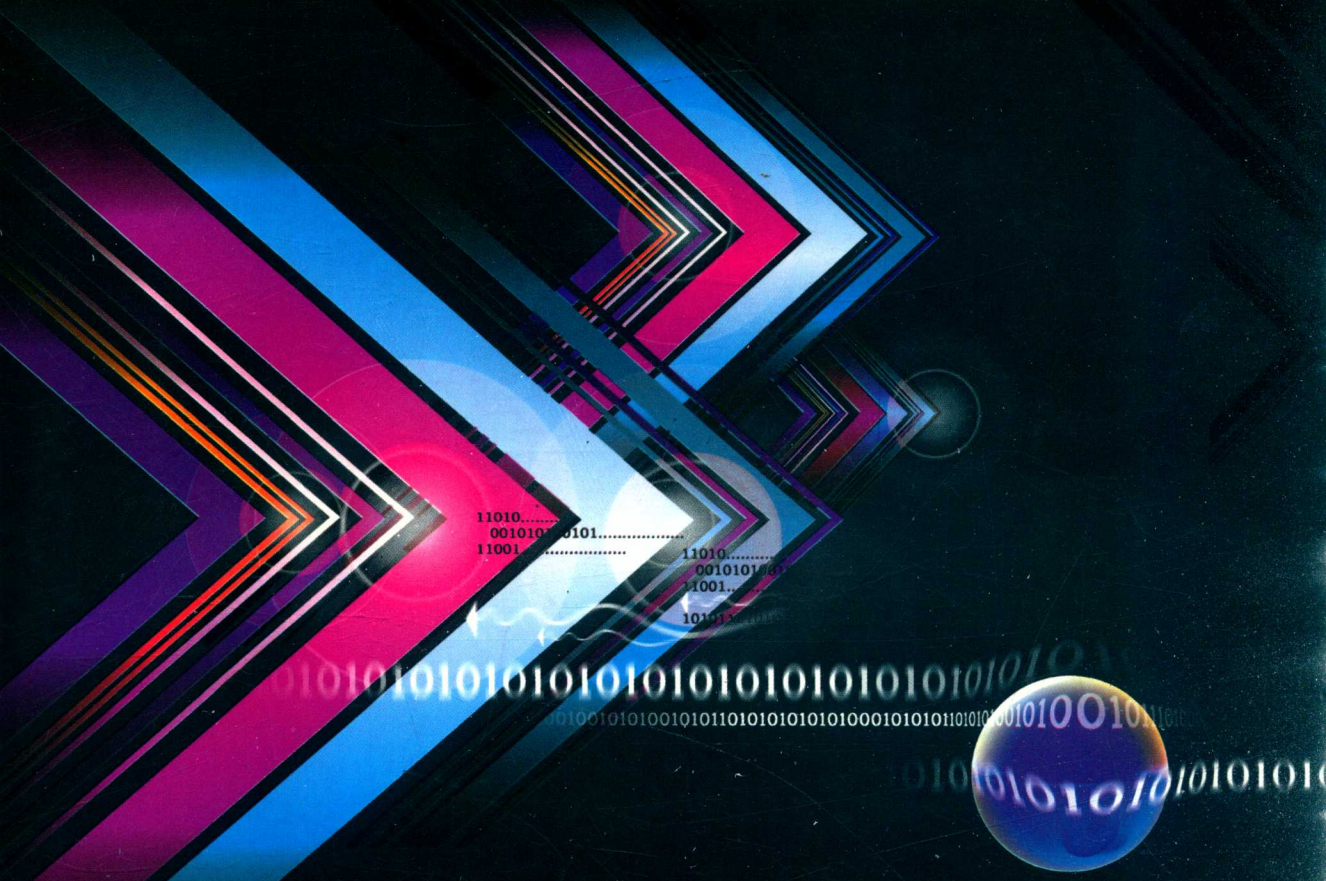
中国电子教育学会高教分会推荐

普通高等教育电子信息类“十三五”课改规划教材



计算机操作系统

刘晓建　岳国华　编著

李占利　李军民　审

西安电子科技大学出版社

http://www.xduph.com

中国电子教育学会高教[redacted]

普通高等教育电子信息[redacted]训教材

计算机操作系统

刘晓建　岳国华　编著

李占利　李军民　审

西安电子科技大学出版社

内 容 简 介

本书是参考教育部颁布的计算机软件专业操作系统教学大纲，结合编者多年积累的教学经验和科研成果，借鉴国内外经典教材的优点编写而成。书中主要介绍了操作系统基本概念、原理和方法，力图体现计算思维、形式化方法和软硬件一体化等基本理念与思维方式在操作系统中的运用。

本书适合作为高等学校计算机相关专业的本科教材，也可作为相关人员的自学参考书。

图书在版编目(CIP)数据

计算机操作系统/刘晓建，岳国华编著. —西安：西安电子科技大学出版社，2017.7
普通高等教育电子信息类“十三五”课改规划教材
ISBN 978-7-5606-4385-4

Ⅰ. ① 计… Ⅱ. ① 刘… ② 岳… Ⅲ. ① 操作系统—教材 Ⅳ. ① TP316

中国版本图书馆 CIP 数据核字(2016)第 293505 号

策　　划　毛红兵
责任编辑　买永莲
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西天意印务有限责任公司
版　　次　2017 年 7 月第 1 版　2017 年 7 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印　张　18
字　　数　424 千字
印　　数　1～3000 册
定　　价　38.00 元
ISBN 978-7-5606-4385-4/TP

XDUP 4677001-1

前　言

操作系统是当今最复杂的系统软件之一，是所有复杂应用软件的基础。操作系统已经深度融入到几乎所有与信息处理相关的系统中，小到各种传感器、智能手机、掌上电脑，大到超级计算机、云计算平台，甚至整个互联网络。

操作系统的种类多种多样，运行的硬件平台、应用目标和行为特征各不相同，从教学来说，不可能一一列举所有这些种类的操作系统。本书不是介绍某种具体的操作系统，而是在回顾计算机和操作系统发展历史的基础上，重点介绍操作系统的基本概念、原理和方法。因此，本书内容具有一般性和普适性。掌握了这些内容之后，可以将这些概念、原理和方法应用到某种具体的操作系统中。

计算机操作系统是计算机相关专业的一门重要的理论基础课，是学习和开发大型复杂软件系统的基础，同时也是学习程序开发方法、并发程序设计，甚至程序语义学的基础。“计算机操作系统”课程以计算机组成原理、数据结构、编译原理、程序设计语言等课程为先行课程，但是在学习过程中，并不要求学生对计算机硬件结构、编译原理等有详细了解，如果需要，则会在相关章节进行抽象和宏观的说明。

本书的编者是在高校长期从事操作系统教学和研究的人员。在编写本书时，参考了多部国内外经典教材，并根据教学活动中学生的反馈意见，进行了内容的合理选择和组织。本书具有以下主要特色：一是注重基本概念、方法和原理的讲解，力求做到概念准确、原理透彻，能够满足教学以及工程开发的基本要求；二是加强操作系统不同知识模块间的联系，便于学习者对操作系统形成一个系统化认识；三是通过例子以及 UNIX 系统调用的介绍，将抽象的概念和原理具体化，变得更容易理解和操作；四是增加了硬件基础知识的介绍，有利于形成软硬件一体化的思维方式，同时便于不具备计算机硬件基础的人员学习。

全书共七章。第一章为计算机操作系统概论，介绍了操作系统的基本功能，总结了操作系统的结构，回顾了操作系统的发展历史并展示了操作系统的全貌；第二章为操作系统的硬件基础，从宏观角度介绍了计算机的基本模块、指令集、指令循环和处理器模式，可为后续操作系统的学习建立硬件基础；第三章是进程管理，介绍了进程的概念、结构和状态迁移模型，进程的控制、进程的调度策略，以及线程的概念；第四章是进程的并发和死锁，主要介绍了实现并发控制的基本机制，如信号量、管程、消息传递等，分析了典型并发设计问题的解决方案，深入讨论了死锁以及处理死锁的各种策略；第五章是内存管理，介绍了虚拟内存的概念，分页式、分段式内存管理方法以及采取的管理策略；第六章为文件管理，介绍了文件的属性、结构、存储空间管理、目录结构、共享以及文件系统的保护等；第七章是输入/输出系统，介绍了 I/O 硬件结构和组织、软件组织、缓冲处理技术、磁盘驱动调度和 I/O 进程控制等。

本书的第一章至第六章由刘晓建编写，第七章由岳国华编写，最后由刘晓建统稿，李占利教授和李军民教授审核了全书。

本书的出版受到陕西省自然科学基础研究计划面上项目(2017JM6105)的资助，同时得到了西安科技大学校级规划教材项目和西安电子科技大学出版社的大力支持，在此表示衷心的感谢。

限于编者水平，书中不妥与疏漏之处在所难免，恳请广大读者批评指正。

编著者

2017 年 4 月

目　录

第一章　计算机操作系统概论

计算机操作系统是当今最复杂的系统软件之一，它是几乎所有复杂应用软件的基础。操作系统已经深度融入到几乎所有与信息处理相关的系统之中，小到各种传感器、手机、掌上电脑，大到整个互联网。它的应用如此广泛，以至于我们感觉不到它的存在。

本章围绕操作系统的基本概念展开，试图从三个角度理解操作系统的概念，即功能角度、资源管理角度和计算环境角度；另外，本章回顾了操作系统的发展历史，总结了操作系统的基本结构，对已经渗透到社会生活方方面面的操作系统给出了一个全景式的概述。

本章学习内容和基本要求如下：

- 从三个角度理解计算机操作系统的概念。
- 在学习操作系统发展历史的基础上，重点掌握多道程序设计的概念、原理和实现；掌握分时系统的基本原理。
- 了解操作系统的典型结构，重点掌握层次结构、微内核结构的优点和不足；对虚拟机结构有所了解。

1.1　操作系统的概念

操作系统本质上是一组程序，它管理和控制着其他程序的执行，并充当应用程序和计算机硬件之间的接口。一般来说，操作系统应满足如下三个应用需求：

(1) 方便性。从用户使用的角度来看，操作系统应当为用户使用计算机提供便利。

(2) 有效性。从计算资源分配和管理的角度来看，操作系统应当为用户程序的执行提供必要的计算环境，并合理有效地调配各种计算机资源，使得系统整体使用效率得到优化和提高。

(3) 可扩展性。从操作系统本身的设计和构造方面来看，它应当满足可扩展性，即在构造操作系统时，应该允许在不妨碍服务的前提下，有效地开发、测试和引进新的系统功能。

当然，随着操作系统的普遍使用，人们对于操作系统的上述需求有所偏重。比如对于嵌入式操作系统，由于其面向专门的应用场合并工作在资源受限的环境下，人们更偏重于操作系统性能的有效性；而对于桌面或手机操作系统而言，由于它们主要完成与用户的交互，因此对操作系统的方便性要求更加突出。

一般来说，可以从以下三个角度来理解操作系统的概念：用户使用角度、计算资源管理和控制角度以及计算环境角度。

1.1.1　从用户使用角度理解操作系统

计算机用户大致可以分为三类：使用者、开发者和维护者。无论是哪一类用户，操作系统都应当为他们使用计算机提供便利。从用户角度来看，操作系统表现为一组可用的功

能和提供这些功能的接口。总的来说，操作系统为用户提供了如下功能：

(1) 程序开发。操作系统提供了各种各样的工具和服务，如编辑器、编译器和调试器，帮助开发者开发程序。这些服务通常以实用工具程序的形式出现，严格来说，它们并不属于操作系统的核心部分。

(2) 程序运行。运行一个程序需要很多步骤，包括必须把指令和数据载入到内存、初始化 I/O 设备和文件以及准备其他一些资源。操作系统为用户程序的执行提供必要的环境，并处理这些调度问题。

(3) I/O 设备访问。对每个 I/O 设备的访问都需要特定的指令集或控制信号，操作系统隐藏这些细节并提供了统一的访问接口，使得程序员可以使用简单的读、写操作访问这些设备。

(4) 文件访问和控制。操作系统能够为用户提供简单、抽象的文件访问和控制方法，屏蔽对文件的具体存储介质(如磁盘、磁带)的访问和控制，隐藏存储介质中文件数据的存储结构等。此外，对于多用户系统，操作系统还可以提供保护机制来控制对文件的访问。

(5) 并发控制和系统保护。对于共享或共有的系统资源，操作系统对这些系统资源和数据进行并发访问控制及安全保护，解决资源竞争的冲突问题，避免未授权用户的非法访问。

(6) 错误检测和响应。计算机系统运行时，可能发生各种各样的错误，包括内部和外部硬件错误(如存储器错误、设备失效或故障)以及各种软件错误(如算术溢出、试图访问被禁止的存储器单元、执行非法指令等)。对每种情况，操作系统都必须给予响应以清除错误条件，减小其对正在运行的应用程序的影响。响应可以是终止引起错误的程序、重新执行操作或简单报告应用程序错误等。

(7) 日志和记账。一个好的操作系统可以收集对各种资源使用的统计信息，监控诸如响应时间之类的性能参数。在任何系统中，这些信息对于预测将来增强功能的需求以及调整系统以提高性能等都大有用处。对于多用户系统，这个信息还可用于记账。

1.1.2 从计算资源管理和控制角度理解操作系统

从资源管理角度来看，操作系统是计算资源的管理者。当多个任务或多个用户同时请求有限的计算资源时，为了防止资源冲突，并且高效地利用这些资源，需要一个管理者从中进行合理有效的分配和协调。计算资源包括：

(1) 处理器。对于单处理器计算机，当多个应用程序同时请求运行时，就会发生处理器资源的争夺。协调多个应用程序的执行，需要使用操作系统的调度机制来解决。即使对于多处理器计算机，仍然存在处理器资源的分配、协调和负载均衡问题。比如，当一个任务到来时，究竟让哪一台处理器执行该任务，或者当一个正在执行的任务需要与另一个运行在其他处理器上的任务通信时，都需要操作系统的处理器调度机制来参与协调。

(2) 内存。当多个应用程序同时运行时，就会争夺内存资源。如何分配内存资源，使得每个应用程序既能相互隔离、避免相互干扰，又能共享特定的区域、方便应用程序之间的通信和联络，同时还能提高内存的使用效率，最大程度上发挥处理器的处理能力，这就需要操作系统的内存管理机制来加以解决。

(3) 外部存储介质。对于大量的数据信息，如何有效地进行存储、访问以及保护，是操作系统存储器管理和文件系统管理必须解决的问题。

(4) I/O 资源。I/O 资源是计算机系统中最为丰富多彩，同时也是控制最为复杂的一部

分资源。

(5) 实用程序、关键数据和文档。除了上面的硬件资源外，实用程序、关键数据和文档也是计算资源的组成部分。

所有这些资源的管理和控制问题，其核心都是一个最优化问题，也就是如何在有限资源的约束下，满足每个应用的需求，同时使整个系统的某个或某些指标达到最优。这是作为资源管理者的操作系统重点需要解决的问题。这些问题将在后续章节详细展开。

操作系统对应用程序及资源的管理和控制方式与普通的自动控制系统有所不同。在自动控制系统中，控制器与被控对象通常是不同的事物，被控对象一般是物理过程，而控制器一般是包含控制算法的计算机系统，控制器根据负反馈原理，通过执行器向被控对象施加操作，实现控制目标。操作系统对资源的控制方式与自动控制系统有三方面不同：

(1) 操作系统与普通的应用程序一样，都是由处理器执行的一段或一组程序。同时，操作系统还要管理和控制其他程序，因此操作系统可以称为“元程序”(Meta program)。

(2) 操作系统不能独立于计算机资源之外而存在，它既是计算资源的使用者，又是计算资源的管理者。操作系统管理和控制着处理器、内存和外存等计算资源，同时它的运行和存储也必须用到这些资源。这一点决定了操作系统必须耗费和占有计算资源。

(3) 操作系统对应用程序的控制并不是由操作系统来解释和执行应用程序，而是通过操作系统对资源的控制和释放来实现的。当一个应用程序需要执行时，操作系统必须放弃处理器资源，让它转而执行该应用程序；当需要操作系统进行控制和管理时，处理器又必须从应用程序被切换到操作系统程序，执行相关控制功能；控制结束之后，处理器又要被切换回应用程序，继续执行有意义的操作。图 1-1 形象地说明了操作系统对处理器的控制和释放过程，其中黑粗线条表示操作系统或应用程序正在占用处理器。

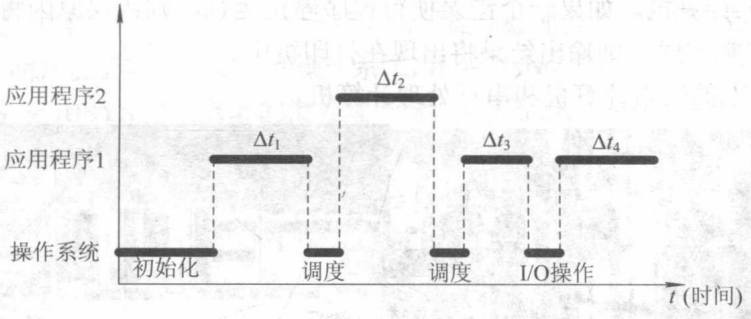


图 1-1　操作系统对处理器的控制和释放

1.1.3　从计算环境角度理解操作系统

从提供计算环境的角度来看，可以把操作系统看作：为相互独立运行的用户或进程，提供隔离的虚拟计算环境的提供者。在后续章节的学习中将会看到，通过处理器调度、虚拟内存、虚拟磁盘以及 I/O 多路复用和磁盘调度等技术的使用，操作系统可以为一个用户或进程创建一个独立、自含的计算环境，并给用户造成自己私人占有整个计算机的假象，我们把这样的计算环境称为虚拟计算环境或虚拟机。

虚拟计算环境能够很好地实现硬件计算资源的共享性和用户之间的逻辑隔离性。当一

个用户需要进行计算时，总需要处理器、内存、存储器和 I/O 等资源，但是在一个多用户系统中，这些资源不可能完全分配给该用户；另一方面，每个用户都希望能够拥有各自独立的计算空间，相互不发生干扰，圆满地完成计算任务。虚拟计算环境就是为了满足这两个看似相互矛盾的需求而提出的。

虚拟计算环境为用户或进程使用计算机提供了一个“视图”(View)，或者说虚拟计算环境就是一个用户或进程所看到的计算机的映像，这些映像是整个计算机系统的一个局部、侧面或片段。要为一个用户创建一个虚拟计算环境，并保证计算环境的正确、可靠和有效，就需要操作系统的支持。

虚拟计算环境的概念在操作系统的设计中运用得较为普遍。比如在分时系统中，运用时分复用原理，为每个用户分配周期性时间片资源，如果周期足够小，就会造成用户自己独占计算资源的假象；再如，在目前广泛使用的 Android(安卓)手机操作系统中，每个应用程序都运行在一个 Dalvik 虚拟机的基础上，使得应用程序相互隔离。当一个程序出错时，错误被限定在自己的计算环境中，不会干扰其他应用程序，从而提高了系统的安全性和可靠性。

1.2　操作系统的发展历史

1.2.1　串行处理

20 世纪 40 年代后期到 50 年代中期，计算机处于早期发展阶段。当时还没有出现操作系统，程序员直接与计算机硬件打交道。这些机器都运行在一个控制台上，控制台包括显示灯、触发器、某种类型的输入设备和打印机等。用机器代码编写的程序通过输入设备(如卡片阅读机)载入计算机。如果一个错误使得程序停止运行，则错误原因将由显示灯指示。如果程序运行正常完成，则输出结果将出现在打印机中。

图 1-2 为早期的纸带计算机和串行处理计算机。

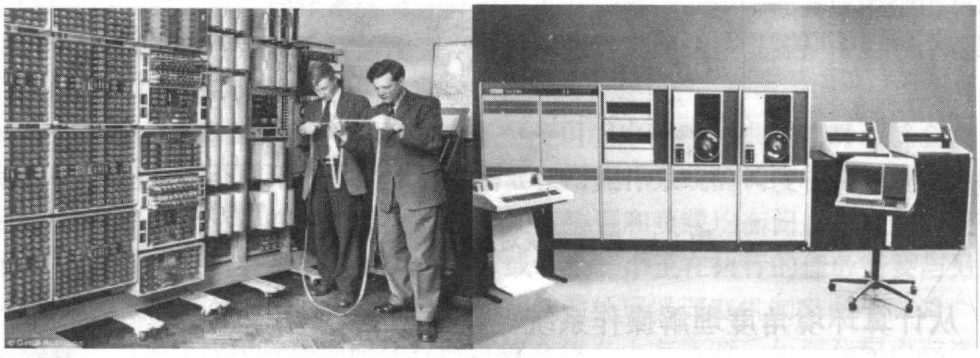


图 1-2　早期的纸带计算机和串行处理计算机

该时期，人工控制和使用计算机的过程大致如下：

(1) 输入源程序：人工把源程序穿孔在卡片或纸带上（卡片和纸带相当于外部存储器），变成计算机能够识别的输入形式。

(2) 加载系统程序：将准备好的汇编解释程序或编译系统（也在卡片或纸带上）装入计算机。

(3) 加载待汇编/编译源程序：汇编程序或编译系统读入人工装在输入机上的穿孔卡片或穿孔带上的源程序。

(4) 执行汇编或编译命令：执行汇编过程或编译过程，产生目标程序，并输出到目标卡片或纸带上。

(5) 装入可执行程序：通过引导程序把装在输入机上的目标程序读入计算机。

(6) 运行可执行程序，加载待处理数据：启动目标程序，从输入机上读入人工装好的数据卡片或数据带上的数据。

(7) 输出结果：产生计算结果，并将结果从打印机上或卡片机上输出。

人工操作的缺点如下：

(1) 用户独占全机资源，造成资源利用率不高，系统效率低下。

(2) 手工操作多，处理机时间浪费严重，也极易产生差错。

(3) 上机周期长，作业调度不合理。大多数装置都使用一个硬拷贝的登记表预定机器时间。通常，一个用户可以以半小时为单位登记一段时间。有可能用户登记了 1 小时，而只用了 45 分钟就完成了工作，在剩下的时间中计算机只能闲置，这时就会导致浪费。另一方面，如果一个用户遇到一个问题，没有在分配的时间内完成工作，在解决这个问题之前就会被强制停止。

这种操作模式称为“串行处理”，反映了用户必须顺序访问计算机的事实。后来，为使串行处理更加有效，人们开发了各种各样的系统软件工具，其中包括公用函数库、链接器、加载器、调试器和 I/O 驱动程序等，作为公用软件，所有的用户都可使用它们。

1.2.2　简单批处理系统

早期的计算机非常昂贵，由于调度和准备而浪费计算机处理器资源是难以接受的，因此最大限度地利用处理器是当时面临的主要问题。为了提高处理器的利用率，General Motors 在 20 世纪 50 年代中期开发了第一个批处理操作系统，并用在 IBM 701 上。运行在 IBM 7090/7094 计算机上的操作系统 IBSYS 是最著名的批处理系统，对其他操作系统有着广泛影响。

简单批处理方案的中心思想是使用一个称作监控程序的软件。通过这个软件，用户不再直接访问机器，用户把卡片或磁带中的作业程序提交给计算机操作员，由他把这些作业按顺序组织成一批，并将整批作业放在输入设备上，供监控程序使用。每个作业处理完成后，返回到监控程序，由监控程序自动加载下一个程序。

监控程序控制作业的顺序，为此大部分监控程序必须常驻内存并且可以执行，这部分程序称作常驻监控程序(Resident monitor)，如图 1-3 所示。用户程序区域包括用户程序以及一些实用程序和公用函数，它们作为用户程序的子程序，在需要用到它们时才被载入。监控程序每次从输入设备(通常是磁带驱动器或卡片阅读机)中读出一个作业，把它放置在用户程序区域，并且把控制权交给这个作业。当作业完成后，控制权被返回给监控程序，监控程序立即读取下一个作业。每个作业的结果被发送到输出设备(如打印机)，交付给用户。

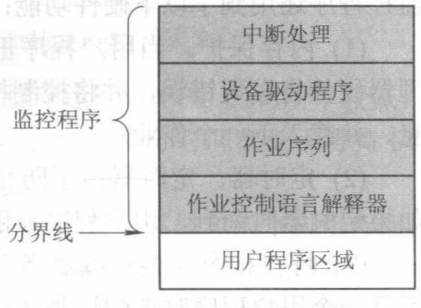


图 1-3　常驻监控程序的内存布局

监控程序完成调度功能。一批作业排队等候，处理器尽可能迅速地执行作业，没有任何空闲时间。监控程序还改善了作业的准备时间。每个作业的执行步骤由作业控制语言(JCL，Job Control Language)所编制的程序来给出，监控程序读取该程序，根据作业控制语句的指示，完成每个执行步骤。

例如，用户提交一个用 FORTRAN 语言编写的程序以及程序需要用到的一些数据，所有 FORTRAN 程序和数据在一个单独打孔的卡片中，或者是磁带中的一个单独的记录。除了 FORTRAN 程序和数据之外，作业中还包括作业控制指令，这些指令以“$”符号开头。作业的整体格式如下：

```
$JOB
$FTN
...
...(FORTRAN 程序)
...
$LOAD
$RUN
...
...(数据)
...
$END
```

为执行这个作业，监控程序读$FTN 行，从磁带中载入合适的语言编译器。编译器将用户程序翻译成目标代码，并保存在内存或磁带中。在编译操作之后，监控程序重新获得控制权，此时监控程序读$LOAD 指令，启动一个加载器，并将控制权转移给它，加载器将目标程序载入内存(在编译器所占的位置中)，之后开始执行目标程序。

在目标程序的执行过程中，任何输入指令都会读入一行数据。目标程序中的输入指令导致调用一个输入例程，输入例程是操作系统的一部分。如果输入过程中发生错误，那么控制权转移给监控程序进行错误处理。用户作业完成后，监控程序扫描输入行，直到遇到下一条 JCL 指令。

可以看出，监控程序或者说批处理操作系统只是一个简单的计算机程序，它依赖于处理器可以从内存中的不同部分取指令并执行的能力，以交替地获取或释放控制权。此外，监控程序还用到了以下硬件功能：

(1) 内存保护。当用户程序正在运行时，不能改变监控程序所在的内存区域，否则处理器硬件将发现错误，并将控制权转移给监控程序，监控程序取消这个作业，输出错误信息，并载入下一个作业。

(2) 定时器。定时器用于防止一个作业独占系统。在每个作业开始时，设置定时器，如果定时器时间到，用户程序被停止，控制权返回给监控程序。

(3) 特权指令。某些机器指令设计成特权指令，只能由监控程序执行。如果处理器在运行一个用户程序时遇到这类指令，则会发生错误，并将控制权转移给监控程序。I/O 指令属于特权指令，因此监控程序可以控制所有 I/O 设备，此外还可以避免用户程序意外地读

到下一个作业中的作业控制指令。如果用户程序希望执行 I/O 操作，它必须请求监控程序为自己执行这个操作。

(4) 中断。这个特性能够让一个用户程序放弃处理器的执行权，将处理器的控制权交给监控程序，让监控程序开始运行。

内存保护和特权指令引入了操作模式的概念。用户程序在用户态或用户模式下执行，在该模式下，有些内存区域是受到保护的。用户程序也不允许执行特权指令，因此可以防止用户程序有意或无意地破坏关键内存区域或越权操纵 I/O 设备。监控程序在系统态或内核模式运行，在这个模式下，可以执行特权指令，可以访问任何内存区域。

1.2.3　多道程序批处理系统

多道程序批处理系统是指允许多个程序同时进入一个计算机系统的主存储器并启动，进行交替执行的方法，即计算机内存中同时存放了多道程序，它们都处于开始与结束点之间。从宏观上看，多道程序并发运行，它们都处于运行过程中，但都未运行结束。从微观上看，多道程序的执行是串行的，各道程序轮流占用处理器，交替执行。多道程序设计技术的硬件基础是中断机制和通道技术。中断机制能够使一个程序在执行过程中被其他程序所打断，进而将处理器的执行权从一个程序切换到另一个程序；通道技术使程序中复杂繁琐的 I/O 操作的控制和具体实施过程被代理到通道处理机，处理器得到了释放，进而可以执行其他应用程序。

下面通过两个例子来说明多道程序处理如何提高处理器的利用率和系统吞吐量。

【例 1-1】 某个数据处理问题 P1，要求从输入机上输入 500 个字符(花费 70 ms)，经处理器(CPU)处理 50 ms 后，将结果的 2000 个字符存到磁带上(花费时间 100 ms)，重复这个过程，直至数据全部处理完毕。试计算这个问题中 CPU 的利用率。

解　先画出数据处理问题 P1 的时序图，如图 1-4 所示。

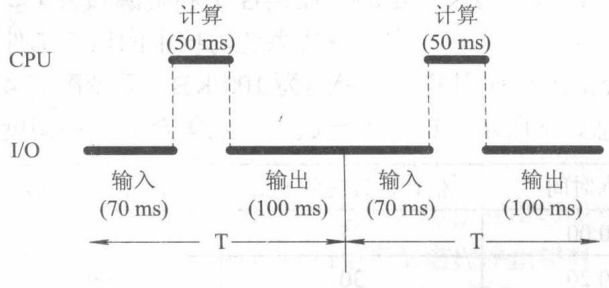


图 1-4　单道程序运行时 CPU 和 I/O 工作的时序图

CPU 的利用率 = 50/(70 + 50 + 100) = 23%。可见单道程序运行时 CPU 的利用率较低，主要原因是 I/O 的执行速度远低于 CPU 的执行速度，使得 CPU 大部分时间都处于等待 I/O 完成的状态，宝贵的 CPU 资源被浪费了。

为了提高 CPU 的利用率，如果内存空间足够大，可以容纳操作系统和两个应用程序，那么当一个作业正在等待 I/O 时，处理器可以切换到另一个可能并不在等待 I/O 的作业。进一步还可以扩展内存以保存三个、四个或更多的程序，并且在它们之间进行切换。这种处理方式就是“多道程序设计”(Multiprogramming)或“多任务处理”(Multitasking)，如图

1-5 所示，这是现代操作系统采用的主要方案。

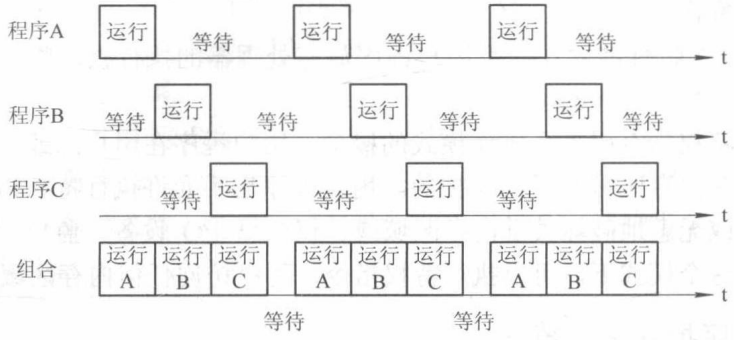


图 1-5　多道程序设计实例

【例 1-2】 在图 1-5 中，假定程序 A、B 和 C 的运行模式是相同的，在一个周期内运行时间为 50ms，输入输出的时间（即等待时间）为 100 ms。试计算单道、两道和三道程序运行时，CPU 的利用率。

解　单道程序运行时，CPU 的利用率为 50/150 = 1/3。两道程序运行时，在一个周期内，运行时间达到 100 ms，因此 CPU 利用率为 100/150 = 2/3。三道程序运行时，在一个周期内，运行时间达到 150 ms，因此 CPU 利用率达到 100%。

从上例可见，多道程序下，处理器的等待时间大大减小，利用率得到提高，单位时间内处理作业的个数也增多了。还可以看出，要使处理器的利用率达到最高，需要考虑多个程序的运行和等待时间，并对这些程序的执行次序进行精心的安排，这就涉及调度问题。

一般的，给定一个作业集合 J 以及资源约束条件 C。如果 S 是对 J 中作业执行的一个编排，并且满足资源约束 C，那么称 S 为作业集合 J 上的一个调度(Schedule)。

对同一作业集合，可以有多个调度。比如，对程序 A、B 和 C，可以进行串行调度，让处理器顺序执行这三个程序，也可以用图 1-5 所示的方式进行并发调度。显然，不同调度的处理性能是不同的，因此对于给定的作业集合 J 和资源约束 C，有必要寻找一个最优调度。调度通常用调度算法来实现，它是操作系统内核中的核心部件之一。

【例 1-3】某系统供用户使用的内存空间为 100 KB，系统配有 4 台磁带机。一批作业的运行和资源需求信息如下所示。试给出对这个作业集合的一种调度。

作业	进入时间	估计运行时间/min	内存需求/KB	磁带机需求/台
J_1	10:00	25	15	2
J_2	10:20	30	60	1
J_3	10:30	10	50	3
J_4	10:35	20	10	2
J_5	10:40	15	30	2

解　作业集合 $J = \{J_1, J_2, J_3, J_4, J_5\}$

约束 C：$M(J_i) + M(J_j) + \cdots + M(J_k) \leqslant 100K$

$P(J_i) + P(J_j) + \cdots + P(J_k) \leqslant 4$

其中，J_i，J_j，…，J_k 是当前内存中的作业，$M(J_i)$和 $P(J_i)$分别表示 J_i 所占的内存和打印机资源。

图 1-6 给出了一个可能的调度。

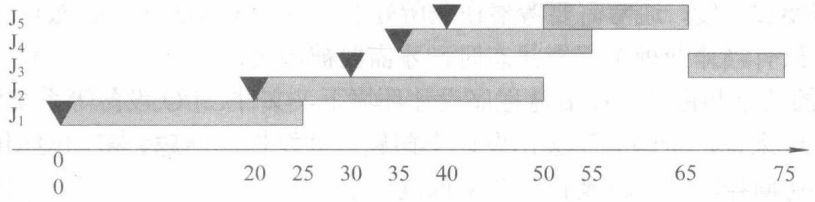


图 1-6　一个可能的调度

开始时，系统中没有任何作业在运行，当 J_1 就绪时，由于它满足约束条件，因此调度器允许其进入计算机系统，这时剩余的内存和打印机资源为<85KB, 2>；当运行到 10:20 时，J_2 就绪请求进入，这时剩余的资源仍然允许它进入，这样 J_1 和 J_2 就开始并发运行，这时剩余的资源为<25KB, 1>；当运行到 10:25 时，J_1 已经结束，因此它释放占用的内存和打印机资源，这时剩余资源为<40KB, 3>；在 10:30 时，J_3 就绪请求进入，但是由于其内存需求得不到满足，因此不能进入系统；按照这样的方式依次编排剩余作业的运行次序，就得到了一个可行的调度。图 1-6 给出的这个可行调度是 $J_1 \rightarrow J_2 \rightarrow J_4 \rightarrow J_5 \rightarrow J_3$。

值得注意的是：可行的调度可能不止一种，图 1-6 仅给出了其中的一种，可能还存在其他可行调度。另外，当多个作业被载入内存并发执行时，由于处理器只有一个，因此还需要按照特定的调度策略，在并发执行的多个作业之间实施调度。比如，在图 1-6 例子中，10:20～10:25 期间，J_1 和 J_2 并发运行，调度器可以采用分时调度策略来调度 J_1 和 J_2 的执行，从宏观的角度来看，它们好像是同时运行的。本例忽略了并发调度对作业执行时间的影响。实际上，由于并发调度的影响，作业的实际运行时间会大于估计运行时间。

总之，采用多道程序设计后，减小了处理器时间的浪费。对计算型作业，由于 I/O 操作较少，处理器浪费的时间很少；而对于 I/O 型作业，例如商业数据处理，I/O 时间通常占到 80%～90%，采用多道程序设计效果明显。如果在主存中存放足够多的作业，可使 CPU 的利用率接近 100%。

1.2.4　多道程序设计的实现

实现多道程序设计的核心是妥善解决计算资源的共享和保护问题。具体来说，必须首先解决以下三个问题：

(1) 存储保护和地址重定位。在多道程序设计环境下，内存中的多道程序与操作系统一起共享同一内存空间，因此必须提供必要的保护手段，防止应用程序对操作系统所占内存区域的侵犯，以及各道程序之间的相互侵犯，特别是当某道程序出错时，不致影响其他程序。

在多道程序运行的环境中，原有程序不断结束退出，新程序不断进入，内存占用状态不断发生变化，因此一个程序员事先无法预知他所写的程序在内存中的确切位置，这就要求程序指令和数据的内存地址延迟到装载时，甚至运行时才被确定下来，而且不能影响程序的执行结果。这就是程序的“地址重定位”问题。这一问题的解决，需要从编译器、加载器和内存管理等多个方面来着手。

(2) 处理机管理和调度。多道程序共享同一个处理器，于是就存在处理器分配问题(或作业调度问题)。为了更好地进行作业调度，需要解决可调度作业集和调度策略这样两个问题。所谓可调度作业集，是指在作业集合中划分出需要被调度的作业子集，以避免不必要的调度开销；调度策略是指当要调度多个作业时，如何选择一个优化调度的问题。认定一