第5章 操作系统

操作系统是计算机系统中最重要的软件，它对计算机系统的软硬件资源进行管理、协调，并代表计算机与外界进行通信，正是有了操作系统才使得计算机硬件系统真正可用。本章将重点阐述操作系统是如何完成上述功能的。首先介绍什么是操作系统及操作系统的简要发展历程，使读者对操作系统中的基本概念、技术和发展有直观、形象的了解，然后从资源管理的视角介绍计算机操作系统的基本功能以及相关技术。

5.1 操作系统概述

现代计算机是一个复杂的硬件系统，如果需要直接面对计算机的硬件设备，那对用户会是一个巨大的挑战，用户可能需要花费相当大的精力去学习如何调用打印机的接口打印文件、如何从内存中的某一块读取数据等一些底层的操作。如果现实是如此，计算机也不会像今天这样普及。另外，管理所有这些设备并使之协调工作也是一个巨大挑战。于是几乎所有的计算机都安装了一种称为操作系统的软件，它最主要的任务就是为用户提供一个方便、简单、清晰的计算机系统使用环境。

在现代计算机系统中，操作系统是计算机系统中最基本的系统软件，是整个计算机系统的控制中心。操作系统通过管理计算机系统的软硬件资源，为用户提供使用计算机系统的良好环境，并且采用合理有效的方法组织多个用户程序共享各种计算机系统资源，最大限度地提高系统资源的利用率。操作系统不仅可以为应用程序提供运行基础，它还管理着计算机硬件，充当计算机硬件和计算机用户的中介。操作系统完成这些任务的方式也是多样化的。大型操作系统设计偏重于充分优化硬件的利用率，个人计算机操作系统主要是为了能够支持商业应用、程序设计、游戏娱乐等方便快捷的运行和使用，智能手机、平板计算机等设备的操作系统则更倾向于为用户提供方便的、交互式执行程序的环境。

5.1.1 操作系统发展简史

从1946年第一台电子计算机诞生到20世纪50年代中期的计算机，都属于第一代计算机。此时的计算机操作是由用户(即程序员)采用人工操作方式直接使用计算机硬件系统，即由程序员将事先已穿孔(对应于程序和数据)的纸带或卡片装入纸带输入机或卡片输入机，再启动它们将程序和数据输入计算机，然后启动计算机运行。当程序运行完毕并

取走计算结果之后，才让下一个用户上机。人工操作方式严重降低了计算机资源的利用率。随着技术的进步，CPU 的运行速度迅速提高，但输入输出设备的运行速度却提高缓慢，这使得 CPU 与输入输出设备之间速度不匹配的矛盾更加突出。为了缓和矛盾，曾先后出现了通道技术、缓冲技术，但都未能很好地解决上述矛盾，直至后来引入了脱机输入输出技术，才获得了令人较为满意的结果。

为了解决人机矛盾及 CPU 和输入输出设备之间速度不匹配的问题，20 世纪 50 年代末出现了脱机输入输出技术。该技术是事先将装有用户程序和数据的纸带或卡片装入纸带输入机或卡片机，在一台外围机的控制下，把纸带或卡片上的数据和程序输入到磁带上。当 CPU 需要这些程序和数据时，再从磁带或卡片上将其高速地调入内存。

20 世纪 50 年代中期发明了晶体管，人们开始用晶体管替代电子管来制造计算机，从而出现了第二代计算机。为了能充分地利用第二代计算机，应尽量让系统连续运行，以减少空闲时间。为此，通常是把一批任务以脱机方式输入到磁带上，并在系统中配上监督程序(Monitor)，在它的控制下使这批任务能一个接一个地连续处理。其自动处理过程是：首先由监督程序将磁带上的第一个任务装入内存，并把运行控制权交给该任务；当该任务处理完成时，又把控制权交还给监督程序，再由监督程序把磁带或磁盘上的第二个任务调入内存。计算机系统就这样对任务逐个进行处理，直至磁带或磁盘上的所有任务全部完成，这样便形成了早期的批处理系统。由于系统对任务的处理都是成批地进行的，并且在内存中始终只保持一个任务，故称此系统为单道批处理系统或简单批处理系统(Simple Batch Processing System)。

在单道批处理系统中，内存中仅有一个任务，无法充分利用系统中的所有资源，致使系统整体性能较差。为了进一步提高资源的利用率和系统吞吐量，在 20 世纪 60 年代中期又引入了多道程序设计技术用于第三代计算机中，由此形成了多道批处理系统(Multiprogrammed Batch Processing System)。在该系统中，用户所提交的任务都先存放在外存上并排成一个队列，称为“后备队列”；然后，由任务调度程序按一定的算法从后备队列中选择若干个任务调入内存，使它们共享 CPU 和系统中的各种资源。图 5-1 给出了多道程序工作示例，当程序 A 需要进行输入操作时，CPU 将会加载程序 B 运行，当程序 B 也需要进行输入操作，并且该操作需要很长时间时，程序 A 将在其输入操作结束后被加载运行。

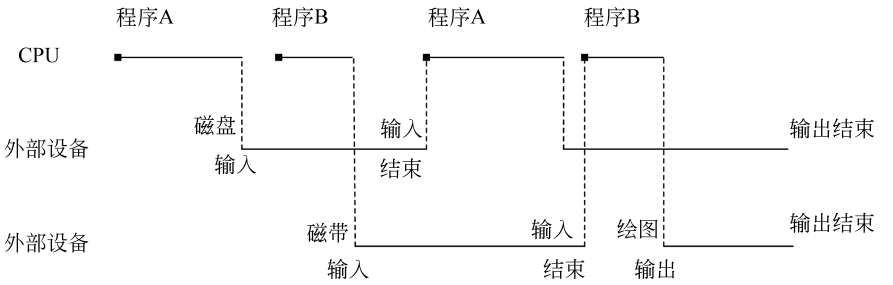

图 5-1　多道程序工作示例

分时系统(Time Sharing System)是指在一台主机上连接了多个带有显示器和键盘

的终端，同时允许多个用户通过自己的终端以交互方式使用计算机，共享主机中的资源。它能很好地将一台计算机提供给多个用户同时使用，提高计算机的利用率。它经常被应用于查询系统中，满足许多查询用户的需要。第一台真正的分时操作系统是由麻省理工学院开发的 CTSS(Compatible Time Sharing System)。继 CTSS 成功后，麻省理工学院又和贝尔实验室、通用电气公司联合开发出多用户多任务操作系统——MULTICS (Multiplexed Information and Computing System)，该机器能支持数百用户。值得一提的是，参加 MULTICS 研制的贝尔实验室的 Ken Thompson，在 PDP-7 小型机上开发出一个简化的 MULTICS 版本，它就是当今广为流行的 UNIX 操作系统的前身。

实时系统(Real Time System)的主要功能是及时(或即时)响应外部事件的请求，在规定的时间内完成对该事件的处理，并控制所有实时任务协调一致地运行。虽然多道批处理系统和分时系统已能获得较为令人满意的资源利用率和响应时间，从而使计算机的应用范围日益扩大，但是它们仍然不能满足实时控制和实时信息处理等某些应用领域的需要。

以多道程序管理为基础的现代操作系统具有以下主要特征。

(1) 并发性：指两个或多个程序在同一时间段内运行。注意与并行的区别，并行指的是两个或多个程序在同一时刻同时运行。

(2) 共享性：指系统中的资源可供内存中多个并发执行的程序共同使用。

(3) 虚拟性：指通过某种技术，把一个物理实体变为若干逻辑上的对应物。物理实体是实的，即实际存在的；而逻辑上的对应物是虚的，仅是用户感觉上的东西。

(4) 异步性：在多道程序环境下，允许多个程序并发执行，但只有程序在获得所需的资源后才能执行。在单处理机环境下，由于系统中只有一个 CPU，因而每次只允许一个任务执行，其余任务只能等待。因此，系统中各个任务会因为等待和中断而不可预知它们的执行进度。

并发和共享是操作系统的两个最基本的特征。资源共享是以程序的并发执行为条件的；但若系统不能对资源共享实施有效管理，不能协调好多个程序对共享资源的访问，也必然影响到程序并发执行的程度，甚至根本无法并发执行。

操作系统已经从最早的简单程序，发展到今天的囊括分时多任务和计算机系统资源管理的复杂软件系统。它是一组控制和管理计算机系统和软件资源，合理地组织计算机工作流程，并为用户使用计算机提供方便的程序和数据的集合。在计算机系统中设置操作系统的目的在于提高计算机系统的效率，增强系统的处理能力，提高系统资源的利用率，方便用户使用计算机。操作系统的发展并没有停止，不断发展的计算机技术对其提出了新的要求。例如，出现了多处理机操作系统、分布式操作系统等技术更加复杂的操作系统。

5.1.2 操作系统基础

操作系统的主要任务是为多道程序的运行提供良好的运行环境，以保证多道程序能有条不紊地、高效地运行，并能最大程度地提高系统中各种资源的利用率，方便用户的使

用。为实现上述任务，操作系统应具有 4 个方面的功能：进程管理、内存管理、设备管理和文件管理。为了方便用户使用操作系统，还需向用户提供方便的用户接口。因此，操作系统具有资源管理者和用户接口两重角色。

1. 资源管理者

计算机系统的资源包括硬件资源和软件资源。从管理角度看，计算机系统资源可分为四大类：CPU、存储器、输入输出设备和信息(通常是文件)。操作系统的目标是使整个计算机系统的资源得到充分有效的利用。为达到该目标，一般通过在相互竞争的程序之间合理有序地控制系统资源的分配，从而实现对计算机系统工作流程的控制。作为资源管理者，操作系统的主要工作是跟踪资源状态、分配资源、回收资源和保护资源。由此，可以把操作系统看成是由一组资源管理器(CPU 管理、内存管理、输入输出设备管理和文件管理)组成的软件系统。

在传统的多道程序系统中，CPU 的分配和运行都是以进程为基本单位的，因而对 CPU 的管理可归结为对进程的管理。进程管理的主要功能是创建和撤销进程，对多个进程的运行进行协调，实现进程之间的信息交换，以及按照一定的调度算法把 CPU 分配给进程。

内存管理的主要任务是为多道程序的运行提供良好的环境，方便用户使用内存，提高内存的利用率，以及能从逻辑上扩充内存。为此，内存管理应具有内存分配、内存保护、地址映射和内存扩充等功能。

设备管理用于管理计算机系统中所有的输入输出设备，设备管理的主要任务是：完成用户进程提出的输入输出请求；为用户进程分配其所需的输入输出设备；提高 CPU 和输入输出设备的利用率；提高输入输出速度；方便用户使用输入输出设备。为实现上述任务，设备管理应具有缓冲管理、设备分配、设备处理和虚拟设备等功能。

文件管理的主要任务是对用户文件和系统文件进行管理，以方便用户使用，并保证文件的安全性。为此，文件管理应具有对文件存储空间的管理、目录管理、文件的读写管理以及文件的共享与保护等功能。

2. 用户接口

对多数计算机而言，直接对输入输出设备操作和编程是相当困难的，需要一种抽象机制让用户在使用计算机时不涉及硬件细节。操作系统正是这样一种抽象，用户使用计算机时，都是通过操作系统进行的，不必了解计算机输入输出设备工作的细节。通过操作系统来使用计算机，操作系统就成为了用户和计算机之间的接口。该接口通常可分为人-机接口和程序接口两类。

(1) 人-机接口：是提供给用户使用的接口，用户可通过该接口取得操作系统的服务。

(2) 程序接口：是提供给程序员在编程时使用的接口，是用户程序取得操作系统服务的唯一途径。

5.2 进 程 管 理

5.2.1 进程与程序

在5.1.2节中提到对CPU的管理可归结为对进程的管理，为了弄清楚操作系统如何对CPU进行管理，首先需要弄清楚CPU是如何工作的。在3.2.2节中已经介绍了CPU的工作过程，可以用4个字概括这个过程，即“取指执行”：CPU从PC(程序计数器)所指向的内存地址取出指令，并存储在IR(指令寄存器)中，而后对指令进行解释和执行，同时PC自动加1。由于任何一个程序员所写的程序都只会使得PC在其内部来回移动(因为其编写程序时不可能知道在未来运行时会与哪个程序同时运行)，因此，可以想象在没有其他技术支持的情况下，当某一个程序在执行时，PC只会在这个程序内部来回移动。也就是说，一段时间之内只能有一个程序在运行。这与用户实际使用计算机的情况有很大区别。计算机用户习以为常的使用场景可能是一边用“迅雷”视频软件下载某个电影，一边用IE浏览器访问某个网页，同时还可能使用“酷狗音乐”播放某首歌曲，这是如何做到的呢？

为了解决这个问题，必须引入一种非常重要的技术手段——中断(Interrupt)。中断是在计算机发展过程中出现的一种技术，中断发生时的处理过程如下：首先由CPU外部的设备(可能是时钟发生器，也可能是一个I/O设备)产生一个中断信号，中断当前CPU正在执行的工作，将CPU内部一些寄存器的值(可统称为程序执行的上下文，Context)保存到内存中的某个位置，并将PC指向事先设定好的某个内存地址(一般位于操作系统所在的内存区域，记为OS_entry)，这样CPU在下一个指令周期执行时，便会从该处(OS_entry处)开始执行。中断发生时保存的上下文保证了在未来的某个时刻可以将CPU恢复到中断发生之前的状态，进而使得中断发生之前正在执行的程序可以继续运行。这里主要讨论了外部中断的执行过程，事实上CPU处理的中断信号还可能来自其内部(即内部中断)，这部分内容在其他课程中会详细介绍，在此不再赘述。

在中断技术的辅助下，当一个程序(如IE浏览器)运行了一段时间后，时钟中断发生，此时PC被设置到操作系统中的某个位置，CPU开始执行操作系统的指令，操作系统执行过程中计算出下一个要执行的程序(这个过程称为调度，Schedule)。例如酷狗音乐，将酷狗音乐被中断时的上下文恢复到CPU中，并将PC设置为酷狗音乐上次被中断处的位置，此后酷狗音乐得以继续执行。每个程序每次能执行的时间称为时间片(Time Slice)。由于CPU的工作速度非常快，上述过程在极短的时间内即被完成，因此人类用户感觉不到中间被中断过，好像所有程序都在同时运行一样。实际上，在任意时间点上，CPU都只能执行一个程序的指令。宏观上看各个程序似乎在“同时”执行，但微观上其实是称为并发(Concurrency)的串行的技术在执行，又因为“同时”执行的多个程序同时都存放在内存中，所以这种现象又称为多道程序设计(Multiprogramming)。在操作系统里，与并发相对应的另一个容易被弄混淆的概念是并行(Parallel)，在并行方式下，同一个时刻确实有

多个程序被处理，但这需要多个 CPU 的支持，本书中主要讨论单 CPU 的情况。

从上述分析可以看出，为了保证多个程序的并发执行，在操作系统中需要保存每个程序被中断的上下文，这些上下文与被执行的程序一一对应，而且每个程序每次被中断的上下文很可能不相同，为了保存这样的一组上下文信息，需要在操作系统内部开辟一块空间，使用某种数据结构进行存储，这样的一种数据结构称为进程控制块（Process Control Block，PCB）。程序的执行还需要一些输入的数据信息（如 word.exe 的执行需要一个.doc 文件作为输入），这样可以把程序、程序执行所需要的数据、PCB 统一视为一个整体，这个整体在操作系统领域里有一个专门的名字，称为进程（Process）。

通过上述分析不难看出进程具有如下特征。

（1）结构特征：通常的程序是不能并发执行的。为使程序（含数据）能独立运行，应为之配置一进程控制块，即 PCB；而由程序、相关的数据和 PCB 3 部分构成了进程实体。

（2）动态性：进程的实质是进程实体的一次执行过程，因此动态性是进程的最基本的特征。动态性还表现在“它由创建而产生，由调度而执行，由撤销而消亡”。可见，进程实体有一定的生命期，而程序则只是一组有序指令的集合，并存放于某种介质上，其本身并不具有动态的含义，而是静态的。

（3）并发性：这是指多个进程实体同时存放于内存中，并且能在一段时间内“同时”运行。并发性是进程的重要特征，同时也成为操作系统的重要特征。引入进程的目的也正是为了使某个进程实体能和其他进程实体并发执行；而程序（没有建立 PCB）是不能并发执行的。

（4）独立性：在操作系统中，独立性是指进程实体是一个能独立运行、独立分配资源和独立接受调度的基本单位。凡未建立 PCB 的程序都不能作为一个独立的单位参与运行，或者说未建立 PCB 的程序根本就不是一个进程。

（5）异步性：这是指进程按各自独立的、不可预知的速度向前推进，或者说进程实体按异步方式运行。

在这里，我们把传统操作系统中的进程定义为：进程是程序的运行过程，是系统进行资源分配和调度的一个独立单位。进程与程序之间的区别和联系有以下 4 个方面。

（1）进程是动态的，程序是静态的。可以将进程视为程序的一次执行，而程序是有序代码的集合。

（2）进程是暂时的，程序是永久的。进程存在于主存，进程运行结束就消亡，而程序可长期保存在外存储器上。

（3）进程与程序的组成不同。进程的组成包括程序、数据和进程控制块。

（4）进程与程序密切相关。同一程序的多次运行对应到多个进程，一个进程可以通过调用激活多个程序。

5.2.2 进程状态

在 5.2.1 节中提到，在实际的计算机系统中，各个进程在时钟中断的作用下呈现交替执行的并发状态，由操作系统通过调度算法选择出下一次要执行的进程。那么是不是所

有的进程在被选择时都处在相同的地位呢？考虑这样一个问题，假如有一个进程 A，在第一个时间片内执行的过程中遇到一条 I/O 指令（如从磁盘读取一组数据），为了提高 CPU 的使用率，操作系统调度另一个进程 B 执行，当 B 的时间片用光时，A 的 I/O 操作并没有完成，此时如果操作系统调度进程 A 执行，那么进程 A 能够继续向前执行么？通常来讲，答案是否定的。由于进程 A 中的后续指令很可能要用到 I/O 操作读入的数据，因此如果进程 A 继续向前执行很可能出现错误。由此可见，操作系统在调度进程执行时需要对各个进程区别对待，或者说进程的调度队列可能不止一个。那么进程依照什么排队呢？这里引入一个新的概念称为进程的状态：进程执行时的间断性决定了进程可能在不同的运行阶段具有不同的状态。一个进程可能具有以下 3 种基本状态。

（1）就绪状态（Ready）：当进程已分配到除 CPU 以外的所有必要资源后，只要再获得 CPU，便可立即运行，进程这时的状态称为就绪状态。在一个计算机系统中处于就绪状态的进程可能有多个，通常将它们排成一个队列，称为就绪队列。

（2）运行状态（Running）：进程已获得 CPU，其程序正在运行。在单处理机系统中，只有一个进程处于运行状态；在多处理机系统中，则有多个进程处于运行状态。

（3）阻塞状态（Blocked）：正在运行的进程由于发生某事件（如输入输出操作）而暂时无法继续运行时，便放弃 CPU 而处于暂停状态，即进程的运行受到阻塞，把这种暂停状态称为阻塞状态，有时也称为等待状态或封锁状态。在阻塞状态下，即使将 CPU 分配给该进程，进程也无法继续执行。致使进程阻塞的典型事件有：请求输入输出操作，申请缓冲空间等。通常将这种处于阻塞状态的进程也排成一个队列。也有一些系统根据阻塞原因的不同而把处于阻塞状态的进程排成多个队列。

处于就绪状态的进程，在调度程序为之分配了 CPU 之后，该进程便可运行，相应地，它就由就绪状态转变为运行状态。正在运行的进程也称为当前进程，如果因分配给它的时间片用完而被暂停运行时，该进程便由运行状态又恢复到就绪状态；如果因发生某事件而使进程的运行受阻（如进程请求使用某 I/O 设备，而该设备正被其他进程使用时），使之无法继续运行，该进程将由运行状态转变为阻塞状态。图 5-2 展示了进程的 3 种基本状态以及各状态之间的转换关系。

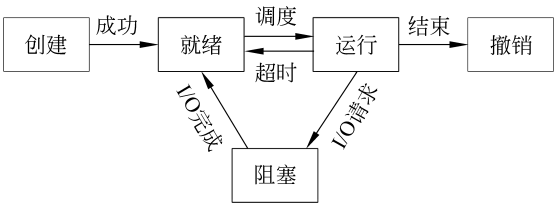

图 5-2　进程的 3 种基本状态及其转换关系

进程在整个生命周期内，就是不断地在这 3 个状态之间进行转换，直到进程被撤销。

（1）就绪→运行：就绪状态的进程一旦被操作系统选中，获得 CPU，便发生此状态变迁。因为处于就绪状态的进程往往不止一个，操作系统根据调度算法把 CPU 分配给其中某个就绪进程，建立该进程运行状态标记，把它由就绪状态变为运行状态，并把 CPU 的控制权转到该进程，这样该进程就投入运行。

(2) 运行→阻塞：运行中的进程需要执行 I/O 请求时，发生此状态变迁。处于运行状态的进程为完成 I/O 操作需要申请新资源(如需要等待数据的读入)而又不能立即被满足时，进程状态由运行变成阻塞。此时，系统将该进程在其等待的设备上排队，形成资源等待队列。同时，操作系统根据调度算法把 CPU 分配给处于就绪状态的其他进程。

(3) 阻塞→就绪：被阻塞进程的 I/O 请求完成时，发生此状态变迁。被阻塞的进程在其被阻塞的原因获得解除后不能立即执行，而必须通过操作系统统一调度获得 CPU 才能运行。因此，系统将其状态由阻塞状态变成就绪状态，放入就绪队列，使其继续等待 CPU。

(4) 运行→就绪。当一个正在运行的进程时间片用完时，发生此状态变迁。一个正在运行的进程，由于规定的运行时间片用完，系统将该进程的状态修改为就绪状态，插入就绪队列。

5.2.3 进程管理与调度

1. 进程控制块

前面提到，为了保存程序执行的断点，引入了一种特殊的数据结构——进程控制块 PCB。实际上，它的作用准确来说是为了描述和控制进程的运行，它是进程实体的一部分，是操作系统中最重要的记录型数据结构。PCB 中记录了操作系统所需的、用于描述进程的当前情况以及控制进程运行的全部信息。进程控制块的作用是使一个在多道程序环境下不能独立运行的程序(含数据)，成为一个能独立运行的基本单位，一个能与其他进程并发执行的进程。或者说，操作系统是根据 PCB 来对并发执行的进程进行控制和管理的。例如，当操作系统要调度某进程执行时，要从该进程的 PCB 中查出其当前状态及优先级；在调度到某进程后，要根据其 PCB 中所保存的 CPU 各寄存器信息，设置该进程恢复运行的现场(中断时的状态)，并根据其 PCB 中保存的程序和数据的内存地址，找到其需要运行的程序和数据；进程在运行过程中，当需要和与之合作的进程实现同步、通信或访问文件时，也都需要访问 PCB；当进程由于某种原因而暂停运行时，需要将其断点的 CPU 环境保存在 PCB 中。可见，在进程的整个生命周期中，系统总是通过 PCB 对进程进行控制，即系统是根据进程的 PCB 而不是任何别的什么而感知到该进程的存在。当系统创建一个新进程时，就为它建立了一个 PCB；进程结束时又回收其 PCB，进程于是也随之消亡。PCB 可以被操作系统中的多个模块读取或修改，例如被调度程序、资源分配程序、中断处理程序以及监督和分析程序等读取或修改。

在进程控制块中，主要包括下述 4 个方面的信息。

(1) 进程标识符：进程标识符用于唯一地标识一个进程。

(2) CPU 状态：CPU 状态信息主要是由 CPU 的各种寄存器中的内容组成的。CPU 在运行时，许多信息都放在寄存器中。当进程被中断时，所有这些信息都必须保存在 PCB 中，以便在该进程重新运行时，能从断点继续运行。

(3) 进程调度信息：在 PCB 中还存放一些与进程调度有关的信息，包括：①进程状

态，指明进程的当前状态；②进程优先级，用于描述进程使用 CPU 的优先级别的一个整数；③进程调度所需的其他信息，它们与所采用的进程调度算法有关，例如进程已等待 CPU 的时间总和、进程已运行的时间总和等；④事件，指进程由运行状态转变为阻塞状态所等待发生的事件，即阻塞原因。

(4) 进程控制信息：包括：①程序和数据的地址，指进程的程序和数据所在的内存或外存(首)地址；②进程同步和通信机制，指实现进程同步和进程通信时必须具有的机制，如消息队列指针、信号量等；③资源清单，即一张列出了除 CPU 以外的、进程所需的全部资源以及已经分配到该进程的资源的清单；④连接指针，它给出了本进程(PCB)所在队列中的下一个进程的 PCB 的首地址。

2. 进程控制

进程控制是进程管理中最基本的功能。它用于创建一个新进程，终止一个已经完成的进程，或者终止一个因出现某事件而使其无法运行下去的进程，还负责进程运行中的状态转换。例如，当一个正在运行的进程因等待某事件而暂时不能继续运行时，将其转换为阻塞状态，而当该进程所期待的事件出现时，又将该进程转换为就绪状态，等等。

1) 进程的创建

一旦操作系统发现了要求创建新进程的事件后，便按下述步骤创建一个新进程。

(1) 申请空白 PCB。为新进程申请获得唯一的数字标识符 PID(Process Identifier)，并从内存区域申请一块未使用区域用于存放该 PCB。

(2) 为新进程分配资源。为新进程的程序和数据以及用户栈分配必要的内存空间。

(3) 初始化进程控制块。PCB 的初始化包括：①初始化标识信息；②初始化 CPU 状态信息；③初始化 CPU 控制信息，将进程的状态设置为就绪状态。

(4) 将新进程插入就绪队列。

2) 进程的终止

如果系统中发生了要求终止进程的某个事件，操作系统将按下述过程终止指定的进程。

(1) 根据被终止进程的标识符，从 PCB 集合中检索出该进程的 PCB，从中读出该进程的状态。

(2) 若被终止进程正处于运行状态，应立即终止该进程的运行。

(3) 若该进程还有子孙进程，还应将其所有子孙进程予以终止，以防它们成为不可控的进程。

(4) 将被终止进程所拥有的全部资源，或者归还给其父进程，或者归还给系统。

(5) 将被终止进程(PCB)从所在队列或链表中移除。

3) 进程的阻塞

进入阻塞状态后，由于此时该进程还处于运行状态，所以应先立即停止运行，把进程控制块中的现行状态由“运行”改为“阻塞”，并将 PCB 插入阻塞队列。如果系统中设置了因不同事件而阻塞的多个阻塞队列，则应将本进程插入具有相同事件的阻塞(等待)队列。最后，运行调度程序进行重新调度，将 CPU 分配给另一个就绪进程并进行切换，即保留

被阻塞进程的 CPU 状态(在 PCB 中),再按新进程的 PCB 中的 CPU 状态设置 CPU 的环境。

4) 进程的唤醒

当某进程被阻塞的原因消失(如获得被阻塞时需要的资源)时,操作系统将其唤醒。唤醒进程的过程是:首先通过进程标识符找到被唤醒进程的 PCB,然后从阻塞队列中移出该 PCB,将 PCB 中的进程状态设置为就绪状态,并插入就绪队列。

3. 进程调度

当 CPU 空闲时,操作系统将按照某种策略从就绪队列中选择一个进程,将 CPU 分配给它,使其能够运行。按照某种策略选择一个进程,使其获得 CPU 的工作称为进程调度。引起进程调度的因素有很多,例如正在运行的进程结束运行、运行中的进程要求 I/O 操作、分配给运行进程的时间片已经用完等。

进程调度策略的优劣将直接影响到操作系统的性能。目前常用的调度策略有以下 4 种。

(1) 先来先服务:按照进程到达就绪列表的先后顺序来调度进程,到达得越早就越先执行。

(2) 时间片轮转:系统把所有就绪进程按先后次序排队,并总是将 CPU 分配给就绪队列中的第一个就绪进程,分配 CPU 的同时分配一个固定的时间片(如 50ms)。当该运行进程用完规定的时间片时,系统将 CPU 和相同长度的时间片分配给下一个就绪进程,如图 5-3 所示。每个用完时间片的进程,如果未遇到任何阻塞事件则将在就绪队列的尾部排队,等待再次被调度运行。

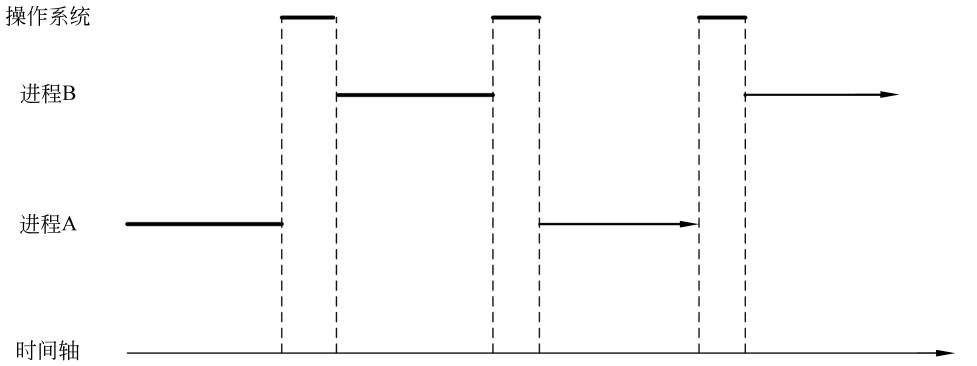

图 5-3 时间片轮转调度示意图

(3) 优先级法:把 CPU 分配给就绪队列中具有最高优先级的就绪进程。根据已占有 CPU 的进程是否可被抢占这一原则,又可将该方法分为抢占式优先级调度算法和非抢占式优先级调度算法。前者当就绪进程优先级高于正在 CPU 上运行的进程优先级时,将会强行停止其运行,将 CPU 分配给就绪进程;而后者不进行这种强制性切换。

(4) 多级反馈队列轮转:把就绪进程按优先级排成多个队列,赋给每个队列不同的时间片,一般高优先级进程的时间片比低优先级进程的时间片小。调度时按时间片轮