第3章 进程管理

处理机是系统的重要资源之一，而处理机的管理实际上是进程的管理。在现代计算机系统中，通常以进程的观点来设计和研究操作系统。因此，只有深刻理解进程的概念，才能够很好地理解操作系统各部分的功能和工作原理。

本章首先引入进程的概念，指出其特点，然后逐步介绍进程的管理，包括进程的建立、调度、控制等。

3.1 进程的概念

进程是现代操作系统中最重要的概念之一。在多道程序系统中程序并发的执行导致其出现了与单道程序顺序执行时一些不同的特征，由此引入了进程的概念。

3.1.1 进程的引入

1. 单道程序的顺序执行

在早期的计算机系统中，只有单道程序执行的功能。也就是说，每一次只允许一道程序运行，在这个程序运行期间，它将独立占用整个计算机系统的资源，而且系统按照程序的步骤顺序地执行，在该程序执行完之前，其他程序只能等待。这种程序的执行方式称为顺序执行方式。程序的顺序执行具有如下特点：

(1) 顺序性

程序的执行过程是一系列严格按程序规定的状态转移过程。上一条指令的执行结果是下一条指令的执行开始的充分必要条件。

(2) 封闭性

程序是在封闭的环境下执行的，即程序执行时独占资源，资源的状态(除初始状态外)只有本程序才能改变它。程序执行得到的最终结果由初始条件决定，不受外界因素的影响。

(3) 可再现性

程序执行结果与它的执行速度无关，只要输入的条件相同，重复执行时，不论它是从头到尾不停顿地执行，还是“停停走走”地执行，都会得到相同的结果。

程序的顺序执行的特点，使系统管理非常方便，程序员检验和校正程序的错误也很容

易。然而，系统的资源利用率却非常低，尤其在对外部设备进行操作的时间内，系统处理器都在等待。

2. 多道程序系统中程序的执行

为提高处理机的效率，人们设想让多个程序同时执行。然而，单处理机系统每一时刻只能执行一条指令，如果要同时执行多条指令，必须具有多个处理机或者处理部件，这就是并行结构和并行处理要解决的问题。

能否在单处理机上实现程序的同时执行呢？这就是程序的并发执行问题。并发执行是基于多道程序的一个概念，即让多道程序在计算机中交替地执行，当一道程序不用处理机时，另一道程序就马上使用，从而大大提高了处理机的利用率。虽然在每一时刻仍然只有一条指令在执行，但在计算机的主存储器中同时存放了多道程序，在同一时间间隔内，这些程序在交替地执行。因此，在微观上指令是顺序的，而在宏观上程序是并发的，从而减少了处理机的等待时间，使得处理机和外设可同时工作，提高了系统的使用效率。

程序的并发执行，虽然提高了系统的使用效率，但由于多道程序在主机中并发执行，共享系统资源，因而产生了一些与顺序程序不同的特点：

(1) 间断性

程序在并发执行时，由于它们共享系统资源，以及为完成同一项任务而相互合作，致使在这些并发执行的程序之间，形成了相互制约的关系。例如，几个并发程序竞争同一资源(如打印机)，得到资源的程序继续运行，而其他的则只有等待，这是间接制约。又如，一程序请求从磁盘中读入一个文件，它就直接受到系统磁盘管理程序何时完成该请求的制约，只有等到后者读入了指定的文件后，该程序才能继续执行与该文件有关的操作，这是直接制约。由于存在制约，就存在等待，因此，并发程序具有“执行—暂停—执行”这种间断性的活动规律。

(2) 失去封闭性

程序在并发执行时，多个程序共享系统中的各种资源，因而这些资源的状态将由多个程序来改变，致使程序的运行失去封闭性。这样，某个程序在执行时，必然会受到其他程序的影响。例如，当处理机这一资源已被某个程序占用时，另一程序必须等待。

(3) 不可再现性

程序在并发执行时，由于失去了封闭性，也将导致其失去可再现性。

例 3-1　设有两道程序 CP 和 PP，它们共享一个变量 n，其初值为 0。CP 程序循环 10 000 次，做 n＝n＋1；PP 程序将 n 的值打印。CP 和 PP 可分别描述如下：

```
CP()                                PP()
{                                   {
  while(n<10000)                      printf("n = %d\n",n);
    n = n + 1;                      }
}
```

显然，如果上例中的 CP 和 PP 程序段进行顺序执行，其执行结果为 n＝10 000(屏幕显示)。但如果让两个程序段并发执行时，程序 CP 和 PP 的执行速度不同，将有可能出现下述几种情况：

(1) 首先程序段 CP 抢占了处理机开始执行,然后执行程序段 PP,执行结果是 n=10 000。

(2) 首先程序段 PP 抢占了处理机开始执行,然后执行程序段 CP,执行结果是 n=0。

(3) 如果在某一分时系统中。首先,程序段 CP 开始执行,执行到某一时间,其时间片到(假设 CP 执行到 n=2000),这时程序段 PP 也开始执行且抢占了处理机,执行结果中为 n=2000 和 n=10 000。

(4) 如果将两个程序放在另外一个相对执行速度较快的分时系统中执行,假设 CP 执行到 n=3000 时,其时间片到,这时程序段 PP 开始执行,其执行结果中有 n=3000 和 n=10 000。

这说明了,程序在并发执行时,有一些程序经过多次执行,虽然它们执行的初始条件相同,但其执行速度不同,结果也不同,即多道程序系统中,程序的执行不再具有可再现性,甚至会出现错误的结果。

3. 进程概念的引入

从上述讨论可以看出,在多道程序环境下,程序的执行属于并发执行,程序的执行具有了许多新的特性,程序与执行结果不再一一对应。而在多道程序系统中,一般情况下,并发执行的各程序段如果共享软、硬件资源,都会造成其执行结果受执行速度影响的局面(例 3-1 中的程序段并发执行出现不同的结果是由于两程序段共享变量 n)。显然,这是程序设计人员不希望看到的。为了使得在并发执行时不出现错误结果,必须采取某些措施来制约、控制各并发程序段的执行速度,这在操作系统程序设计中尤为重要。

为此,为了控制和协调各程序执行过程中的软、硬件资源的共享和竞争,在操作系统中引入了进程的概念来反映和刻画系统和用户程序的活动。

3.1.2 进程的定义

1. 进程的定义

进程的概念是 20 世纪 60 年代初期,首先在 MIT 的 Mulitics 系统和 IBM 的 TSS/360 系统中引入的。从那时以来,人们对进程下过各式各样的定义。现列举其中几种:

(1) 进程是可以并行执行的计算部分(S. E. Madnick, J. T. Donovan);

(2) 进程是一个独立的、可以调度的活动(E. Cohen, Jonfferson);

(3) 进程是一抽象实体,当它执行某个任务时,将要分配和释放各种资源(P. Denning);

(4) 行为的规则叫程序,程序在处理机上执行的活动称为进程(E. W. Dijkstra);

(5) 一个进程是一系列逐一执行的操作,而操作的确切含义则有赖于以何种详尽程度来描述进程(Brinch Hansen)。

以上进程的定义,尽管各有侧重,但本质是相同的,即主要注重进程是一个程序的执行过程这一概念。进程是程序的一次运行活动,即程序是一种静态的概念,而进程是一种动态的概念,它是“活动的”。

进程和程序之间的区别是很微妙的,但却非常重要。一个类比可以使我们更容易理解

这一点。想像一位有一手好厨艺的计算机科学家正在为他的女儿烘制生日蛋糕。他有做生日蛋糕的食谱，厨房里有所需的原料：面粉、鸡蛋、糖、香草汁等等。在这个比喻中，做蛋糕的食谱就是程序（即用适当形式描述的算法），计算机科学家就是处理机（CPU），而做蛋糕的各种原料就是输入的数据。进程就是厨师阅读食谱、取来各种原料以及烘制蛋糕的一系列动作的总和。

现在假设计算机科学家的女儿哭着跑了进来，说她被一只蜜蜂螫了。计算机科学家就记录下自己照着食谱做到哪儿了（保存进程的当前状态），然后拿出一本急救手册，按照其中的指示处理螫伤。这里，我们看到处理机从一个进程（做蛋糕）切换到另一个高优先级的进程（实施医疗救治），每个进程拥有各自的程序（食谱和急救指示）。当蜜蜂蜇伤处理完之后，计算机科学家又回来做蛋糕，从他离开时的那一步继续做下去（继续执行做蛋糕程序）。

这里的关键思想是：一个进程是某种类型的一个活动，它有程序、输入、输出及状态。单个处理机被若干进程共享，它使用某种调度算法决定何时停止一个进程的工作，并转而为另一个进程提供服务。

因此，在此给出进程的定义：一个具有独立功能的程序对某个数据集在处理机上的执行过程和分配资源的基本单位。

2. 进程的特征

在操作系统中，进程是进行系统资源分配、调度和管理的最小独立单位，操作系统的各种活动都与进程有关。为了进一步明确进程的概念，我们给出进程的一些突出特征。

（1）动态性

动态性是进程的最基本特征之一。进程是程序的一次运行过程，具有一个从静止到活动的过程，即从诞生、运行到消失的过程，有一定的生命的周期，它与程序并不一一对应。程序是静态的，只是指令的集合，作为一种文件可长期存放在存储装置中。一个进程可以对应一个程序，或者对应一段程序（一个程序的部分）。一个程序可以对应一个进程，也可以对应多个进程（此时称这个程序被多个进程所共享），可共享的程序代码被称为可重入代码或者纯代码，纯代码在运行过程中不能被改变。

（2）并发性

这是指多个进程同时驻留内存，且能在一段时间内交替运行。并发性是进程的最基本特征之一，同时也是操作系统的重要特征。引入进程的目的也正是为了使多个进程能并发运行，而程序是不能并发运行的。

（3）独立性

进程是操作系统中一个可以独立运行的基本单位，也是分配资源和进行调度的基本单位。进程在获得其必需的资源后即可运行，因不能得到某个资源时便停止运行。在具有并发活动的系统中，未建立进程的程序不能作为一个独立单位而运行。

（4）异步性

指进程的执行起始时间的随机性和执行速度的独立性。

（5）结构性

为了记录、描述、跟踪和控制进程的变化过程，系统建立了一套重要的数据结构，每个进程有其对应的数据结构。

3.1.3 引入进程的利弊

引入进程是多道程序和分时系统的需要，也是描述程序并发执行活动的需要。在操作系统中，通过为每道程序建立进程，使它们彼此间能够并发执行，从而提高系统资源的利用率和系统的吞吐量。因此，目前几乎所有的操作系统中都引入了进程的概念，支持多进程的操作。然而，引入进程后，同时也带来如下的问题。

1. 空间开销

由于系统必须为每个进程建立必需的数据结构和管理数据结构的机构，它们将占据一定的存储器空间。在主存容量较小的情况下，它自身也成为一个包袱，会影响主存空间使用率。如果主存空间足够大，空间影响就会降低。

2. 时间开销

系统为了管理和协调进程的运行，要不断跟踪进程的运行过程，不断地更新有关的系统和进程数据结构，进行进程间的运行工作切换、现场保护等。这些都需要占用处理机的时间，使系统付出时间上的开销。要减少时间开销，对操作系统设计和数据结构的选择，以及高速处理机的采用都有直接关系。

3.2 进程控制块和进程的状态

由前面的叙述可知，进程在执行过程中，具有“执行—暂停—执行”这种间断性的活动规律。那么，系统如何区别这些活动规律？显然，需要有一个专门的机制能够管理进程的这种变化过程。

3.2.1 进程的状态及其变化

一个进程在其生命周期内，由于各进程并发执行及相互制约，使得它们的状态不断发生变化。一般而言，进程具有以下三种最基本的状态：

（1）就绪状态

当进程已分配到除处理机以外的所有必要资源后，只要再获得处理机，便可立即执行，进程这时的状态称为就绪状态。

（2）运行状态

已经获得处理机及其他的运行资源，正在处理机上运行的进程处于运行状态。

（3）等待状态

正在执行的进程由于某种运行条件不具备而暂停执行时，在等待某一事件的发生，此时进程处于等待状态，有时也被称为阻塞状态。致使进程等待的典型事件有：请求 I/O、申请缓冲空间等。

在单处理机系统中，处于运行状态的进程只有一个。正在运行的进程如果因分配给它

的时间片到而被暂停运行时，该进程便由运行状态又回到就绪状态。处于就绪状态的进程可以有多个，它们都具备了运行的所有条件，仅仅未获得处理机控制权，如果有多个处理机，这些就绪进程都可以转入运行状态。如果需要等待某一事件的发生而使运行受阻（例如，进程请求访问某种独享资源，而该资源正在被其他进程占用，必须等到该资源被释放），该进程将由运行状态转变为等待状态。当引起等待的原因解除后，即回到就绪状态。图 3-1 给出了三个基本状态之间转换关系。

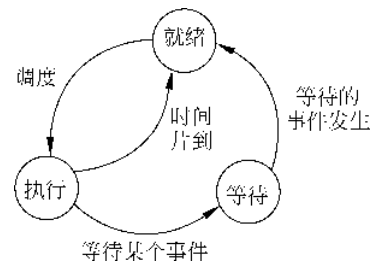

图 3-1　进程状态转换

进程状态转换发生变化的原因和条件归根到底是由于进程之间的相互制约关系引起的。对进程状态的转换过程，需要注意如下三点：

（1）进程从等待状态到运行状态，必须经过就绪状态，而不能直接转换到运行状态。这是因为此进程阻塞的原因被解除后，系统中可能有多个进程都处于可运行状态（就绪状态），因此系统必须按照一定的算法选择一个就绪进程占用处理机。这种选择过程被称为进程调度（scheduler）。

（2）一个进程由运行状态转变为等待状态一般是由运行进程自己主动提出的。例如，进程在运行过程中需要某一条件而不能满足时，就主动放弃处理机而使进程进入等待状态。

（3）一个进程由等待状态转变为就绪状态总是由外界事件引起的，而不是由该进程自己引起的。例如某一 I/O 操作完成，由 I/O 结束中断来解除等待此 I/O 完成的进程的等待状态，将其转换为就绪状态。

以上三种状态是进程最基本的状态，在实际的操作系统中往往不止这三种。进程的状态设置和规定与实际的操作系统设计有关。例如，还可以设置自由态、睡眠态、接收态、停止态等。对三种基本状态也可以再细分，例如，可分为静止就绪、活动就绪、静止等待、活动等待等。这些状态的设立和状态之间的转换均与系统进程的调度需要有关，根据操作系统的设计目标不同而不同。

3.2.2　进程控制块

1. 进程的静态描述

进程既然是一个动态的概念，那么如何表示一个进程，又如何知道进程的存在呢？显然在系统中需要有描述进程存在和能够反映其变化的物理实体，即进程的静态描述。进程的静态描述由三部分组成：

（1）程序：指进程运行所对应的执行代码。

（2）数据集合：指程序加工的对象和场所，是进程运行中必须的数据资源，包括对 CPU 占用，存储器、I/O 通道等的需求信息。

（1）和（2）两部分内容与进程的执行有关，在大多数操作系统中，把这两部分内容放在外存中，直到该进程执行时再调入内存。

（3）进程控制块（PCB）：系统为每个进程定义的一个数据结构。它包含了有关进程的

描述信息、控制信息和资源信息，是进程动态特征的集中反映。

2. 进程控制块的作用

为了描述和控制进程的运行，系统为每个进程定义了一个数据结构——进程控制块(process control block，PCB)，它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、用于描述进程当前情况以及进程控制运行的全部信息。进程控制块的作用是使一个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时，从该进程的 PCB 中查出其现行状态及优先级；在调度到某进程后，要根据 PCB 中所保存的处理机状态信息，设置该进程恢复运行的现场，并根据其 PCB 中程序和数据的内存始址，找到其程序和数据；进程在执行过程中，当需要和与之合作的进程实现同步、通信和访问文件时，也需要访问 PCB；当进程因某种原因而暂停执行时，又须将其断点的处理机环境保存在 PCB 中。可见，在进程的整个生命期中，系统总是通过 PCB 对进程进行控制的，亦即系统是根据进程的 PCB 而感知进程的存在的。所以说，PCB 是进程存在的唯一标志。

当系统创建一个新进程时，就为它建立了一个 PCB，进程结束时又收回其 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，如调度程序、资源分配程序、中断处理程序以及监督和分析程序等。因为 PCB 经常被系统访问，因此，几乎在所有的多道操作系统中，一个进程的 PCB 结构全部或部分常驻内存。

3. 进程控制块中的信息

一般来说，根据操作系统的要求不同，进程的 PCB 所包含的内容会有所不同。但是，下面所示的基本信息内容是必需的。

(1) 描述信息

① 进程标识符：在所有的操作系统中，创建一个进程时，系统都为每个进程赋予一个唯一的内部标识符，以便系统区别每个进程。

② 用户名或用户标识符：每个进程都隶属于某个用户，用户名或用户标识符有利于资源共享与保护。

③ 家族关系：在有的系统中，进程之间互成家族关系。PCB 中用相应的项描述其家族关系。

(2) 控制信息

① 进程当前状态：进程当前状态说明进程当前处于何种状态。

② 进程优先级：进程优先级是选取进程占用处理机的重要依据。与进程优先级有关的 PCB 表项有占用 CPU 时间、进程优先级偏移、占据内存时间等。

③ 程序开始地址：指出该进程程序以此内存地址开始执行。

④ 各种计时信息：给出进程占用和利用资源的有关情况。

⑤ 通信信息：记录进程在运行过程中与其他进程通信的有关信息。

(3) 资源管理信息

① 占用内存大小及其管理用数据结构指针，例如，后述内存管理中所用到的进程页表

指针等。

② 在某些复杂系统中，还有对换或覆盖用的有关信息，如对换程序段长度、对换外存地址等。这些信息在进程申请、释放内存时使用。

③ 共享程序段大小及起始地址。

④ 输入/输出设备的设备号，所要传送的数据长度、缓冲地址、缓冲长度及所用设备的有关数据结构指针等。这些信息在进程申请释放设备进行数据传输时使用。

⑤ 指向文件系统的指针及有关标识等。进程可使用这些信息对文件系统进行操作。

(4) CPU 现场保护机构

处理机状态信息主要是由处理机的各种寄存器中的内容组成的。当处理机被中断时，所有这些信息都必须保存在 PCB 中，以便在该进程重新执行时，能从断点继续执行。这些寄存器包括：

① 通用寄存器：又称为用户可视寄存器，它们是用户程序可以访问的，用于暂存信息。在大多数处理机中，有 8～32 个通用寄存器，在 RISC 结构的计算机中可超过 100 个。

② 指令计数器：其中存放了要访问的下一条指令地址。

③ 程序状态字 PSW：其中含有状态信息，如条件码、执行方式、中断屏蔽标志等。

④ 用户栈指针：指每个用户进程都有一个或若干与之相关的系统栈，用于存放过程和系统调用参数及调用地址。

例 3-2　一种用 C 语言描述的 PCB 结构。

```
struct pentry{
        int pid;              //进程标识符
        int pprio;            //进程优先级
        char pstate;          //进程状态
        int pname;            //进程用户标识符
        int msg;              //进程通信信息
        int paddr             //进程对应的程序执行地址
        int pregs[SIZE];      //现场保护区大小
        …
        }pcb[];               //定义进程控制块结构数组
```

可见，通过这些表项内容，标识了进程的存在和运行，集中反映了进程的动态特征，系统是通过 PCB 对进程进行管理和控制的。

4. PCB 的组织方式

在一个系统中，通常可拥有数十个，数百个乃至数千个 PCB。为了能对它们加以有效的管理，应该用适当的方式将这些 PCB 组织起来。目前常用的组织方式有两种。

(1) 链接表方式：处于相同状态的 PCB 组成队列，形成运行、就绪和阻塞队列。

每个 PCB 增加一个链指针表项，指向队列中下一个 PCB 的起始地址，系统中设置固定单元指出各单元的头，即第一个 PCB 的始址。运行队列实际上只有一个成员，用运行队列指针指向它即可。就绪队列的排队原则与调度策略有关。阻塞队列可多个，可根据阻塞原因的不同而把处于阻塞状态的进程的 PCB 排成等待 I/O 操作完成的队列和等待分配内存的队列等。图 3-2 给出了一种链接队列的组织方式。

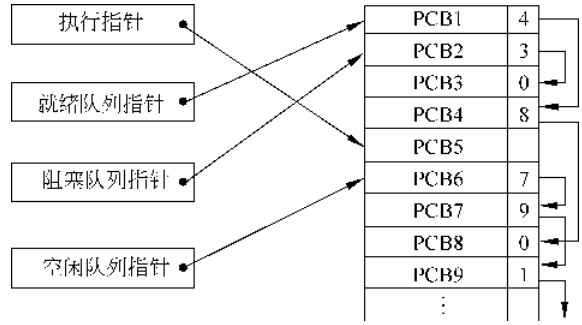

图 3-2　PCB 链接队列示意图

(2) 索引方式：系统根据所有进程的状态建立几张索引表，如就绪索引表、阻塞索引表等，并把各索引表在内存的首地址记录在内存的一些专用单元中。在每个索引表的表目中，记录具有相应状态的某个 PCB 在 PCB 表中的地址。图 3-3 给出了索引方式的 PCB 组织。

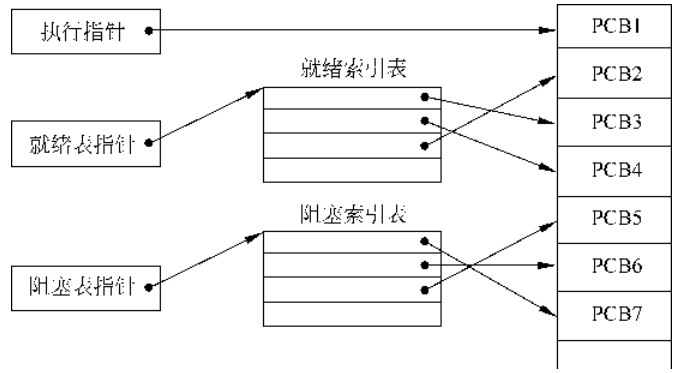

图 3-3　按索引方式组织 PCB

3.3　进程的控制

进程从它创建到其消失的整个生命周期中状态是不断发生变化的。那么，如何控制进程状态的变迁呢？如何创建和撤销一个进程呢？

操作系统的进程控制机构控制进程的状态变换。进程的控制机构首先表现在建立、撤销、阻塞、唤醒等方面。通常操作系统内核提供了称作原语的具有特定功能的程序段来完成进程的建立、撤销以及完成进程各状态间的转换。

原语被认为是机器语言的延伸，是在系统核心态下执行的、由一条或若干条机器指令组成的、具有特定功能的程序段。它一旦被启动，在执行期间是不可中断的。操作系统原语对用户是透明的，一般不允许用户直接使用，以避免对操作系统内核的干扰和破坏。但随着系统的发展，为系统程序员的方便使用，有的原语被作为一种特殊的系统调用，既提供给系统进程，也提供给用户进程，通过系统调用方式使用。

3.3.1　进程的创建原语

1. 进程图

一个进程可以创建若干个新进程，新创建的进程又可以继续创建进程，这个创建过程形成了一种树型结构，如图 3-4 所示，操作系统中称为进程图(process graph)。进程图是一棵有向树，其结点代表进程，分枝代表创建。若进程 A 创建了进程 B，称 A 是 B 的父进程(parent process)，而 B 称为 A 的子进程(progeny process)。进程树形成了一个进程“家族”，根结点为该家族“祖先”(ancestor)。必须注意，在进程图中，进程 A 创建了进程 B，但并不意味着只有进程 A 执行完进程 B 才能执行，而是 A 和 B 可以并发执行。

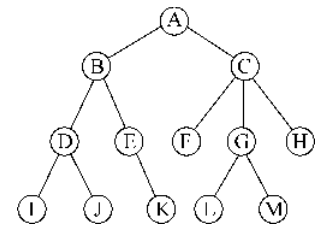

图 3-4　进程树

了解进程间的这种关系是十分重要的。因为子进程可以继承父进程所拥有的资源。例如，继承父进程打开的文件，继承父进程所分配到的缓冲区等。当子进程被撤销时，应将其从父进程那里获得的资源归还给父进程。此外，在撤销父进程时，也必须同时撤销其所有的子进程。为了标识进程之间的家族关系，在 PCB 中设置了家族关系表项，以标明自己的父进程及所有的子进程。

2. 引起创建进程的事件

在多道程序环境中，程序只有作为进程时才能在系统中运行。因此，为使程序能运行，就必须为它创建进程。导致一个进程去创建另一个进程的典型事件，可有以下 4 类。

(1) 用户登录。在分时系统中，用户在终端输入登录命令后，如果是合法用户，系统将为该终端建立一个终端进程，并把它插入就绪队列中。

(2) 作业调度。在批处理系统中，当作业调度程序按一定的算法调度到某作业时，便将该作业装入内存，为它分配必要的资源，并立即为它创建主进程，再插入就绪队列中。

(3) 提供服务。当运行中的用户程序提出某种请求后，系统将专门创建一个进程来提供用户所需要的服务。例如，用户程序要求进行文件打印，操作系统将为它创建一个打印进程，这样，不仅可使打印进程与该用户进程并发执行，而且还便于计算出为完成打印任务所花费的时间。

(4) 应用请求。在上述三种情况下，都是由系统内核为其创建一个新进程，而第 4 类事件则是基于应用进程的需求，由它自己创建一个新进程，以便使新进程以并发运行方式完成特定任务。例如，某应用程序需要不断地从键盘终端读入输入数据，继而又要对输入数据进行相应的处理，然后，再将处理结果以表格形式在屏幕上显示。该应用进程为使这几个操作能并发执行，以加速任务的完成，可以分别建立键盘输入进程、数据处理进程和表格输出进程。

3. 进程创建原语

一旦操作系统发现了要求创建新进程的事件后，便调用进程创建原语(creation)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,并从 PCB 集合中索取一个空白 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。显然,此时操作系统必须知道新进程所需内存的大小。对于批处理作业,其大小可在用户提出创建进程要求时提供;若是为应用进程创建子进程,也应是在该进程提出创建进程的请求中给出所需内存的大小;对于交互型作业,用户可以不给出内存要求而由系统分配一定的空间;如果新进程要共享某个已在内存的地址空间(即已装入内存的共享段),则必须建立相应的链接。

(3) 初始化进程控制块。PCB 的初始化包括:①初始化标识信息,将系统分配的标识符和父进程标识符,填入新 PCB 中;②初始化处理机状态信息,使程序计数器指向程序的入口地址,使栈指针指向栈顶;③初始化处理机控制信息,将进程的状态设置为就绪状态或静止就绪状态,对于优先级,通常是将它设置为最低优先级,除非用户以显式方式提出高优先级要求。

(4) 将新进程插入就绪队列,如果进程就绪队列能够接纳新进程,便将新进程插入就绪队列。

3.3.2 进程的撤销原语

1. 引起进程撤销的事件

(1) 正常结束

在任何计算机系统中,都应有一个用于表示进程已经运行完成的指示。例如,在批处理系统中,通常在程序的最后安排一条 Holt 指令来终止程序的执行。当程序运行到 Holt 指令时,将产生一个中断,去通知操作系统本进程已经完成。在分时系统中,用户可利用 Logs off 去表示进程运行完毕,此时同样可产生一个中断,去通知操作系统进程已运行完毕。

(2) 异常结束

在进程运行期间,由于出现某些错误和故障而迫使进程终止。这类异常事件很多,常见的有:①越界错误。这是指程序所访问的存储区,已越出该进程的区域。②保护错。进程试图去访问一个不允许访问的资源或文件,或者以不适当的方式进行访问,例如,进程试图去写一个只读文件。③非法指令。程序试图去执行一条不存在的指令。出现该错误的原因,可能是程序错误地转移到数据区,把数据当成了指令。④特权指令错。用户进程试图去执行一条只允许操作系统执行的指令。⑤运行超时。进程的执行时间超过了指定的最大值。⑥等待超时。进程等待某事件的时间,超过了规定的最大值。⑦算术运算错。进程试图去执行一个被禁止的运算,例如,被 0 除。⑧I/O 故障。这是指在 I/O 过程中发生了错误等。

(3) 外界干预

外界干预并非指在进程运行中出现了异常事件,而是指进程应外界的请求而终止运行。这些干预有:①操作员或操作系统干预。由于某种原因,例如,发生了死锁,由操作员或操