# 第八章 操作系统支持

8.1操作系统概述

8.1.1操作系统的目标和功能

8.1.2操作系统的类型

8.2调度

8.2.1长期调度

8.2.2中期调度

8.2.3短期调度

8.3内存管理

8.3.1交换

8.3.2分区

8.3.3分页

8.3.4虚拟存储器

8.3.5 快表（转译后备缓冲器，TLB）

分段

8.4英特尔x86内存管理

地址空间

分段

分页

8.5臂存储器管理

存储系统组织

虚拟内存地址转换

内存管理格式

访问控制

8.6个关键术语、复习问题和问题

299

M08SARS5658Y10GEYC08IDED 299 下午3:19/15

300章8 /操作系统支持

学习目标

学习本章后，你应该能够：

在顶层概述操作系统（OS）的关键功能。

讨论早期简单批处理系统到现代复杂系统的操作系统的演变。

解释长期、中期和短期计划之间的差异。

了解内存分区的原因，并解释所使用的各种技术。

评估分页和分割的相对优势。定义虚拟内存。

虽然本文的重点是计算机硬件，但有一个软件领域需要解决：计算机的操作系统。操作系统是管理计算机资源、为程序员提供服务和调度其他程序执行的程序。了解操作系统对于理解CPU控制计算机系统的机制至关重要。一般来说，中断的影响和记忆层级的管理的解释在此上下文中得到了最好的解释。

本章从操作系统的概述和简要历史开始。本章的大部分内容着眼于与计算机组织和体系结构研究最相关的两个OS功能：调度和内存管理。

8.1操作系统概述

操作系统的目标和功能

操作系统是一个控制应用程序执行的程序，它充当应用程序和计算机硬件之间的接口。可以认为它有两个目标：

一个操作系统使计算机更方便使用。方便：

■操作系统允许以有效的方式使用计算机系统资源。效率：

让我们依次研究OS的这两个方面。

作为用户的操作系统用于向用户提供应用程序的硬件和软件可以分层或分层的方式查看，如图8.1所示。这些应用程序的用户，即最终用户，通常不关心计算机的体系结构。因此，最终用户根据应用程序查看计算机系统。该应用程序可以用编程语言表示，并由应用程序员开发。将应用程序开发为一组处理器指令

M08SARS5658Y10GEYC08IDED 300 下午3:19/15

8.1 /操作系统概述301

应用 应用程序

编程接口 图书馆/公用事业

应用 软件

二进制接口 操作系统

指令集

建筑学

执行硬件

记忆

系统互连

翻译 硬件

（公共汽车）

输入输出设备 主要的

和

记忆

网络

图8.1计算机软硬件结构

完全负责控制计算机硬件是一项极其复杂的任务。为了简化这一任务，提供了一套系统程序。这些程序中的一些被称为实用程序。这些实现经常使用的功能，这些功能有助于程序创建、文件管理和I/O设备的控制。程序员在开发应用程序时使用这些工具，应用程序在运行时调用实用程序来执行某些功能。最重要的系统程序是操作系统。操作系统从程序员那里屏蔽了硬件的细节，并为程序员提供了使用系统的方便接口。它充当中介者，使程序员和应用程序更容易访问和使用这些设施和服务。

简言之，操作系统通常提供以下领域的服务：

■操作系统提供各种工具和服务，如编辑器和调试器，以帮助程序员创建程序。通常，这些服务是以实用程序的形式提供的，这些实用程序实际上不是OS的一部分，但是可以通过OS访问。程序创建：

■执行程序需要执行许多步骤。指令和数据必须加载到主存储器中，I/O设备和文件必须初始化，并且必须准备其他资源。操作系统为用户处理所有这些操作。程序执行：

■每个I/O设备需要它自己的一组特定的仪器或控制信号来操作。操作系统负责处理细节，以便程序员能够根据简单的读写进行思考。访问I/O设备：

■对于文件，控制必须包括不仅了解I/O设备（磁盘驱动器、磁带驱动器）的性质，而且了解存储介质上的文件格式。再次，操作系统担心细节。此外，在具有多个同时用户的系统的情况下，OS可以提供保护机制来控制对文件的访问。对文件的受控访问：

M08SARS5658Y10GEYC08IDED 301 下午3:19/15

302章8 /操作系统支持

■在共享或公共系统的情况下，操作系统控制对系统整体和特定系统资源的访问。访问功能必须对资源和数据提供保护，使其免受未授权用户的攻击，并且必须解决资源争用的冲突。系统访问：

■■在计算机系统运行时，可能发生各种错误。这些错误包括内部和外部硬件错误，例如内存错误或设备故障或故障；以及各种软件错误，例如算术溢出、试图访问禁止的存储器位置，以及操作系统不能准许应用程序的请求。在每种情况下，OS都必须做出清除错误条件的响应，对正在运行的应用程序影响最小。响应的范围可以从结束导致错误的程序、重试操作到简单地将错误报告给应用程序。错误检测和响应：

■一个好的操作系统收集各种资源的使用统计信息，并监视性能参数，如响应时间。在任何系统上，这些信息在预测未来增强的需求和调整系统以提高性能方面都是有用的。在多用户系统中，信息可以用于计费目的。图8.1还表示典型计算机系统中的三个关键接口：会计：

■ISA定义了计算机可以遵循的机器语言指令集。这个接口是硬件和软件之间的边界。请注意，应用程序和实用程序都可以直接访问ISA。对于这些程序，指令指令集的子集可用（用户ISA）。操作系统可以访问处理管理系统资源（系统ISA）的其他机器语言指令。指令集体系结构（ISA）：

■ABI定义了跨程序的二进制可移植性的标准。ABI定义了操作系统的系统调用接口和通过用户ISA在系统中可用的硬件资源和服务。应用二进制接口（ABI）：

■该API通过用户ISA加上高级语言（HLL）库调用，使程序访问系统中可用的硬件资源和服务。任何系统调用通常是通过库执行的。使用API使得应用程序软件能够通过重新编译轻松地移植到支持相同API的其他系统。应用程序编程接口（API）：

作为资源管理器的操作系统计算机是用于移动、存储和处理数据以及控制这些功能的一组资源。操作系统负责管理这些资源。

我们能说操作系统控制数据的移动、存储和处理吗？从一个角度来看，答案是肯定的：通过管理计算机的资源，操作系统控制着计算机的基本功能。但这种控制是以一种奇怪的方式进行的。通常，我们认为控制机制是被控制的事物之外的东西，或者至少是被控制的事物的独立和单独的部分。（例如，住宅供暖系统）

M08SARS5658Y10GEYC08IDED 302 3/19 / 15下午3点47分

8.1 /操作系统概述303

由温控器控制，温控器完全不同于发热和热分配装置。)OS的情况并非如此，它作为控制机制在两个方面是不寻常的：

■操作系统的功能与普通计算机软件相同，即它是由处理器执行的程序。

■操作系统经常放弃控制，并且必须依赖处理器才能重新获得控制。

和其他计算机程序一样，操作系统为进程sor提供指令。关键差异在于程序的意图。操作系统指导处理器使用其他系统资源和执行其他程序的定时。但是，为了让处理器完成这些事情，它必须停止执行OS程序并执行其他程序。因此，操作系统放弃控制让处理器做一些“有用”的工作，然后恢复控制足够长的时间，使处理器做好准备进行下一项工作。随着本章的进行，涉及所有这些的机制应该变得清晰。

图8.2显示了OS管理的主要资源。OS的一部分在主存储器中。这包括内核或内核，内核包含操作系统中最常使用的函数，以及在给定时间，当前使用的操作系统的其他部分。主存储器的其余部分包含用户程序和数据。正如我们将看到的，这个资源（主内存）的分配由处理器中的操作系统和内存管理硬件共同控制。操作系统决定程序何时可以使用I/O设备，并控制对

计算机系统

记忆 输入输出设备

操作 输入输出控制器 打印机

系统 键盘，

数码相机，

软件 输入输出控制器 等。

程序 • •

• •

数据 • •

输入输出控制器

处理器 •••• 处理器

保管部

操作系统

程序

数据

图8.2作为资源管理器的操作系统

M08SARS5658Y10GEYC08IDED 303 下午3:19/15

304章8 /操作系统支持

文件的使用。处理器本身是一种资源，操作系统必须确定要花费多少处理器时间来执行特定的用户程序。在多处理器系统的情况下，这个决策必须跨越所有处理器。

操作系统的类型

某些关键特性用于区分不同类型的操作系统。这些特征沿着两个独立的维度下降。第一个维度指定系统是批处理还是交互。在交互系统中，用户/程序员通常通过键盘/显示终端直接与计算机交互，以请求执行作业或执行事务。此外，根据应用程序的性质，用户可以在执行作业期间与计算机通信。批处理系统是互操作的对立面。用户的程序与其他用户的程序一起批处理并由计算机操作员分送。在程序完成后，将结果打印给用户。纯批处理系统在今天是罕见的，然而，简要地研究批处理系统对于描述当代操作系统是有用的。

独立的维度指定系统是否采用多重编程。使用多道程序设计，试图通过让处理器一次处理多个程序来使处理器尽可能繁忙。几个程序被加载到内存中，处理器在它们之间快速切换。另一种选择是单编程系统，它一次只运行一个程序。

早期系统使用最早的计算机，从20世纪40年代末到中期

20世纪50年代，程序员直接与计算机硬件交互，没有操作系统。这些处理器从控制台运行，控制台由显示灯、切换开关、某种形式的输入设备和打印机组成。处理器代码中的程序通过输入设备（例如，读卡器）加载。如果一个错误使程序停止，则错误条件由灯指示。程序员可以继续检查寄存器和主存储器以确定错误的原因。如果程序继续正常完成，则输出出现在打印机上。

这些早期系统提出了两个主要问题：

■大多数安装使用注册表来预约处理器时间。通常，用户可以以以大约半个小时的倍数注册一段时间。用户可能注册一个小时并在45分钟内完成；这将导致浪费计算机空闲时间。另一方面，用户可能会遇到问题，没有在指定的时间内完成，并且在解决问题之前被迫停止。行程安排：

■一个单独的程序，称为作业，可以包括将编译器和高级语言程序（源程序）加载到内存中，保存已编译的程序（目标程序），然后将目标程序和公共函数加载和连接在一起。这些步骤中的每一个都可能涉及安装或拆卸磁带，或设置卡片。如果发生错误，那么不幸的用户通常必须回到设置序列的开始。因此，在设置要运行的程序上花费了大量的时间。安装时间：

M08SARS5658Y10GEYC08IDED 304 3/19 / 15下午3点47分

8.1 /操作系统概述305

这种操作模式可以称为串行处理，反映了用户能够连续访问计算机的事实。随着时间的推移，各种系统软件工具被开发出来，试图使串行处理更有效。这些库包括通用函数、链接器、加载器、调试器和I/O驱动程序例程，这些程序可以作为所有用户的通用软件使用。

简单批处理系统早期的处理器非常昂贵，因此最大化处理器利用率非常重要。由于调度和设置时间而浪费的时间是不可接受的。

为了提高利用率，开发了简单的批量操作系统。有了这样的系统，也称为监视器，用户就不能直接访问进程了。而是，用户将卡片或磁带上的作业提交给计算机操作员，计算机操作员顺序地将作业批处理在一起，并将整个批处理放置在输入设备上，供监视器使用。

为了理解这个方案是如何工作的，让我们从两个角度来看待它：监视器和处理器。从监视器的角度来看，监视器控制事件的顺序。为此，大部分监视器必须始终位于主内存中，并且可用于执行（图8.3）。该部分被称为驻留监视器。监视器的其余部分由实用程序和公共函数组成，这些实用程序和公共函数在需要它们的任何作业开始时作为子例程加载到用户程序上。监视器一次从输入设备（通常是读卡器或磁带驱动器）读取一个作业。当读取时，将当前作业放置在用户程序区域中，并将控制传递给该作业。当作业完成时，它将控制返回给监视器，监视器立即读取下一个作业。每个作业的结果都打印出来，以便交付给用户。

中断

处理

监测器

装置

驱动程序

边界

工作

测序

控制语言

口译译员

用户

程序

地区

图8.3内存布局

驻留监视器

M08SARS5658Y10GEYC08IDED 305 下午3:19/15

306章8 /操作系统支持

现在从处理器的角度考虑这个序列。在某个时间点，处理器正在执行来自包含监视器的主存储器部分的指令。这些指令导致下一个作业被读入主存储器的另一部分。一旦读入了作业，处理器将在监视器中遇到一条分支指令，该指令指示处理器在用户程序开始时继续执行。然后，处理器将在用户的程序中执行指令，直到它遇到结束或错误条件。任一事件都导致处理器从监视程序获取其下一条指令。因此，短语“控制被传递给作业”仅仅意味着处理器现在在用户程序中获取和执行指令，而“控制被返回到监视器”意味着处理器现在从监视器程序获取和执行指令。

应该清楚的是，监视器处理调度问题。一批作业被排队，作业被尽可能快地执行，没有中间的空闲时间。

工作安排时间如何？监视器也处理这个问题。对于每个作业，指令都包含在作业控制语言（JCL）中。这是一种特殊的编程语言，用于向监视器提供指令。一个简单的例子是用户提交用FORTRAN编写的程序加上一些要被程序使用的数据。每个FORTRAN指令和每个数据项都在单独的穿孔卡片或磁带上的单独记录上。除了.-TRAN和数据行之外，作业还包括作业控制指令，这些指令由开头“$”表示。工作的整体形式看起来是这样的：

美元的工作 6

$FTN

F FORTRAN指令

美元负荷 6

美元运行 数据

F

美元的末端

为了执行此作业，监视器读取$FTN行，并从其大容量存储器（通常是磁带）加载适当的编译器。编译器将用户的程序翻译成对象代码，存储在内存或大容量存储器中。如果该操作存储在内存中，则称为“编译、加载和转到”。如果该操作存储在磁带上，则需要$LOAD指令。该指令由监视器读取，该监视器在编译操作之后重新获得控制。监视器调用加载器，该加载器将目标程序加载到内存中，而不是编译器，并将控制传递给它。以这种方式，可以在不同子系统之间共享大段主存储器，尽管一次只能驻留和执行一个这样的子系统。

我们看到，监视器或批处理OS，只是一个计算机程序。它依赖于处理器从main的各个部分获取指令的能力。

M08SARS5658Y10GEYC08IDED 306 3/19 / 15下午3点47分

8.1 /操作系统概述307

存储器，以便交替地夺取和放弃控制。还需要某些其他硬件特性：

■当用户程序执行时，它必须不改变包含监视器的存储区域。如果进行了这样的尝试，则处理器硬件应该检测错误并将控制传送到监视器。然后，监视器将中止作业，打印出错误消息，并加载下一个作业。内存保护：

■定时器用于防止单个作业垄断系统。计时器设置在每个作业的开头。如果定时器到期，则发生中断，并且控制返回到监视器。Timer：

某些指令被指定为特权，只能由监视器执行。如果处理器在执行用户程序时遇到这样的指令，则会发生错误中断。在特权指令中有I/O指令，以便监视器保持对所有I/O设备的控制。例如，这防止了用户程序从ACCI上读取来自下一个作业的作业控制指令。如果用户程序希望执行I/O，则它必须请求监视器对其执行操作。如果处理器在执行用户程序时遇到了一个特权指令，处理器硬件认为这是一个错误，并将控制权传递给监视器。特权指令：

早期的计算机模型没有这种能力。这个特性使操作系统在放弃控制和从用户程序中恢复控制方面更具灵活性。中断：

处理器时间在用户程序的执行和监视器的执行之间交替。已经有两个牺牲：一些主内存现在被移交给监视器，并且一些处理器时间被监视器消耗掉。这两者都是开销的形式。即使有这样的开销，简单的批处理系统也提高了计算机的利用率。

多程序批处理系统即使由简单批处理OS提供的自动作业排序，处理器也常常是空闲的。问题在于，与处理器相比，I/O设备速度较慢。图8.4详细描述了一个有代表性的计算。计算涉及一个程序，该程序处理记录文件，并且平均每条记录执行100个处理器指令。在这个例子中，计算机花费超过96%的时间等待I/O设备完成数据传输！图85A说明了这种情况。处理器消耗一定量的

从LE读取一条记录 15

执行100条指令 1

把一条记录写下来 15

合计 31

CPU利用率311 0.032 3.2%= =

图8.4系统利用实例

M08SARS5658Y10GEYC08IDED 307 下午3:19/15

308章8 /操作系统支持

程序A

程序A

程序B

组合的

程序A

程序B

程序C

组合的

跑 等待 跑 等待

时间

（a）单程序设计

跑 等待 跑 等待

等待 跑 等待 跑 等待

跑 跑 等待 跑 跑 等待

一 B 一 B

时间

（b）两个程序的多道程序设计

跑 等待 跑 等待

等待 跑 等待 跑 等待

等待 跑 等待 跑 等待

跑 跑 跑 等待 跑 跑 跑 等待

一 B C 一 B C

时间

（c）用三个程序进行多道程序设计

图8.5多道程序设计实例

执行时间，直到达到I/O指令为止。然后，它必须等待直到I/O指令结束之后才能继续。

这种低效率是不必要的。我们知道必须有足够的内存来保存OS（常驻监视器）和一个用户程序。假设操作系统和两个用户程序还有空间。现在，当一个作业需要等待I/O时，进程可以切换到另一个作业，该作业可能不等待I/O（图8.5b）。此外，我们可能会扩展内存来容纳三个、四个或更多的程序，并在它们之间切换（图8.5c）。这种技术被称为多程序设计（multi.-ming）或多任务处理（multitasking），是现代操作系统的中心主题。1

1术语多任务处理有时被保留为表示同一程序内的多个任务，这些任务可以由操作系统同时处理，而多重编程指的是来自多个程序的多个进程。然而，将术语多任务和多编程等同于大多数标准字典（例如，IEEE Std 100-1992，电气和电子术语新IEEE标准字典）是更常见的。

M08SARS5658Y10GEYC08IDED 308 下午3:19/15

8.1 /操作系统概述309

例8.1这个例子说明了多道程序设计的好处。考虑具有250M字节可用内存（操作系统不使用）、磁盘、终端和打印机的计算机。同时提交三个程序JOB1、JOB2和JOB3以供执行，属性如表8.1所示。我们假设JOB1和JOB2以及JOB3连续使用磁盘和打印机的处理器需求最小。对于简单的批处理环境，这些作业将按顺序执行。因此，JOB1在5分钟内完成。作业2必须等到5分钟结束，然后15分钟后完成。作业3在20分钟后开始，在最初提交时30分钟后完成。平均资源利用率、吞吐量和响应时间显示在表8.2的uniprogramming列中。图8.6a显示了每个设备的利用率。很显然，当在需要的30分钟时间段内进行平均时，所有资源的利用率都严重不足。

现在假设作业在多道程序设计OS下并发运行。因为作业之间几乎没有资源争用，所以这三个作业可以在几乎最小时间内运行，同时与计算机中的其他作业共存（假设为JOB2和JOB3分配了足够的处理器时间来保持它们的输入和输出操作是活动的）。JOB1仍然需要5分钟才能完成，但是到那个时间结束时，JOB2将完成三分之一，而JOB3将完成一半。所有三个工作将在15分钟内完成。当检查从图8.6b所示的直方图获得的表8.2的多道程序设计列时，这种改进是明显的。

与简单的批处理系统一样，多道程序设计批处理系统必须依赖于某些计算机硬件特性。对多道编程有用的最显著的附加特性是支持I/O中断的硬件

表8.1示例程序执行属性

JOB1 作业2 作业3

工作类型 重计算 重输入输出 重输入输出

持续时间（分钟） 5 15 10

所需内存（m） 50 100 80

需要磁盘吗？ 不 不 是的

需要终端吗？ 不 是的 不

需要打印机吗？ 不 不 是的

表8.2多道程序设计对资源利用的影响

单程序设计 多道程序设计

处理器使用率（%） 20 40

内存使用（%） 33 67

磁盘使用率（%） 33 67

打印机使用率（%） 33 67

经过时间（分钟） 30 15

吞吐率（作业/人力资源） 6 12

平均响应时间（min） 18 10

M08SARS5658Y10GEYC08IDED 309 下午3:19/15

310章8 /操作系统支持

100% 100%

中央处理器 中央处理器

0% 0%

100% 100%

记忆 记忆

0% 0%

100% 100%

磁盘 磁盘

0% 0%

100% 100%

终点站 终点站

0% 0%

100% 100%

打印机 打印机

0% 0%

工作历史 JOB1 作业2 作业3 工作历史 JOB1

作业2

0 5 10 15 20 25 30

作业3

分钟

0 5 10 15

时间

分钟 时间

（a）单程序设计 （b）多道程序设计

图8.6利用直方图

和DMA。通过中断驱动的I/O或DMA，处理器可以对一个作业发出I/O命令，并在I/O被设备控制器从车中取出时继续执行另一个作业。当I/O操作完成时，处理器被中断，控制被传递给操作系统中的中断处理程序。操作系统然后将控制传递给另一个作业。

与单程序或单程序系统相比，多程序操作系统是相当复杂的。为了准备运行多个作业，必须将作业保存在主存储器中，这需要某种形式的内存管理。此外，如果准备运行多个作业，则处理器必须决定运行哪个作业，这需要一些调度算法。本章后面将讨论这些概念。

使用多道程序设计，批处理可以非常有效。然而，对于许多作业，希望提供一种用户直接与计算机交互的模式。的确，对于一些作业，例如事务处理，交互模式是必不可少的。

今天，对交互式计算设备的要求可以通过，并且经常通过使用专用的微型计算机来满足。在20世纪60年代，当大多数计算机都是大而昂贵的时候，这个选项是不可用的。取而代之的是时间共享。

正如多道程序设计允许处理器一次处理多个批处理作业一样，多道程序设计也可以用于处理多个交互式作业。在后一种情况下，该技术称为分时，因为proces sor的时间在多个用户之间共享。在分时系统中，多个用户

M08SARS5658Y10GEYC08IDED 310 下午3:19/15

8.2 /调度311

表8.3批处理与时间共享

批多道程序设计 分时

主要目标 最大化处理器使用 最小化响应时间

指令源 作业控制语言命令 命令输入

操作系统 提供工作 终端

同时通过终端访问系统，OS在短时间内或计算量内交错每个用户程序的执行。因此，如果同时有n个用户主动请求服务，则每个用户将只看到平均1/n的有效计算机速度，而不计算OS开销。然而，给定相对缓慢的人类反应时间，在适当设计的系统上的响应时间应该与专用计算机上的响应时间相当。

批量多道程序设计和时间共享都使用多道程序。表8.3列出了关键差异。

8.2调度

多道程序设计的关键是调度问题。事实上，通常涉及四种类型的调度（表8.4）。我们现在将探讨这些。但首先，我们介绍了过程的概念。这个术语最早是在20世纪60年代由Multics OS的设计者使用的。这个术语比job更通用。已经为术语过程给出了许多定义，包括

执行中的程序

一个程序的“动画精神”

一个处理器分配给的实体

随着我们的进行，这个概念应该变得更加清晰。

长期调度

长期调度程序确定哪些程序被允许进入系统进行处理。因此，它控制多重编程的程度（内存中的进程数）。一旦被接纳，作业或用户程序就变成一个过程，并被添加到短期调度器的队列中。在某些系统中，新创建的过程以交换状态开始，在这种情况下，它被添加到中期调度器的队列中。

表8.4调度类型

长期调度 向要执行的进程池中添加的决定。

中期调度 增加部分或

完全在主内存中。

短期调度 关于哪些可用进程将由

处理器。

输入输出调度 关于哪一个程序未决I/O请求的决定是韩-

通过可用的I/O设备进行DLED。

M08SARS5658Y10GEYC08IDED 311 下午3:19/15

312章8 /操作系统支持

在批处理系统中，或者对于通用操作系统的批处理部分，新提交的作业被路由到磁盘并被保存在批处理队列中。长期调度器可以在队列中创建进程。这里有两个决定。首先，调度器必须决定OS可以承担一个或多个附加进程。第二，调度程序必须决定接受哪个作业或作业，并将其转换为进程。所使用的标准可以包括优先级、预期执行时间和I/O要求。

对于分时系统中的交互式程序，当用户试图连接到系统时，生成进程请求。分时用户并不只是排队等待系统接受他们。相反，OS将接受所有授权的角落，直到系统饱和，使用一些预先定义的饱和度量。此时，连接请求将收到一条消息，该消息指示系统已满，用户应该稍后再试一次。

中期调度

中期调度是交换功能的一部分，如8.3节所述。通常，交换决策是基于管理多重编程程度的需要。在不使用虚拟内存、内存人的系统上

管理也是一个问题。因此，交换入决策将考虑交换出过程的内存需求。

短期调度

长期调度程序执行相对不频繁，并且使

是否采用新流程，以及采用哪个流程的详细决策。短期调度器（也称为调度器）执行频繁，并对接下来要执行的作业做出细粒度的决定。

进程状态为了理解短期调度器的操作，我们需要考虑过程状态的概念。在过程的生命周期中，它的状态将改变很多次。它在任何时间点的状态被称为状态。之所以使用“状态”这个术语，是因为它意味着存在某些信息，这些信息定义了此时的状态。至少，流程有五个已定义的状态（图8.7）：

■一个程序被高级调度程序接受，但是还没有准备好执行。操作系统将初始化进程，将其移动到就绪状态。新的：

承认 派遣 释放

新的 准备好 运行 出口

超时

事件

发生的

等待

此路不通

图8.7 五态过程模型

M08SARS5658Y10GEYC08IDED 312 下午3:19/15

8.2 /调度313

■该进程准备执行，并且正在等待对处理器的访问。准备好：

这个过程是由处理器执行的。运行：

■进程暂停执行，等待某些系统资源，如I/O。等待：

■进程已经结束，将被操作系统破坏。停止：

对于系统中的每个进程，操作系统必须维护指示进程状态的信息以及进程执行所需的其他信息。为此，每个进程在OS中由一个进程控制块（图8.8）表示，它通常包含：

每个当前进程都有唯一的标识符。标识符：

■过程的当前状态（新的、就绪等）。状态：

相对优先级。优先：

■要执行的程序中下一个指令的地址。程序计数器：

■进程在内存中的开始和结束位置。内存指针：

■这些是在进程执行时存在于处理器中的寄存器中的数据，它们将在第三部分中讨论。现在，只要说这些数据表示流程的“上下文”就足够了。当进程离开运行状态时，将保存上下文数据和程序计数器。它们由处理器在恢复执行进程时检索。上下文数据：

识别器

状态

优先

程序计数器

内存指针

上下文数据

输入输出状态

信息

会计核算

信息

•

•

•

图8.8过程控制块

M08SARS5658Y10GEYC08IDED 313 下午3:19/15