2.1 进程概念

1. 进程概念

进程概念是操作系统中最基本、最重要的概念之一。进程是一个正在执行中的程序或者可以描述为可并发执行的程序在一个数据集合上的运行过程。进程不只是程序代码，程序代码有时称为文本段（或代码段）。进程还包括当前活动、程序计数器的值和处理器寄存器的内容。另外，进程通常还包括进程堆栈段（包括临时数据，如函数参数、返回地址和局部变量）和数据段（包括全局变量）。进程还可能包括堆(heap)，它是在进程运行期间动态分配的内存。内存中的进程结构如图 2.1 所示。

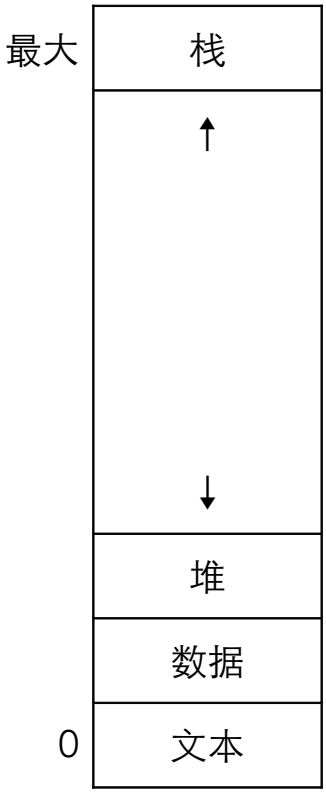

图 2.1 内存中的进程

从操作系统管理角度出发，进程由数据结构以及在其上执行的程序组成，是程序在

这个数据集合上的运行过程，也是操作系统进行资源分配和调度的基本单位。因此进程需要一些资源，比如 CPU 时间、内存空间、文件以及 I/O 设备。这些资源可以在进程创建时分配，或者可以在进程运行过程中动态分配。

作为描述程序执行过程的概念，进程具有动态性、独立性、并发性和结构化等特征。

- 动态性是进程的最基本特征，它是程序执行过程，它是有一定的生命期。它由创建而产生、由调度而执行，因得不到资源而暂仃，并由撤消而死亡。
- 独立性是指各进程的地址空间相互独立，除非采用进程间通信手段，否则不能相互影响。
- 并发性也称为异步性，是指从宏观上看，各进程是同时独立运行的。
- 结构化是指进程地址空间的结构划分，如代码段、数据段等。

进程和程序是两个密切相关的不同概念，它们在以下几个方面存在区别和联系:

- 进程是动态的，程序是静态的。程序是有序代码的集合；进程是程序的执行。进程通常不可以在计算机之间迁移；而程序通常对应着文件、静态和可以复制。
- 进程是暂时的，程序是永久的。进程是一个状态变化的过程；程序可长久保存。
- 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（即进程状态信息）。
- 进程与程序是密切相关的。通过多次执行，一个程序可对应多个进程；通过调用关系，一个进程可包括多个程序。进程可创建其他进程，而程序并不能形成新的程序。

在很多操作系统原理书中，描述进程的活动时，有的时候会以作业、程序、任务等概念出现，这需用通过上下文进行理解。

2.进程的状态与转换

进程与程序的最大区别是其动态性。随着进程的推进，它会不断改变状态。进程状态由具体操作系统根据进程当前活动而定义。例如，每个进程可处于如下状态：创建、就绪运行、等待（阻塞）、终止。通常讲的进程三个基本状态是指就绪状态、运行状态和等待或阻塞状态。

- 创建（新）：进程正被创建；
- 运行：进程的指令正被执行；
- 等待：进程正在等待发生一些事件（如I/O 完成或接收一个信号），也称阻塞状态；
- 就绪：进程正等待分配处理器；
- 终止：进程结束运行。

当前不在执行的进程会放在某个等待队列中。究竟处于哪个等待队列，取决于进程正在等待什么资源或什么事件。其中，就绪队列有其特殊性，就绪队列中的进程正在等待 CPU 资源；而且，进程等待 CPU 之时，它一定已经得到了其它一切必需的资源。

当进程状态发生变化时，导致进程状态转换。如图 2.2

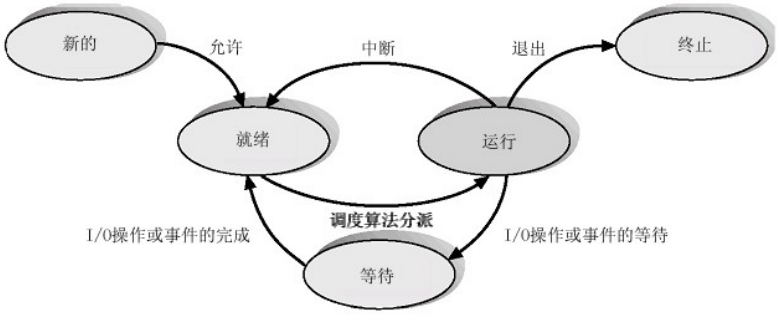

图 2.2　进程状态转换图

此时，必然引起资源的释放或申请。例如，进程得到 CPU 时，从就绪状态变成运行状态；进程需要一个资源却暂时得不到该资源时，从运行状态变成等待状态。

并发运行的每个进程不断地从一个状态转换到另一个状态，就绪、运行、等待三个基本状态之间可能转换和转换原因如下：

- 就绪状态→运行状态：当处理机空闲时，进程调度程序必将处理机分配给一个处于就绪状态的进程，该进程便由就绪状态转换为运行状态。
- 运行状态→等待状态：处于运行状态的进程在运行过程中需要等待某一事件发生后（例如因 I/O 请求等待 I/O 完成后），才能继续运行，则该进程放弃处理机，从运行状态转换为等待状态。
- 等待状态→就绪状态：处于等待状态的进程，若其等待的事件已经发生，于是进程由等待状态转换为就绪状态。
- 运行状态→就绪状态。处于运行状态的进程在其运行过程中，因分给它的处理机时间片已用完或被抢占，而不得不让出处理机，于是进程由运行状态转换为就绪状态。

3.进程组织

进程是个抽象的概念，它无法亲自挂在等待队列里“排队”。在计算机这种“0/1”世界里，进程需要一个具体的代表参与管理。这就是进程控制块（Process Control Block，PCB）。每个进程都有唯一的 PCB 与之对应。进程在各种队列之间转换，其实是

进程的 PCB 在对应队列里转换；PCB 拥有指针，处于同一队列的 PCB 由这些指针实现相互间的链接。PCB 可以说是操作系统中最复杂的数据结构，它包含了所有描述其对应进程的独有信息。对进程的控制，实际上就是对进程 PCB 的控制。

操作系统在管理进程过程中需要负责以下工作：进程的创建和删除；进程的调度；进程的同步、进程的通信、进程死锁的处理等。当然，都通过管理 PCB 而得以实现。

进程控制块存储了某一具体进程的信息，包括：

- 进程状态：该状态可能是新、就绪、运行、等待、终止等等。
- 程序计数器：计数器指明了该进程要执行的下一条指令的地址。
- CPU寄存器：基于计算机体系结构，这些寄存器的数量和类型不相同。包括累加器、变址寄存器、栈指针、通用寄存器等。和程序计数器一样，在中断发生时必须要保存这些状态信息，这样便于后来进程继续正确执行。
- CPU 调度信息：包括进程优先权、指向调度队列的指针和其它的调度参数。
- 内存管理信息：包括基址寄存器和界限寄存器值、页表或段表、虚拟地址空间管理信息等。
- 记账信息：包括CPU时间、实际使用时间、账户数目、作业或进程数目等。
- I/O 状态信息：包括分配给该进程的I/O设备列表、打开文件的列表等。

存放在磁盘上的可执行文件的代码和数据的集合称为可执行映象(Executable Image)。当一个程序装入系统中运行时，它就形成了一个进程。进程是由程序代码、数据

和进程控制块组成。进程上下文（context）实际上是进程执行活动全过程的静态描述。具体说，进程上下文包括系统中与执行该进程有关的各种寄存器（例如：通用寄存器、程序计数器 PC、程序状态寄存器 PS 等）的值，程序的机器指令代码集（或称正文段）、数据集及各种堆栈值和 PCB 结构，如图 2.3 所示。

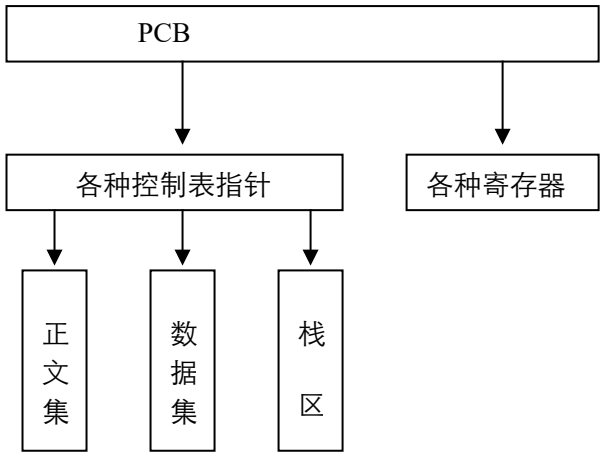

图 2.3 进程上下文结构

在 Unix 操作系统中，进程上下文（也称进程映象）包括三个组成部分：

- 用户级上下文：由用户程序代码、用户数据段（含共享数据块）和用户堆栈组成的进程地址空间。
- 系统级上下文：包括进程的标识信息、现场信息和控制信息，进程环境块，以及

系统堆栈等组成的进程地址空间。

- 寄存器上下文：由程序状态字寄存器和各类控制寄存器、地址寄存器、通用寄存器组成。

4. 上下文切换

当进程调度程序选择一个新进程并为其分配CPU时，要将CPU切换到另一个进程，要求首先保存原来进程的上下文，然后装入新进程的上下文。将CPU切换到另一个进程需要保存当前进程的状态并恢复另一个进程的状态，这一任务称为上下文切换(context switch) 。也就是说，当发生上下文切换时，内核会将旧进程的上下文保存在其PCB中，然后装入经调度要执行的新进程的已保存的上下文，如图2.4所示。上下文切换时间是额外开销，因为切换时系统并不能做任何用户进程要求的工作。

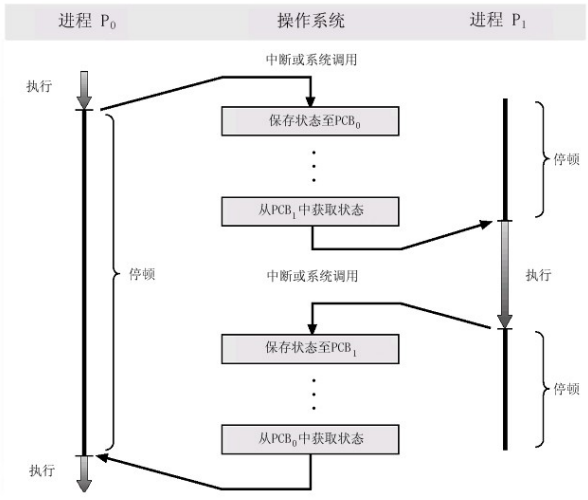

图 2.4　进程上下文切换过程

进程上下文切换通过中断实现，当发生一个中断时，系统需要保存当前运行在CPU

中进程的上下文，从而在其处理完后能恢复上下文，即先中断进程，之后再继续。进程上下文用进程的PCB表示，它包括CPU寄存器的值、进程状态和内存管理信息等。通常，通过执行一个状态保存(state save) 来保存CPU 当前状态(不管它是内核模式还是用户模式)，之后执行一个状态恢复(state restore) 重新开始运行。