第5章 操作系统

操作系统是计算机系统中最重要、最复杂的软件，它对计算机系统的软件和硬件资源进行协调管理，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。

本章首先介绍什么是操作系统以及操作系统的简要发展历程，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 概 述

在早期的计算机上，程序员通过打孔纸带的方式编程，当程序员编程完毕后，将编好的程序（即一组纸卡或一卷纸带）交给计算机操作员，由操作员将程序放入计算机的输入设备读入，计算机将程序运行完毕后，再由输出设备输出到一组纸卡或纸带上。随着计算机技术的发展和进步，程序的运行越来越快，操作员手动放入纸带的速度显得过慢，人工操作成为整个系统运行效率的瓶颈。因此，需要一个软件来负责自动读入程序，并交给计算机运行。某种意义上说，该软件代替了操作员来操作真正的计算机，这个软件就是操作系统（Operating System，OS）。

当今世界，几乎所有的计算机都安装了一个底层软件，即操作系统。操作系统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，并使得计算机中各类软件和硬件资源能够高效协调工作起来。

5.1.1 操作系统的发展

操作系统的发展历程与计算机硬件的发展息息相关。

1. 无操作系统时代

以 ENIAC 为代表的第一代电子管计算机，其操控方式是人工手动操作实现编程，即通过人工扳动开关进行编程，如图 5-1 所示。

程序员在 ENIAC 上通过手动扳动开关、插拔线缆的方式编程，因此，该时代的计算机并没有操作系统的存在。

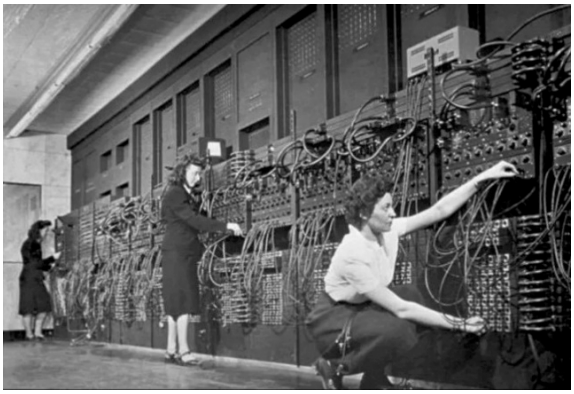

图 5-1　人工手动操作实现编程

2. 单道批处理系统

在晶体管计算机时代，以 IBM 7094 为代表的计算机的编程变为用打孔机在纸卡（纸带）上编程，然后将纸卡（纸带）拿到计算机上进行执行。当计算机的运行速度比较快时，希望通过某个程序操控纸卡（纸带）的装入和执行，此时出现第一代操作系统——单道批处理系统。

批处理系统负责监控程序的执行和任务（纸带）的输入，当一个任务执行完毕或出错时，自动执行下一个。为了能充分地利用晶体管计算机，应尽量让计算机连续运行，以减少空闲时间。为此，程序员把一批程序先输入到磁带上，并在系统中配上监督程序（Monitor），在监督程序的控制下使这批程序能一个接一个自动地连续处理。

早期批处理系统自动处理的过程是：首先，由监督程序将磁带上的第一个程序 A 装入内存，并把 CPU 的使用控制权交给程序 A；当程序 A 处理完成时，又把 CPU 的控制权交还给监督程序，再由监督程序把磁带（盘）上的第二个程序 B 调入内存；计算机系统就这样逐个对程序进行处理，直至磁带（盘）上的所有程序全部完成。

由于系统对程序的处理都是成批地进行，且在内存中始终只保持一个程序，故称此系统为单道批处理系统（Simple Batch Processing System）。历史上 IBM 7094 机就是一个典型的单道批处理系统，如图 5-2 所示。

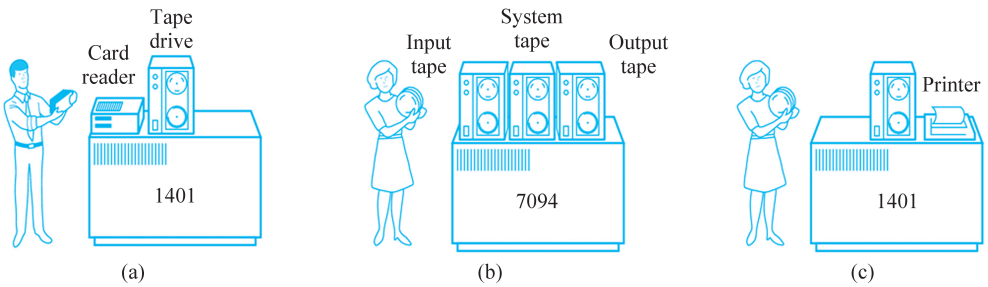

图 5-2　IBM 7094 上的一种早期单道批处理系统

图 5-2(a)为程序员将一批卡片拿到 1401 机处，然后 1401 机将批处理作业读到磁带上；再由图 5-2(b)中操作员将输入纸带送至 7094 机，然后 7094 机进行计算；结束后图 5-2(c)中

操作员将输出磁带送到 1401 机，最后 1401 机打印输出。

3. 多道批处理系统

在单道批处理系统中，内存中仅有一个程序，无法充分利用系统中的所有资源，致使系统性能较差。为了进一步提高资源的利用率和系统吞吐量，20 世纪 60 年代中期又引入了多道程序设计技术，并将其用于第三代计算机中，由此而形成了多道批处理系统（Multiprogrammed Batch Processing System）。

在多道批处理系统中，各个程序在调度程序的控制下，按一定的算法交替使用 CPU，使它们共享 CPU 和系统中的各种资源，如图 5-3 所示，从而显著提高了系统的整体效率和性能。

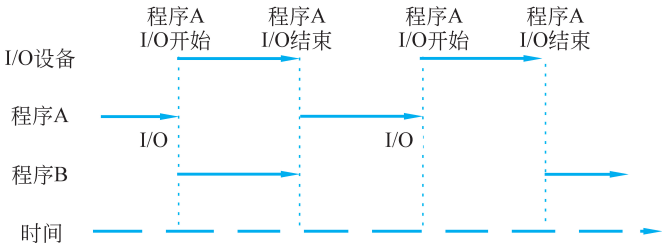

图 5-3　多道批处理系统工作示例

在图 5-3 的多道程序示例中，当程序 A 需要进行 I/O（输入/输出）操作时，调度程序会将 CPU 的使用权调度给程序 B，此时 I/O 设备同时也在处理 A 的 I/O 命令，当程序 A 的 I/O 结束后，调度程序将 CPU 的使用权重新交还给 A 程序。这样，在逻辑上形成了“多道”的运行结果，在 A 进行 I/O 时，CPU 并没有空闲，而是在处理 B 程序。

与单道程序设计相比，多道程序设计的 CPU 利用率得到提高。同时，由于 I/O 设备也在工作，因此，采用多道程序设计方法不仅提高了 CPU 利用率，也提升了 I/O 设备的使用效率。

1973 年，北京大学与“738 厂”联合成功研制出的我国首款每秒百万次运算计算机 150 机，这台计算机运行了中国第一个多道操作系统。负责开发 150 机多道运行操作系统的北京大学杨芙清教授团队在 1981 年又成功研发出 240 机操作系统，240 机操作系统是国内首个全部使用高级语言书写的大型操作系统。图 5-4 是杨芙清教授向比尔·盖茨介绍北大青鸟。

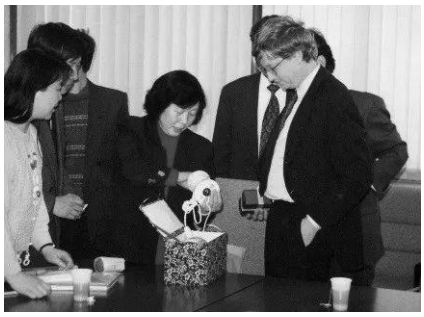

图 5-4　杨芙清教授向比尔·盖茨介绍北大青鸟

4. 分时操作系统

计算机发展的初期阶段，计算机造价都非常昂贵，以 1962 年 IBM 推出的 7094 为例，一台 7094 计算机的造价就大约 250 万美元！而一台机器同一时间只能服务一个用户，造成资源的极大浪费。为了解决这一问题，人们提出了分时系统（Time Sharing System，TSS），即一台计算机可以在不同时间段内服务于多个用户，从而提高了设备的利用率，降低了成本，推动了计算机技术的进一步发展。

分时系统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该计算机，共享主机中的资源，从而顺利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。

第一台真正的 CTSS（Compatable Time Sharing System，分时操作系统）是由美国麻省理工学院开发成功的。继成功后，麻省理工学院又和美国贝尔实验室、通用电气公司联合开发出多用户多任务操作系统——MULTICS，该系统使得一台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肯·汤普逊（Kenneth Lane 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。

5. 实时操作系统

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。随着计算机应用场景的增多，在军事、航空航天等领域出现了对系统响应实时性的需求，于是实时系统（Real Time System，RTS）应运而生。

实时系统的主要功能是“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。

当前主流的实时操作系统包括风河公司（Wind River System）的 VxWorks 系统以及国内的 RT-Thread 等。VxWorks 以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如卫星通信、军事演习、弹道制导、飞机导航等。在美国的 F-16、FA-18 战斗机、B-2 隐形轰炸机和爱国者导弹上，甚至连 1997 年 4 月在火星表面登陆的火星探测器、2008 年 5 月登陆的凤凰号和 2012 年 8 月登陆的好奇号也都使用了 VxWorks。

6. 其他操作系统

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，出现了嵌入式操作系统、网络操作系统、分布式操作系统、多处理器操作系统等。其中，生活中最为常见的是嵌入式操作系统，例如智能手机和平板电脑里的 Android、iOS、鸿蒙等。

随着科技的飞速发展，全球科技巨头之间的竞争愈发激烈。华为公司作为中国科技行业的领军企业，一直在自主研发的道路上不断突破。2012 年，华为公司开始规划自

有操作系统——鸿蒙操作系统(Harmony OS)。2019 年 8 月 9 日华为公司在广东东莞举行的华为开发者大会上正式发布鸿蒙操作系统。鸿蒙操作系统是一款全新的、面向全场景的分布式操作系统,能够创造一个超级虚拟终端互联的世界,将人、设备、场景有机地联系在一起。

5.1.2 操作系统的特征

1. 多道程序管理特征

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

并发是指两个或多个程序在同一时间段内交替运行。并发与并行是有区别的,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

共享是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

虚拟是指通过某种技术,把一个物理实体变为若干逻辑上的对应物。前者物理实体是实的,即实际存在的;而后者是虚的,仅是用户感觉上的东西。

(4) 异步性

异步是指在多道程序环境下,允许多个程序并发执行,但只有在获得所需的资源后方能执行。在单 CPU 环境下,由于计算机中只有一个 CPU,因而每次只能允许一个程序执行,其余程序只能等待。因此,系统中各程序的执行因等待时间未知导致不可提前预知它们的推进速度。

2. 当代操作系统特征功能

操作系统已经从最早的简单监控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,但其基本功能仍然可以简单地概括为:操作系统是一套控制和管理计算机软件、硬件等计算机资源的程序,通过合理地组织计算机工作流程,为用户使用计算机提供方便的用户接口。

从冯·诺依曼体系结构出发,计算机硬件包括 CPU、内存、输入输出设备等部分,因此,操作系统对硬件的管理功能包括 CPU 管理、内存管理和 I/O 设备管理。硬盘作为使用最为频繁、使用方法最为复杂的外部设备,操作系统为其单独抽象出了文件管理的功能。同时,为了方便用户与计算机的交互,操作系统还提供了方便的用户接口管理。下面将介绍操作系统的上述功能。

5.2 CPU 管 理

5.2.1 进程与程序

1. 进程的概念

(1) 指令周期

为了弄清楚操作系统是如何对 CPU 进行管理的，首先需要弄清楚 CPU 是如何工作的。前面已经介绍了 CPU 的工作原理，可以用“取指执行”4 个字概括这个过程，CPU 的指令周期如图 5-5 所示。

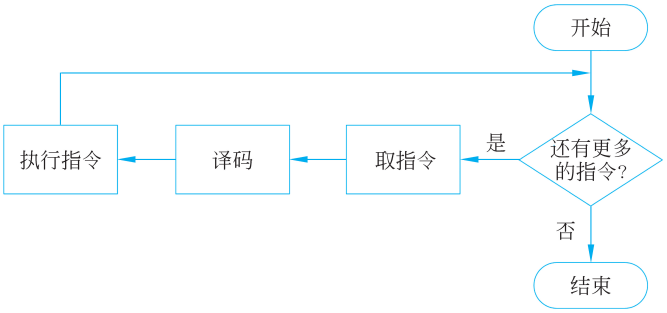

图 5-5 CPU 的指令周期

CPU 从程序计数器(PC)所指向的内存地址取出指令，并存储在指令寄存器(IR)中，而后对指令进行译码和执行，同时 PC 自动加 1。

(2) 中断

假设用户使用计算机时，需要一边制作 PPT，一边用浏览器访问网页，同时用迅雷软件下载资料，还使用酷狗音乐播放歌曲。如果没有其他技术支持，当某个程序正在执行时，PC 只能在这个程序的内部来回移动。也就是说，一段时间内只能有一个程序在运行。为了解决这个问题，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。简单地讲，中断就是暂停某个程序的执行，CPU 转去处理其他程序，之后再恢复这个程序的执行。带中断检测的 CPU 的指令周期如图 5-6 所示。

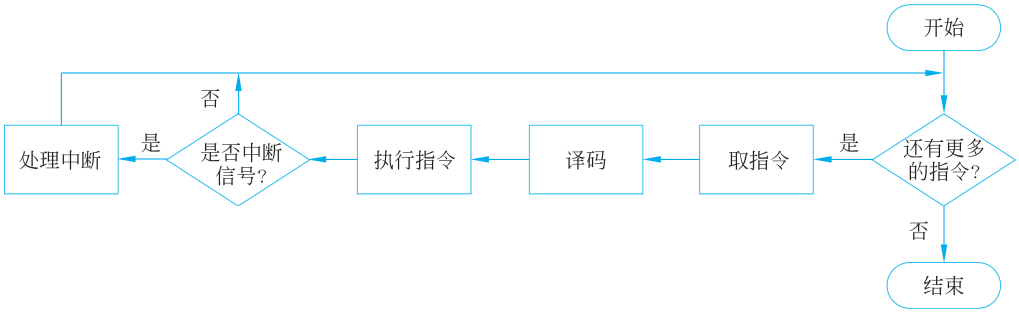

图 5-6 带中断检测的 CPU 指令周期

中断发生时的处理过程是：首先由 CPU 之外的设备(可能是时钟发生器，也可能是一个 I/O 设备)向 CPU 产生一个中断信号，中止当前 CPU 正在执行的工作；CPU 将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 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。

这里主要讨论的是外部中断的执行过程，事实上 CPU 处理的中断信号也可能来自其内部，即内部中断，在此不再赘述。

(3) 并发

在中断技术的辅助下，可以实现多个程序的交替执行。

例如，当前内存中有操作系统及 4 个正在运行的应用程序：酷狗音乐、360 浏览器、迅雷和 PowerPoint，不同时刻 PC 的指向如图 5-7 所示。

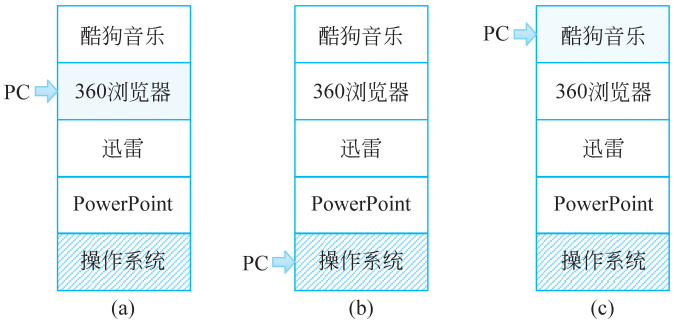

图 5-7　不同时刻 PC 的指向

CPU 在同一时刻只能处理一个程序，假设当前仅 360 浏览器正在运行，即 PC 指向 360 浏览器，如图 5-7(a)所示。

当 360 浏览器程序运行了一段时间后，时钟中断发生，此时 PC 跳转到操作系统中的某个位置，如图 5-7(b)所示。

CPU 开始执行操作系统的指令，操作系统通过调度算法计算出下一个要执行的程序是酷狗音乐，那么操作系统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，如图 5-7(c)所示，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。这种宏观上各个程序似乎在“同时”执行，但微观上其实是串行的技术称为并发(Concurrency)，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)。

在操作系统中，与并发相对应的另一个容易被弄混淆的概念是并行(Parallel)。并行在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，本书主要讨论单 CPU 的情况。

(4) 进程

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

执行过程中被中断时的上下文(程序执行被中断时 CPU 的快照,即 CPU 中若干关键寄存器的信息),这些上下文与被执行的程序一一对应,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被保存的上下文信息,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,这样的一个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 统一看作一个整体,这个整体称为进程(Process)。

本书中依据传统操作系统中对进程的定义:进程是进程实体的运行过程,是系统进行资源分配和调度的一个独立单位。

进程在 Windows 中可以通过任务管理器查看和管理,如图 5-8 所示。

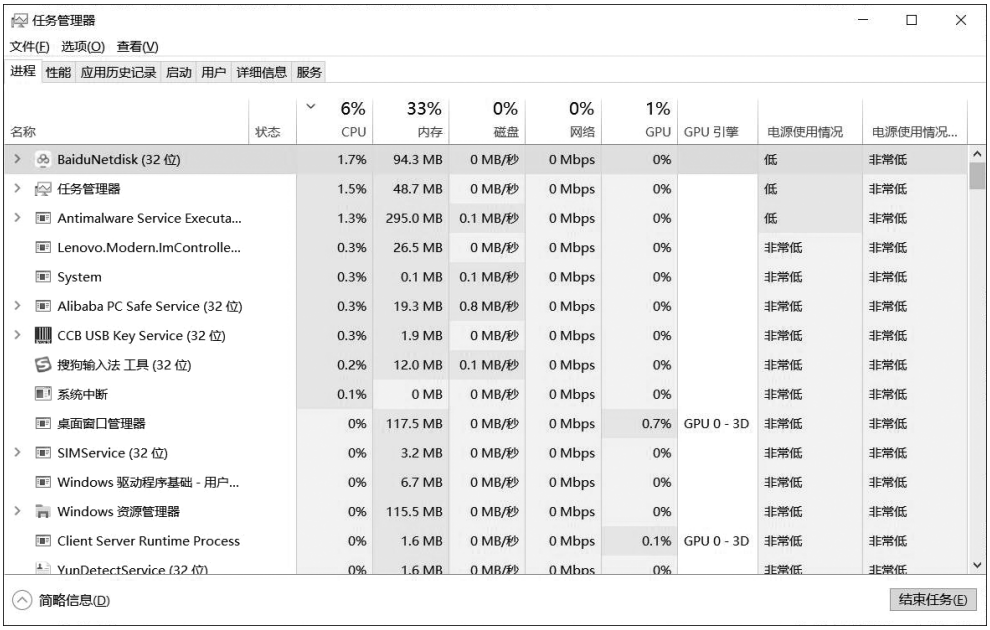

图 5-8　任务管理器中的进程

2. 进程的特征

进程是程序的一次执行,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,应为之配置一个进程控制块,即 PCB;而由程序、相关的数据和 PCB 三部分便构成了进程实体。

(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:进程由创建而产生,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,其本身并不具有运动的含

义，因而是静态的。

（3）并发性

并发是指多个进程实体同存于内存中，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使某个进程实体能和其他进程实体并发执行。

而程序因为没有建立 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，换句话说，未建立 PCB 的程序根本就不是一个进程。

（5）异步性

异步性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。

3. 进程与程序的关系

进程与程序是两个完全不同的概念，综上定义和特征分析也能分辨出二者之间具有明显的区别，同时又存在一定的联系。进程与程序的关系如表 5-1 所示。

表 5-1　进程与程序的关系

属　性	进　程	程　序
状态	程序的一次执行，动态	有序代码的集合，静态
生命周期	由创建而产生，由调度而执行，由撤销而消亡，暂时	代码文件可长期保存在外存储器，永久
组成	程序、数据、进程控制块 PCB	代码
联系	同一个程序的多次运行则对应到多个进程；而一个进程可以通过调用激活多个程序	

5.2.2　进程状态

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，在其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 A 执行，A 能够继续向前执行么？通常来讲，答案是否定的。由于 A 进程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 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。

1. 进程状态

为了解决进程队列的调度，首先引入一个新的概念——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三种基本状态。

（1）就绪状态

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

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

正在运行的进程由于发生某事件（如 I/O 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。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（I/O）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，有的系统则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。

2. 进程状态的调度

处于就绪状态的进程，在调度程序为它分配了 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已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。进程在整个生命周期内，就是不断地在这三个状态之间进行转换，直到进程被撤销，如图 5-9 所示。

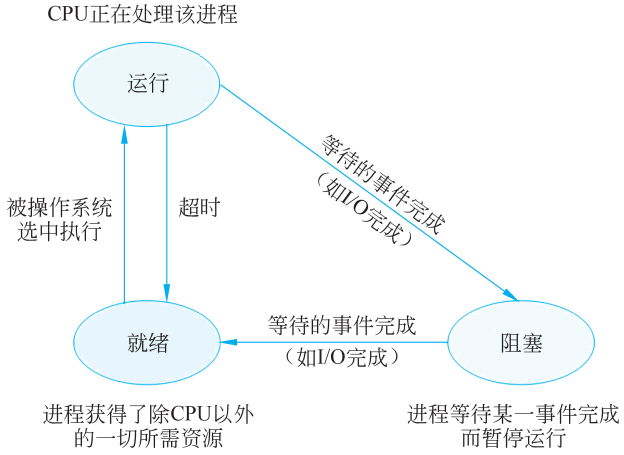

图 5-9　进程的三种基本状态及其转换

（1）就绪状态→运行状态

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