄存器、程序状态字寄存器）时钟等内容，用于恢复断点，让进程继续执行。

4) 管理调度信息：进程优先数；进程队列指针；消息队列指针；进程使用的资源清单；进程家族关系；进程当前打开的文件。

3、系统并发度（PCB 表）

为便于管理，系统把所有 PCB 组织在一起，并把它们放在内存的固定区域，就构成了 PCB 表。PCB 表的大小决定了系统中最多可同时存在的进程个数，个数也叫系统的并发度。

注：多道程序中的多道与系统并发度中的 PCB 个数不同。如有十个用户（多道）在上机，而每个用户有 ≥ 1 个进程。系统并发度 $>$ 多道。

4、进程的组成：程序 + 数据 + PCB（构成进程三要素）

四、进程的创建和撤消

进程是动态实体，有生命周期。

1、进程的创建

系统为一个程序分配一个工作区（存放程序处理的数据），并为该程序建立一

个进程控制块后，进程创建完成。

完成以下工作：

- (1) 建立一个 PCB (找一个空 PCB)。
- (2) 为进程分配内存等必要资源 (进程的工作区，存放程序处理的数据集)。
- (3) 填写 PCB 中各项目，例如，初始状态为“就绪态”。
- (4) 把 PCB 插入进程就绪队列。

2、进程的撤消

当进程完成特定的任务后，做以下工作：

- 1、收回进程所占有的资源 (工作区)。
- 2、撤消该进程的 PCB。

操作系统中往往设计一些进程控制原语：创建、撤消、阻塞、唤醒等。

原语：完成特定功能的、不可中断的过程。

原语：是 (由若干条机器指令构成的) 完成某种特定功能的一段程序，具有不可分割性。即原语的执行必须是连续的，不允许被中断。

【思考】

1. 如果系统中有 N 个进程，运行的进程最多几个，最少几个；就绪进程最多几个，最少几个；等待进程最多几个，最少几个？
2. 有没有这样的状态转换：等待—运行；就绪—等待？
3. 一个转换发生，有无另一个转换一定发生，找出所有的可能。
4. 日常生活中进程的例子。

五、进程队列

多个进程，单 CPU，一个进程处于运行态，其他进程处于就绪或等待态。

1、进程队列

把处于相同状态的进程的 PCB 链接在一起，构成进程队列。

- 就绪队列：一个
- 等待队列：若干 (不同等待原因不同等待队列)

2、队列组织管理

队列链接形式：单向链接，双向链接 (画图)

队首指针

出队：队首进程出队、非队首 (非队尾) 进程出队、队尾进程出队

入队：入队首、入队尾、插队 (考虑原来队列为空)

入队、出队模型：P47 页图 (本)，P18 页图 (专)

六、进程的特征

- 1、并发性：任何进程都可以同其它进程一起向前推进。
- 2、动态性：进程对应程序的执行
 - 进程是动态产生，动态消亡的。
 - 进程在生存周期内在三种态间转换。
- 3、独立性：进程是 CPU 调度的一个独立单位。

- 4、交互性：进程在执行过程中可与其它进程产生直接或间接的相互作用。
- 5、异步性：每个进程都与其相对独立的不可预知的速度向前推进。
- 6、结构性：进程程序、数据、PCB 组成。

3.3 中断系统

一、基本概述

中断是实现多道程序操作系统的前提，没有中断，操作系统无法改变 CPU 的状态（即目态 - 管态之间的互换）

中断是现代计算机系统的基本设施之一，它起着通讯联络作用，协调系统对各种外部事件的响应和处理。中断是实现多道程序的必要条件。

操作系统是由中断驱动的。

1、中断定义

CPU 在执行一个程序时，对系统发生的某个事件（程序自身或外界的原因）作出的一种反应：CPU 暂停正在执行的程序，保留现场后自动转去处理相应的事件，处理完该事件后，到适当的时候返回断点，继续完成被打断的程序。（如有必要，被中断的程序可以在后来某时间恢复，继续执行。）

事件：如读盘，盘有问题，无法读，产生中断，解决后，程序恢复，软件错误也会中断。


特点：1) 中断随机的


2) 中断是可恢复的

3) 中断是自动进行处理的

2、中断系统的有关概念

中断由软件（操作系统）硬件协同完成，硬件机构称中断装置。

 中断装置：指发现中断，响应中断的硬件。

 中断处理程序是由软件来完成的。

以上合称中断系统

中断源：引起中断发生的事件

中断寄存器：硬件为每个中断源设置寄存器，中断发生时信息被记录在寄存器中，以便分析处理（记录中断）

中断字：中断寄存器中的内容

程序状态字：控制指令执行顺序，并保留和指示与程序相关的系统状态。


基本内容

- ◆ 程序基本状态（指令地址，条件码，目态/管态，等待计算）
- ◆ 中断码：保存程序执行时，当前发生的中断事件，以便操作系统分析处理（设置中断码）
- ◆ 中断屏蔽位

程序状态字寄存器（CPU 按照其内容执行）

系统堆栈：在内存开辟的一块区域用来临时保存进程运行现场

二、中断类型

 强迫性中断

正在运行程序所不期望的，如外部请求或某些意外事故。

特点：这类中断是随机发生的，不知何时、何地发生，进程的断点可能在任意位置。

1) 硬件故障中断

电源电压故障（超出规定范围），内存出错（奇偶校验错）

2) 程序性中断：运行程序本身出现的问题引起的中断

如非法操作、定点溢出、缺页、缺段、地址越界、除数为“0”等等

3) 外部中断：由外部事件引起的中断

如时钟中断（时间片到时等）

控制台中断（由控制台发出控制信息，产生中断）

用户输入命令

4) 输入/输出(I/O)中断：主要来自输入输出控制系统

设备传输出错，传输正常结束等

🔔 自愿性中断

又称访管中断

用户在程序中有意识安排的中断，这是由于用户在编制程序时要求操作系统提供服务，有意使用“访管指令”或系统调用，所引起的中断。

特点：断点位置确定。

例如：

用户请求系统分配内存空间，请求分配设备，请求启动外围设备等

各计算机系统都提供“访管指令”作为调用操作系统功能的手段，但其名称和格式可能不同。当 CPU 执行到访管指令时，产生中断。

UNIX：trap

DOS：INT

三、中断响应

发现中断、接收中断的过程（由中断装置完成）

1) 发现中断源(识别中断原因)

提出中断请求，当有多个中断源存在时，选择优先级高的中断源。

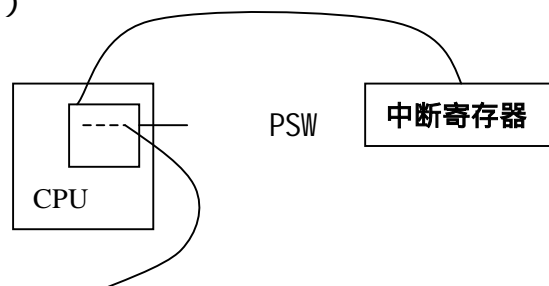
设置中断码。

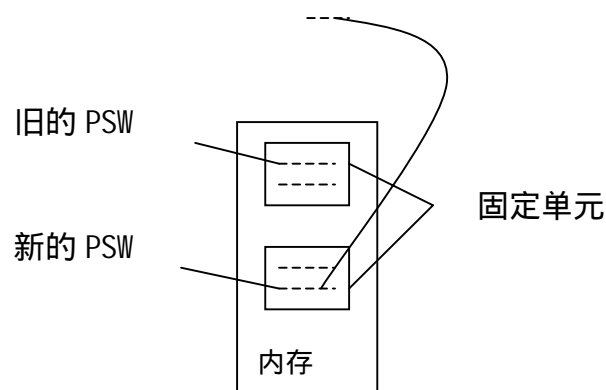
2) 保存现场，将中断向量推入系统堆栈，使中断处理结束后，程序能继续运行

3) 引出中断处理程序

具体做法：CPU 在执行每条指令后，硬件中断装置扫描中断寄存器，检查有无中断请求。如果没有，则执行下一指令；如果有中断请求，则暂停现行进程的执行，通过交换程序状态字引出中断处理程序，让操作系统的中断处理程序占用 CPU。

（交换程序状态字进入中断处理程序：保留现行 PSW 内容，将它送入内存单元中相应中断源的旧状态字；将内存中相应的新状态字单元送入 PSW 内，形成新的现行程序状态字。）





四、中断处理（中断事件的处理）

- 1) 保存被中断进程的现场(保存未被硬件保存的现场)
- 2) 识别中断具体原因
- 3) 根据中断原因处理中断事件
教材 P40 页（本）P21 页（专）
- 4) 中断返回
从程序中断处继续执行
启动一个新的程序

五、中断优先级、中断屏蔽和中断嵌套

问题：多个中断同时发生时，中断硬件装置如何响应？

1、中断优先级

中断优先级是由硬件规定的，系统根据引起中断事件的重要性与紧迫程度，将中断源划分为若干个级别。当有多个中断同时发生时，系统根据优先级决定响应中断的次序，优先响应级别高的中断，对同级中断按硬件规定次序响应。

（由硬件设计时规定的，不同系统对中断优先级的划分是不一样的。中断优先级规定了中断响应的次序，不可改变，但可调整）

中断优先级（高低顺序）：

硬件故障中断，自愿性中断，程序性中断，外部中断，输入输出中断

2、中断屏蔽

暂时禁止响应一个或多个中断请求。

中断发生时，CPU 不予响应的状态，常用于必须连续运行的程序，防止任务被中断干扰；或执行处理某一类中断时，防止其它中断干扰。

例如：中断处理程序可屏蔽比自己级别低的中断事件。

在 PSW 中设置一个中断屏蔽位，通过设置中断屏蔽指令完成开中断与关中断来进行中断屏蔽。

（中断屏蔽技术：让程序状态字中的中断屏蔽位与一些中断事件相对应，当某位有屏蔽标志时，表示封锁对相应事件的响应。于是，当中断装置检查到有中断事件后，再要查看当前 PSW 的中断屏蔽标志。若没有屏蔽，则可响应该中断；若有屏蔽标志，则暂不响应该中断，待屏蔽标志消除后再响应。）

注意：自愿性中断不能屏蔽。

3、中断嵌套及中断嵌套处理

在处理一个中断事件时，系统又响应了新的中断事件（通过堆栈来完成）。

（此时，前一个中断处理程序的执行被打断，由处理后一个事件的中断处理程序先插入执行。

出现的两个问题：

1) 优先级低的中断事件的处理打断了优先级高的中断事件的处理，使得中断事件的处理顺序与中断的响应顺序不一致。

2) 会形成多重嵌套处理，中断的嵌套处理使现场保护、程序返回等工作变得复杂。）

3.4 进程调度（CPU 调度）

问题的提出：多个进程竞争一个 CPU，怎样解决进程竞争 CPU 的问题

要解决的问题

WHAT：按什么原则分配 CPU—进程调度算法的确定

WHEN：何时分配 CPU—进程调度时机的选定

HOW：如何分配 CPU—CPU 调度过程

一、进程调度算法

1、进程调度（程序）

进程调度（程序）的职责（任务）是控制协调进程对 CPU 的竞争，即按选定的调度算法从就绪队列中选择一个进程，让它占用 CPU（即把 CPU 的使用权交给被选中的进程，把 CPU 分配给被选中进程）执行。

进程调度算法涉及到系统性能和效率，应着重考虑整个算法设计。

2、进程调度算法的选择

算法确定的原则

☆ CPU 利用率

应尽可能使 CPU 处于忙碌状态。但不希望频繁调度。

☆ 考虑等待时间，具有公平性，防止饥饿现象

等待时间：一段时间内进程在就绪队列中等待的总时间，应尽量减少这种时间。

☆ 在交互式系统（如分时系统）中追求响应时间，对用户的请求应尽快予以响应，越短越好。

☆ 在批处理系统中追求系统吞吐量

单位时间里有更多的进程完成工作，提高单位时间的处理能力。

在设计系统时如何选择进程调度算法？很难评价哪种算法是最好的，各自有自己的特性。

提问：哪一种算法？

3、各种进程调度算法

1) 先来先服务调度算法 (FCFS)

按照进程就绪的先后次序来调度进程。其优点是实现简单；缺点是没有考虑进程的优先级，没有考虑系统效率。

例子：三个进程 A、B、C，占用处理机时间分别为 3 毫秒、3 毫秒和 24 毫秒。

2) 基于优先数的调度算法 (HPF—Highest Priority First)

优先选择就绪队列中优先级最高的进程投入运行，优先级根据优先数来决定。

优先数大不一定优先级高，要求给每个进程赋予一个优先数，放入 PCB 中。

如何确定进程的优先数？（任务的紧迫性和系统效率两方面考虑）

确定优先数的方法：

静态优先数法：

在进程创建时指定优先数，在进程运行过程中优先数不变。

优点：简单，开销较小

缺点：公平性差，可能造成低优先级进程长期等待

动态优先数法：

在进程创建时指定一个优先数，但在其生命周期内优先数可以动态变化。如等待时间长优先数可变大。

优点：具有公平性

缺点：开销较大

两种占用 CPU 的方式：

(1) 可剥夺式（可抢占式）：当有比正在运行的进程优先级更高的进程就绪时，系统可强行剥夺正在运行进程的 CPU，提供给具有更高优先级的进程使用。

(2) 不可剥夺式（非抢占式）：某一进程被调度运行后，除非由于它自身原因不能继续运行，否则一直运行下去。

3) 时间片轮转调度算法 (RR—Round Robin)

规定进程一次使用处理器的最长时间：时间片。

让就绪进程按就绪的先后次序排成队列，每次总是选择就绪队列中的第一个进程占用处理器，但规定只能使用一个“时间片”。如果一个时间片用完，进程工作尚未结束，则它也必须让出处理器给其他进程使用，自己被重新排到就绪队列的末尾，等待再次运行。

把 CPU 划分成若干时间片，并且按顺序赋给就绪队列中的每一个进程，进程轮流占有 CPU，当时间片用完时，进程未执行完毕，则系统剥夺该进程的 CPU，将该进程插在就绪队列末尾；同时选择另一个进程运行。

分时系统中采用时间片轮转法

时间片选择问题：

固定时间片：大小相等

可变时间片

时间片大小的选择：

时间片过大，每个进程在一个时间片中完成，退化为先进先出算法；

时间片过小，进程切换频率高，系统开销大

与时间片大小有关的因素：

系统响应时间

就绪进程个数

CPU 能力

例子：一个分时系统 10 个终端用户同时工作，时间片为 100 毫秒，每个终端用户请求需要花费 300 毫秒处理，终端响应时间为 3 秒。

4) 分级调度算法 (多队列反馈调度算法)(本)

- * 系统中设置多个就绪队列
- * 每个就绪队列分配给不同大小的时间片，优先级高的为第一级队列，给它时间片小，随着队列级别的降低，时间片加大
- * 各个队列按时间片轮转调度算法
- * 一个新进程就绪后进入第一级队列末尾
- * 进程由于等待而放弃 CPU 后，进入等待队列，一旦等待的事件发生，则回到第一级就绪队列
- * 当时间片到后，进程放弃 CPU，进入下一级队列
- * 当第一级队列空时，就去调度第二级队列，依次类推

几点好处：

- * 运行时间短的进程只需经过前面几级队列就能得到结果，且它们被优先调度，有利于提高系统的吞吐率。
- * 运行时间长的进程在进入了低级就绪队列后可得到较长的时间片，以减少调度次数来保证系统效率。
- * 对经常使用外围设备的进程来说，每次等待外围设备传输结束后总是排入第一级就绪队列，它们会被优先调度，有利于处理器与外围设备以及外围设备之间的并行工作，从而提高资源的使用效率。

存在的问题：运行时间长的进程进入了低级就绪队列之后，有可能长时间得不到处理器。解决方案：确定一种原则，在适当的时机把低级就绪队列的进程移到高一级的就绪队列中去，提高被调度的优先级。

二、进程调度的时机

1、进程切换：一个进程让出 CPU，由另一个进程占用 CPU 的过程。

如果在时刻 T_1 进程 P_1 在运行，在时刻 T_2 进程 P_2 在运行，那么 P_1 P_2 ，就说时刻 $T_1 T_2$ 之间进行了进程切换。

进程的切换使系统中各进程都有机会占用 CPU。

进程调度的结果：原来的进程 或 新的进程运行

2、引起进程调度的时机：

- 当一个进程运行完毕（进程切换）
- 当一个进程在运行中变成等待状态（进程切换）
- 当一个进程从运行状态变成就绪状态（分时系统：时间片到）

- 当一个进程从等待态到就绪态
 - 可抢占方式：当有一个优先级更高的进程就绪
 - * 新产生一个进程
 - * 等待变成就绪
- 闲逛进程：如果没有就绪进程，系统会安排一个闲逛进程，没有其它进程时闲逛进程一直运行，在执行过程中可接收中断。
- 恢复现场：最后一步是恢复 PSW

3、进程调度过程

把选中进程的 PCB 中有关的现场信息（通用寄存器、控制寄存器和程序状态寄存器的内容）送入 CPU 相应寄存器中。

中断⇒进程状态变化⇒进程切换 P46 图（本）P24 图（专）

第四章 存储管理

4.1 概述

一、概述

1、存储器

计算机系统存储器可以分成两类：内存（简称内存）和辅助存储器（简称辅存）。内存可被处理器直接访问，但处理器不能直接访问辅助存储器。在输入输出控制系统管理下，辅助存储器与内存之间可以进行信息传送。

内存是可被处理器直接访问的，处理器是按绝对地址访问内存的。

2、存储体系 - 信息的二级存储

由于 CPU 只能直接访问内存，所以进程运行时，必须把其程序和数据存放到内存中，为了提高处理器的利用率和系统的工作效率，内存中经常存放了多个程序和数据。

各种应用程序需要占用的内存空间越来越大，内存空间就变得紧张。许多计算机系统都采用二级存储的办法，利用辅助存储（磁盘、磁带等）提供的大容量存储空间，存放准备运行的程序和数据，当需要时或内存空间允许时，随时将它们读入内存。

（日常生活中的例子：衣物保管）

目的：（使计算机系统）能存储更多的信息，更快地处理信息。

3、内存空间划分

两部分：系统区和用户区。

系统区：用来存放

- 操作系统与硬件的接口信息
- 操作系统的管理信息和程序、标准子程序等。

用户区：用来存放用户的程序和数据。

存储管理是对用户区进行管理，其目的是尽可能地方使用户和提高内存空间

的利用率。

二、存储管理的功能

计算机系统对内存的要求：容量足够；存取速度快；稳定可靠。

对内存使用的要求：合理、有效。

1、内存空间的管理、分配和回收（去配）

数据结构：“内存空间分配表”，记录内存空间的分配情况。

进行分配时查找内存空间分配表，找出足够的空闲区域分配给请求使用者。若当时的空闲区不能满足申请要求，则让申请者成为等待内存资源状态，直到有足够内存空间时再分配给它。进行去配时也要修改“内存空间分配表”，在分配表中将回收区域的标志置成“空闲”状态。

当内存中某个作业撤离或主动归还内存空间时，则收回它所占有的全部或部分的内存空间。收回存储空间的工作称“去配”。

2、实现地址转换（重定位）

（1）物理地址和逻辑地址

● 物理地址和物理地址空间

内存以字节（每个字节为8个二进制位）为单位编址，每个字节都有一个地址与其对应。假定内存的容量为 n ，则该内存就有 n 个字节的存储空间，其地址编号为 $0, 1, 2, \dots, n-1$ 。这些地址称为内存的“物理地址”（绝对地址），由物理地址构成的内存空间称“物理地址空间”。

● 逻辑地址和逻辑地址空间

在多道程序设计的系统中，内存中同时存放了多个用户作业。操作系统根据内存的使用情况为用户分配内存空间。因此，每个用户不可能预先知道其作业被存放到内存的什么位置。这样，在用户程序中就不能使用内存的绝对地址。为了方便用户，每个用户可认为自己作业的程序和数据存放在一组“0”地址开始的连续空间中。用户程序中使用的地址称为“逻辑地址”，由逻辑地址对应的存储空间称“逻辑地址空间”。

（2）重定位

用户的逻辑地址与分到的内存空间的绝对地址经常不一致，程序执行时不能按照其逻辑地址到内存中存取信息，处理器必须按照实际地址去访问内存才能保证程序的正确执行。

因此，为了保证作业的正确执行，必须根据分配给作业的内存区域对作业中指令和数据的存放地址进行重定位，即要把逻辑地址转换成绝对地址。把逻辑地址转换成绝对地址的工作称“重定位”或“地址转换”。

重定位的方式：

● 静态重定位

在装入一个作业时，把作业中的指令地址和数据地址全部转换成绝对地址。在作业执行过程中无需再进行地址转换工作，这种地址转换方式称“静态重定位”。

● 动态重定位

在装入作业时，不进行地址转换，而是直接把作业装入到分配的内存区域中。在作业执行过程中，每当执行一条指令时都由硬件地址转换机制将指令中的逻辑

地址转换成绝对地址。这种方式的地址转换是在作业执行时动态完成的，故称为“动态重定位”。

动态重定位应由软件和硬件相互配合来实现。硬件要有一个地址转换机制，该机制可由一个基址寄存器和一个地址转换线路组成。

作业所占的内存区域的起始地址被存放到“基址寄存器”中。作业执行时，处理器每执行一条指令都会把指令中的逻辑地址与基址寄存器中的值相加得到绝对地址，然后按绝对地址访问内存。

采用动态重定位时，由于装入内存的作业仍保持原来的逻辑地址，所以，必要时可改变作业在内存中的存放区域。作业在内存中被移动位置后，只要把新区域的起始地址代替原来在基址寄存器中的值，这样，作业执行时，硬件地址转换机制将按新区域的起始地址与逻辑地址相加，转换成新区域中的绝对地址，使作业仍可正确执行。

（3）程序浮动

若作业执行时，被改变了存放区域仍能正确执行，则称程序是可浮动的。采用动态重定位的系统支持“程序浮动”。

采用静态重定位时，由于被装入内存的作业信息都已经用绝对地址，故作业执行过程中是不能改变存放位置的。可见，采用静态重定位的系统不支持“程序浮动”。

存储管理必须配合硬件进行地址转换工作，把一组逻辑地址转换成绝对地址，以保证处理器的正确执行。

3、内存空间的共享和保护

在多道程序的系统中，若干个作用同时装入内存，它们共享了一个内存，在其中各自占用了某些内存区域。这些作业在执行时可能要调用共同的程序。例如，都要调用编译程序进行编译，把这个编译程序存放在某个内存区域中，各个作业要调用该编译程序时就要访问这个内存区。因此，这个内存区域又成为各作业的共享区域。

为了防止各作业相互干扰和保护各区域内的信息不被破坏，必须实现存储保护。存储保护的工作由硬件和软件配合实现，操作系统把程序可访问的区域通知硬件，程序执行时由硬件检查是否允许访问，若允许则执行，否则产生“地址越界”中断，由操作系统的中断处理程序进行处理。一般地说，对内存区域的保护可采取如下措施：

- （1）程序执行时访问属于自己内存区域中的信息，则允许它既可读，又可写。
- （2）对共享区域中的信息只可读，不可修改。
- （3）程序执行时，不允许访问其他程序的内存区域，即对于非共享区或非自己的内存区域中的信息既不可读，也不可写。

4、内存空间的扩充

如果用户编制程序时，可以不考虑内存的实际容量，即允许程序中的逻辑地址空间大于内存的绝对地址空间，那么，会使用户感到极大的方便。存储管理利用磁盘作为内存的后援，当一个大型的程序要装入内存时，仅把当前需要的部分装入，其余部分暂留在磁盘上。程序执行中要用到不在内存中的信息时，再由操作系统将其装入内存。如果内存空间不够，则可由操作系统采用覆盖技术。这样，

用户就感到计算机系统提供了容量极大的内存空间。实际上,这个容量极大的内存空间不是物理意义上的内存,而是操作系统中的一种存储管理方式,这种方式为用户提供的是一个虚拟的存储器。虚拟存储器比实际内存的容量大,起到了扩充内存空间的作用。

4.2 分区存储管理

分区存储管理是把内存中的用户区作为一个连续区或分成若干个连续区进行管理。

一、单一连续区存储管理（一个分区的存储管理）

最简单的存储管理方式。

操作系统占用一部分存储空间,其余的用户区作为一个连续的分区分配给一个作业使用。适用于单用户情况,个人计算机和专用计算机系统可采用。

1、地址转换

静态重定位

硬件支持:界限寄存器,内容为当前可供用户使用的内存区域的起始地址。

绝对地址 = 逻辑地址 + 界限寄存器值

2、存储保护

作业执行时,CPU对每条指令中的绝对地址进行检查。

若 绝对地址 \geq 界限地址 且 \leq 内存最大地址

则可执行,否则有地址错误,形成“地址越界”程序性中断事件。

限定作业在规定的内存区内执行,避免破坏操作系统的信息,达到“存储保护”的目的。

3、覆盖技术

使一个大作业能在小的空间中运行

若作业的地址空间大于用户区,要求用户把作业划分成若干段,采用覆盖技术控制作业的执行。主段是不能覆盖的,主段所占的内存区称为驻留区。其他的段保留在辅助存储器(磁盘)上,当需要某段执行时,操作系统把它装入内存的可覆盖区。

4、分时系统的存储管理技术

对换技术:让多个用户作业轮流进入内存执行。

5、单一连续分区存储管理的主要缺点

不能充分利用CPU

不能充分利用内存(低了内存的利用率)

不能充分利用外设

应用例子:IBM 7094 FORTRAN 监督系统、CP/M 系统、DJS 0520 系统。

二、固定分区管理方式

1、基本思想

把内存中可分配的用户区预先划分成若干个连续区,每个连续区的大小可以相同,也可以不同。

分区个数固定,每个分区的大小固定。

一个分区中装入一个作业，作业在执行过程中不会改变存放区域。
适用于多道程序设计系统。

2、内存空间的管理、分配与回收（去配）

- 内存分配表：说明各分区的分配情况，记录分区的起始地址和长度，并为每个分区设一个标志位。当标志位为“0”时，表示对应的分区是空闲分区，当标志为非“0”时，表示对应的分区已被占用（被哪一个程序占用）。
- 内存空间分配：顺序分配算法。

顺序查找内存分配表，找标志为“0”的分区，比较装入作业的逻辑地址空间的长度和分区长度。当能容纳作业时，则把此分区分配给该作业。当找到的分区不能容纳该作业时，则重复上述过程。

3、地址转换和存储保护

采用静态重定位方式。

由装入程序把作业中的逻辑地址与分区的下限地址相加，得到相应的绝对地址。装入程序在地址转换时要检查其绝对地址是否在指定的分区范围中。

硬件支持：下限寄存器 与 上限寄存器

装入程序对每条指令中的地址都要进行核对：

下限地址 \leq 绝对地址 \leq 上限地址

如果绝对地址在上、下限地址范围内，则可按绝对地址访问内存；如果不等式不成立，则形成“地址越界”的程序性中断事件，达到存储保护的目的。

4、内存空间的利用率的提高

固定分区存储管理方式总会使分区中有一部分区域闲置，影响了内存空间的利用率。

采用如下方法可使内存空间利用率得到改善：

- 划分分区时按分区大小顺序排列。
- 根据经常出现的作业的大小和频率划分分区。
- 按作业对内存空间的需求量排成多个作业队列，规定每个作业队列中的各作业只能依次装入对应的分区中；不同的分区中可同时装入作业；某作业队列为“空”时，该作业队列对应的分区也不能用来装其他作业队列中的作业。

这样可以防止小作业进入大分区，从而减少了闲置的内存空间量。（缺点：分区划分不合适，会造成某作业队列经常为空，使对应分区经常没有作业被装入，反而使分区的利用率不高。）

三、可变分区存储管理

1、基本思想

内存不是预先划分好的，而是当作业装入时，根据作业的需求和内存空间的使用情况来决定是否分配。若有足够的空间，则按需要分割一部分分区给该作业；否则令其等待内存空间。

分区长度不固定，且分区个数不确定。

2、内存空间的管理、分配与回收（去配）

内存分配表：两张表格组成。

已分配区表：记录已装入的作业在内存中占用分区的始址和长度，用标志位

指出占用分区的作业名

空闲区表：记录内存中可供分配的空闲区的始址和长度，用标志位指出该分区是未分配的空闲区

由于已占分区和空闲区的个数不定，因此，两张表格中都应设置适当的空栏目，分别用以登记新内存分配表。

- 常用的内存分配算法

- (1) 最先适应分配算法

每次分配时，总是顺序查找空闲区表，找到第一个能满足作业长度要求的空闲区，分割这个找到的空闲区，一部分分配给作业，另一部分仍为空闲区。

这种分配算法实现简单，但可能把大的内存空间分割成许多小的分区，形成许多不连续的空闲区，“碎片”。碎片的长度有时不能满足作业的要求，因此，碎片过多时使内存空间的利用率降低。

作为改进，可把空闲区按地址顺序从小到大登记在空闲区表中。有利于大作业的装入。

- (2) 最优适应分配算法

最优适应分配算法总是按作业要求挑选一个能满足作业要求的最小空闲区，这样保证可以不去分割一个更大的区域，使装入大作业时比较容易得到满足。在实现这种算法时，可把空闲按长度以递增顺序登记在空闲区表中。

采用最优适应分配算法，有时找到的一个分区可能只比作业所要求的长度略大一些。这样经分割后剩下的空闲区就很小了，这种极小的空闲区往往是无法使用的，影响了内存空间的使用率。

- (3) 最坏适应分配算法

最坏适应分配算法总是挑选一个最大的空闲区分割一部分给作业使用，使剩余部分的空闲空间不致于太小，仍可供分配使用。这种分配算法对中小型作业是有利的。

实现最坏适应分配算法时，空闲区表中的登记项可按空闲区长度以递减顺序排列。

- 空闲区回收

回收内存空间时，应检查是否有与归还区相邻的空闲区。若有，则应合并成一个空闲区登记。

一个归还区可能有上邻空闲区，也可以有下邻空闲区，或既有上邻又有下邻空闲区，或既无上邻又无下邻空闲区。假定作业归还的分区始址为 S ，长度为 L ，则：

- (1) 归还区有下邻空闲区

如果 $S + L$ 正好等于空闲区表中某个登记栏目（假定为第 j 栏）所示分区的始址，则表明归还区有一个下邻空闲区。这时只要修改第 j 栏记项的内容：

始址： $= S$

长度： $= \text{原长度} + L$

则第 j 栏指示的空闲区是归还区与下邻空闲区合并后的大空闲区。

- (2) 归还区有上邻空闲区

如果空闲区表中第 j 个登记栏中的“始址 + 长度”正好等于 S ，则表明归还区有一个上邻空闲区。这时要修改第 j 栏登记项的内容：始址不变，长度为原长度加上 L 。于是，归还区便与上邻空闲区合在一起了。

(3) 归还区既有上邻空闲区又有下邻空闲区

如果 $S = \text{第 } j \text{ 栏始址} + \text{长度}$

$S + L = \text{第 } k \text{ 栏始址}$

则表明归还区既有上邻空闲区，又有下邻空闲区。可以进行如下修改：第 j 栏始址不变；第 j 栏长度为“ j 栏中原长度 + k 栏中长度 + L ”；第 k 栏的标志应修改成“空”状态。于是，第 j 栏中登记的空闲区就是合并后的空闲区，而第 k 栏成为空表目了。

(4) 归还区既无上邻又无下邻空闲区

如果在检查空闲区表时，无上述三种情况出现，则表明归还区既无上邻又无下邻空闲区。这时，应找一个标志为“空”的登记栏，把归还区的始址和长度登记入表，且把该栏目中的标志位修改成“未分配”，表示该登记栏中指示了一个空闲区。

3、地址转换和存储保护

一般均采用动态重定位方式装入作业。

有硬件地址转换机构作支持。

硬件支持：设置两个专用控制寄存器。

基址寄存器：用来存放作业所占分区的起始地址

限长寄存器：用来存放作业所占分区的长度

作为现场信息的分区始址和长度被送入基址寄存器和限长寄存器中。作业执行过程中，处理器每执行一条指令都要取出该指令中的逻辑地址，当逻辑地址小于限长寄存器中的限长值时，则逻辑地址加基址寄存器值就可得到绝对地址。

当逻辑地址大于限长寄存器中的限长值时，表示欲访问的内存地址超出了所分配的分区范围。这时就不允许访问，形成一个“地址越界”的程序性中断事件，达到存储保护的目。

4、移动技术

可采用移动技术使分散的空闲区集中起来，以容纳新的作业。

移动技术也为作业执行过程中扩充内存空间提供方便，一道作业在执行中要求增加内存量时，只要适当移动邻近的作业就可增加它所占的分区长度。

移动可以集中分散的空闲区，提高内存空间的利用率，移动也为作业动态扩充内存空间提供了方便。但是：

- 采用移动技术时必须的问题

(1) 移动会增加系统开销

(2) 移动是有条件的。移动一道作业时，应先判定它是否在外围设备交换信息。若为否，则可以移动该作业；若是，则暂不能移动该作业，交换结束后才可移动。

4.3 页式存储管理

不连续存储：把逻辑地址连续的作业分散存放到几个不连续的内存区域中，并能保证作业的正确执行。既可充分利用内存空间，又可减少移动所带来的开销。

一、基本原理

页式存储管理需要硬件的支持。

1、内存划分

内存分成大小相等的许多区域，每个区称为“块”(内存块)。

2、程序划分

程序中的逻辑地址也进行分页，页的大小与块的大小一致。

3、内存分配

以块为物理单位进行内存空间分配，即把作业信息按页存放到块中。

进行存储空间分配时，根据作业的长度可确定它的页面数，一个作业有多少页，那么在把它装入内存时就给它分配多少块内存空间。这些内存块可以是不相邻的。

4、逻辑地址

两部分组成：页号和页内地址。其格式为

页号	页内地址
----	------

二、管理上的考虑

1、数据结构

● 内存分配表

位示图：指出哪些块已分配，哪些块尚未分配以及当前剩余的空闲块数。

P66 页 图

● 页表

装入作业时，为作业建立一张“页表”。

页表：给出逻辑地址中的页号与内存块号的对应关系。

作业执行时，按逻辑地址中的页号查“页表”，得到该页对应的块号，可知该页在内存中的位置，再按页内地址计算出要访问的绝对地址。

绝对地址的计算公式：

绝对地址 = 块号 × 块长 + 页内地址

2、存储空间的分配与回收（去配）

进行内存分配时，先查看空闲块数是否能满足作业要求。或不能满足，则不进行分配，作业不能装入内存；若能满足，则根据需求从位示图中找出一些为“0”的位，且把这些位置成“1”，从空闲块数中减去本次占用块数，按找到的位计算出对应的块号。

(1) 块号计算

当找到一个为“0”的位后，根据它所在的字号、位号，按如下公式可计算出对应的块号

块号 = 字号 × 字长 + 位号

(2) 块号 i 在位示图对应位置的计算

当一个作业执行结束，则应收回作业所占的内存块。根据归还的块号计算出该块在位示图中对应的位置，将占用标志修改成“0”，把归还块数加入到空闲块数中。

假定归还块的块号为 i，则在位示图中对应的位置为：

字号 = $\left[\frac{i}{\text{字长}} \right]$ ，位号 = $i \bmod \text{字长}$

三、地址转换和存储共享、保护

1、地址转换

页表指出该作业逻辑地址中的页号与所占用的内存块号之间的对应关系。页表的长度由作业拥有的页面数而定。

页式存储管理采用动态重定位方式装入作业,因而要有硬件的地址转换机构作支持。页表又是硬件进行地址转换的依据,每执行一条指令时按逻辑地址中的页号查页表,若页表中无此页号,则产生一个“地址错”的程序性中断事件;或页表中有此页号,则可得到对应的内存块号,按计算公式可转换成访问的内存绝对地址。

绝对地址的计算公式“块号 \times 块长+页内地址”,计算的结果是把块号和为绝对地址的高位地址,而页内地址作为它的低地址部分。

2、快表

为了提高存取速度,通常都设置一个小容量高速缓冲存储器。查找速度极快,但造价很高,故一般都是小容量的。

利用高速缓冲存储器存放页表的一部分,把存放在高速缓冲存储器中的部分页表称“快表”。快表中登记了页表中的一部分页号与内存块号的对应关系。根据程序执行局部性的特点,在一段时间内总是经常访问结页,若所这些页登记在快表中,则可快速查找并提高指令执行速度。

3、页的共享和保护

采用页式存储管理能方便地实现程序和数据的共享。

处理器执行指令时要核对操作要求,若想向只读块写入信息,则指令停止执行。同样地,对共享程序块不允许读或写,只能调用执行,否则也停止指令的执行。当违反规定的访问权限时,将产生一个“非法操作”的程序性中断事件。

4.3 段式存储管理

用户希望他的程序是由若干段组成的:一个程序可以由一个主程序、若干子程序、符号表、栈以及数据等若干段组成。每一段都有完整的逻辑意义,每一段的程序都可独立编制,且每一段的长度可以不同。

段式存储管理支持用户的分段观点,以段为单位进行存储空间的分配。

一、基本原理

1、用户程序划分

按程序自身的逻辑关系划分为若干个程序段,每个程序段都有一个段名,且有一个段号。段号从0开始,每一段也从0开始编址,段内地址是连续的。

2、逻辑地址

段号	段内地址
----	------

3、内存划分

内存空间被动态的划分为若干个长度不相同的区域,这些区域被称为物理段,每个物理段由起始地址和长度确定。

4、内存分配

以段为单位分配内存,每一个段在内存中占据连续空间(内存随机分割,需要多少分配多少),但各段之间可以不连续存放。

* 段式存储管理的地址结构与页式存储管理的地址结构实质上的不同：

页式存储管理提供连续的逻辑地址，由系统自动地进行分页；段式存储管理中作业的分段是由用户决定的，每段独立编程，因此，段间的逻辑地址是不连续的。

二、管理上的考虑

1、数据结构

- 内存空间分配表

同可变分区存储管理（不同点：以段为单位）

- 段表

它记录了段号，段的首（地）址和长度之间的关系。

2、内存空间的分配和回收（去配）

同可变分区存储管理

三、地址转换与存储保护

段式存储管理要有硬件的地址转换机构作支撑，段表的表目起到了基址/限长寄存器的作用。每执行一条指令时地址转换机构按逻辑地址中的段号查段表，得到该段对应的表目。

当逻辑地址中的段内地址不超过表目中设置的长度，则把表目中的起始地址与段内地址相加，就得到欲访问的内存绝对地址。如果段内地址超过了限定的长度，则产生一个“地址越界”程序性中断事件而暂停作业的执行。

地址转换机构要根据该作业的段表进行地址转换。为此，硬件设置一个“段表控制寄存器”，用来存放当前占用处理器的作业的段表始址和长度。

四、段页式存储管理

兼用分段和分页的方法，构成可分页的段式存储管理，通常被称为是“段页式存储管理”。段页式存储管理兼顾了段式在逻辑上清晰和页式在管理上方便的优点。

段页式存储管理为每一个装入内存的作业建立一张段表，且对每一段要建立一张页表。段表的长度由作业分段的个数决定，段表中的每一个表目指出本段的页表始址和长度。页表的长度由对应段所分的页的个数决定，页表中的每一个表目指出本段的逻辑页号与内存块号的对应关系。

执行指令时，地址机构根据逻辑地址中的段号查段表，得到该段的页表始址，然后根据页号查页表，得到对应的内存块号，由内存块号与逻辑地址中的页内地址可形成可访问的绝对地址。如果逻辑地址中的段号超出了段表中的最大段号或者页号超出了该段页表中的最大页号，都要形成“地址越界”的程序性中断事件。

4.4 虚拟存储器

一、虚拟存储管理的基本思想

1、问题的提出

- 一些作业太大，以至于无法将作业的全部信息装入内存。
- 装入内存的程序有些部分根本执行不到，但又占用内存空间，造成浪费。

2、解决

只装入作业的部分信息就让作业开始执行，那么当内存空间小于作业需求量

时，也可以接收作业。进而允许逻辑地址空间大于实际的内存空间。

3、好处

- 使内存空间能充分地利用
- 从用户的角度来看，好像计算机系统提供了容量很大的内存，一般称它为“虚拟存储器”，简称“虚存”。

虚拟存储器实际上是为扩大内存容量而采用的一种设计技巧，**虚拟存储器的容量是由计算机的地址结构决定的。**

4、实现虚存的前提——程序的局部性原理

- (1) 时间局部性
- (2) 空间局部性

二、页式虚拟存储管理

1、基本原理

页式虚拟存储管理是在页式存储管理的基础上实现虚拟存储器的，首先把作业信息作为副本存放在磁盘上，作业执行时，把作业信息的一部分页面装入内存，作业执行时若所访问的页面已在内存中，则进行地址转换，得到欲访问的内存绝对地址，若欲访问的页面不在内存中，则产生一个“缺页中断”，由操作系统把当前所需的页面装入内存中。

为此，在装入作业时，就应在该作业的页表中指出哪些页已在内存，哪些页还没有装入内存。

2、页面调度

(1) 基本概念

当内存中无空闲块时，为了装入一个页面而必须按某种算法从已在内存的页中选择一页，将它暂时调出内存，让出内存空间，用来存放所需装入的页面，这个工作称为“页面调度”。

(2) 常用的页面调度算法

- 先进先出调度算法 (FIFO)
先进先出调度算法总是选择最先装入内存的那一页调出。
FIFO 算法简单，易实现。
- 最近最少用调度算法 (LRU)
最近最少用算法总是选择距现在最长时间内没有被访问过的页面先调出。
- 最近最不常用调度算法 (LFU)
这种算法总是选择被访问次数少的页面调出。
- 最佳页面调度算法 (OPT)
理想的调度算法。选择以后再也不使用的页或者是距当前最长时间以后才使用的页面调出。
优点：能使缺页中断率最低，可作为衡量各种具体算法的标准
缺点：无法实现

(3) 性能问题

颠簸 或 抖动：

刚被调出的页面又立即要用,因而又要把它装入,而装入不久又被选中调出,调出不久又被装入,如此反复,使调度非常频繁。这种现象称为“抖动”或称“颠簸”。一个好的调度算法应减少和避免抖动现象。

(4) 缺页中断率

假定一个作业共有 n 页,系统分配给它的内存块是 m 块 (m 、 n 均为正整数,且 $1 \leq m \leq n$)。因此,该作业最多有 m 页可同时被装入内存。如果作业执行中访问页面的总次数为 A ,其中有 F 次访问的页面尚未装入内存,故产生了 F 次缺页中断。现定义: $f = F/A$, f 称为“缺页中断率”。

显然,缺页中断率与缺页中断的次数有关。

影响缺页中断率的因素:

✓ 分配给作业的内存块数

分配给作业的内存块数多,则同时装入内存的页面数就多,反之,缺页中断率就高。

✓ 页面的大小

页面大,就减少了缺页中断的次数,降低了缺页中断率。反之,缺页中断率就高。

✓ 程序编制方法

程序编制的方法不同,对缺页中断的次数有很大影响。缺页中断率与程序的局部化程度密切相关。一般说,希望编制的程序能经常集中在几个页面上进行访问,以减少缺页中断率。

例子:教材 P77 (本)

✓ 页面调度算法

三、段式虚拟存储管理

段式虚拟存储管理仍以段式存储管理为基础,为用户提供比内存实际容量大的虚拟空间。段式虚拟存储管理在段表中增设段是否在内存的标志以及各段在磁盘上的位置,已在内存的段要指出该段在内存中的起始地址和占用内存区长度。

作业执行中访问某段时,由硬件的地址转换机构查段表,若该段在内存,则立即把逻辑地址转换成绝对地址。若该段不在内存中,则形成“缺断中断”,由操作系统处理这个中断。处理的办法是:

(1) 查内存分配表,找出一个足够大的连续区以容纳该分段,如果找不到足够大的连续区则检查空闲区的总和,若空闲区总和能满足该段要求,那么进行适当移动将分散的空闲区集中;若空闲区总和不能满足该段要求,可把内存中的一段或几段调出,然后把当前要访问的段装入内存。

(2) 段被移动、调出和装入后都要对段表中的相应表目作修改。

(3) 新的段被装入后应让作业重新地被中断的指令,这时就能找到要访问的段,可以继续执行下去。

第五章 文件管理

5.1 概述

文件管理的主要工作：管理用户信息的存储、检索、更新、共享和保护。
操作系统为用户提供“按名存取”的功能。

一、概述

1、文件

文件：逻辑上具有完整意义的信息集合。

每个文件都要用一个名字作标识，称为“文件名”。

2、文件分类

文件系统在管理文件时还要识别和区分文件的类型，如果是文件系统所确认的文件类型，则可根据类型对文件进行合理的操作。

(1) 按用途分类

系统文件、库文件和用户文件。

(2) 按保护级别分类

根据限定的使用文件的权限：执行文件、只读文件和读写文件等。

(3) 按信息流向分类

物理设备的特性决定了文件信息的流向：输入文件、输出文件和输入输出文件。

(4) 按存放时限分类

根据系统保留文件的时间：临时文件、永久文件和档案文件。

(5) 按设备类型分类

根据文件存储介质的设备类型：磁盘文件、磁带文件、卡片文件和打印文件等。

(6) 按文件的组织结构分类

由用户组织的文件称逻辑文件：流式文件和记录式文件。

文件在存储介质上的组织方式称文件的物理结构（物理文件）：顺序文件、链接文件和索引文件等。

二、文件存取方式

1、文件存取方式

用户在一个文件上的操作：“读”和“写”。

用户要求写一个文件时，文件系统便把用户组织好的逻辑文件保存到存储介质上；用户要求读一个文件时，文件系统就要从存储介质上取出文件信息并把它存放到内存中。（存储介质 内存 或 内存 存储介质）

从对文件信息的存取次序考虑，存取方式有：顺序存取和随机存取。

顺序存取：对文件中的信息按顺序依次进行读写的存取方式；

随机存取：可以按任意次序随机地读写文件中的信息。

2、存储介质

存储介质：可用来记录信息的磁带、硬磁盘组、软磁盘片、卡片等。

存储介质和存储设备是不同的概念：存储介质可以从存储设备上卸下来，也可以把存储介质装到相应的存储设备上。

存储介质的物理单位：卷。

例如，一盘磁带、一张软盘片、一个磁盘组都可称为一个卷。

单文件卷：一个卷上可以保存一个文件；

多文件卷：一个卷上可以保存多个文件；
多卷文件：可以把一个文件保存在多个卷上；
多卷多文件：多个文件保存在多个卷上。

存储设备与内存之间进行信息交换的物理单位：块。

块（物理记录）：存储介质上连续信息所组成的一个区域。

每次总是把一块或几块（整数块）信息读入内存，或是把内存中的信息写到一块或几块中。

块大小的划分应考虑：用户要求、存储设备类型、信息传输效率等多种因素。

磁盘上：块的长度一般定在 32 到 4096 字节之间，在 MS DOS 和 UNIX 系统中，磁盘上的块长都定为 512 个字节。

磁带上：块的大小可以根据用户的要求来划分，块的大小原则上没有限制，块的长度应适中。块长过小时不易区分是干扰信息，还是记录信息，块长过大时对产生的误码就难以发现和校正。一般地说，磁带上的分块以 10 至 32768 个字节长度为宜。

对磁盘机、磁带机等外围设备，由于启停机械动作的要求，两个相邻块之间必须留有间隙。间隙是块之间不记录代码信息的区域，可以利用间隙确定“块”所在的位置。

3、存取方式的采用所取决的因素

（1）文件的使用

文件的性质决定了文件的使用，也决定了存取方式的选择。

例如，源程序 或 学生成绩档案文件。

（2）存储介质的特性

磁带机：适合顺序存取。

（总是从磁头的当前位置开始读写磁带上的信息。当磁头读写了第 i 块的信息后，走过其后的间隙就到达了第 $i+1$ 块的位置。当磁带机继续工作时，一定是读写第 $i+1$ 块的信息。）

磁盘：既可采用顺序存取方式，又可采用随机存取方式。

磁盘机是一种可按指定的块地址进行信息存取的设备。

磁盘地址：用“柱面号、磁头号、扇区号”三个参数表示。（磁盘机能根据给定的地址带动读写磁头到达指定柱面后，让指定的磁头存取指定扇区上的信息。）

在建立文件时，应定义好存取方式，使用文件时必须与预定义的存取方式一致。

三、文件系统的功能

1、文件系统

是操作系统中统一管理信息资源的一种软件，管理文件的存储、检索、更新，提供安全可靠的共享和保护手段，并且方便用户使用。

2、文件系统的功能

(1) 目录管理

文件目录是实现按名存取的一种手段。

建立一个新文件，应把与该文件有关的一些属性登记在文件目录中；

读一个文件，应从文件目录中查找指定文件是否存在并核对是否有权使用。

一个好的目录结构应既能方便检索，又能保证文件的安全。

(2) 文件的组织

用户按信息的使用和处理方式组织文件，称为文件的逻辑结构或称为逻辑文件。把逻辑文件保存到存储介质上的工作由文件系统来做，这样可减轻用户的负担。根据用户对文件的存取方式和存储介质的特性，文件在存储介质上可以有多种组织形式。把文件在存储介质上的组织方式称为文件的物理结构或称为物理文件。因此，当用户要求保存文件时，文件系统必须把逻辑文件转换成物理文件，而当用户要求读文件时，文件系统又要把物理文件转换成逻辑文件。

(3) 文件存储空间的管理

要把文件保存到存储介质上时，必须记住哪些存储空间已被占用，哪些存储空间是空闲的。文件只能保存到空闲的存储空间中，否则会破坏已保存的信息。当文件没有必要再保留而被删除时，该文件所占的存储空间应成为空闲空间。

(4) 文件操作

为了保证文件系统正确地存储和检索文件，规定了在一个文件上可执行的操作，这些可执行的操作统称为“文件操作”。文件系统提供的基本文件操作有建立文件、打开文件、读文件、写文件、关闭文件和删除文件等。“文件操作”是文件系统提供给用户使用文件的一组接口，用户调用“文件操作”提出对文件的操作要求。

(5) 文件的共享、保护和保密

在多道程序设计的系统中，有些文件是可以共享的，例如，编译程序、库文件等。实现文件共享既节省文件的存放空间，又可减少传送文件的时间，但必须对文件采取安全保护措施。既要防止有意或无意地破坏文件，又要避免随意地剽窃文件。

5.2 文件目录

一、概述

1、目录项

记录了系统为管理文件所需的有关信息。包括文件名、存储介质上的位置，以及如何和管理文件等信息。（文件控制块）

一般地说，目录项应包含如下内容：

(1) 有关文件存取控制的信息。例如，用户名、文件名、文件类型、文件属性（可写、只能读、只可执行等）。

(2) 有关文件结构的信息。例如，文件的逻辑结构、文件的物理结构、记录个数、文件在存储介质上的位置等。

(3) 有关文件管理的信息。例如，文件的建立日期、文件被修改的日期、文件保留期限和记账信息等。

2、文件目录

- ◆ 把所有的目录项有机地组织在一起，就构成了文件目录。
- ◆ 文件目录是用于检索文件的，它是文件系统实现按名存取的重要手段。
- ◆ 当用户要求使用某个文件时，文件系统可顺序查找文件目录中的目录项，通过比较文件名，可找到指定文件的目录项，根据该目录项中给出的有关信息可进行核对使用权限等工作，并读出文件供用户使用。
- ◆ 文件目录的组织和管理应便于检索和防止冲突。

3、目录文件

系统中有许多的文件目录（或子目录），文件目录需要长期保存，为了对文件目录进行管理，通常把文件目录也作为文件保存在外存中。

目录文件：由文件目录组成的文件。

值班目录：把当前正在使用文件的文件目录存放在内存中，称为值班目录。

（如，UNIX 等系统。因为任何一个用户，在一段时间里只使用少数文件，涉及到少量文件目录。）

（目录文件可以像其他文件一样进行读写等处理。当需要时可把目录文件中的有关文件目录读到内存进行检索或修改，也可把内存中的文件目录写回到外存上的目录文件中。这样，既不占用太多的主存空间，又可减少搜索目录的时间。）

二、一级目录结构

所有文件都登记在同一个文件目录中。

一级目录结构简单，管理方便。

要求在文件目录中登记的各个文件都有不同的文件名。（重名问题）

三、二级目录结构

为每个用户建立一个独立的文件目录。

第一级为主文件目录，主文件目录以用户名为索引，对每个用户都设置一个指向用户文件目录的指针。第二级为用户文件目录，用户文件目录为本用户的每一个文件设置一个目录项。

四、树形目录结构——多级目录结构

允许用户在自己的文件目录中根据不同类型的文件再建立子目录。

1、路径名

在树形目录结构中，每一个文件都有一个从根到叶的路径。

路径名：从根目录出发到某个文件的通路上所有各级子目录名和该文件名的顺序组合称为文件的路径名，在各级子目录名和文件名之间可用“/”隔开。

每个文件都有一个惟一的路径名，用户存取文件时必须给出文件所在的路径名。文件系统根据用户指定的路径名检索各级目录，从而确定文件所在的位置。

2、当前目录

（1）为什么引入当前目录？

由于查找文件总是从根目录开始，因而查找的时间较长。事实上，用户在一段时间内会经常访问一个子目录下的文件。为了提高效率和方便用户，文件系统引进了“当前目录”的概念。

(2) 当前目录

为了提高文件检索速度，文件系统向用户提供了一个当前正在使用的目录，称为当前目录。

系统初始启动后，当前目录就是根目录。当前目录可根据需要任意改变，用户可以用“改变当前目录”命令指定自己当前的工作目录。查找一个文件可从当前目录开始，使用相对路径名；当前目录一般存放在内存。（文件系统可以根据用户的要求从目录文件中找出用户的当前目录，把当前目录读入内存作为值班目录。）

3、绝对路径名和相对路径名

有了当前目录后，文件系统把路径各分成两类：绝对路径名和相对路径名。

绝对路径名：指出了从根目录开始跟随的一条指向指定文件的路径；

相对路径名：指出了从当前目录出发到指定文件的路径。

（如果文件就在当前目录中，则存取文件时只要指出文件名就行，文件系统将在当前目录中寻找该文件。如果文件不在当前目录中，但在当前目录的下级目录中，则可用相对路径名指定文件，文件系统就从当前目录开始沿着指定的路径查找该文件。）

使用相对路径名可以减少查找文件所花费的时间。

4、普通文件和目录文件的目录项

在树形目录结构中，根目录或子目录中的目录项可能指向文件，也可能指向下一级子目录。由于每个目录项都有相同的形式，怎样区分它指向的是文件，还是子目录呢？我们可以在目录项中用一个二进制位区分该目录项所指向的文件（当二进制位取值为“0”时），还是子目录（当二进制位取值为“1”时）。

4、树形目录结构的优点

(1) 解决了重名问题

允许在不同的子目录中使用相同的名字命名文件或下级子目录。这样，系统在检索时使用的路径名是不同的，故对同名文件不会引起混淆。

(2) 有利于文件的分类

系统或用户可以把不同类型的文件登录在不同的子目录下，并可按层次建立子树。

(3) 提高检索文件的速度

利用当前目录和相对路径不仅方便用户，而且系统从当前目录开始检索文件，缩短了检索路径，提高了检索速度。

(4) 能进行存取权限的控制

在子目录中可规定存取权限，在检索文件时核对存取权限，避免一个用户未经授权就存取另一个用户的文件，保证了用户文件的私有性，可实现对文件的保护和保密。

5.3 文件的组织结构

一、文件的组织结构

1、文件的组织结构

是指文件的构造方式。用户和文件系统往往从不同的角度对待同一个文件。

2、用户的角度

- 用户：从使用的角度，按信息的使用和处理方式组织文件。
- 文件的逻辑结构：由用户构造的文件称为文件的逻辑结构或称为逻辑文件。

3、文件系统的角度

- 文件系统：从文件的存储和检索的角度，根据用户对文件的存取方式和存储介质的特性组织文件，决定用户文件存放在存储介质上的方式。
- 文件的物理结构：文件在存储介质上的组织方式称为文件的物理结构或称为物理文件。

文件的物理结构对用户来说是不必关心的，但对文件系统来说却是至关重要的，因为它直接影响存储空间的使用和检索文件信息的速度。

把逻辑文件保存到存储介质上的工作由文件系统来做，这样可减轻用户的负担。根据用户对文件的存取方式和存储介质的特性，文件在存储介质上可以有多种组织形式。

用户按逻辑结构使用文件，文件系统按物理结构管理文件。因此，当用户请求读写文件时，文件必须实现文件的逻辑结构与物理结构之间的转换。

二、文件的逻辑结构

1、流式文件

构成文件的基本单位是字符，文件是有逻辑意义的一串信息组成。

对流式文件，用户常常以长度或特殊字符提出读取文件信息的要求。

例如，源程序文件。

好处：提供很大的灵活性

2、记录式文件

用户对文件内的信息按逻辑上独立的含义再划分信息单位，每个单位称为一个逻辑记录（简称记录）。一个逻辑文件是由若干个逻辑记录组成的，称为记录式文件。记录式文件中的逻辑记录可依次编号，其序号称为逻辑记录号（简称记录号）。

例如，学生成绩文件。

对记录式文件，逻辑记录是文件内可以独立存取的最小信息单位。

主键：惟一标识某个逻辑记录的数据项。

次键：逻辑记录中除主键外的其他各个数据项，利用次键能快速找出具有某一特性的所有记录。

二、文件的物理结构

1、磁带文件的组织

顺序结构：一个文件在逻辑上连续的信息存放到存储介质的依次相邻的块上，称为顺序结构。

采用顺序结构的文件称为“顺序文件”。磁带上的文件都采用顺序结构。

例子，磁带上的每个文件都有文件头标、文件信息和文件尾标三个组成部分。为了能方便、快速地检索磁带文件，在磁带上的各类信息之间用一个称做“带标”的特殊字符将其隔开，最后用两个带标表示磁带上的有效信息到此结束。

2、磁盘文件的组织

(1) 顺序结构 (顺序文件 或 连续文件)

一个文件在逻辑上连续的信息被存放到磁盘上依次相邻的块上。把顺序结构的文件称为“顺序文件”或“连续文件”。

逻辑记录顺序与磁盘块的顺序相一致。

优点：存取速度快。

支持顺序存取和随机存取。

存在的问题：

- (1) 磁盘存储空间的利用率不高。
- (2) 对输出文件很难估计需多少磁盘块。
- (3) 影响文件的扩展。

(2) 链接结构 (链接文件 或 串联文件)

一个文件的信息存放在若干不相邻的物理块中，各块之间通过指针连接，前一个物理块指向下一个物理块。

链接结构的特点是每个磁盘块中都要有一个指针指向链接文件中的上一个磁盘块，最后一块中的指针可用特殊字符（例如“-1”）表示文件到此结束。

把链接结构的文件称为“链接文件”或“串联文件”。

优点：解决了顺序结构中的所有问题。

磁盘上所有空闲块都可以被利用；

建立文件时也不必事先考虑文件的长度，文件可继续扩展；

便于在文件的任何位置插入一个记录或删除一个记录。

缺点：

采用随机存取方式是低效的。

可靠性问题，如指针出错。

链接指针占用一定的空间。

读出一块信息时，应将其中的指针分离出来，保证用户使用信息的正确性。

(3) 索引结构 (索引文件)

一个文件的信息存放在若干不相邻物理块中，系统为每个文件建立一个专用数据结构--索引表，并将指示每个逻辑记录存放位置的指针集中在索引表中。

采用索引结构的文件称为索引文件。

对索引文件既可采用顺序存取方式，又可采用随机存取方式。

优点：能方便地实现文件的扩展、记录的插入和删除。

缺点：必须增加索引表占用的空间和读写索引表的时间。

索引表的管理：当索引表非常大时，需要多个磁盘块存放，各磁盘块之间可用指针链起来。当随机存取某个记录时，可能要沿链搜索才能找到该记录的存放地址，很费时间。

解决方案：UNIX 的多级索引结构。

三、记录的成组和分解

问题的提出：逻辑记录的大小往往与存储介质分块的大小不一致。

1、记录的成组

把若干个逻辑记录合成一组存放一块的工作称“记录的成组”，每块中的逻辑记录个数称“块因子”。

2、记录的分解

从一组逻辑记录中把一个逻辑记录分离出来的操作称“记录的分解”。

由于读写存储介质上的信息以块为单位，而用户处理信息要以逻辑记录为单位，所以当逻辑记录成组存储后，用户要处理记录时必须执行记录的分解操作。

记录的成组与分解操作都要使用内存缓冲区。

记录的成组和分解是以设立内存缓冲区和操作系统增加成组分解操作的功能为代价，来提高存储介质的利用率和减少启动设备的次数。

5.4 磁盘存储空间的管理

一、位示图

对每个磁盘可以用一张位示图指示磁盘空间的使用情况。一个磁盘的分块确定后，根据总块数决定位示图由多少字组成，位示图中的每一位与一个磁盘块对应，某位为“1”状态表示相应块已被占用，为“0”状态的位所对应的块是空闲块。假设：每个盘面8磁道，4个扇区。

$$\text{块号} = i \times 32 + j$$

$$\text{已知块号} : M = [\text{块号}/32], N = \text{块号} \bmod 32$$

- 柱面号 = M
- 磁头号 = $[N/4]$
- 扇区号 = $N \bmod 4$

$$\text{块号} = \text{柱面号} \times 32 + \text{磁头号} \times 4 + \text{扇区号}$$

$$\text{字号} = [\text{块号}/32]$$

$$\text{位号} = \text{块号} \bmod 32$$

二、空闲块表

系统为每个磁盘建立一张空闲块表，表中每个登记项记录一组连续空闲块的首块号和块数，空闲块数为“0”的登记项为“空”登记项。

适合采用顺序结构的文件。

三、空闲块链表

1、单块连接

把所有空闲块用指针连接起来，每一个空闲块中都设置一个指向另一个空闲块的指针，所有的空闲块就构成了一个空闲块链。系统设置一个链首指针，指向链中的第一个空闲块，最后一个空闲块中的指针为“0”。

效率较低，麻烦费时。

2、成组连接

把空闲块分成若干组，把指向一组中各空闲块的指针集中一起。

这样既可方便查找，又可减少为修改指针而启动磁盘的次数。

UNIX 系统：采用空闲块成组连接的方法。

UNIX 系统把每 100 个空闲块作为一组，每一组的第一个空闲块中登记下一组空闲块的块号和空闲块数，余下不足 100 块的那部分空闲块的块号及块数登记在一个专用块中，登记最后一组块号的那个空闲块其中第 2 个单元填“0”，表示该块中指出的块号是最后一组的块号，空闲块链到此结束。

系统初始化时先把专用块内容读到内存，当需分配空闲块时，就直接在内存中可找到哪些块是空闲的，每分配一块后把空闲块数减 1。但要把一组中的第一个空闲块分配出去之前应把登记在该块中的下一组的块号及块数保存到专用块中。

当一组空闲块被分配完后，则再把专用块的内容读到内存，指出另一组可供分配的空闲块。当归还一块时，只要把归还块的块号登记到当前组中且空闲块数加 1。如果当前组已满 100 块，则把内存中的内容写到归还的那块中，该归还块作为新组的第一块。假设初始化时系统已把专用块读入内存 L 单元开始的区域中，分配和回收的算法如下：

- 分配一个空闲块

查 L 单元内容（空闲块数）：

当空闲块数 > 1 $i := L + \text{空闲块数}$ ；

从 i 单元得到一空闲块号；

把该块分配给申请者；

空闲块数减 1。

当空闲块数 = 1 取出 L + 1 单元内容（一组的第一块块号或 0）；

其值

= 0 无空闲块，申请者等待

不等于零 把该块内容复制到专用块；

该块分配给申请者；

把专用块内容读到主存 L 开始的区域。

- 归还一块

查 L 单元的空闲块数；

当空闲块数 < 100 空闲块数加 1；

$j := L + \text{空闲块数}$ ；

归还块号填入 j 单元。

当空闲块数 = 100 把主存中登记的信息写入归还块中；

把归还块号填入 L + 1 单元；

将 L 单元置成 1。

采用成组连接后，分配回收磁盘块时均在内存中查找和修改，只是在的一组空闲块分配完或空闲的磁盘块构成一组时才启动磁盘读写。比单块连接方式效率高。

5.5 文件的使用

用户应遵照系统规定和提供的手段使用文件。

一、文件操作

1、文件操作

文件系统提供给用户使用文件的一组接口，用户通过调用“文件操作”提出对文件的操作要求。

2、六种主要的文件操作

- (1) 建立文件
- (2) 打开文件
- (3) 读文件
- (4) 写文件
- (5) 关闭文件
- (6) 删除文件

二、文件的使用

1、文件使用

为保证对文件的正确管理和文件信息的安全可靠，规定了使用文件的操作步骤。

“打开文件”、“建立文件”：验证用户对文件的使用权，并为用户做好使用文件前的准备工作。

“关闭文件”：让用户向系统归还文件的使用权。

2、使用文件的操作步骤

- (1) 读文件
- (2) 写文件
- (3) 删除文件

3、隐式使用文件的方法

5.6 文件的共享、保护和保密

一、文件的共享

1、文件共享

指一个文件可以让指定的多个用户共同使用。

好处：节省存储空间

免除系统复制文件的工作

2、共享文件的管理

共享文件的使用

- (1) 不允许同时使用
- (2) 可以同时使用

二、文件的保护

1、文件保护

防止文件被破坏。

2、造成文件被破坏的原因

(1) 系统故障

硬件故障，软件失误

(2) 用户在使用共享文件时发生错误引起的

3、文件系统提供的保护措施

(1) 防止因系统故障造成的破坏

建立副本：把同一个文件保存到多个存储介质上

缺点：p104

定时转储：每隔一段时间把文件转储到其他存储介质上，当文件系统发生故障时，可用转储的文件进行复原。

UNIX 采用定时转储方法保护文件，提高文件系统的可靠性。

(2) 防止用户因共享文件造成的破坏

防止对共享文件的非法使用：提供文件的使用权限（只读，读写，执行，删除）

- 采用树型目录结构

- 存取控制表

用二维矩阵列出每个用户对每个文件或子目录的存取权限。

A_{ij} ：第 i 个用户对第 j 个文件的存取权限

缺点：系统开销大

- 文件使用权限

用户分类：文件主，伙伴，一般用户。

对每一类用户规定使用文件的权限。

UNIX 采用这种方法。

`rw-rw-rw-`

三、文件的保密

1、文件保密

防止未经文件拥有者授权而窃取文件。

（规定文件的使用权限只能在一定程度上起到文件保密的作用）

2、文件保密措施

(1) 隐藏文件目录

(2) 设置口令

(3) 使用密码（文件加密）

第六章 设备管理

6.1 概述

一、概述

1、设备管理与文件系统是密切相关

文件系统：按名存取文件，用户把信息组织成逻辑文件，由文件系统实现逻辑文

件与物理文件之间的映射，完成文件信息的存取必须启动相应的外围设备。

设备管理：启动外围设备和控制其工作

文件管理实现文件存取前的准备工作，而文件的物理存取由设备管理实现。

2、设备分类

(1) 存储型设备和输入输出型设备

程序执行中经常启动外围设备，把存储介质上的信息读到内存进行处理，或把内存的信息传送到存储介质上。

输入输出操作：主存储器与外围设备之间的信息传送操作

存储型设备：磁带机、磁盘机等，输入输出操作的信息传输单位为“块”

输入输出型设备：显示器、输入机、打印机等，输入输出操作的信息传输单位为“字符”

(2) 独占设备、共享设备和虚拟设备

从使用角度看。

独占设备：只能让一个作业独占使用的设备，例如，输入机，磁带机和打印机等。通常采用静态分配方式，利用率较低。

共享设备：可以让几个作业同时使用的设备，例如，磁盘。“同时使用”的含义是指多个作业可以交替地启动共享设备。

(理解对共享设备可同时使用的含义：对共享设备允许多个作业同时使用，而不是让一个作业在整个执行期间独占。这里“同时使用”的含义是指多个作业可以交替地启动共享设备，当一个作业共在使用设备时其他作业暂不能使用，即每一时刻仍只有一个作业占用，但当一个作业暂时不用时其他作业就可使用。)

虚拟设备：用共享设备来模拟独占设备的工作，把独占设备改造成可共享的，以提高设备的利用率，这种模拟的独占设备称为虚拟设备。

二、独占设备的分配

1、设备的绝对号和相对号

(1) 设备绝对号和相对号

绝对号：计算机系统为每一台设备确定一个唯一编号，以便区分和识别不同的设备。

相对号：为了避免使用时的混乱，用户把自己使用的若干台同类设备给出编号，即由用户在程序中定义的设备编号。

(为什么用户使用设备时采用“相对号”？在多道程序设计系统中，因为用户无法知道哪台设备被其他用户占用了，哪台设备是空闲的。所以，一般情况下用户不直接使用设备的绝对号。用户可以向系统说明所要使用的设备类型。至于实际使用哪一台，由系统根据该类设备的分配情况来决定。有时用户可能要求同时使用几台同类设备，为了避免使用时的混乱，用户可以把自己要求使用的若干台同类设备给出编号，由用户在程序中定义的设备编号称设备的“相对号”。)

(2) 独占设备的指定方式

两种

第一种：绝对号

如果用绝对号来指定设备,系统应该把与绝对号对应的那台设备分配给作业。如果指定的这台设备已经被其他作业占用或者有故障时,则作业的申请要求就得不到满足。于是,该作业就暂时不能装入主存储器。

第二种：设备类、相对号

通常,申请设备时不是具体指定要哪台设备,而是提出要申请哪类设备多少台,且在用户程序中用“设备类、相对号”提出使用设备的要求。好处:使设备分配的适应性好、灵活性强(因为,1、系统只要从指定的哪一类设备中找出“好的且尚未分配的”设备来进行分配。2、万一分配给用户的设备在使用中出了故障,系统可以从同类设备中找另一台“好的且尚未分配的”设备来替换。)

系统为用户分配了具体设备后,建立“绝对号”与“设备类、相对号”的对应关系。这样,系统根据用户程序执行时给出的使用要求就能知道实际应启动哪台设备。

2、设备独立性

用户编制程序时不必指定特定的设备,而是在程序中使用“设备类、相对号”定义逻辑设备。程序执行时系统将用户指定的逻辑设备转换成与其对应的具体物理设备,并启动该物理设备工作。于是,用户编制程序时使用的设备与实际使用哪台设备无关,这种特性称为“设备的独立性”。

3、独占设备分配策略

静态分配策略

设备分配表:用来记录计算机系统所配置的独占设备类型、台数以及分配情况等

由“设备类表”和“设备表”两部分组成。

设备类表:设备类、台数、现存台数、设备表始址

设备表:绝对号、状态(好/坏)、分配情况(已/未分配)、占用作业名、相对号

当作业申请某类设备时,系统先查“设备类表”,如果该类设备的现存台数可以满足申请要求,则从该类设备的“设备表”始址开始依次查该类设备在设备表中的登记项,找出“好的且未分配的”设备分配给作业,分配后要修改设备类表中现存台数。把分配给作业的设备标志改成“已分配”且填上占用该设备的作业名和作业程序中定义的相对号,然后,把设备的绝对号与相对号的对应关系通知用户,以使用户在分配到的设备上装上存储介质。

当作业执行时向系统提出使用设备要求,系统根据使用要求中的设备类得到该类设备的设备表始址,再根据作业名和相对号比较设备表中的登记项可得到设备的绝对号,然后启动该设备把作业需要的读出,或把作业产生的结果保存起来。

当作业执行结束撤离时应归还所占的设备,系统根据作业名查设备表,找出作业占用设备的登记栏,把标志修改成“未分配”,清楚作业名。同时把收回的设备台数加到设备类表中的“现存台数”中,这样就收回了作业使用设备的权利。

6.2 磁盘驱动

一、磁盘结构

1、磁盘块地址

柱面号、磁头号、扇区号

2、输入输出操作时间

启动磁盘执行输入输出操作时，要把移动臂移动到指定的柱面，再等待指定的扇区旋转到磁头位置下，然后让指定的磁头进行读写，完成信息传送。启动磁盘完成一次输入输出操作所花的时间包括：

寻找时间--磁头在移动臂带动下移动到指定柱面所花的时间。

延迟时间--指定扇区旋转到磁头下所需的时间。

传送时间--由磁头进行读写完成信息传送的时间。

其中，传送信息所花的时间是硬件设计时固定的，而寻找时间和延迟时间是与信息在磁盘上的位置有关。

3、计算公式

磁盘号编号顺序：按柱面（从0号柱面开始）、磁头、扇区顺序。

假定用 t 表示每个柱面上的磁道数，用 s 表示每个盘面上的扇区数，则第 i 柱面， j 磁头， k 扇区所对应的块 b 有如下公式确定：

$$b = k + s \times (j + i \times t)$$

已知块号，确定该块在磁盘上的位置：

在上述的假定下，每个柱面上有 $s \times t$ 个磁盘块，为了计算第 P 块在磁盘上位置，可以令 $D = s \times t$ ，设 $M = [P/D]$ ， $N = P \bmod D$ 。于是，第 P 块在磁盘上位置为：

柱面号 = M

磁头号 = $[N/S]$

扇区号 = $[N \bmod S]$

二、磁盘调度

1、问题的提出

有若干个访问者请求磁盘执行输入输出操作，应先让哪一个访问者完成操作？（为了保证信息的安全，系统在任时刻只允许一个访问者启动磁盘执行输入输出操作，其余的访问者必须等待。）

为了提高系统效率，降低若干个访问者执行输入输出操作的总时间（平均服务时间），增加单位时间内的输入输出操作次数，应根据移动臂的当前位置使寻找时间和延迟时间尽可能小的那个访问者优先得到服务。

驱动调度：系统采用一定的调度策略来决定各个等待访问磁盘者的执行次序，这一工作称为磁盘的“驱动调度”，采用的调度策略称为“驱动调度算法”。

2、移臂调度

根据访问者指定的柱面位置来决定执行次序的调度

目的：尽可能地减少输入输出操作中的寻找时间。

（1）先来先服务调度算法

按照访问者提出访问请求的先后次序进行调度。（公平）

缺点：未考虑访问者要求访问的物理位置，可能使移动臂来回地移动，花费的寻找时间较长，进而执行输入输出操作的总时间增长。

(2) 最短寻找时间优先调度算法

从若干访问者中挑选寻找时间最短的那个请求进行调度。

优点：减少了寻找时间，因而缩短了为各请求访问者服务的平均时间，即提高了系统效率。

不公平：未考虑访问者到来的先后次序。

(3) 电梯调度算法

从移动臂当前位置开始，沿着磁臂的移动方向选择距离当前移动臂最近的那个访问者进行调度，如果沿磁臂的移动方向不再有请求访问时，就改变方向再选择。

目的：尽量减少移动臂移动时所花的时间。

两者比较：“最短寻找时间优先”不考虑臂的移动方向，可能导致移动臂来回改变移动方向。由于移动臂改变方向是机械动作，速度相对较慢。相比之下，电梯调度算法是一种简单、实用且高效的调度算法，但是，实现时除了要记住读写磁头的当前位置外，还必须记住移动臂的移动方向。

(4) 单向扫描调度算法

总是从 0 号柱面开始由外向里扫描，依次服务所遇到的访问请求；移动臂到达最后一个柱面时，立即带动读写磁头快速返回到 0 号柱面，返回时不为任何访问者服务，返回后再次进行扫描。

3、旋转调度

旋转调度：根据延迟时间来决定执行次序的调度。

【即当移动臂（磁头）定位后，决定同一柱面访问者的执行次序？从减少输入输出操作总时间为目标考虑，应优先选择延迟时间最短的访问者进行服务。】

进行旋转调度时应分析下列情况：

- (1) 若干等待访问者请求访问同一磁道上的不同扇区。
- (2) 若干等待访问者请求访问不同磁道上的不同编号的扇区。
- (3) 若干等待访问者请求访问不同磁道上具有相同编号的扇区。

对于前两种情况，旋转调度总是让首先到达读写磁头位置下的扇区先进行传送操作。对于第三种情况，这些扇区同时到达读写磁头位置下，旋转调度可任意选择一个读写磁头进行传送操作。

4、信息的优化分布

记录在磁道上的排列方式会影响输入输出操作的时间。

记录的优化分布有利于减少延迟时间，从而缩短了输入输出操作的时间。所以，对于一些能预知处理要求的信息采用优化分布可以提高系统的效率。

例子：教材（本）p117-119 页

6.3 外围设备启动

作业执行中经常要求启动各种外围设备：

- 信息读入主存储器进行处理
- 主存储器中的信息传送到存储介质上

启动外围设备：

- 要根据设备特性的编制复杂且繁琐的输入输出程序

- 执行“启动 I/O”指令使外围设备工作

设备管理：完成以上这些复杂的、与硬件有关的工作，目的：

- 减轻用户负担
- 防止用户错误地使用外围设备而影响系统的可靠性

一、通道和通道程序

1、通道结构

(1) 通道

又称输入输出处理机，完成内存与外设之间的传送信息。只要 CPU 启动了通道，通道就按指定的要求独立完成输入输出操作，而 CPU 可以做与输入输出操作无关的其他工作，从而使计算机系统获得了 CPU 与外设之间并行工作的能力。

现代计算机系统采用自成独立系统的通道结构。

(2) 通道的连接

图：教材（本）p120 页，（专）p75 页

一个中央处理器可以连接多个通道，一个通道可以连接多个设备控制器，一个设备控制器可以连接同类型的多台设备。有的系统还可以把一台设备连接到几个设备控制器上，一个设备控制器连接到几个通道上，实现多路交叉连接。

(3) 通道工作原理

当有输入输出请求时，CPU 先执行“启动 I/O”指令，启动指定通道上的指定设备（把要求通道“做什么和怎样去做”告诉通道）。当启动成功，通道按规定的要求通过设备控制器控制外设进行操作。这时，CPU 就可执行其他任务并与通道并行工作，直到输入输出操作完成，由通道发出操作结束的“输入、输出中断”时 CPU 才暂停当前的工作，转去处理输入输出中断事件。

2、通道程序

(1) 通道命令与通道程序

为了使操作系统能使用种类繁多、特性各异的外围设备，计算机硬件提供“通道命令”，操作系统用一组通道命令来规定通道执行一次输入输出操作应做的工作，这一组通道命令就组成了一个“通道程序”。通道被启动后，就依次执行预定的通道程序中的一条条通道命令，从而实现对外围设备的操作控制。

(2) 通道命令组成

每一条通道命令规定了设备的一种操作。

由命令码、数据主存地址、传送字节个数及标志码等部分组成。

命令码：规定了外围设备应执行的操作。

分成三类：数据传输类（例如，读，反读，写等）：要求外围设备执行信息传送

通道命令转移类：用来改变通道程序的执行顺序

设备控制类：要求外围设备执行某些辅助操作，如磁带反绕到始点、磁盘搜索和查找、打印走纸和换页等。

数据主存地址：

(对数据传输类命令)规定了本通道命令进行数据传送的主存地址,由传送字节个数决定了这个主存区域的大小。

(当命令码为“读”操作,这个区域用来存放从外围设备读入主存的信息;当命令码为“写”操作,这个区域用来存放输出到外界设备的信息。)

(对通道命令转移类命令)即为指定的转移地址,通道转向转移地址所指示的通道命令执行。

(对设备控制类命令)指出与外围设备有关的控制信息所在单元,无控制信息时,数据主存地址可缺省。

标志码:通道命令的连接标志。

当标志码为非“0”时,表示通道程序尚未结束,还有后继的通道命令要执行;当标志码为“0”时,表示本条通道命令是通道程序的最后一条命令,执行完本条命令后,一次输入输出操作就结束。

传送字节个数:(对数据传输类命令)表示本命令应传输的字节数。

通道执行该命令时,每传送一个字节就把传送字节数减1,当传送字节个数为“0”时,表示命令执行结束,由于通道执行通道命令时总是先检查传送字节个数是否“0”,若为“0”就认为该命令执行结束,如果有后继命令则继续执行下一条命令。

对非数据传输类命令,虽不需要传送数据,也要把传送字节个数填上任意一个非“0”数。

通道程序:教材(本)p121页,(专)p76页

(3) 通道地址字(CAW)

内存中设置的一个固定单元,用来存放当前启动外围设备时要求通道执行的通道程序首地址。

通道被启动后从CAW指示的内存单元中可取到要执行的第一条通道命令,并把存放通道程序的地址记录下来,以后就可顺序地从内存中取到该通道程序中的所有通道命令,逐条解释执行。

(4) 通道状态字(CSW)

记录了通道在执行通道程序时通道和设备执行操作的情况。

包括:通道命令地址,设备状态,通道状态,剩余字节数

二、外围设备启动

操作系统启动和控制外设完成输入输出操作的过程可分成三个阶段。

1、准备阶段

用户调用文件操作请求存取文件信息,文件系统根据用户给定的参数可以确定应启动哪个通道上的哪台设备以及信息存放的内存地址和存储介质上的位置等。然后,文件系统把这些存取要求告诉设备管理,请求协助。设备管理接到文件系统传来的请求后按存取要求组织好通道程序,且把通道程序的首地址存放到通道地址字(CAW)中。

2、启动I/O阶段

计算机硬件提供：“启动 I/O”指令，指出要启动的通道号和设备的绝对号。

当设备管理中负责启动外设的进程占有 CPU 后，首先把指定的通道号和设备号组织到“启动 I/O”指令中。当 CPU 执行“启动 I/O”指令后，指定的通道就会根据通道和设备的工作情况用“条件码”向 CPU 回答（存放在程序状态字寄存器的条件码位置）。分析条件码可知道是否启动成功。若条件码表示为“通道忙或设备忙”则本次启动不成功，应该再重新启动；若条件码表示启动成功则通道控制设备进行输入输出操作，而 CPU 可继续执行程序，与通道并行工作。

3、结束处理阶段

通道发现通道状态字（CSW）中有控制器结束、设备结束、通道结束、设备出错或设备特殊等情况时，发“输入输出中断”（I/O 中断），一方面把产生中断的通道号和设备绝对号存入中断寄存器，另一方面把通道状态字存入内存的一个固定单元中。中断装置响应中断后，操作系统的中断处理程序根据通道号、设备绝对号和通道状态字作相应的处理。

教材（本）P123 页图 6-2-4：一次成功的输入输出操作过程示意。

设备处理的一致性：具有通道结构的计算机系统，从启动外围设备到完成输入输出操作，没有考虑不同类型的物理设备的特性，都采用统一的方法进行处理。这种不考虑具体特性（实际上设备特性已隐含在通道程序中）的处理方法称为设备处理的一致性。

采用设备处理一致性技术使得输入输出操作的处理既简单又不易出错。

三、I/O 中断处理

I/O 中断是使 CPU 和通道协调工作的一种手段，通道借助 I/O 中断请求 CPU 进行干预，CPU 根据产生的 I/O 中断事件了解输入输出操作的执行情况。

1、操作正常结束

通道状态字（CSW）：通道结束、控制器结束、设备结束

对独占设备：等待该设备的进程从等待变为就绪

对共享设备：使用设备的进程与等待该设备的进程变为就绪

2、操作异常结束

设备故障、设备特殊情况

（1）设备故障

表示通道或设备不正常。

硬件的故障：接口错、控制错、通道程序错（如通道命令没及时链接上）以及数据错（如校验码不符合）等

处理原则：先组织通道程序复执。经复执后故障可能排除，若故障排除则通道可继续执行通道程序；若经多次复执故障仍未排除，则操作系统只能输出一些信息，请维护人员人工排除故障。

（2）设备特殊情况

各种设备在工作时的一些特殊情况：打印机纸用完，往磁带上写信息时磁带机到了末点，在读磁带文件时读到了带标等

操作系统将分别处理。

6.4 虚拟设备

一、虚拟设备引入

1、独占设备的不足：不能充分利用，不利于提高系统效率

具体表现：

(1) 占有输入机和打印机的作业，只有一部分时间在使用它们，在其余时间里这些设备处于空闲状态。在设备空闲时不允许其他作业去使用它们，因此，**不能有效地利用这些设备**。

(2) 当系统只配有一台输入机和一台打印机时，就不能接受两个以上要求使用输入机和打印机的作业同时执行，**不利于多道并行工作**。

(3) 这些独占设备大都是**低速设备**，在作业执行中往往由于等待这些设备的信息传输而延长了作业的执行时间。

2、解决方法：虚拟设备

用共享的磁盘来模拟输入机和打印机的工作，使作业都感到各自拥有独占使用的设备且它们的传输速度与磁盘的传输速度一样快。这种用一类物理设备模拟另一类物理设备的技术，使各作业在执行期间只使用虚拟的独占设备而不直接使用物理的独占设备。

好处：使独占设备变成了共享设备，设备的利用率和系统效率都能得到提高。

二、虚拟设备的实现

脱机外围设备操作（成本高、增加操作员的手工操作，费时且易出错、作业周转时间长） 引入联机同时外围设备操作

1、实现原理

(1) 基本条件

必须要有一定的硬件和软件条件为基础。

硬件：大容量的磁盘，中断装置和通道，具有 CPU 与通道并行工作的能力。

操作系统：采用多道程序设计技术。

(2) 工作原理

把一批作业的全部信息通过输入设备预先传送到磁盘上。在多道程序设计系统中，可从磁盘上选择若干个作业同时装入内存，并让它们同时执行。由于作业的信息已全部在磁盘上，故作业执行时不必启动输入机读信息，而从共享的磁盘上读取各自的信息。把作业产生的结果也暂时存放在磁盘上而不直接启动打印机输出。直到一个作业得到全部结果且执行结束时，才把该作业的结果从打印机输出。

2、实现技术

(1) 输入井和输出井

在磁盘上划出专用存储空间，称为“井”用以存放作业的初始信息和作业的执行结果。

为了便于管理把“井”又分成两部分：“输入井”和“输出井”。“输入井”中存入作业的初始信息，“输出井”中存放作业的执行结果。

(2) 斯普林 (SP00L) 系统

操作系统中实现虚拟设备的功能模块。

联机的外围设备同时操作 (simul taneous peripheral operati on on line 缩写 SP00L)。

斯普林系统由三部分程序组成：

- 预输入程序

把一批作业组织在一起形成作业流,由预输入程序把作业流中每个作业的初始信息传送到“输入井”保存,以备作业执行时使用。

- 井管理程序

作业执行过程中要求启动输入机 (或打印机) 读文件信息 (或输出结果) 时, 操作系统根据作业请求调出“井管理程序”工作, 转换成从“输入井”读信息 (或把结果写入“输入井”)。

对系统来说, 从“井”中存取信息可以缩短信息的传输时间, 从而加快作业的执行。对用户来说, 只要保证信息的正确存取就行, 至于信息是从“井”中存取还从独占设备上存取无关紧要。由于磁盘是可共享的, 因此从“井”中存取信息可以同时满足多个用户的读写要求, 从而使每个用户都感到有供自己独立使用的输入机 (或打印机) 且速度与磁盘一样快。

“井管理程序”包括“井管理读程序”和“井管理写程序”。

当作业请求从输入机上读文件信息时, 就把任务转交给“井管理读程序”, 从“输入井”读出信息供用户使用。

当作业请求从打印机上输出结果时, 就把任务转交给“井管理写程序”, 把产生的结果保存到“输出井”中

- 缓输出程序

缓输出程序负责查看“输出井”中是否有待输出的结果信息, 若有, 则启动打印机把作业的结果文件打印输出。

(3) 数据结构

- 作业表

SP00L 系统设置一张作业表, 用来登记进入“输入井”的各个作业的作业名、作业状态、作业拥有的文件数以及预输入表和缓输出表的位置等。

输入井中的作业可有四种状态：

输入状态 预输入程序启动了输入机正在把该作业的信息传输到“输入井”。

收容状态 该作业的信息已经存放在“输入井”中, 但尚未被选中执行。

执行状态 作业已被选中并装入内存开始执行。

完成状态 作业已执行结束, 其执行结果在“输出井”中等待打印输出。

- 预输入表

每个作业有一张预输入表, 用来登记该作业初始信息的各个文件。指出各文件的文件名、传输文件信息时使用的设备类型、文件的长度以及文件存放位置等。

作业的初始信息通常是源程序、数据等。作业执行是总是采用顺序存取方式。因此, 把这些文件信息存入“输入井”时可采用链接结构, 在预输入表中只要给出文件在“输入井”中的位置。

- 缓输出表

每个作业设置一张缓输出表, 用来登记该作业产生的结果文件。

作业产生的结果按链接结构组织成文件存放在“输出井”中。

(4) 功能实现

当用户提交了一批作业：

- 操作员键入“预输入命令”启动预输入程序工作。预输入程序查看作业表中是否有空登记项，若有空登记项则再检查“输入井”中是否有空闲空间，如果有空闲空间则可接纳新的作业进入“输入井”。
- 启动输入机读出并分析作业的标识信息，把作业名、文件个数登记入作业表，且置成“输入状态”。接着依次读出作业的文件信息，寻找“输入井”中的空间存放这些信息，把它们组织成链接文件的形式登记到预输入表中。直到该作业信息全部输入，把作业状态修改成“收容状态”，将预输入表的位置填入作业表。
- 当作业流中还有后继作业时，预输入程序继续工作，只要能接纳新作业，就按上述过程把作业信息存入“输入井”，直到“输入井”的空间已占满或作业表中无空登记项时就暂不能再接纳新作业。当不能接纳新作业或当前作业流中信息已全部进入“输入井”，则预输入程序工作结束，当需要时可再次启动预输入程序工作。
- 当主存储器中可以装入作业时，就从“输入井”中选择处于“收容状态”的作业执行，被选中的作业其状态应改为“执行状态”。作业执行过程中要求启动输入机读文件时，系统并不实际地启动输入机，而是调出“井管理读程序”工作，根据作业名先找到该作业的预输入表，再根据文件名可从预输入表中得到文件存放的起始位置，沿着链接指针可依次读出存放在“输入井”中的文件信息。在这里，“井管理读程序”模拟从输入机读文件的工作。由于输入机把读出的信息传送给作业后就不再保留已读出的信息，所以，“井管理读程序”从“输入井”读出文件信息后也不必保留该信息。于是，当文件信息传送给作业后，应把文件占用“输入井”的存储空间归还，归还后的空间可以用来存放其他的文件。作业执行中要求启动打印机输出结果时，系统也不实际地启动打印机，而是调出“井管理写程序”工作。首先根据作业名找到缓输出表，同时查找“输出井”中的空闲空间，把结果信息组织成链接文件存入“输出井”，并在缓输出表中登记。作业执行结束后，把作业状态修改成“完成状态。”
- 当处理器空闲时缓输出程序可以占用处理器，缓输出程序查看作业表，找出处于完成状态的作业，再查看相应的缓输出表得到结果文件的存放位置，读出文件信息并把它们转换成符合打印要求的格式，然后启动打印机将其打印输出。文件信息被打印输出后，文件占用“输出井”的存储空间应归还。一个作业结果文件均被输出后，应将其在作业表中除名，相应的登记项成为空登记项，可以用来登记新进入“输入井”的作业。

第七章 作业管理

7.1 概述

一、作业的基本概念

1、作业

把用户要求计算机处理的一个计算问题称为一个“作业”。

2、典型的作业控制过程（图 7-1-1）

编制编辑源程序 - 编译 - 连接装配 - 运行

3、作业步

任何一个作业都要经过若干加工步骤才能得到结果,作业的每一个加工步骤称为一个“作业步”;每个作业步都对应一个程序的执行,前一个作业步的结果信息往往作为后一个作业步的输入。

实际上每个作业所经历的加工步骤是可以不同的。

二、作业控制方式

所谓“作业控制方式”是指用户根据操作系统提供的手段来说明作业加工步骤的方式。

1、批处理方式

自动控制方式、脱机控制方式

用户使用操作系统提供的“作业控制语言”对作业执行的控制意图写好一份,“作业控制说明书”,连同该作业的源程序和初始数据一同提交给计算机系统,操作系统将按照用户说明的控制意图来控制作业的执行。于是,作业执行过程中用户不必在计算机上进行干预,一切由操作系统按作业控制说明书的要求自动地控制作业执行。

对作业的控制意图是事先说明的,不必联机输入。采用这种控制方式的作业完全由操作系统自动控制,因此,适合成批处理,在成批处理时操作系统按各作业的作业控制说明书中的要求分别控制相应的作业按指定的步骤去执行。

采用批处理控制方式的作业称为“批处理作业”。

2、交互方式

联机控制方式

用户使用操作系统提供的“操作控制命令”来表达对作业执行的控制意图。用户逐条输入命令,操作系统每接到一条命令,就根据命令的要求控制作业的执行,一条命令所要求的工作做完后,操作系统把命令执行情况通知用户且让用户输入下一条命令,以控制作业继续执行,直至作业执行结束。

在作业执行过程中操作系统与用户之间不断交互信息。采用交互方式时用户必须在计算机上直接操作。交互方式也适合终端用户使用,终端用户通过终端设备把操作控制命令传送给操作系统,操作系统把命令执行情况也通过终端设备通知用户,最终从终端上输出结果。

采用交互控制方式的作业称为“交互式作业”,对于来自终端的作业也称为“终端作业”。

7.2 批处理作业的管理

一、批处理作业的组织

1、作业的组成

源程序，数据，作业说明书

作业说明书：用作业控制语言书写，刻画了用户对作业的基本描述，规定量用户对作业执行的控制要求。

作业控制语言：由若干控制语句组成，每个控制语句除包含了表示特征的关键字外，还有指示控制要求的若干参数。

例如：

//作业名 JOB 参数

JOB 语句是每个作业的第一个控制语句，它标志一个新作业的开始。其中“作业名”是用户给作业定义的名称，“JOB”是语句特征的关键字，“参数”通常包括用户名、记帐信息、作业类别、估计的计算时间、优先数以及主存空间要求等。在JOB语句中“作业名”和“JOB”是不可缺省的，其他参数可缺省，由用户选定。

//步名 EXEC PGM = 程序名，参数

EXEC 语句是标志一个作业步的语句，它告诉操作系统本作业步要执行程序。语句中的EXEC语句特征的关键字，“PGM = 程序名”是不可缺省的，其他参数可缺省。

二、作业输入

每个用户根据自己的解题要求组织作业，把准备好的作业交给操作员，操作员把一批作业组织成作业流向系统成批输入。

采用批处理控制方式的计算机系统一般均提供SP00L操作技术。

操作员只要用“预输入命令”启动SP00L系统中的“预输入程序”工作就可把 workflow 中的作业信息存放到“输入井”中。预输入程序根据作业控制说明书中的作业标识语句可以区分各个作业，把作业登记入作业表，把作业中的各个文件存到“输入井”且登记到预输入表中。这样，就完成了作业的输入工作，被输入的作业处于“收容状态”在“输入井”中等待处理。

三、作业调度

1、作业调度

操作系统根据允许并行工作的道数和一定的调度算法从“输入井”中选取若干作业，把它们装入内存，使它们有机会获得CPU运行，这一工作称为“作业调度”。

由“作业调度进程”实现。

2、设计作业调度时考虑的原则

◆ 公平性

公平对待每个用户、使用户满意，不能无故或无限制地拖延一个作业的执行。

◆ 均衡使用资源

使同时装入内存的作业在执行时尽可能利用系统中的各种不同资源，从而极大提高资源的使用率。

◆ 极大的流量

缩短作业的周转时间，在单位时间内尽可能为更多的作业服务，提高计算机

系统的吞吐能力。

这些原则不能兼顾。应根据系统的设计目标来决定调度原则。

理想的调度算法 :能提高系统效率 ,能使进入系统的作业及时得到计算结果。

必需遵循一个必要条件：系统所有的尚未分配的资源可以满足被选作业的资源要求。

3、作业调度算法

◆ 作业 P_i 周转时间：

$$T_i = E_i - S_i$$

S_i ：作业 P_i 进入“输入井”的时间， E_i ：得到计算结果的时间为。

一般，希望周转时间 T_i 尽可能地小，但是，在批处理控制方式下实现多道并行工作时，不可能让每个用户都得到理想的效果。

从系统的角度，希望作业的平均周转时间尽可能地小。

◆ 作业平均周转时间 T

$$T = \left(\sum_{i=1}^n T_i \right) \times \frac{1}{n}$$

周转时间和平均周转时间的大小与选用的调度算法有关。

(1) 先来先服务算法

它是按照作业进入“输入井”的先后次序来挑选作业，先进入的作业优先被挑选。

但要注意，不是先进入的一定被先选中，这要根据资源的分配情况来决定。

优点：公平性，容易实现。

缺点：当计算时间长的作业先进入“输入井”而被选中执行时，就可能使计算时间短的作业长期等待。不仅使这些用户不满意，而且使计算时间短的作业周转时间变长，从而平均周转时间也变长，降低了系统的吞吐能力。

(2) 短作业优先算法

采用这种算法时，要求用户对自己的作业需要计算的时间预先作出一个估计，在作业控制说明书中加以说明。作业调度时依据在输入井中的作业提出的计算时间为标准。优先选择计算时间短且资源能得到满足的作业。

优点：易于实现，强调了资源的充分利用，保证了系统的最大吞吐量（单位时间里处理作业的个数）。

缺点：不公平，会造成长作业长期等待。

◆ 假设系统中所有作业同时到达，可以证明采用短作业优先算法能得到最短的作业平均周转时间。

采用这种算法应注意两个问题：

- 该算法是以用户估计的计算时间为标准，有些用户为了使自己的作业能优先执行，可能故意把计算时间估计的低一些。为了避免这一现象，若作业执行超过所估计的时间，则可加价收费。
- 由于系统可能不断接受新作业进入“输入井”，如果新进入“输入井”的作

业估计的计算时间比较短，则将会使进入“输入井”较早但要求计算时间长的作业等待时间太长，使这些用户不满意。

(3) 最高响应比优先算法

响应比定义：

$$\text{响应比 } R = \frac{\text{等待时间}}{\text{计算时间}}$$

采用响应比最高者优先算法进行调度时，必须对“输入井”中的所有作业计算出它的响应比，从资源能得到满足的作业中选择响应比高的作业优先装入内存。

该算法是前两种算法结合，克服了前面两种算法的缺点

优点：公平，吞吐率大

缺点：增加了计算，增加了开销

(4) 基于优先数的调度算法

为每个作业确定一个优先数，资源能满足且优先数高的作业优先被选取，当几个作业有相同优先数时，对这些具有相同优先数的作业再按照先来先服务原则进行调度。

确定优先数的方法：

- ✓ 由用户来提出自己作业的优先数
- ✓ 由操作系统根据作业的缓急程度、作业估计的计算时间、作业的类型、资源申请情况等因素综合考虑，分析这些因素在实现系统设计目标中的影响，决定各因素的比例，综合得出作业的优先数。有的系统还可以根据作业在“输入井”中的等待时间动态地改变其优先数。提高等待时间长的作业优先数，以缩短作业的周转时间和平均周转时间。

(5) 均衡调度算法

根据作业对资源的要求进行分类。作业调度从各类作业中去挑选，尽可能使用不同资源的作业同时执行。这样不仅可以使系统的各种资源都在忙碌，而且可以减少作业等待使用相同资源的时间，从而加快作业的执行。

4、作业调度与进程调度

作业调度按一定的算法从“输入井”中选择资源能得到满足的作业装入内存，使作业有机会去占用处理器执行。但是，一个作业能否占用处理器则由进程调度来决定。

所以，作业调度选中了一个作业且把它装入内存时，就应为该作业创建一个进程，若有多个作业被装入主存储器，则同时存在多个进程，这些进程的初始状态为就绪状态，然后，由进程的调度来选择当前可占用处理的进程，进程运行中由于某种原因状态发生变化，当它让出处理器时，进程调度就再选另一个作业的进程去运行。

可见，只有作业调度与进程调度相互配合才能实现多道作业的并行执行。

四、作业控制

主要负责控制作业的运行，具体解释执行作业说明书的每一个作业步。
由作业控制进程完成（一个系统进程）。

一个作业被作业调度选中后，操作系统按照作业控制说明书中所规定的控制要求去控制作业的执行。一个作业往往要分几个作业步执行，一般说，总是按作业步的顺序控制作业的执行，一个作业步执行结束后，就顺序取下一个作业步继续执行，直到最后一个作业步完成，整个作业就执行结束。这时，系统收回作业所占资源且撤离该作业，作业执行的结果在“输入井”中等待输出。

如果作业执行到某个作业步时发生了错误，则要分析错误的性质。如果是某些用户估计到的错误，且用户已在作业控制说明书给出了处理办法，系统应按用户的说明转向指定的作业步继续顺序执行，直至作业执行结束。

当作业执行结束后，应让作业调度程序再选择作业装入内存。

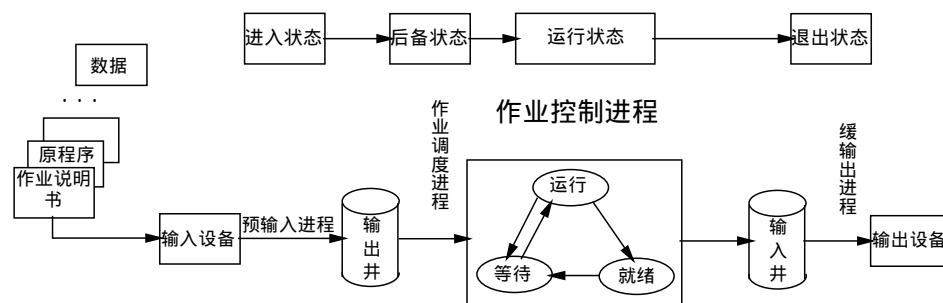
怎样才能完成作业步的执行呢？不同的作业步要完成不同的工作，都要有不同的程序去解释、执行。一般来说，按作业控制说明书中的作业步控制语句（例如 EXEC 语句）中参数指定的程序，把相应的程序装到内存，然后创建一个相应的作业步进程，使它的状态为“就绪”。当被进程调度选中运行时，该进程就执行相应的程序，在执行中可用“访管指令”提出“系统功能调用”，请求操作系统服务，操作系统的功能模块完成了用户提出的调用要求后，让用户进程返回原来的断点（调用点）继续执行，当一个作业步的进程执行结束，需向操作系统报告执行结束的信息，操作系统接到信息后可输出“×××作业结束”把作业执行进展情况通知操作员，同时撤消该进程，再继续取下一个作业步的控制语句，控制作业继续执行，当取到一个表示作业结束的控制语句时，操作系统收回该作业占有的全部内存和外围设备等资源，然后让作业调度再选取下一个可执行的作业。

作业调度选中一个作业后，按作业控制说明书中的第一个作业步控制语句的要求创建作业的第一个进程，例如，如果第一个作业步是要求编译，就先创建一个“编译进程”。

一个作业步执行结束后，操作系统顺序取下一个作业步控制语句，系统为不同的作业步创建不同的进程，以完成作业步要求的工作。

五、作业退出

1. 把输出结果送到输出设备上（启动缓输出进程完成）。
2. 回收各种资源。



7.3 交互式作业的管理

一、交互式作业

由用户使用操作系统提供的操作控制命令(也称命令语言)或使用会说话语言系统提供的会话语句直接提出对作业的控制要求,每当用户输入一条命令或一个会话语句后,系统立即解释执行且及时给出应答。用户根据作业执行情况决定输入的下一条命令或下一个会话语句,以控制作业的继续执行。

在使用分时操作系统的计算机系统中,终端用户通过终端设备输入作业的程序和数据,且直接在终端设备上输入各种命令或会话语句,来表达对作业的控制意图。系统把作业的执行情况也通过终端设备通知用户,最终从终端上输出结果。

交互式作业的特点:交互性,采用人机对话的方式工作。

因此,用户能从系统给出的应答中及时掌握作业的执行情况,以决定下一个作业步应该做什么,从系统给出的应答中可及时发现作业执行中的问题且予以纠正,能方便地实现对程序的联机调和修改。

二、交互式作业的控制

1. 操作使用接口

常用的操作使用接口有操作控制命令、菜单技术和窗口技术等。

✓ 操作控制命令

一般格式为:

命令名 参数 1, 参数 2.....;

命令名是作为请求完成指定功能的标识,它是不可缺省的,参数是用来表示指定功能时所需的各种信息,在某些情况下参数可以不分缺省或全部缺省。

命令语言:系统提供的操作控制命令的集合。

基本分成以下类型;

- (1)“注册”和“注销”命令。
- (2)编辑类命令。
- (3)文件类命令。
- (4)编译、装配和运行类命令。
- (5)目录操作类命令。
- (6)调试类命令。
- (7)操作文式转换命令。

✓ 菜单技术

当一个程序具有若干项可供用户选择功能时,由该程序先显示出自己具有的各功能的名称及其含意,然后用户根据需要指出希望完成的功能,再由该程序按用户的指定要求调出相应的功能模块进行处理。这种方法像菜馆的点菜方式,故称为“菜单技术”。

菜单技术为用户提供一种“友好的使用接口”。

✓ 窗口技术

2. 命令解释的执行

命令解释程序接收来自用户的命令并对命令进行分析。

一般说,可把命令分成两大类:一类是由操作系统中的相应处理模块直接解释执行;另一类必须创建用户进程去解释执行。

三、终端作业的管理

终端用户控制终端作业的执行大致分四个阶段：终端的连接、用户注册、控制作业执行和用户退出。

✓ 终端的连接

近程终端：直接与计算机系统连接的终端。

远程终端：借助于租用专线或交换线接到计算机系统，在终端加电后用户还需通过电话拨号进行呼叫

当终端与计算机系统在线路上接通后，计算机系统会在终端上显示信息告诉用户。

✓ 用户注册

✓ 控制作业执行

✓ 用户退出

在分时操作系统控制下，对终端用户均采用“时间片轮转”的方法使每个终端作业都能在一个“时间片”的时间内去占用处理器。当一个时间片用完后，它必须让出处理器给另一个终端作业占用处理器。这样，可保证从终端用户输入命令到计算机系统给出应答只是几秒钟的时间，使端用户感到满意。

在一个具有分时兼批处理的计算机系统中总是优先接纳终端作业，仅当终端作业数小于系统可允许同时工作的作业数时，可以调度批处理作业，允许终端作业与批处理作业混合同时执行。为了使终端用户有满意的响应时间，可采用分级调度的方法分配处理器，可把有关终端作业的就绪进程排成一个就绪队列，而把有关批处理作业的就绪进程排入另外的就绪队列中，当有终端作业就绪进程时，总让终端作业的就绪进程按“时间片轮转”的方法先占用器，当无终端作业的就绪进程时，才去查看批处理作业的就绪队列，按确定的算法从中选择一个就绪进程让它占用处理器，这样，既可使终端用户满意，又能提高系统效率。

第八章 进程同步与通信

8.1 进程的顺序性与并发性

一、进程的顺序性

1、进程的顺序性

进程的顺序性：进程在 CPU 上的执行是严格按顺序进行的，前一个操作结束之后才能开始后继操作。

在计算机系统中只有一个程序在运行，这个程序独占系统中的全部资源，其执行不受外界影响。

2、特征

1) 程序的封闭性

独占资源，执行过程中不受外界影响，执行状态由程序自身决定，程序运行结果与程序执行速度无关。

2) 程序结果的可再现性

只要初始状态相同，执行结果应相同。

二、进程的并发性

1、并发进程

并发进程：系统中存在一组可同时执行的进程，这些进程称为并发进程。

在一定时间内有两个或两个以上程序在计算机系统中同处于开始执行但尚未结束的状态。（一个程序结束之前，另一个已经开始执行，并且次序不是事先确定的。）

并行：在物理机器上有多个程序在运行。

2、特征

1) 程序执行结果的不可再现性

进程执行的速度不能由自己来控制，所以，并发程序执行的结果与其执行的相对速度有关，是不确定的。

2) 并发进程的执行是间断性的，与自身原因和进程调度策略的有关。

3) 制约性（相互作用）

资源共享：系统中资源不为某个进程独占。

并发程序之间存在着制约关系，这些制约关系在某些情况限制了程序的运行速度。

制约关系分为两类：

直接制约：当一个程序到达某些点时，其继续运行可能取决于另一程序是否完成某一任务程序之间相互通信，传递信息由同步操作引起的。

间接制约：由竞争相同资源而引起的，称互斥关系。

三、与时间有关的错误

书写规则：

Begin

 变量说明（变量名：数据类型）

 变量初值（变量名=初值）

Cobegin

Process 1

Begin

L1:

 Goto L1

End

Process 2

Begin

L2:

 Goto L2

End

.....

Coend.

1、例子 1： 观察者进程 与 报告者进程
共享变量：count

2、例子 2： 飞机订票系统
共享变量：Aj

3、例子 3： get , copy , put 三个并发进程

8.2 进程间的相互作用

一、并发进程间的联系

- ◆ **无关进程（无交往进程）**：彼此完全独立，无任何联系的进程。
如果一个进程的执行不影响其他进程的执行，且与其他进程的进展情况无关，即它们是各自独立的，则说这些并发进程的相互之间是无关的。
无关的并发进程一定没有共享的变量，它们分别在各自的数据集合上操作。
- ◆ **有交往进程**：指多个并发进程具有某种联系。
如果一个进程的执行依赖其他进程的执行情况，或者说，一个进程的执行可能影响其他进程的执行结果，则这些并发进程相互之间是有交往的。
有交往的并发进程一定共享某些资源。
- ◆ **进程的相互作用**
直接作用：进程间的相互联系是有意识的安排的。
间接作用：进程间需要通过某种中介（共享变量）发生联系，一般是无意识安排的。

二、进程的互斥（间接作用）

1、进程的互斥

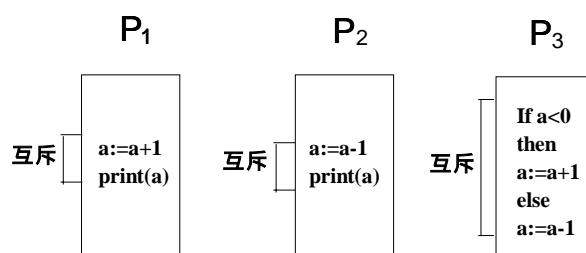
由于各进程要求共享资源，而有些资源需要互斥使用，因此各进程竞争使用这些资源，进程的这种关系为进程的互斥。

共享变量（临界资源或互斥资源）：一次只允许一个进程使用。

2、临界区（互斥区 critical section/critical region）

在并发进程中涉及到共享变量的程序段叫临界区。

临界区分散在不同进程中，但这些临界区对同一共享变量进行操作，这些临界区称为相关临界区。



例：

在它们不为互斥时，如果 a 初值为 5， P_1 先执行，当读完 $a=a+1$ 时 a 值为 6，由于某种原因(时间片到)去。这时 a 值为 5，当执行 P_2, P_3 后去执行 P_1 这时 PRINT (a) 出来的值为 5。这显然不对，如果使它们成为互斥区则在不完成对 a 的操作不会停止。

3、使用互斥区的原则

在“任何进程无权停止其它进程的运行，进程之间相对运行速度无硬性规定”的条件下遵循以下原则：

(1) 一次最多让一个进程在临界区执行，当有进程在临界区时，其他想进入临界区执行的进程必须等待。

有空让进：当无进程在互斥区时，任何有权使用互斥区的进程可以进入。

无空等待：不允许两个以上的进程同时进入互斥区。

(2) 任何一个进入临界区的进程必须在有限的时间内退出临界区，即任何一个进程不应该无限逗留在临界区中。

(3) 不能强迫一个进程无限地等待进入临界区，即有进程退出临界区时应让一个等待进入临界区的进程进入。

有限等待：任何进入互斥区的要求应在有限的时间内得到满足。

解决：标志方式、上锁开锁方式、PV 操作方式、管程方式

三、进程的同步（直接作用）

进程的同步：指系统中一些进程需要相互合作，共同完成一项任务。具体地说，一个进程运行到某一点时，要求另一伙伴进程为它提供消息；在未获得消息之前，该进程变为等待状态；获得消息后被唤醒进入就绪态。

例 1：司机进程 P_1	售票员进程 P_2
REPEAT	REPEAT
启动车辆	关门
正常运行	售票
到站停车	开门
UNTIL FALSE	UNTIL FALSE
关门--启动--行驶--停--开门	

例 2：教材 P160 (本) P103 (专) A、B 两个进程和一个缓冲区

四、进程的同步机制 信号量及 P、V 操作

解决进程同步互斥问题。

1、信号量：

是一个数据类型。定义如下：

```
TYPE semaphore= RECORD
  Value : integer;
  Queue : Pointer_PCB;
```

END

信号量说明：

VAR : S : semaphore;

在信号量上只能进行 P、V 操作。

2、P.V 操作

P 操作：

$P(s) :$

$s := s - 1;$

IF $s < 0$ then

该进程状态置为等待状态, 将该进程的 PCB 插入相应的等待队列末尾。

V 操作：

$V(s) :$

$s := s + 1;$

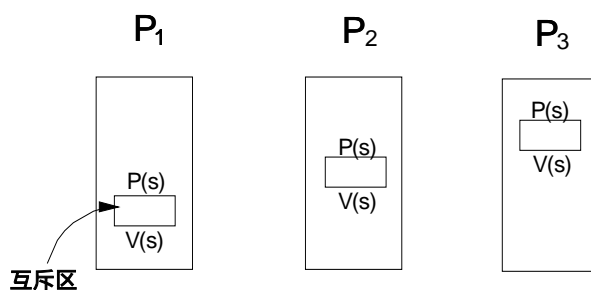
IF $s \leq 0$ then

唤醒相应等待队列中等待的一个进程, 改变其状态为就绪态, 并将其插入就绪队列。

P、V 操作是原语操作 (primitive, atomic action)

原语：是 (由若干条机器指令构成的) 完成某种特定功能的一段程序, 具有不可分割性。即原语的执行必须是连续的, 不允许被中断。

3、用 P.V 操作解决进程间互斥问题



说明：

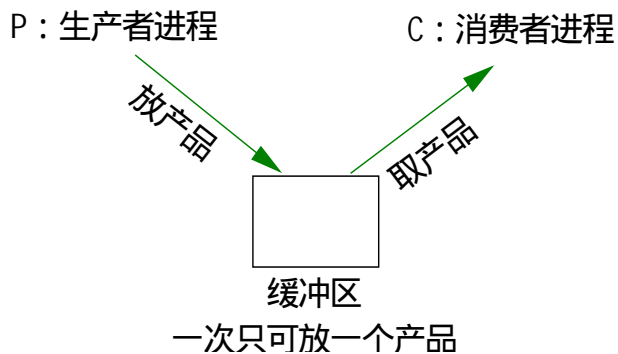
设置信号量：互斥信号量 S , S 初值为 1。

在进入互斥区前进行 P 操作, 出互斥区后进行 V 操作可解决上题问题。

假定 P_1 先操作, $P(s) = 0$ 则往下操作, 假设在互斥区中时间片到了, 则去执行 P_2 , 经过 $P(s)$ 时, 其值为 -1 则等待去执行 P_3 , 经过 $P(s)$ 时, 其值为 -2 并等待去执行 P_1 。当执行完毕经过 $V(s)$ 后, s 值为 -1, 同时唤醒一个相应等待队列中的进程假如唤醒 P_3 (此时不经过 $P(s)$ 直接进入互斥区), 退出后执行 $V(s)$, s 值为 0, 并唤醒 P_2 , 同样进行操作, 退出时经过 $V(s)$, s 值又转为初值 1。

4、用 P.V 操作解决进程同步问题

经典的生产者 消费者问题



同步问题：

P 进程不能往“满”的缓冲区中放产品。两次生产之间，必须有一次消费：设置信号量为 S_1 ， S_1 初值为 1。

Q 进程不能从“空”的缓冲区中取产品。两次消费之间，必须有一次生产，设置信号量 S_2 ， S_2 初值为 0。

P Repeat	Q Repeat
...	...
生产一个产品;	P (s_2);
P (s_1);	取一个产品;
送一个产品到缓冲区;	V (s_1);
V (s_2);	消费产品;
...	...
Until false	Until false

【思考题】要不要对缓冲区（临界资源）进行互斥操作？

解：取送不能同时，设置信号量 mutex ，初值为 1。

P Repeat	Q Repeat
...	...
P (s_1);	P (s_2);
生产一个产品;	P (mutex);
P (mutex);	取一个产品;
送一个产品到缓冲区;	V (mutex);
V (mutex);	V (s_1);
V (s_2);	消费产品;
...	...
Until false	Until false

【扩充 1】消息缓冲区问题（一个生产者，一个消费者， k 个缓冲区）

P :	Q :
$i := 0$	$j := 0$
REPEAT	REPEAT
生产产品 ;	P(S_2)
P(S_1)	从 Buffer[j]取产品 ;
往 Buffer [i]中放产品	V(S_1)
V(S_2)	消费产品
$i := (i+1) \bmod k$	$j := (j+1) \bmod k$

S_1 初值为 n , S_2 初值为 0

【扩充 2】消息缓冲区问题 (n 个生产者, m 个消费者, k 个缓冲区)

互斥信号量: mutex1 , mutex2 , 初值均为 1。

$P_x : (x=1,2,\dots,n)$

$i:=0$

REPEAT

生产产品;

$P(S_1)$

$P(\text{mutex1})$

往 Buffer $[i]$ 中放产品

$i:=(i+1) \bmod k$

$V(\text{mutex1})$

$V(S_2)$

UNTIL false

$Q_y : (y=1,2,\dots,m)$

$j:=0$

REPEAT

$P(S_2)$

$P(\text{mutex2})$

从 Buffer $[j]$ 取产品;

$j:=(j+1) \bmod k$

$V(\text{mutex2})$

$V(S_1)$

消费产品

UNTIL false

S_1 初值为 n , S_2 初值为 0

5、P、V 操作讨论

1) 信号量的物理含义

$S > 0$ 表示有 S 个资源可用;

$S = 0$ 表示无资源可用或表示不允许进程再进入临界区;

$S < 0$ 则 $|S|$ 表示在等待队列中进程的个数或表示等待进入临界区的进程个数。

$P(S)$: 表示申请一个资源; $V(S)$ 表示释放一个资源。

信号量的初值应该大于等于 0。

2) P 、 V 操作必须成对出现, 有一个 P 操作就一定有一个 V 操作。当为互斥操作时, 它们同处于同一进程。当为同步操作时, 则不在同一进程中出现。

如果两个 P 操作相邻, 那么它们的顺序至关重要, 而两个相邻的 V 操作的顺序无关紧要。一个同步 P 操作与一个互斥 P 操作在一起时, 同步 P 操作在互斥 P 操作前。

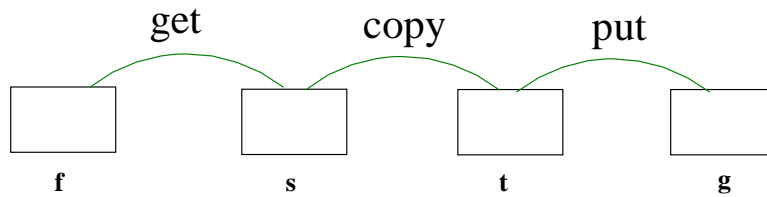
3) P 、 V 操作的优缺点

优点: 简单, 而且表达能力强 (用 PV 操作可解决任何同步互斥问题。)

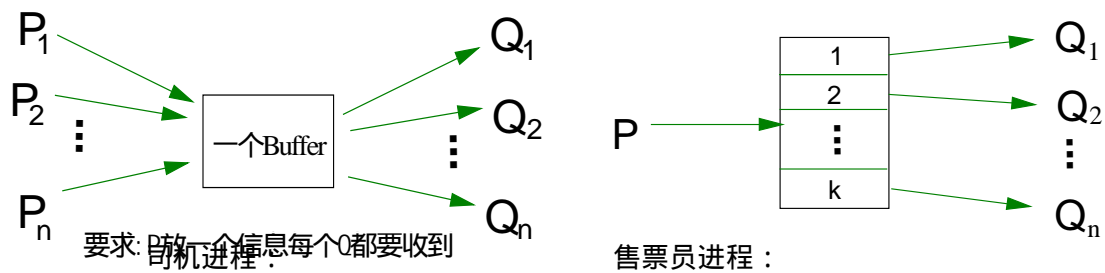
缺点: 不够安全; P 、 V 操作使用不当会出现死锁; 实现复杂。

【作业】

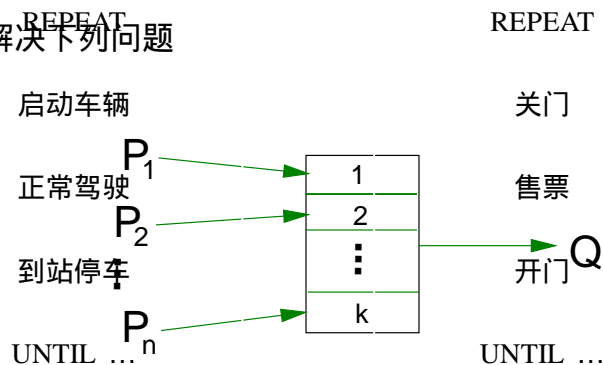
1、用 P.V 操作解决下图之同步问题。



2、用 P.V 操作解决司机与售票员的同步问题。



3、用 P.V 操作解决下列问题



6. 关于 P.V 操作的经典例子

读者与写者问题

有两组并发进程：读者和写者，共享一组数据区，要求：

- 1) 允许多个读者同时执行读操作。
- 2) 不允许多个写者同时操作。
- 3) 不允许读者，写者同时操作。

第一类：读者优先

如果读者到：1) 无写者，新读者可以读。

2) 无写者，有其它读者则新读者可以读。

3) 有其它读者，且有写者等候，新读者也可以读。

如果写者到：1) 无读者，新写者可以写。

2) 有读者，新写者等待。

3) 有其它写者，新写者等待。

互斥信号量：w, mutex; 初值均为 1。

教材上的例子：本科教材 P163, 专科教材 P106

读者：

教材上的例子：本科教材 P166, 专科教材 P109

readcount:=readcount+1;

8.3 进程通信

then (Pw);

写者：

P(w);

一、概念

读

写

1、为什么引入进程通信

P、V 操作实现的是进程之间的低级通信，所以 P、V 为低级通信原语。它只能传递简单的信息，不能传递交换大量信息。如果要在进程间传递大量信息，则要有专门的通信机制，即高级通信原语。

V(mutex);

2、进程通信机制

(1) 共享内存

相互通信的进程间设有公共内存，一组进程向该公共内存中写，另一组进程从公共内存中读，通过这种方式实现两组进程间的信息交换。

- 系统要提供公共内存
- 并为公共内存解决同步机制（读时不能写等）
（读者与写者问题的原型）

(2) 信件（消息）传递机制

信件（消息）：

发送者；消息（或地址长度）；等不等回信；回信存放地址；等等

高级通信原语：

系统为进程提供了两个高级通信原语 send 和 receive。

send：当要进行消息传递时，发送进程执行 send。

参数：信件，发送地址

receive：当接收者要接收时，接收进程执行 receive。

参数：信件存放地址，取信地址

信件（消息）传递机制又分：

* 消息缓冲

* 信箱通信

（3）管道通信

3、进程通信方式

（1）直接通信

发消息时要指定接收进程的名字和地址，反过来，接收时要指明发送进程的名字和地址。

1：1

（2）间接通信

1：M 或 M：1

二、信箱通信

1、信箱组成

信箱说明 与 信箱体

可存放信件数，已存放信件数，指针

2、信箱使用规则

（1）若发送信件时信箱已满，则发送进程被置为“等信箱”状态，直到信箱有空时才被释放。

（2）若取信件时信箱中无信，则接收进程被置为“等信件”状态，直到有信件时才被释放。

3、Send 实现

send (N, M)：把信件 M 送到指定的信箱 N 中。

实现：

查找指定信箱 N；

若信箱未满，则把信件 M 送入信箱且释放“等信件”者；

若信箱已满置发送信件进程为“等信箱”状态。

4、Receive 实现

receive (N, X)：从指定信箱 N 中取出一封信，存放到指定的地址 X 中。

实现：

查找指定信箱 N；

若信箱中有信，则取出一封信存于 X 中且释放“等信箱”者；
若信箱中无信件则置接收信件进程“等信件”状态。

5、应用实例

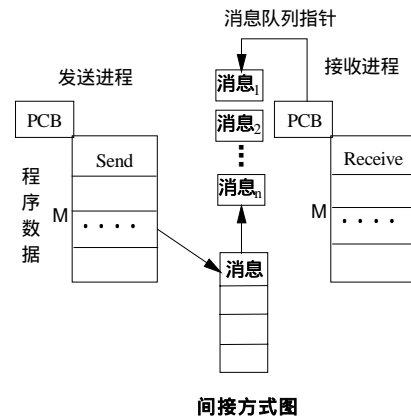
磁盘管理进程：从信箱中收取信件，处理，启动磁盘工作

其他进程：要求访问磁盘，向磁盘管理进程发一封信

三、消息缓冲

1、有界缓冲区工作原理

在操作系统空间设有一组缓冲区，当发送进程需要发送消息时，执行 send 系统调用，产生自愿性中断，进入操作系统，操作系统为发送进程分配一个空缓冲区，并将所发送的消息从发送进程空间 copy 到缓冲区中，然后将该载有消息的缓冲区连接到接收进程的消息队列末尾，如此就完成了发送过程，发送进程返回到用户态继续执行；在以后某个时刻，当接收进程需要接收消息时，执行 Receive 原语，也产生自愿性中断进入操作系统，由操作系统将载有消息的缓冲区从消息队列中取出，并把消息内容 copy 到接收进程空间，之后回收空缓冲区，如此完成了消息的接收，接收进程返回到用户态继续进行。



2、消息缓冲区结构

包括消息长度，消息正文，发送者，消息指针。

3、用 P、V 操作来实现 Send 原语

```
Send ( R , M )
Begin
  根据 R 找接收进程，如果没找到出错返回；
  申请空缓冲区 P ( s-b )；
  P ( b-mutex )；
  摘空缓冲区；
  V ( b-mutex )；
  把消息从 M 处 copy 到空缓冲区；
  P ( m-mutex )；
  把缓冲区挂到接收进程的消息链链尾；
  V ( m-mutex )；
  V ( s-m )；
END
```

其中 s-b 初值: n ； s-m 初值: 0

四、用进程通信实现进程同步

8.4 线程

一、线程的概念

1、线程

又称为“轻型进程”，线程是进程中可独立执行的子任务。

2、线程的特征

- (1) 每个线程有一个唯一的标识和一张线程描述表。
线程描述表：记录了线程执行时的寄存器和栈等现场信息。
- (2) 不同的线程可以执行相同的程序。
- (3) 同一进程中的各个线程共享该进程的内存空间，进程中的所有线程对进程的整个主存空间都有存取权限。
- (4) 线程是 CPU 调度的独立单位，多个线程可以并发执行，各线程可交替占用。
- (5) 线程亦有生命周期，线程在生命周期内会经历各种状态的变化。

线程与进程的根本区别：进程是资源的分配单位，而线程是调度和执行的独立单位。

二、为什么引入线程

1、进程机制的缺陷

- (1) 进程要占用一个进程控制块和一个私有的主存空间，开销较大
- (2) 进程之间的通信必须要由通信机制来完成，速度较慢
- (3) 进程增多会给调度和控制带来复杂性，增加了死锁的机会。

结论：应尽量避免设计过多的进程。

2、多线程技术的优越性

- (1) 创建线程不需另外分配资源，创建线程的速度比创建进程的速度快，且系统的开销也少。
- (2) 线程间的通信在同一地址空间中进行，不需要额外的通信机制，通信简便，快捷。
- (3) 线程能独立执行，能充分利用和发挥 CPU 与外围设备并行工作能力。

第九章 死锁

9.1 概述

一、死锁的基本概念

1、死锁的定义：

若系统中存在一组进程（两个或多个进程），它们中的每个进程都无限等待

被该组进程中另一进程所占有的因而永远无法得到的资源,这种现象称为进程死锁,这一组进程就称为死锁进程。

2、关于死锁的一些结论

- ☆ 参与死锁的所有进程都在等待资源。
- ☆ 参与死锁的进程至少有两个（两个以上进程才会出现死锁）
- ☆ 参与死锁的进程至少有两个已经占有资源。
- ☆ 参与死锁的进程是当前系统中所有进程的子集。

3、为什么会产生死锁？

资源

- ☆ 永久性资源：
可以被进程多次使用（可再用资源）
- ☆ 临时性资源：
可消耗性资源，只可使用一次。如信号量，中断信号，同步信号等

死锁产生的因素：

操作系统：资源管理/分配

（申请 分配 使用 释放）

资源数目有限，竞争资源

资源分配策略：资源动态申请、动态分配（申请主存、外设、读写文件等）

- ☆ 在进程竞争资源时，系统的资源管理和分配策略不当
- ☆ 并发进程的执行速度

4、死锁的严重性

如果死锁发生，会浪费大量系统资源，不仅涉及到死锁的哪些进程无法执行下去，而还将妨碍其他进程的执行，甚至导致系统崩溃。

5、几点假设

假设

我们只考虑由于操作系统对资源管理不当引起的死锁问题,而避免与硬件故障及其他程序性错误纠缠在一起。

- 任何一个进程要求资源的最大数量不超过系统能提供的最大量
- 任何一个进程在执行中所申请的资源能得到满足，那么它一定能在有限的时间内执行结束且归还它所占的全部资源。
- 一个进程在申请资源得不到满足时才处于等待资源状态。

不属于要本章讨论的死锁问题：

某个进程申请系统中不存在的资源或申请的资源数超过了系统拥有的最大资源；由于硬件故障或程序性错误引起的循环等待。

二、死锁产生的例子

1、申请不同类型资源产生死锁

P1：

...

申请打印机

申请扫描仪

使用

释放打印机

释放扫描仪

P2 :

...

申请扫描仪

申请打印机

使用

释放打印机

释放扫描仪

2、申请同类资源产生死锁

经典的哲学家就餐问题（本科教材 P179 页）

3、P、V 操作不当引起死锁

9.2 死锁的特征

一、产生死锁的四个必要条件

只要死锁发生，则这四个条件都存在，如果有一个条件不存在，那么，一定没有死锁发生。但四个条件都存在，不一定发生死锁。

1、互斥使用（资源独占）

一个资源每次只能给一个进程使用。

如果某一个进程请求一资源，该资源被另外进程占有时，则申请者等待，直到资源被释放。

2、占有并等待（部分分配，占有申请）

进程占有某些资源的同时又申请新资源，但得不到满足而处于等待资源状态，且不释放已经占有资源。

2、不可抢夺（不可剥夺）

资源申请者不能强行的从资源占有者手中夺取资源，资源只能由占有者使用完后自愿释放。

4、循环等待

存在一个进程等待队列 $\{P_1, P_2, \dots, P_n\}$ ，其中 P_1 等待 P_2 占有的资源， P_2 等待 P_3 占有的资源，... P_n 等待 P_1 占有的资源，形成一个环路。

二、资源分配图

用有向图描述进程的死锁（准确、形象）

系统由若干类资源构成，一类资源称为一个资源类（ m 个资源类）；每个资源类中包含若干个同种资源，称为资源实例（ R_i ）。

1、资源分配图（见下图）



资源类（资源的不同类型）

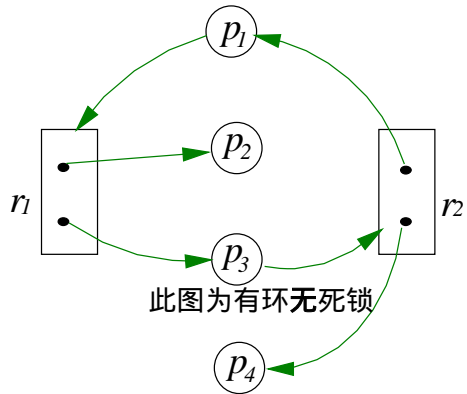
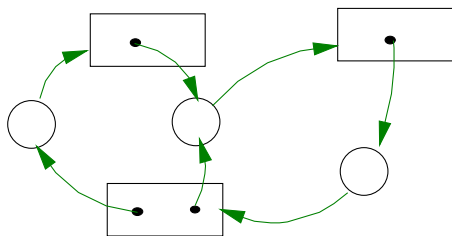
资源实例（存在于每个资源中）

进程

分配边：资源实例 进程有一条有向边

申请边：进程 资源类的一个有向边

用 “” 表示
用 “.” 表示
用 “” 表示



2、

(1) 如果图中没有环路则系统中没有死锁。

(2) 死锁定理

如果图中存在环路则系统中可能存在死锁,如果每个资源类中只包含一个资源实例,则环路是死锁的充分必要条件。

3、资源分配图化简法

(如果经过化简后图中仍有环路则有死锁否则无死锁)

方法：

1) 找一个非孤立的且只有分配边的进程结点,去掉分配边,将其变为孤立结点。

2) 把相应的资源分配给一个等待该资源的进程,即把一个申请边 分配边。

9.3 死锁解决方案

死锁的解决方案(预防,避免,解除)

解决死锁的方法

三种观点：

☆ 不考虑死锁问题(鸵鸟政策)

☆ 不让死锁发生：静态方式：对进程申请资源加以限制,不让死锁发生
-----死锁预防(资源分配算法原则上不让死锁发生。)

动态方式：进程在申请资源时,系统审查是否会产生死锁,若会产生死锁则不分配-----死锁避免

☆ 让死锁发生：然后解决死锁-----死锁检测与解除

一、死锁预防

1、定义：

在系统设计时确定资源分配算法,限制进程对资源的申请,从而保证不发生死锁。具体的做法是破坏产生死锁的四个必要条件之一。

(产生死锁的必要条件被破坏,使系统不进入死锁状态,只要破坏四个必要条件之一,则死锁不出现)

2、破坏“占有并等待”条件

(一) 静态分配资源

要求每一个进程在开始执行前就申请它所需要的全部资源，仅当系统能满足进程的资源申请要求且把资源分配给进程后，该进程才能开始执行。

采用静态分配资源的策略后，进程在执行过程中不再申请资源，故不可能出现占有了某些资源再等待其他资源的情况，即能使得“占有并等待”的条件不成立，从而防止死锁的发生。

缺陷：降低资源的利用率。

(二) 释放已占资源

仅当进程没有占用资源时才允许它去申请资源，如果进程已经占用了某些资源而又要再申请资源，则它应先归还所占的资源后再申请新资源。

这种资源分配策略仍会使进程处于等待状态，这是因为进程所申请的资源可能已被其他进程占用，只能等占用者归还资源后才可分配给申请者，但是，申请者是在归还资源后才申请新资源的，故不会出现占有了部分资源再等待其他资源的现象。

3、破坏“不可抢夺”条件

约定：如果一个进程已经占有了某些资源又要申请新资源，而新资源不能满足（已被其他进程占用）必须等待时，系统可以抢夺该进程已占有的资源。

具体做法如下：

- (1) 一个进程申请的资源尚未被占用，则系统可把资源分配给该进程。
- (2) 若进程 A 申请的资源 r 已被进程 B 占用，则查看进程 B 的状态，如果进程 B 处于等待另一个资源的状态，那么就抢夺进程 B 已占用的资源 r 并把 r 分配给进程 A；如果进程 B 不是处于等待状态，则让进程 A 处于等待资源 r 的状态。
- (3) 一个等待资源的进程只有在得到自己所申请的新资源和所有被抢夺的老资源后才能继续执行。

这种资源分配策略并不是对所有资源都适用。例如，打印机、磁带机等就不能采用抢夺的方式，否则造成使用混乱。抢夺式的分配策略只适用于主存空间和处理器。

缺点：实现起来困难，而且付出一定代价。

4、破坏“循环等待”条件

采用资源有序分配法：把系统中所有资源排序编号，进程在申请资源时必须严格按资源编号的递增次序进行，否则操作系统不予分配。

把按序分配的策略应用到 5 个哲学家问题中，规定每个哲学家想吃面时总是先取编号小的叉子再取编号大的叉子就不会形成死锁。根据按序分配的要求，可修改第 5 个哲学家的程序（本科教材 P187 页）。

例如：1, 2, 3, ..., 10

P1：

申请 1

申请 3

申请 9

...
P2 :
申请 1
申请 2
申请 5
...
P3 :

P10

二. 死锁避免

1、定义：系统中对进程发出的每一个系统能够满足的资源申请进行动态检查，并根据检查结果决定是否分配资源，如果分配后系统可能发生死锁，则不予分配，否则予以分配。

是一种保证系统不进入死锁状态的动态策略

2、安全状态与不安全状态

- 😊 安全状态：如果存在一个由系统中所有进程构成的安全序列 P_1, \dots, P_n ，则系统处于安全状态。
- 😊 安全序列：一个进程序列 $\{P_1, \dots, P_n\}$ 是安全的，如果对于每一个进程 $P_i (1 \leq i \leq n)$ ，它以后尚需要的资源量不超过系统当前剩余资源量与所有进程 $P_j (j < i)$ 当前占有资源量之和。
(有安全序列则一定是没有死锁发生)
- 😊 不安全状态：不存在一个安全序列。不安全序列一定导致死锁，但不安全状态不一定是死锁状态。

3、银行家算法

死锁避免：在进行资源分配时动态检查资源请求，如果发现分配资源后系统进入不安全状态，则不予分配，否则予以分配。

	目前占有量	最大需求量	尚需要量
P1	1	4	3
P2	4	6	2
P3	5	8	3
系统剩余量	2		

其安全序列为 P2, P1, P3 或 (P2, P3, P1)

三. 死锁的检测与解除

死锁防止和死锁的避免都是要对资源的分配加以限制，操作系统解决死锁的问题的另一条途径是“死锁检测”方法，这种方法对资源的分配不加限制，只要有剩余的资源，就可把资源分配给申请者。

1、死锁检测

允许死锁发生，但操作系统不断监视系统进展情况，判断死锁是否真的发生，一旦发现死锁，则采取专门的措施解除死锁并以最小的代价恢复系统运行。

2、检测时机

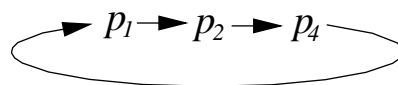
- 当进程等待时检测死锁，其缺点是系统的开销大。
- 定时检测，但有可能检测出死锁时死锁已发生很久了。
- 系统资源利用率下降时检测死锁。

3、死锁检测算法

(1) 每个资源类中只有一个资源

如果每个资源类中只有一个资源，可设置两张表格来记录进程使用资源的情况。

判断有无死锁发生的思路：先看进程等待表，再看资源分配表。如果有环路产生则有死锁。如上表， P_1, P_2, P_4 可构成一个环路，所以必有死锁发生。



资源分配表		进程等待表	
资源	进程	资源	进程
R_1	P_1	P_1	R_2
R_2	P_2	P_2	R_3
R_3	P_4	R_4	P_4
R_4	P_3		

第一步，找出资源已经满足的进程，即不再申请资源的进程，若有这样的进程则它们一定能在有限的时间内执行结束且归还所占的资源，所以，可以把它们所占的资源与系统中还剩余的资源加在一起作为“可分配的资源”，同时对这些进程置上标志。

第二步，检测所有无标志的进程，找出一个尚需资源量不超过“可分配的资源”量的时程，若能找到，则只要把资源分配给该进程就一定能在有限时间内收回它所占的全部资源，故可把该进程已占的资源添加到“可分配的资源”中，同时为该进程置上一个标志。重复第二步的执行直到所有进程均有标志或无标志的进程尚需资源量均超过“可分配的资源”量。

第三步，若进程均有标志，表示当前不存在永远等待资源的过程，也即系统不处于死锁状态；若存在无标志的进程，表示当前系统已有死锁形成，这些无标志的进程就是一组处于死锁状态的进程。检测结束，清除检测时所设置的所有标志。

4、死锁的解除

重要的是以最小的代价恢复系统的运行。

两种方法：

一种是终止一个或几个进程的执行为破坏循环等待；另一种是从涉及死锁的进程中抢夺资源。

撤消进程（中止死锁进程，收回所占用资源）；

剥夺资源；

（重新启动，代价太大；进程回退）

（1）终止进程

终止涉及死锁的进程，系统可收回被终止进程所占的资源进行再分配，以达到解除死锁的目的。

- 终止涉及死的所有进程

代价大

- 一次终止一个进程

进程的重新执行：从头开始 或 从某一校验点开始。

（2）剥夺资源

从涉及死锁的一个或几个进程抢夺资源，把夺得的资源再分配给卷入死锁的其他进程直到死锁解除。

应考虑三个问题：

- 抢夺哪些进程的哪些资源

总是希望能以最小的代价结束死锁，因而必须考虑涉及死锁的进程所占有的资源数，以及它们已经执行的时间等因素。

- 被抢夺者的恢复
- 进程的“饿死”

用死锁检测及解除方法解决死锁，适用于不经常发生死锁的系统中。

【作业：】

问题

1. 此状态是否为安全状态，如果是找出安全序列。

2. 在其基础上

1) P_2 申请 (1, 0, 2) 能否分配？为什么？

2) P_5 申请 (3, 3, 0) 能否分配？为什么？

3) P_1 申请 (0, 2, 0) 能否分配？为什么？

	已分配的资源			最大需求量		
	A	B	C	A	B	C
P_1	0	1	0	7	5	3
P_2	2	0	0	3	2	2
P_3	3	0	2	9	0	2
P_4	2	1	1	2	2	2
P_5	0	0	2	4	3	3
剩余资源	A	B	C			
	3	3	2			

