第 1 章 绪　　论

现代计算机系统中，都无一例外地配置了操作系统。操作系统是伴随着计算机系统的发展，逐步形成和完善起来的，经历了一个从无到有、从简单到复杂的过程。操作系统已经成为现代计算机系统中不可缺少的系统软件，是其他所有系统软件和应用软件的运行基础，负责控制和管理整个计算机系统中的软硬件资源，并为用户使用计算机提供一个方便灵活、安全可靠的工作环境。

本章将从操作系统的概念、发展、类型、功能、特征、运行环境及操作系统安全等方面进行介绍，使读者对操作系统有个大致的认识。

1.1 操作系统的概念

一个完整的计算机系统由两大部分组成：计算机硬件和计算机软件。硬件是所有软件运行的物质基础，软件能充分发挥硬件潜能和扩充硬件功能，完成各种系统及应用任务。两者互相促进，相辅相成，缺一不可。

计算机硬件是指计算机系统中由电子、机械、电气、光学和磁学等元器件构成的各种部件和设备，这些部件和设备依据计算机系统结构的要求组成一个有机整体。计算机软件是指由计算机硬件执行以完成一定任务的程序和数据。计算机软件包括系统软件和应用软件两部分，系统软件包括操作系统、编译程序、编辑程序、数据库管理系统等；应用软件是为各种应用目的而编制的程序。

1.1.1 操作系统的地位和作用

1)操作系统的地位

没有配置软件的计算机称为裸机，它仅仅由硬件构成，而实际呈现在用户面前的计算机

系统是经过若干层软件改造的计算机。图 1-1 显示了一个计算机系统的软硬件层次结构。其中，每一层具有一组功能并提供相应的接口，接口对层内掩盖了实现的细节，对层外则提供了使用约定。

硬件层提供了基本的可计算性资源，包括处理器、寄存器、存储器，以及各种 I/O 设施和设备，是操作系统和上层软件赖以工作的基础。

操作系统是裸机上的第一层软件，是对硬件功能的首次扩充，主要完成资源的调度和分配、信息的存取和保护、并发活动的协调和控制等许多工作。操作系统是上层其他软件运行的基础，为编译程序和数据库管理系统等系统程序的设计者提供了有力支撑。

系统程序层的工作基础建立在操作系统改造和扩充过的机器上，利用操作系统提供的扩展指令集，可以较为容易地实现各种语言处理程序、数据库管理系统和其他系统程序。此外，还提供种类繁多的实用程序，如连接装配程序、库管理程序、诊断排错程序、分类/合并程序等供用户使用。

应用程序层解决用户特定的或不同应用需要的问题，应用程序开发者借助于程序设计语言来表达应用问题，开发各种应用程序，既快捷又方便。而最终用户则通过应用程序与计算机系统交互来解决其应用问题。

图 1-1　计算机系统的层次关系

从图 1-1 中可以看出，计算机的硬件和软件以及软件的各部分之间形成了一种层次结构的关系。裸机在最下层，它的上面是操作系统，经过操作系统提供的资源管理功能和各种服务功能把裸机改造成为功能更强、使用更方便的机器，通常将裸机之上覆盖了软件的机器称为虚拟机或扩展机，各种实用程序和应用程序便运行在操作系统之上，它们以操作系统为支撑环境，同时向用户提供完成其工作所需的各种服务。

2)操作系统的作用

操作系统的作用主要体现在以下 3 个方面。

(1)提供了用户与计算机硬件之间的接口。操作系统是对计算机硬件系统的第一次扩充，用户通过操作系统来使用计算机系统。经过操作系统改造和扩充过的计算机不但功能更强，使用也更为方便，用户可以直接调用操作系统提供的各种功能，而无须了解诸多软硬件本身的细节，对于用户来讲，操作系统就是用户与计算机硬件之间的一个接口。

(2)为用户提供了虚拟机。人们很早就认识到必须找到某种方法把硬件的复杂性与用户隔离开来。经过不断地探索和研究，目前采用的方法是在计算机裸机上加上一层又一层的软件来组成整个计算机系统，同时，为用户提供一个容易理解和便于程序设计的接口。每当在计算机上覆盖了一层软件，系统的功能便增加一点，使用就更加方便一点，用户可用的运行环境就更加好一点。因此，当计算机上覆盖了操作系统后，可以扩展基本功能，为用户

提供一台功能显著增强、使用更加方便、安全可靠性好、效率明显提高的机器，就好像可以使用的是一台与裸机不同的虚拟计算机(virtual machine)。

(3)充当计算机系统的资源管理者。在计算机系统中，能分配给用户使用的各种硬件和软件设施统称为资源。操作系统的重要任务之一是对资源进行抽象研究，找出各种资源的共性和个性，有序地管理计算机中的硬件、软件资源，跟踪资源使用情况，监视资源的状态，满足用户对资源的需求，协调各程序对资源的使用冲突，研究使用资源的统一方法，为用户提供简单、有效的资源使用手段，最大限度地实现各类资源的共享，提高资源利用率，从而使计算机系统的效率大大提高。

引入操作系统的目的是：提供一个计算机用户与计算机硬件系统之间的接口，使计算机系统更易于使用；有效地控制和管理计算机系统中的各种硬件和软件资源，使之得到更有效的利用；合理地组织计算机系统的工作流程，以改善系统性能。

在操作计算机的过程中，操作系统提供了正确使用资源的方法。可通过用户和系统的观点来研究操作系统。

1.1.2 用户观点

操作系统的用户观点根据所使用计算机种类的不同而异。绝大多数用户使用的是个人计算机(PC)，个人计算机有显示器、键盘、鼠标和主机。这类计算机操作系统的设计是为了让用户单独使用其资源，优化其所进行的工作。对于这种情况，操作系统的设计目的主要是方便用户使用，性能反而是次要的。

有些用户使用与大型机或小型机相连的终端，用户通过终端访问同一计算机。这些用户共享资源并可交换信息。这类计算机操作系统的设计目的是使资源利用率最大化。

有些用户使用工作站，工作站与其他工作站和服务器相连。这些用户不仅可以使用专用资源，而且还可以使用共享资源。这类操作系统的设计是个人可用性和资源利用率的折中。

近年来，出现了许多类型的手持计算机，它们大多为单个用户所独立使用。有的也通过有线或无线与网络相连。由于受电源和接口限制，它们只能执行相对少的远程操作。这类计算机操作系统的设计目的主要是个人可用性和电源管理。

有的计算机几乎没有或根本没有操作系统用户观点，如家电和汽车中所使用的嵌入式计算机，这些设备及其操作系统通常设计成无须用户干预就能执行。

1.1.3 系统观点

从计算机的角度看，操作系统是与计算机硬件最为密切的程序。可以将操作系统看做资源分配器，它是资源管理者。

在计算机系统中有两类资源：硬件资源和软件资源。按作用又可以将资源分为4类：处理机、存储器、外部设备和文件(程序和数据)。这些资源构成了操作系统本身以及用户作业赖以执行的物质基础和工作环境。资源的使用方法和管理策略决定了整个操作系统的规模、类型、功能和实现。例如，面对许多甚至冲突的资源请求，操作系统必须决定如何为各个程序和用户分配资源，以便计算机能公平有效地运行。

1.2 操作系统的发展

要想更清楚地理解操作系统的实质，了解操作系统的发展历史很有必要，因为操作系统的许多基本概念都是在操作系统的发展过程中出现并逐步得到发展和完善的。

1.2.1 手工操作阶段

从1946年诞生第一台计算机起到20世纪50年代末，计算机处于第一代。此时构成计算机的主要元器件是电子管，计算机运算速度慢（只有几千次/秒），硬件价格昂贵，没有操作系统，甚至没有任何软件，人们采用手工操作方式操作计算机。在手工操作方式下，用户一个接一个地轮流使用计算机，每个用户的使用过程大致为：先将程序纸带（或卡片）装到输入机上，然后启动输入机把程序和数据送入计算机，接着通过控制台开关启动程序运行，当程序运行结束时，由用户取走纸带和计算结果。

从上述操作过程中可以看出，程序运行期间计算机系统中的所有资源为一个用户独占，并且在程序运行过程中需要人工干预，以完成装卸纸带、拨动开关等操作。由此可见，手工操作方式具有用户独占计算机资源、资源利用率低及CPU等待人工操作的特点。

随着CPU运行速度的大幅提高，人工操作的低速与CPU运行的高速之间出现了矛盾，这就是所谓的人机矛盾。例如，一个用户程序在速度为1万次/秒的计算机上运行需要1小时，人工操作时间需要3分钟，这种情况下操作时间和运行时间的比为1∶20；若机器运行速度提高到60万次/秒，则该用户程序的运行时间降低为1分钟，而人工操作的速度不会有多大的提高，仍假定为3分钟，此时人工操作时间和运行时间的比为3∶1。这就是说，人工操作时间远远超过了机器运行时间。由此可见，缩短人工操作时间就显得非常必要了。另一方面，CPU与I/O设备之间速度不匹配的矛盾也日益突出。为了缓和这些矛盾，引入了批处理技术及脱机输入/输出技术。

1.2.2 早期的批处理系统

为了解决程序运行过程中的人工干预问题，需要缩短建立作业和人工操作的时间，为此人们提出了从一个作业到下一个作业的自动过渡方式，从而出现了批处理技术。完成作业自动过渡的程序称为监督程序，监督程序是一个常驻内存的程序，它管理作业的运行，负责装入和运行各种系统程序来完成作业的自动过渡。监督程序是最早的操作系统雏形。

批处理技术是指计算机系统对一批作业自动进行处理的一种技术。早期的批处理分为联机批处理和脱机批处理两种类型。

1）联机批处理

在早期的联机批处理系统中，操作员把用户提交的若干作业集中成一批，由监督程序先

把它们输入磁带上，当该批作业输入完成之后，监督程序就开始执行。它自动把磁带上该批作业的第一个作业调入内存，并对该用户程序进行汇编或编译，然后由装配程序把汇编或编译结果装入内存，再启动执行。计算机完成该作业的全部计算或处理后，输出计算或处理结果。第一个作业完成之后，监督程序又自动调入该批作业中的第二个作业，并重复上述执行过程，一直到该批作业全部完成为止。在完成了一批作业之后，监督程序又控制输入另一批作业到磁带上，并按上述步骤重复处理。

2)脱机批处理

在联机批处理中，作业的输入/输出都是联机的。也就是说，作业信息先送到磁带，再由磁带调入内存，并将计算结果在打印机上输出，这些都是由 CPU 来处理的，这种联机输入/输出的缺点是速度慢，并且随着 CPU 运行速度的提高，高速的 CPU 与低速的 I/O 设备之间的矛盾更加突出。为此，在批处理技术中引进了脱机输入/输出技术。

在脱机批处理系统中，除主机之外另设了一台外围机(又称卫星机)，该机只与外部设备打交道，不与主机直接连接，如图 1-2 所示。用户作业通过外围机输入磁带上，而主机只负责从磁带上把作业调入内存，并予以执行。作业完成后，主机负责把结果输出到磁带上，然后再由外围机将磁带上的信息在打印机上输出。

脱机输入是指将用户程序和数据在外围机的控制下，预先从低速输入设备输入磁带上，当 CPU 需要这些程序和数据时，再直接从磁带机高速输入内存。脱机输出是指当程序运行完毕或告一段落，CPU 需要输出时，无须直接把计算结果送至低速输出设备，而是高速地把结果送到磁带上，然后在外围机的控制下，由相应的输出设备输出磁带上的计算结果。

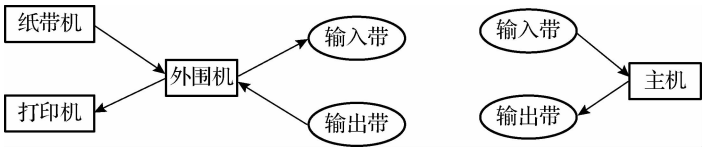

图 1-2　脱机输入/输出方式

若输入/输出操作在主机控制下进行，则称之为联机输入/输出。

脱机批处理系统中采用脱机输入/输出技术后，低速 I/O 设备上数据的输入/输出都在外围机的控制下进行，而 CPU 只与高速的磁带机打交道，从而有效地减少了 CPU 等待低速设备输入/输出的时间。

1.2.3　多道程序设计技术

在早期的批处理系统中，内存中仅有一道用户程序运行，这种程序运行方式称为单道程序运行方式。图 1-3 给出了单道程序运行方式的工作情况。

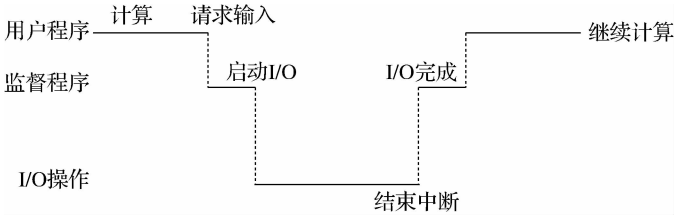

图 1-3　单道程序运行方式

从图 1-3 中可以看出，每当程序发出 I/O 请求时，CPU 便处于等待 I/O 完成的状态，致使 CPU 空闲。为进一步提高 CPU 的利用率，引入了多道程序设计技术。

多道程序设计的基本思想是在内存中同时存放多道程序，这些程序在管理程序的控制下交替运行，共享处理机及系统中的其他资源。现代计算机系统一般都基于多道程序设计技术。图 1-4 给出了多道程序运行的工作情况。

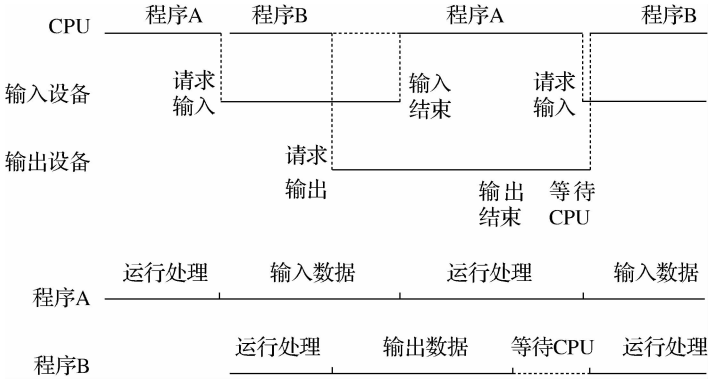

图 1-4　多道程序运行方式

图 1-4 中的计算机系统中有 A、B 两道程序运行，它们的运行过程如下。

(1)程序 A 先在 CPU 上运行，当程序 A 请求输入时，程序 A 停止运行；系统管理程序启动设备进行输入工作，并将 CPU 分配给程序 B。此时，程序 A 利用输入设备进行输入，而程序 B 正在 CPU 上执行。

(2)当程序 B 请求输出时，程序 B 停止运行；系统管理程序启动输出设备进行输出工作。此时，输入设备和输出设备都在工作，而 CPU 处于空闲状态。

(3)当程序 A 请求的输入工作完成时，向系统发出 I/O 结束中断，系统管理程序进行中断处理，然后调度程序 A 运行。此时，程序 A 在 CPU 上执行，程序 B 利用输出设备进行输出。

(4)当程序 B 请求的输出工作完成时，向系统发出 I/O 结束中断，系统管理程序进行中断处理，由于此时程序 A 在 CPU 上运行，因此程序 B 处于等待 CPU 的状态。

(5)当程序 A 再次请求输入时，程序 A 让出 CPU；系统管理程序启动设备进行输入工作，并再次调度程序 B 运行……

从 A、B 程序的执行过程中可以看出，两个程序可以交替运行，若安排合适就会使 CPU 保持忙碌状态，而 I/O 设备也可满负荷工作。与单道程序运行方式相比，两道程序运行时系

统资源利用率提高了,在一段给定的时间内计算机所能完成的总工作量也增加了。

综上所述,在单处理机计算机系统中多道程序运行的特点如下。

(1)多道。计算机内存中同时存放多道相互独立的程序。

(2)宏观上并行。同时进入系统的多道程序都处于运行过程中,即它们先后开始了各自的运行,但都未运行完毕。

(3)微观上串行。内存中的多道程序轮流占有CPU,交替执行。

多道程序设计技术能有效提高系统的吞吐量和改善资源利用率,但实现多道程序系统时,由于内存中同时存在多道作业,因此还需要妥善解决以下一系列问题。

(1)处理机管理问题。要解决如何在多道程序之间分配处理机,以使处理机既能满足各程序运行的需要又有较高的利用率以及将处理机分配给某程序后,应何时收回等问题。

(2)存储器管理问题。要解决如何为每道程序分配必要的内存空间,以使它们获得各自需要的存储空间却又不致因相互重叠而丢失信息,以及如何防止因某道程序出现异常而破坏其他程序等问题。

(3)设备管理问题。多道程序共享系统中的多类I/O设备,要解决如何分配这些I/O设备,如何做到既方便用户使用设备,又能提高设备的利用率等问题。

(4)文件管理问题。现代计算机系统通常都存放有大量文件,要解决如何组织这些文件才能既方便用户使用又能保证文件的安全性和一致性等问题。

多道程序设计的缺点是延长了作业的周转时间。从表面上看,似乎道数越多越能提高效率,但由于系统的开销和用户的要求,程序的道数不是任意增加的,它往往因系统的资源以及用户的要求而定。

1.2.4 操作系统的进一步发展

批处理系统缺少人机交互能力,因此用户使用不方便。为了解决这个问题,人们开发出分时系统。在分时系统中,一台主机可以连接几台乃至上百台终端,每个用户可以通过终端与主机交互,方便地编辑和调试自己的程序,向系统发出各种控制命令,请求完成某项工作;系统完成用户提出的要求,输出计算结果及出错、警告或提示等必要的信息。

为了满足某些应用领域对实时处理的需求,人们又开发出了实时系统。实时系统具有专用性,不同的实时系统用于不同的应用领域。

大约到20世纪60年代中期以后,随着磁盘的问世,相继出现了多道批处理操作系统和分时操作系统、实时操作系统,标志着操作系统正式形成。

操作系统是一组控制和管理计算机硬件和软件资源,合理地组织计算机工作流程,以及方便用户使用的程序的集合。

值得说明的是,操作系统是一个系统软件,它由一组程序组成;操作系统的基本职能是控制和管理计算机系统内的各种资源,有效地组织多道程序的运行;同时操作系统还提供众多服务功能,以方便用户使用计算机,并扩充硬件功能。

近些年来,又开发出了个人计算机操作系统、网络操作系统、分布式操作系统和嵌入式操作系统等。随着硬件技术的飞速发展和应用领域的急剧扩大,操作系统不仅种类越来越多,而且功能更加强大,给用户提供了更为舒适的应用环境。

从操作系统形成至今的40多年间，其性能、规模、应用等方面都取得了飞速发展。推动操作系统发展的因素很多，但主要可归结为以下几个方面。

1)硬件技术更新

从电子管到晶体管、集成电路、大规模集成电路，直至当今的超大规模集成电路，计算机系统的性能得到快速提高，每隔18个月其性能就要翻一番。硬件技术的快速发展，也促使操作系统的性能和结构有了显著提高，如内存管理支撑硬件由分页或分段设施代替了界寄存器以后，操作系统中便增加了分页或分段存储管理功能；图形终端代替逐行显示终端后，操作系统中增加了窗口管理功能，允许用户通过多个窗口在同一时间提出多个操作请求；引进了中断和通道等设施后，操作系统中引入了多道程序设计功能。此外，硬件成本的下降也极大地推动了计算机技术的应用推广和普及。

2)计算机体系结构的发展

计算机体系结构的发展，从单处理器系统到多处理器系统，从指令串行结构到流水线结构、超级标量结构，从单总线到多总线应用等，这些发展有力地推动了操作系统的更大发展，如从单CPU操作系统发展到对称多处理器系统(SMP)，从主机系统发展到个人机系统，从单独自治系统到网络操作系统以及分布式操作系统等。

3)应用需求扩大

应用需求的扩大促进了计算机技术的发展，也促进了操作系统的不断更新升级。为了充分利用计算机系统内的各种宝贵资源，形成了早期的批处理系统；为了方便多个用户同时上机、实现友好的人机交互，形成了分时系统；为了实时地对特定任务进行可靠的处理，形成了实时系统；为了实现远程信息交换和资源共享，形成了网络操作系统及分布式操作系统等。在当今信息时代，信息处理离不开计算机，也就离不开操作系统这个软件平台。可以预见操作系统将会以更快的速度更新换代。

1.3 操作系统的基本类型

根据操作系统具备的功能、特征、规模和所提供应用环境等方面的差异，可以将操作系统划分为不同的类型。操作系统有3种基本类型，即批处理操作系统、分时操作系统和实时操作系统，分别简称为批处理系统、分时系统和实时系统。后来随着计算机体系结构的发展及应用需求的扩大，又出现了许多种新型的操作系统，主要有嵌入式操作系统、个人计算机操作系统、多处理机操作系统、网络操作系统和分布式操作系统。这些系统各有特点，适用于不同的应用领域。

1.3.1 批处理系统

描述任何一种操作系统都要用到作业的概念。所谓作业，就是用户在一次解题或一个事务处理过程中要求计算机系统所做工作的集合，包括用户程序、所需的数据及命令等。

单道批处理系统是早期计算机系统中配置的一种操作系统，其工作流程大致为：用户将作业交给系统操作员，系统操作员将多个用户作业组成一批输入并传送到外存储器；然后批处理系统按一定的原则选择其中的一个作业调入内存并使之运行；作业运行完成或出现错误而无法再进行下去时，由系统输出有关信息并调入下一个作业运行。重复上述过程，直至这批作业全部处理完为止。

单道批处理系统大大减少了人工操作的时间，提高了机器的利用率。但对于某些作业来说，当它发出输入/输出请求后，CPU 必须等待 I/O 的完成，这就意味着 CPU 空闲，特别是当 I/O 设备的速度较慢时，将导致 CPU 的利用率很低。为了提高 CPU 的利用率，引入了多道程序设计技术。

在单道批处理系统中引入多道程序设计技术就形成了多道批处理系统。在多道批处理系统中，不仅在内存中可以同时有多道作业运行，而且作业可随时（不一定集中成批）被接受进入系统，并存放在外存中形成作业队列，然后由操作系统按一定的原则从作业队列中调度一个或多个作业进入内存运行。多道批处理系统一般用于计算中心的大型计算机系统中。

多道批处理系统的主要特征如下。

（1）用户脱机使用计算机。用户提交作业之后直到获得结果之前几乎不再和计算机打交道。

（2）成批处理。操作员将各用户提交的作业组织成一批进行处理，由操作系统负责每批作业间的自动调度。

（3）多道程序运行。按多道程序设计的调度原则，从一批后备作业中选取多道作业调入内存并组织它们运行。

由于多道批处理系统中的资源为多个作业所共享，操作系统实现一批作业的自动调度执行，且运行过程中用户不能干预自己的作业，从而使多道批处理系统具有系统资源利用率高和作业吞吐量大的优点。多道批处理系统的不足之处是无交互性，即用户提交作业后就失去了对作业运行的控制能力，这使用户感觉不方便。

1.3.2 分时系统

在批处理系统中，用户以脱机操作方式使用计算机，用户在提交作业以后就完全脱离了自己的作业，在作业运行过程中，不管出现什么情况都不能加以干预，只能等待该批作业处理结束，用户才能得到计算结果，根据计算结果再作下一步处理，若作业运行出错，还得重复上述过程。这种操作方式对用户而言极不方便，人们希望能以联机方式使用计算机，这种需求导致了分时系统的产生。

在操作系统中采用分时技术就形成了分时系统。所谓分时技术，就是把处理机的运行时间分成很短的时间片，按时间片轮流把处理机分配给各联机作业使用。若某个作业在分配给它的时间片内不能完成计算，则该作业暂时停止运行，把处理机让给另一个作业使用，等待下一轮时再继续运行。由于计算机速度很快，作业运行轮转得也很快，给每个用户的感觉是好像自己独占一台计算机。

在分时系统中，一台计算机和许多终端设备连接，每个用户可以通过终端向系统发出命令，请求完成某项工作，而系统则分析从终端设备发来的命令，完成用户提出的要求，然后用

户再根据系统提供的运行结果，向系统提出下一步请求，这样重复上述交互会话过程，直到用户完成预计的全部工作为止。

分时系统也是支持多道程序设计的系统，但它不同于多道批处理系统。多道批处理系统是实现作业自动控制而无须人工干预的系统，而分时系统是实现人机交互的系统，这使得分时系统具有与批处理系统不同的特征，其主要特征如下。

(1)同时性。也称多路性，是指允许多个终端用户同时使用一台计算机，即一台计算机与若干终端相连接，终端上的这些用户可以同时或基本同时使用该计算机。

(2)交互性。用户能够方便地与系统进行人—机对话，即用户通过终端采用人—机对话的方式直接控制程序运行，与程序进行交互。

(3)独立性。系统中各用户可以彼此独立地进行操作，互不干扰，即各用户都感觉不到别人也在使用这台计算机，好像只有自己单独使用这台计算机一样。

(4)及时性。用户请求能在很短时间内获得响应。分时系统采用时间片轮转方式使一台计算机同时为多个终端用户服务，通常能够在 2～3 s 内响应各用户的请求，使用户对系统的及时响应感到满意。

1.3.3 实时系统

在计算机的某些应用领域内，要求对实时采样数据进行及时处理，作出相应的反应，如果超出限定的时间就可能丢失信息或影响到下一批信息的处理。例如，生产控制过程中，必须及时对出现的各种情况进行分析和处理，这种系统是专用的，它对实时响应的要求是批处理系统和分时系统无法满足的。于是，人们引入了实时系统。

实时系统能及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，并控制所有实时设备和实时任务协调一致地工作。实时系统对响应时间的要求比其他操作系统更高，一般要求秒级、毫秒级甚至微秒级的响应时间，对响应时间的具体要求由被控制对象决定。

实时系统有两类典型的应用形式，即实时控制系统和实时信息处理系统。

(1)实时控制系统。实时控制系统是指以计算机为中心的生产过程控制系统，又称计算机控制系统。在实时控制系统中，要求计算机实时采集现场数据，并对它们进行及时的处理，进而自动控制相应的执行机构，使某参数(如温度、压力、流量等)能按预定规律变化或保持不变，以达到保证产品质量、提高产量的目的。例如，钢铁冶炼的自动控制、炼油生产过程的自动控制、飞机飞行过程中的自动控制等。

(2)实时信息处理系统。在实时信息处理系统中，计算机及时接收从远程终端发来的服务请求，根据用户提出的问题对信息进行检索和处理，并在很短时间内对用户作出正确的响应。航空订票系统、情报检索系统等，都属于实时信息处理系统。

实时系统的主要特征是响应及时和可靠性高。系统必须保证对实时信息分析和处理的速度要快，而且系统本身要安全可靠，因为像生产过程的实时控制系统、航空订票系统等实时信息处理系统，信息处理的延误或丢失往往会带来巨大的经济损失，甚至可能引发灾难性的后果。

实时系统与分时系统的区别如下。

(1)许多实时系统是专用系统,而批处理系统与分时系统通常是通用系统。

(2)实时系统用于控制实时过程,要求对外部事件的响应迅速,具有较强的中断处理机构。

(3)实时系统用于控制重要过程,要求高度可靠,具有较高冗余。

(4)实时系统的工作方式是接受外部消息,分析消息,调用相应处理程序进行处理。

批处理系统、分时系统和实时系统是3种基本的操作系统类型。一个实际的操作系统往往兼有批处理、分时和实时系统三者或其中两者的功能。

1.3.4 其他操作系统类型

1)嵌入式操作系统

现今,手持移动设备、智能机械、信息家电等发展迅速,日益普及。这些设备中都嵌入了各种微处理器或控制芯片,因此必然需要相应的系统软件来管理。

对整个智能芯片以及它所控制的各种部件模块等资源进行统一调度、指挥和控制的系统软件称为嵌入式操作系统。嵌入式操作系统具有高可靠性、实时性、占有资源少、成本低等优点,其系统功能可针对需求进行裁减、调整和编译生成,以满足最终产品的设计要求。

嵌入式操作系统支持嵌入式软件的运行,由于手持移动设备、智能机械、信息家电等面向普通家庭和个人用户,其应用市场比传统的计算机市场大很多,所以嵌入式软件可能成为21世纪信息产业的支柱之一,嵌入式操作系统也必将成为软件厂商争夺的焦点,成为操作系统发展的另一个热门方向。

2)个人计算机操作系统

20世纪80年代个人计算机开始出现,它逐渐进入家庭、办公室、车间、仓库等各种可能的应用场所,成功地渗透到人们的事务处理、办公、学习、生活、娱乐等领域中。个人计算机的出现使计算机技术得到了极大的普及,改变了人们的工作、学习和生活方式。

个人计算机操作系统主要供个人使用,它功能强大,价格便宜,几乎可以在任何地方安装,能满足一般人工作、学习、游戏等方面的需求。由于个人计算机操作系统主要是个人使用,因此在处理机调度、存储保护方面比其他类型的操作系统简单得多。它的主要特点是计算机在某一段时间内为单个用户服务,采用图形界面人机交互的工作方式,界面友好,使用方便。

3)网络操作系统

信息时代离不开计算机网络,特别是互联网的广泛应用正在改变着人们的观念和社会生活的方方面面。每天有成千上万人通过网络传递邮件、查阅资料、搜寻信息、网上订票、网上购物等。

虽然个人计算机大大推动了计算机的普及,但单台计算机的资源毕竟有限,为了实现计算机之间的数据通信和资源共享,可将分布在各处的计算机和终端设备通过数据通信系统连接在一起,构成计算机网络。

计算机网络是通过通信设施将物理上分散的具有自治功能的多个计算机系统互连起来,按照网络协议交换数据、实现信息交换及资源共享的系统。

计算机网络具有以下特点。

(1)分布性。计算机网络是一个互连的计算机系统群体。这些计算机系统在物理上是分散的,它们可以在一个房间、一个单位、一个城市或几个城市,甚至是全国或全球范围内。

(2)自治性。网络上的每台计算机都有自己的内存、I/O 设备和操作系统等,各自独立完成自己承担的工作。网络系统中的各资源之间多是松散耦合的,不具备整个系统统一任务调度的功能。

(3)互连性。利用通信设施将不同地点的资源(包括硬件资源和软件资源)连接在一起,在网络操作系统的控制下,实现网络通信和资源共享。

(4)可见性。计算机网络中的资源对用户是可见的。用户任务通常在本地计算机上运行,利用网络操作系统提供的服务可共享其他主机上的资源。

网络操作系统是基于计算机网络的,是在各种计算机操作系统上按网络体系结构协议标准开发的软件,包括网络管理、通信、资源共享、系统安全和各种网络应用服务,其目标是互相通信和资源共享。

4)分布式操作系统

分布式系统是指多个分散的处理单元经互连网络连接而形成的系统,其中每个处理单元既具有高度自治性又相互协同,能在系统范围内实现资源管理、任务动态分配,并能并行地运行分布式程序。

分布式系统具有如下主要特点。

(1)结构模块性:分布式系统的资源单位形成相对独立的模块,它们互连成一个单一系统。模块在一定范围内的增减或替换不影响系统的整体性。

(2)资源分散性:系统资源分布于物理上分散的若干场点中。在对用户透明的基础上实现资源共享,使单个用户的可用资源成倍地增长。

(3)协同自治性:系统资源的操作是高度自治的,既不存在全系统的主/从控制关系,又能利用处理局部化的原则减少各场点间的通信量。

(4)工作并行性:分布式计算机系统中分散的资源单位可以相互协作,一起解决同一个问题。在分布式操作系统控制下,实现按任务资源重复或按功能时间重叠等不同形式的并行性。

(5)系统透明性:系统对用户是透明的。用户可以像使用单机系统一样使用分布式计算机系统。

(6)整体强健性:系统资源的余和自治控制方式使系统具有动态重构能力,即使系统受到局部性破坏也能继续工作。所以具有可靠性和容错性。

配置在分布式系统上的操作系统称为分布式操作系统。分布式操作系统(distributed operating system,DOS)是为分布式系统配置的操作系统。它在这种多机环境下,负责控制和管理以协同方式工作的多种系统资源,进程的同步和执行,处理机间的通信、调度等控制事务,自动实行全系统范围内的任务分配和负载平衡。分布式操作系统是具有高度并行性的一种高级软件系统。它与单机系统(集中式操作系统)以及网络操作系统都有不同程度的区别,其复杂程度也明显高于它们。分布式操作系统和单机操作系统的主要区别在于进程管理、资源管理和系统结构。

1.4　操作系统的特征和功能

操作系统是系统软件的核心。虽然现在操作系统种类繁多，有各自的特点，但它们也有一些共同特征，这就是并发性(concurrence)、共享性(sharing)、虚拟性(virtual)和不确定性(nondetermistic)。有些学者也把不确定性称为异步性(asynchronism)。

从资源管理的角度来看，操作系统对计算机资源进行控制和管理的功能主要分为处理机管理、存储器管理、设备管理和文件管理等。

1.4.1　操作系统的特征

现代操作系统有以下 4 个基本特征。

1)并发性

并发性和并行性是两个容易混淆的概念。并行性是指两个或多个事件同时发生；而并发性是指两个或多个事件在同一时间间隔内发生。从宏观上看，多道程序环境下并发执行的程序在同时向前推进，但在单处理机系统中，每一时刻仅有一道程序在处理机上执行，故从微观上看这些程序交替在处理机上执行。

程序的并发执行能有效改善系统资源利用率，但由于多道程序对系统资源的共享和竞争，使系统的控制和管理复杂化，因此操作系统必须具有控制和管理各种并发活动的能力。

2)共享性

资源共享是指系统中的硬件和软件资源不再为某个程序所独占，而是供多个用户共同使用。例如，多道程序在内存中并发执行，它们共享内存资源的同时也共享处理机资源；在这些程序执行期间，可能需要进行输入/输出操作或读写文件，因此它们也会共享设备及文件。

并发性和共享性是操作系统的两个最基本特征，二者之间互为存在条件。一方面，资源共享以程序的并发执行为条件，若系统不允许程序并发执行，自然不存在资源共享问题；另一方面，若系统不能对资源共享实施有效的管理，则必将影响到程序的并发执行，甚至根本无法并发执行。

3)虚拟性

虚拟是指把一个物理实体变为若干逻辑实体。物理实体是实际存在的，逻辑实体是虚拟的，其实现思想是通过对物理实体的分时使用，达到让用户感觉有多个实体存在的效果。

例如，在单处理机系统中引入多道程序设计技术后，虽然在系统中只有一个处理机，每次只能执行一道程序，但通过分时使用，在一段时间间隔内，宏观上这台处理机能同时运行多道程序，给用户的感觉是每道程序都有一个处理机为其服务。也就是说，多道程序设计技术可以把一台物理处理机虚拟为多台逻辑处理机。

4)不确定性

不确定性不是指操作系统本身的功能不确定,也不是指在操作系统控制下运行的用户程序的结果不确定(同一程序对相同的输入数据在两次或两次以上运行有不同的结果),而是指系统中各种事件发生的时间及顺序是不可预测的。

在多道程序环境中,由于程序的并发执行及资源共享等原因,程序的执行具有“走走停停”的性质。系统中每个执行着的程序既要完成自己的事情,又要与其他执行着的程序共享系统资源,彼此之间会直接或间接相互制约,每个程序何时执行、多个程序间的执行顺序以及完成每道程序所需的时间都是不可预知的。例如,从外围设备传入的中断、I/O 请求、程序运行时发生的故障等都是不可预测的。这是造成不确定性的基本原因。

1.4.2 操作系统的功能

操作系统已经成为现代计算机系统不可分割的重要组成部分,它为人们建立各种各样的应用环境奠定了基础。

资源管理是操作系统的一项主要任务,而控制程序执行、扩充机器功能、提供各种服务、方便用户使用、组织工作流程、改善人机界面等都可以从资源管理的角度理解。下面从资源管理的观点出发,讨论操作系统具有的几个主要功能。

1)处理机管理

处理机管理的主要任务是对处理机的分配和运行实施有效的管理。从传统的意义来讲,进程是处理机和资源分配的基本单位,因此对处理机的管理可以归结为对进程的管理。进程管理应实现以下 4 种主要功能。

(1)进程控制。进程是系统中活动的实体,进程控制包括进程的创建、撤销以及进程状态的转换。

(2)进程同步。多个进程在活动过程中会产生相互依赖或相互制约的关系,为保证系统中所有进程能够正常活动,必须对并发执行的进程进行协调。

(3)进程通信。相互合作的进程之间往往需要交换信息,为此系统要提供进程通信机制。

(4)作业和进程调度。一个作业通常需要经过两级调度才能在处理机上执行。作业调度将选中的一个或多个作业放入内存,为它们分配必要的资源并建立进程。进程调度按一定的算法将处理机分配给就绪队列中的合适进程。

2)存储器管理

存储器管理的主要任务是分配、保护和扩充内存,以及地址映射。

(1)内存分配。按一定的策略为每道程序分配一定的内存空间。为此,操作系统应记录整个内存的使用情况,当用户程序提出内存空间申请要求时应按照某种策略实施内存分配,当程序运行结束时应回收其占用的内存空间。

(2)内存保护。系统中存在多道并发执行的程序,因此系统应保证各程序在自己的内存区域内运行而不相互干扰,更不能干扰和侵占操作系统空间。

(3)内存扩充。一个计算机系统中的内存容量有限,而所有用户程序对内存的需求量通

常大于实际内存的容量。为了允许大程序或多个程序运行,应借助虚拟存储技术以获得增加内存的效果。

(4)地址映射。通常源程序经过编译链接后形成可执行程序,可执行程序的起始地址都从0开始,程序中的其他地址相对于起始地址计算。这样,在多道程序环境下,用户程序中的地址就有可能与它装入内存后实际占用的物理地址不一样,因此需要将程序中的地址转换为内存中的物理地址。

3)设备管理

计算机外部设备的管理是操作系统中最庞杂、最琐碎的部分。设备管理的主要任务是对计算机系统内的所有设备实施有效的管理。设备管理应具有以下3种主要功能。

(1)设备分配。根据用户程序提出的I/O请求和相应的设备分配策略,为用户程序分配设备,当设备使用完后还应回收设备。为了缓解设备慢速与处理机快速之间的矛盾,使设备与处理机并行工作,还需要使用缓冲技术。

(2)设备驱动。当CPU发出I/O指令后,应启动设备进行I/O操作。当I/O操作完成后,应向CPU发出中断信号,由相应的中断处理程序进行传输结束处理。

(3)设备独立性。设备独立性又称为设备无关性,是指用户程序中的设备与实际使用的物理设备无关。这样,用户程序不必涉及具体的物理设备,由操作系统完成用户程序中的逻辑设备到具体物理设备的映射,使得用户能更加方便灵活地使用设备。

4)文件管理

计算机系统中的程序和数据通常以文件的形式存放在外部存储器上,操作系统中负责文件管理的部分称为文件系统。文件系统的主要任务是有效地支持文件的存储、检索和修改等操作,解决文件的共享、保密和保护等问题。文件管理应具有下以下4种功能。

(1)文件存储空间的管理。文件存放在磁盘上,因此文件系统需要对文件存储空间进行统一管理,包括为文件分配存储空间,回收释放的文件空间,提高外存空间的利用率和文件访问效率。为此,文件系统应设置专门的数据结构记录文件存储空间的使用情况。

(2)目录管理。外存上存放着成千上万的文件,为了方便用户查找自己需要的文件,通常由系统为每个文件设置一个目录项,目录项中包含文件名、文件属性及文件在外存的存放地址,以提供按名存取的功能。

(3)文件操作管理。为方便用户使用文件,系统提供了一组文件操作功能,包括文件的创建、删除和读写等。

(4)文件保护。为了保证文件的安全性,防止系统中的文件被非法使用及遭到破坏,文件系统应提供文件保护功能。

1.5 操作系统的接口

操作系统除了对计算机系统中的软硬件资源实施管理外,还为用户提供了各种使用其服务功能的手段,即接口。

不同的操作系统为用户提供的服务不完全相同,但有许多共同点。操作系统提供的共性服务使得编程任务变得更加容易。操作系统提供给程序和用户的共性服务大致有下6种。

(1)创建程序。提供各种工具和服务,如编辑程序和调试程序,帮助用户编程并生成高质量的源程序。

(2)执行程序。将用户程序和数据装入主存,为程序运行做好一切准备工作并执行程序。当程序编译或运行出现异常时,应能报告发生的情况,终止程序执行或进行适当的处理。

(3)数据I/O。程序运行过程中需要I/O设备上的数据时,可以通过I/O命令或I/O指令请求操作系统的服务。操作系统不允许用户直接控制I/O设备,而是能让用户以简单的方式实现I/O控制和读写数据。

(4)信息存取。文件系统让用户按文件名来建立、读写、修改、删除文件,使用方便,安全可靠。当涉及多用户访问或共享文件时,操作系统将提供信息保护机制。

(5)通信服务。在许多情况下,一个进程要与另外的进程交换信息,这种通信发生在两种场合,一是在同一台计算机上执行的进程之间通信;二是在被网络连接在一起的不同计算机上执行的进程之间通信。

(6)错误检测和处理。操作系统能捕捉和处理各种硬件或软件造成的差错或异常,并将这些差错或异常造成的影响缩小在最小范围内,必要时及时报告给操作员或用户。

操作系统为用户提供了以下几类接口。

1.5.1 命令接口

使用命令接口进行作业控制的主要方式有脱机控制方式和联机控制方式两种。脱机控制方式是指用户将对作业的控制要求以作业控制说明书的方式提交给系统,由系统按照作业说明书的规定控制作业的执行。在作业执行过程中,用户无法干涉作业,只能等待作业执行结束之后才能根据结果信息了解作业的执行情况。联机控制方式是指用户利用系统提供的一组键盘命令或其他操作命令和系统会话,交互式地控制程序的执行。其工作过程是用户在系统给出的提示符下键入特定命令,系统在执行完该命令后向用户报告执行结果,然后用户决定下一步的操作,如此反复,直到作业执行结束。

按作业控制方式的不同,可将命令接口分为联机命令接口和脱机命令接口两种。

1)联机命令接口

联机命令接口又称为交互式命令接口,它由一组键盘操作命令组成。用户通过控制台或终端键入操作命令,向系统提出各种服务要求。用户每输入一条命令,控制权就转入操作系统的命令解释程序,然后命令解释程序对键入的命令解释执行,完成指定的功能。之后,控制权又转回到控制台或终端,此时用户又可以键入下一条命令。

在微机操作系统中,通常把键盘命令分成内部命令和外部命令两大类。

(1)内部命令。这类命令的特点是完成命令功能的程序短小,使用频繁。它们在系统初始启动时被引导至内存并且常驻内存。

(2)外部命令。完成这类命令功能的程序较长,各自独立地作为一个文件驻留在磁盘

上,当需要它们时,再从磁盘上调入内存运行。

2)脱机命令接口

脱机命令接口也称为批处理命令接口,它由一组作业控制命令(或称作业控制语言)组成。脱机用户不能直接干预作业的运行,而应事先用相应的作业控制命令写成一份作业操作说明书,连同作业一起提交给系统。当系统调度到该作业时,由系统中的命令解释程序对作业说明书上的命令或作业控制语句逐条解释执行。

1.5.2 程序接口

程序接口由一组系统调用命令(简称系统调用)组成。用户通过在程序中使用这些系统调用命令来请求操作系统提供的服务。用户在程序中可以直接使用这组系统调用命令向系统提出各种服务要求,如使用各种外部设备、进行有关磁盘文件的操作、申请分配和回收内存以及其他各种控制要求;也可以在程序中使用过程调用语句和编译程序将它们翻译成有关的系统调用命令,再调用系统提供的各种功能或服务。

1)系统调用

所谓系统调用,就是用户在程序中调用操作系统所提供的一些子功能。具体来讲,系统调用就是通过系统调用命令中断现行程序,而转去执行相应的子程序,以完成特定的系统功能;系统调用完成后,控制又返回到系统调用命令的逻辑后继指令,被中断的程序将继续执行下去。

实际上,系统调用不仅可以供用户程序使用,还可以供系统程序使用,以此实现各类系统功能。对于每个操作系统而言,其所提供的系统调用命令条数、格式以及所执行的功能等都不尽相同,即使是同一个操作系统,其不同版本所提供的系统调用命令条数也会有所增减。通常,一个操作系统提供的系统调用命令有几十乃至上百条之多,它们各自有一个唯一的编号或助记符。这些系统调用按功能大致可分为如下几类。

(1)设备管理。该类系统调用完成设备的请求、释放和设备启动等功能。

(2)文件管理。该类系统调用完成文件的读写、创建及删除等功能。

(3)进程控制。该类系统调用完成进程的创建、撤销、阻塞及唤醒等功能。

(4)进程通信。该类系统调用完成进程之间的消息传递或信号传递等功能。

(5)内存管理。该类系统调用完成内存的分配、回收,获取作业占用内存区大小及起始地址等功能。

系统调用命令是作为扩充机器指令提供的,目的是增强系统功能,方便用户使用。因此,在一些计算机系统中,把系统调用命令称为广义指令。广义指令与机器指令在性质上是不同的,机器指令是用硬件线路直接实现的,而广义指令则是由操作系统提供的一个或多个子程序模块实现的。

2)系统调用的执行过程

虽然系统调用命令的具体格式因系统而异,但用户程序进入系统调用的步骤及其执行过程大体上是相同的。

用户程序进入系统调用是通过执行调用指令(在有些操作系统中称为访管指令或软中

断指令)实现的,当用户程序执行到调用指令时,就中断用户程序的执行,转去执行实现系统调用功能的处理程序。系统调用处理程序的执行过程如下。

(1)为执行系统调用命令作准备。主要工作是保留用户程序的现场,并把系统调用命令的编号等参数放入指定的存储单元。

(2)执行系统调用。根据系统调用命令的编号,访问系统调用入口表,找到相应子程序的入口地址,然后转去执行。这个子程序就是系统调用处理程序。

(3)系统调用命令执行完后的处理。主要工作是恢复现场,并把系统调用的返回参数送入指定存储单元,以供用户程序使用。

3)系统调用与过程(函数)调用的区别

在程序中执行系统调用或过程(函数)调用,虽然都是对某种功能或服务的需求,但两者从调用形式到具体实现都有很大区别。

(1)调用形式不同。过程(函数)使用一般调用指令,其转向地址是固定不变的,包含在跳转语句中;系统调用中不包含处理程序入口,而仅仅提供功能号,按功能号调用。

(2)被调用代码的位置不同。过程(函数)调用是一种静态调用,调用者和被调用代码在同一程序内,经过连接编辑后作为目标代码的一部分。当过程(函数)升级或修改时,必须重新编译链接。而系统调用是一种动态调用,系统调用的处理代码在调用程序之外(在操作系统中),这样一来,系统调用处理代码升级或修改时,与调用程序无关。而且,调用程序的长度也大大缩短,减少了调用程序占用的存储空间。

(3)提供方式不同。过程(函数)往往由编译系统提供,不同编译系统提供的过程(函数)可以不同;系统调用由操作系统提供,一旦操作系统设计好,系统调用的功能、种类与数量便固定不变了。

(4)调用的实现不同。程序使用一般机器指令(跳转指令)来调用过程(函数),是在用户态运行的;程序执行系统调用是通过中断机制来实现的,需要从用户态转变到核心态,在管理状态执行,系统调用结束时,返回到用户态。

1.5.3 图形用户接口

通过命令接口方式来控制程序的运行虽然有效,但却给用户增加了很大的负担,即用户必须记住各种命令,并从键盘键入这些命令以及所需的参数,以控制用户程序的运行。随着大屏幕高分辨率图形显示器和多种交互式输入/输出设备(如鼠标、触摸屏等)的出现,图形用户接口于 20 世纪 80 年代后期出现并广泛应用。

图形用户接口的目标是通过对出现在屏幕上的对象直接进行操作来控制和操纵程序的运行。例如,用键盘或鼠标对菜单中的各种操作进行选择,使命令程序执行用户选定的操作;用户也可以通过拖动滚动条上的滑块在列表框中的选项上滚动,以使所要的选项出现在屏幕上,并用鼠标选取的方式来选择操作对象(如文件);用户还可以用鼠标拖动屏幕上的对象(如某图形或图标)使其移动位置或旋转、放大和缩小。这种图形用户接口大大减少或免除了用户的记忆工作量,其操作方式从原来的记忆并键入改为选择并点取,极大地方便了用户,受到用户普遍欢迎。目前图形用户接口是最为常见的人机接口形式,可以认为图形接口是命令接口的图形化。

1.6 操作系统的运行环境和内核结构

任何程序在计算机上运行都需要一定的运行环境,操作系统也不例外。现代操作系统的内核结构通常可分为模块结构、层次机构和微内核结构3类。

1.6.1 操作系统的运行环境

计算机硬件所提供的支持构成现代操作系统的硬件环境,如中央处理器(CPU)、主存储器、缓冲、时钟和中断等,其中,中断技术是推动操作系统发展的重要因素之一。事件引发中断,中断必须加以处理,从而驱动操作系统。

操作系统是一个众多程序模块的集合。根据运行环境的不同,这些程序模块大致可分为3类。

(1)在系统初启时便与用户程序一起主动参与并发运行的程序模块,如作业管理程序、输入/输出程序等。它们由时钟中断、外设中断驱动。

(2)直接面对用户态(亦称常态或目态)程序的程序模块,这是一些“被动”地为用户服务的程序。这类程序的每一个模块都与一条系统调用指令对应,仅当用户执行系统调用指令时,对应的程序模块才被调用、执行。系统调用指令的执行是经过陷入中断机构处理的,因此,从这个意义上来说,该类程序模块也是由中断驱动的。

(3)既不主动运行也不直接面对用户程序,而是隐藏在操作系统内部,由前两类程序模块调用的程序模块。既然前两类程序模块是由中断驱动的,那么该类程序模块也是由中断驱动的。应当注意,操作系统本身的代码运行在核心态(亦称管态、特态)。从用户态进入核心态的唯一途径就是中断。

操作系统控制和管理其他系统软件,并与其共同支持用户程序的运行,构建成用户的运行环境;同时,操作系统的功能设计也受到这些系统软件的功能强弱和完备与否的影响。

1.6.2 操作系统的内核结构

1)模块结构

模块结构也称为单内核模型。整个系统是一个大模块,可以分为若干逻辑模块,即处理器管理、存储器管理、设备管理和文件管理,模块间的交互是通过直接调用其他模块中的函数实现的。

模块结构是基于结构化程序设计的一种软件结构设计方法。早期操作系统(如IBM S/360操作系统)采用的就是这种结构设计方法,其主要设计思想和步骤为:把模块作为操作系统的基本单位,按照功能需要而不是根据程序和数据的特性把整个系统分解为若干模块(还可以再分成子模块),每个模块具有一定的独立功能,若干关联模块协作完成某个功能;明确各个模块之间的接口关系,各个模块间可以不加控制自由调用(所以又叫无序调

用法),数据多数作为全程量使用;模块之间需要传递参数或返回结果时,参数个数和返回方式也可以根据需要随意约定;然后分别设计、编码、调试各个模块;最后把所有模块连接成一个完整的系统。

模块结构设计方法的主要优点是:结构紧密、组合方便,对不同环境和用户的不同需求,可以组合不同模块来满足,从而灵活性大;针对某个功能可用最有效的算法和任意调用其他模块中的过程来实现,因此,系统效率较高;由于划分成模块和子模块,设计和编码可齐头并进,能加快操作系统的研制过程。

模块结构的主要缺点是:模块独立性差,模块之间牵连过多,形成了复杂的调用关系,甚至有很多循环调用,造成系统结构不清晰,正确性难保证,可靠性降低,系统功能的增、删、改十分困难。随着系统规模的扩大,采用这种结构的系统复杂性迅速增长,这就促使人们研究操作系统新的结构概念及设计方法。

2)层次结构

为了能让操作系统的结构更加清晰,使其具有较高的可靠性,较强的适应性,易于扩充和移植,在模块结构的基础上产生了层次式结构的操作系统。所谓层次结构,就是将操作系统划分为内核和若干模块(或进程),这些模块(或进程)按功能的调用次序排列成若干层次,各层之间只能是单向依赖或单向调用关系,即低层为高层服务,高层可以调用低层的功能,反之则不能,这样不但系统结构清晰,而且不构成循环调用。

层次结构可以有全序和半序之分。如果各层之间是单向依赖的,并且每层中的各个模块(或进程)之间也保持独立,没有联系,则这种层次结构被称为全序的。如果各层之间是单向依赖的,但在某些层内允许有相互调用或通信的关系,则这种层次结构称为半序的。

用层次结构构造操作系统目前还没有一个明确固定的分层方法,只能给出若干原则,供划分层次中的模块(或进程)时参考。

(1)应该把与机器硬件有关的程序模块放在最底层,以便起到把其他层与硬件隔离开的作用。在操作系统中,中断处理、设备启动、时钟等反映了机器的特征,因此,与这些特征有关的程序都应该放在离硬件尽可能近的层次中,这样安排既增强了系统的适应性,也有利于系统的可移植性,因为只需把这层内容按新机器硬件的特征加以改变后,而其他层的内容都可以基本不动。

(2)为进程(和线程)的正常运行创造环境和提供条件的内核程序,如 CPU 调度、进程(和线程)控制和通信机构等,应该尽可能放在低层,以支撑系统其他功能部件的执行。

(3)用户可能需要不同的操作方式,如选取批处理方式、联机控制方式或实时控制方式。为了能使操作系统从一种操作方式改变或扩充到另一种操作方式,在分层时就应把反映系统外特性的软件放在最外层,这样改变或扩充时,只涉及对外层的修改,内层共同使用的部分保持不变。

(4)应尽量按照实现操作系统命令时模块间的调用次序或按进程间单向发送信息的顺序来分层。这样,最上层接收来自用户的操作系统命令,随之根据功能需要逐层往下调用(或传递消息),自然而有序。例如,文件管理要调用设备管理,因此,文件管理各个模块(或进程)应该放在设备管理各个模块(或进程)的外层;作业调度程序控制用户程序执行时,要调用文件管理的功能,因此,作业调度模块(或进程)应该放在文件管理模块(或进程)的外层等。操作系统按照层次结构的原则,从底向上可以被安排为裸机、CPU 调度及其他内核功

能、内存管理、设备管理、文件管理、作业管理、命令管理、用户。

3)微内核结构

微内核是指把操作系统结构中的内存管理、设备管理、文件系统等高级服务功能尽可能地从内核中分离出来,变成几个独立的非内核模块,而在内核中只保留少量最基本的功能,使内核变得简洁可靠,因此叫做微内核。

微内核实现的基础是操作系统理论层面的逻辑功能划分。几大功能模块在理论上是相互独立的,形成了比较明显的界限,其优点如下。

(1)充分的模块化,可独立更换任一模块而不会影响其他模块,从而方便第三方开发、设计模块。

(2)未被使用的模块功能不必运行,因而能大幅度减少系统的内存需求。

(3)具有很高的可移植性,从理论上只需要单独对各微内核部分进行移植修改即可。由于微内核的体积通常很小,而且互不影响,因此工作量很小。

但是,因为各个模块与微内核之间是通过通信机制进行交互的,微内核的明显缺点是系统运行效率较低。

1.7 操作系统安全概述

操作系统运行在计算机硬件之上,要了解操作系统的安全,应先了解计算机安全。

1.7.1 计算机安全

有人将计算机安全定义为:计算机的硬件、软件和数据受到保护,不因偶然和恶意的原因而遭到破坏、更改和泄露,系统连续正常运行。

1)计算机安全的属性

计算机安全具有5个属性:可用性(availability)、可靠性(reliability)、完整性(integrity)、保密性(confidentiality)和不可抵赖性(non-repudiation)。

可用性是指无论何时,只要用户需要,信息系统就必须是可用的,也就是说信息系统不能拒绝服务。

可靠性是指系统在规定条件下和规定时间内完成规定功能的概率。

完整性是指系统中的信息不能在未经授权的前提条件下被有意或无意地篡改或破坏。

保密性是指只有授权用户才能以对应的授权存取方式访问系统中的相应资源和信息。

不可抵赖性是面向通信双方信息真实统一的安全要求,包括收、发双方均不可抵赖。

2)计算机安全的层次

通常可将计算机安全分为3个层次:物理安全(physical security)、运行安全(operation security)和信息安全(information security)。

物理安全是指如何保护计算机硬件和软件本身免遭自然灾害和其他环境事故(如电磁

污染等)破坏的措施、过程。特别是避免由于电磁泄漏产生信息泄露,从而干扰他人或受他人干扰。物理安全包括环境安全、设备安全和媒体安全 3 个方面。

运行安全是指计算机能在良好的环境中持续工作,为保障系统功能的安全实现,提供一套安全措施来保护信息处理过程的安全。它侧重于保证系统正常运行,避免因为系统的崩溃和损坏而对系统存储、处理和传输的信息造成破坏和损失。

信息安全是指如何防止信息被故意或偶然地非授权泄露、更改、破坏或使信息被非法的系统辨识、控制,即确保信息的完整性、保密性、可用性和可控性。防止攻击者利用系统的安全漏洞进行窃听、冒充、诈骗等有损于合法用户的行为,本质上是保护用户的利益和隐私。信息安全包括操作系统安全、数据库安全、网络安全、病毒防护、访问控制、加密与鉴别 7 个方面。

1.7.2 操作系统安全的重要性和面临的安全威胁

长期以来,我国广泛应用的主流操作系统都是从国外引进并直接使用的产品。从国外引进的操作系统,其安全性难以保证。目前,虽然我国在安全操作系统方面已经开展了一些工作,但还缺乏对操作系统安全模型和安全评估准则的深入研究,这使得安全操作系统方案缺乏整体性。

操作系统是计算机软件的运行基础,因此操作系统安全是整个计算机系统安全的基础,没有操作系统的安全,就不可能真正解决数据库安全、网络安全和其他各类软件的安全问题。

计算机系统的资源分为硬件、软件、数据等几种。每种类型的资源都面临着安全威胁。

对计算机系统硬件的威胁主要表现在可用性方面,包括对设备的有意或无意的破坏及偷窃。个人计算机和工作站的急剧增加以及局域网的日益广泛使用增加了这方面的潜在破坏,需要物理上的和行政管理上的安全措施来对付这些威胁。

软件所面临的威胁除了可用性威胁之外,还有保密性威胁。软件,尤其是应用软件,非常容易被修改、破坏和删除,从而失效;除此之外,软件的非法修改导致程序仍能运行但其行为却发生了变化。

数据安全性是一个更普遍的安全问题。与数据有关的安全性涉及面广,包括可用性、保密性和完整性。

操作系统作为系统软件,其所面临的安全威胁主要来自计算机病毒和黑客攻击。

操作系统的各种安全威胁导致的最终后果,其实就是对一般信息系统或计算机系统应该拥有的可用性、可靠性、完整性、保密性和不可抵赖性 5 个方面的安全特性的破坏。

第 11 章会具体介绍操作系统所采用的安全策略和保护机制,以应对各种安全威胁。

习　　题

1)选择题

(1)在计算机系统中,操作系统是(　　)。

A. 一般应用软件　　B. 核心系统软件　　C. 用户应用软件　　D. 硬件

(2)实时操作系统必须在(　　)内处理来自外部的事件。

A. 一个机器周期　　B. 被控制对象规定时间

C. 周转时间　　D. 时间片

(3)操作系统提供给编程人员的接口是(　　)。

A. 库函数　　B. 高级语言　　C. 系统调用　　D. 子程序

(4)操作系统中最基本的两个特征是(　　)。

A. 并发性和不确定性　　B. 并发性和共享性

C. 共享性和虚拟性　　D. 虚拟性和不确定性

(5)下述关于并发性的叙述中正确的是(　　)。

A. 并发性是指若干事件在同一时刻发生

B. 并发性是指若干事件在不同时刻发生

C. 并发性是指若干事件在同一时间间隔内发生

D. 并发性是指若干事件在不同时间间隔内发生

(6)多道批处理系统提高了计算机系统的资源利用率,同时(　　)。

A. 减少各个作业的执行时间　　B. 增加了单位时间内作业的吞吐量

C. 减少了部分作业的执行时间　　D. 减少单位时间内作业的吞吐量

(7)分时系统追求的目标是(　　)。

A. 充分利用 I/O 设备　B. 快速响应用户　　C. 提供系统吞吐率　　D. 充分利用内存

(8)批处理系统的主要缺点是(　　)。

A. 系统吞吐量小　　B. CPU 利用率不高　　C. 资源利用率低　　D. 无交互能力

(9)从用户的观点看,操作系统是(　　)。

A. 用户与计算机之间的接口

B. 控制和管理计算机资源的软件

C. 由若干层次的程序按一定的结构组成的有机体

D. 合理地组织计算机工作流程的软件

(10)所谓(　　),是指将一个以上的作业放入内存,并且同时处于运行状态,这些作业共享处理机的时间和外围设备等资源。

A. 多重处理　　B. 多道程序设计　　C. 实时处理　　D. 共行执行

2)填空题

(1)操作系统是计算机系统中的一个_____,它管理和控制计算机系统中的_____。

(2)如果一个操作系统兼有批处理、分时和实时操作系统三者或其中两者的功能,这样的操作系统称为_____。

(3)没有配置_____的计算机称为裸机。

(4)在主机控制下进行的输入/输出操作称为_____操作。

(5)操作系统的基本特征是_____、_____、_____、_____。

(6)实时系统按应用的不同分为_____和_____两种。

(7)在单处理机系统中,多道程序运行的特点是多道、_____和_____。

(8)操作系统作为系统软件,其所面临的安全威胁主要来自_____和_____。

3)解答题

(1)什么是操作系统？从资源管理的角度看，操作系统应具有哪些功能？

(2)操作系统有哪几种基本类型？它们各有什么特点？

(3)什么是多道程序设计技术？多道程序设计技术的特点是什么？

(4)简述并发与并行的区别。

(5)简述操作系统在计算机系统中的位置。

(6)操作系统有哪些特征？

(7)操作系统是随着多道程序设计技术的出现逐步发展起来的，要保证多道程序的正确运行，在技术上要解决哪些基本问题？

(8)实现分时系统的关键问题是什么？应如何解决？

(9)用户与操作系统之间存在哪几种接口？

4)应用题

(1)有一台计算机，具有 1 MB 内存，操作系统占用 200 KB，每个用户进程各占 200 KB。如果用户进程等待 I/O 的时间为 80%，若增加 1 MB 内存，则 CPU 的利用率提高多少？

(2)一个计算机系统，有一台输入机和一台打印机，现有两道程序投入运行，且程序 A 先开始运行，程序 B 后开始运行。程序 A 的运行轨迹为：计算 50 ms，打印 100 ms，再计算 50 ms，打印 100 ms，结束。程序 B 的运行轨迹为：计算 50 ms，输入 80 ms，再计算 100 ms，结束(假设开始时刻为 0)。试说明：

①两道程序运行时，CPU 有无空闲等待？若有，在哪段时间内等待？为什么会等待？

②程序 A、B 有无等待 CPU 的情况？若有，指出发生等待的时刻。

第 2 章 进程与线程

进程是计算机操作系统中最基本的概念之一。操作系统的基本任务是通过进程对资源进行管理。操作系统必须有效控制进程执行，给进程分配资源，允许进程之间共享和交换信息，保护每个进程在运行期间免受其他进程干扰，控制进程的互斥、同步和通信。

进程是资源分配的基本单位，往往也是独立运行的基本单位。本章主要介绍进程的引入、进程的定义、进程的状态及转换、进程控制及线程等内容。

2.1 进程的引入

在早期的计算机系统中，一次只允许运行一道程序，因此程序运行时完全占用了系统中的所有资源。在现代计算机系统中，内存中通常会存放多道程序，这些程序并发执行。并发执行的程序，其特征不同于顺序执行的程序，为了描述这些特征而引入了进程。事实上，在多道程序环境下，程序是不能独立运行的，而作为独立运行的基本单位往往是进程。下面通过对程序的顺序执行和并发执行特征的描述来引入进程这一重要概念。

2.1.1 前趋图

在描述一个程序的各部分(程序段或语句)间的依赖关系，或者一个大的计算任务的各个子任务间的因果关系时，常常采用前趋图的方式。前趋图是一个有向无循环图，用于描述程序、程序段或语句执行的先后次序，图中的每个节点可以表示一条语句、一个程序段或一个进程，节点间的有向边表示两个节点之间存在的前趋关系“→”。

$\rightarrow = \{(P_i, P_j) \mid P_i$ 必须在 P_j 开始执行之前完成$\}$

如果$(P_i, P_j) \in \rightarrow$，可以写成 $P_i \rightarrow P_j$，则称 P_i 是 P_j 的直接前驱，P_j 是 P_i 的直接后继。若

存在一个序列 $P_i \rightarrow P_j \rightarrow \cdots \rightarrow P_k$，则称 P_i 是 P_k 的前驱。在前趋图中，没有前驱的节点称为初始节点，没有后继的节点称为终止节点。图 2-1 给出了一个具有 5 个节点的前趋图示例，图中所描述的语句执行次序为：S_1 首先启动执行，当 S_1 执行完成后才能启动 S_2 及 S_3 执行，S_2 及 S_3 执行完成后才能启动 S_4 执行，S_2 及 S_4 执行完成后才能启动 S_5 执行。

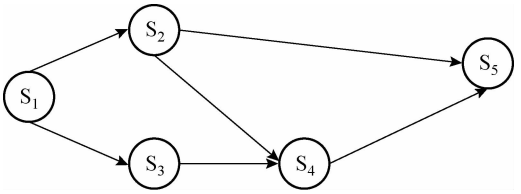

图 2-1　前趋图示例

在前趋图中至少有一个节点没有直接前趋，至少有一个节点没有直接后继，也就是说至少有一个起点和终点。

2.1.2　程序的顺序执行

人们利用计算机解题时，总要使用“程序”这一概念。程序的基本特性是它的顺序性，即一个程序通常由若干操作组成，这些操作必须按照某种先后次序执行，仅当前一个操作执行完成后才能执行后继操作，这类计算过程就是程序的顺序执行过程。例如，系统中有 n 个作业，在执行每个作业时总是先输入程序和数据，然后进行计算，最后将所得的结果打印输出。若用 I_i、C_i 和 P_i 分别表示作业 i 的输入、计算及输出操作，则在顺序执行模式下这些操作的执行次序如图 2-2 所示。

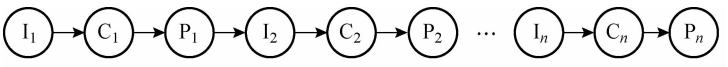

图 2-2　程序的顺序执行

程序顺序执行时具有以下特征。

(1)顺序性。处理机的操作严格按照程序所规定的顺序执行，只有当上一个操作完成后，下一个操作才能开始执行。除了人为的干预造成机器暂时停顿外，前一个动作的结束就意味着后一个动作的开始。

(2)封闭性。程序一旦开始运行，其执行结果就不受外界因素影响。因为程序在运行时独占系统的全部资源，除初始状态外，这些资源的状态只能由本程序改变，并不受任何外界因素的影响。

(3)可再现性。只要程序执行时的初始条件和执行环境相同，则当程序重复执行时，都将获得相同的结果(程序的执行结果与时间无关)。

2.1.3　程序的并发执行

如果在计算机系统中任何时刻都只能运行一道作业，则系统的处理能力无法提高，系统

资源利用率低下。为了提高计算机系统的处理能力和资源利用率，现代计算机系统中普遍采用了多道程序设计技术，这样使得系统内的多道程序可以并发执行。在图 2-2 中，虽然同一作业的输入、计算和打印操作必须顺序执行，但对 n 个作业而言，有些操作是可以并发执行的，如作业 1 的输入操作完成后即可进行该作业的计算操作；与此同时可以进行作业 2 的输入操作，即作业 1 的计算操作和作业 2 的输入操作可以并发执行。

图 2-3 给出了一批作业并发执行时的情况。在图 2-3 中，I_1 先于 C_1 和 I_2，C_1 先于 P_1 和 C_2，P_1 先于 P_2；而 I_2 与 C_1，I_3、C_2 和 P_1 则可以并发执行。程序的并发执行提高了资源利用率，从而提高了系统效率。

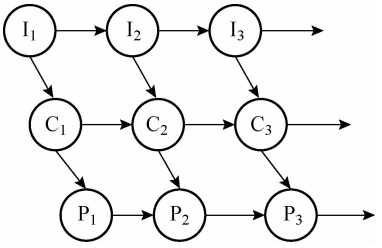

图 2-3　程序的并发执行

程序的并发执行是指若干程序或程序段同时在系统中运行，这些程序或程序段的执行在时间上是重叠的，一个程序或程序段的执行尚未结束，另一个程序或程序段的执行已经开始。

程序的并发执行虽然提高了系统的处理能力和资源利用率，但也带来了一些新问题，产生了一些与顺序执行时不同的特征。

(1)间断性。程序在并发执行时，由于它们共享资源或为完成同一项任务而相互合作，致使并发程序之间形成了相互制约关系。例如，在图 2-3 中，若 C_1 未完成则不能进行 P_1，因此作业 1 的打印操作暂时不能进行，这是由相互合作完成同一项任务而产生的直接制约关系；若 I_1 未完成则不能进行 I_2，则作业 2 的输入操作暂时不能进行，这是由共享资源而产生的间接制约关系。这种相互制约的关系导致并发执行程序具有“执行—暂停执行—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。这样一个程序在执行时，必然会受到其他程序的影响。例如，当处理机被某程序占用时，其他程序必须等待。

(3)不可再现性。程序并发执行时，由于失去了封闭性，也将导致失去其运行结果的可再现性。例如，有两个程序 A 和 B，它们共享一个变量 N。程序 A 对变量 N 执行 $N=N+1$ 的操作；程序 B 对变量 N 执行 print(N)的操作。由于程序 A 和程序 B 都以各自独立的执行速度向前推进，故程序 A 的 $N=N+1$ 操作既可以发生在程序 B 的 print(N)操作之前，也可以发生在 print(N)操作之后。假设某时刻 N 的值为 n，对于上述两种情况，执行完程序 A 和 B 的相应语句后，打印出来的 N 值分别为 $n+1$ 和 n。

2.1.4 程序并发执行的条件

程序并发执行时具有结果不可再现的特征，这并不是用户希望看到的结果。为此，要求程序在并发执行时必须保持封闭性和可再现性，这是正确性的要求。于是我们需要寻找程序并发执行时具有可再现性的条件。

1966 年，Bernstein 首先给出了程序并发执行的条件。为了描述方便起见，先定义一些表示方法。

$R(P_i)=\{a_1,a_2,\cdots,a_m\}$：表示程序段 P_i 在执行期间所需引用的所有变量的集合，称为读集。

$W(P_i)=\{b_1,b_2,\cdots,b_n\}$：表示程序段 P_i 在执行期间要改变的所有变量的集合，称为写集。

若两个程序段 P_1 和 P_2 能满足下述 3 个条件，则它们就能并发执行并且其结果具有可再现性。因该条件由 Bernstein 提出，故又称 Bernstein 条件。

(1)$R(P_1)\cap W(P_2)=\{\ \}$。

(2)$R(P_2)\cap W(P_1)=\{\ \}$。

(3)$W(P_1)\cap W(P_2)=\{\ \}$。

其中，前两个条件保证一个程序在两次读操作之间存储器中的数据不会发生变化；最后一个条件保证程序写操作的结果不会丢失。只要同时满足这 3 个条件，并发执行的程序就可以保持封闭性和可再现性。但这并没有解决所有问题，在实际程序执行过程中很难对这 3 个条件进行检查。

2.2 进程的定义及描述

在多道程序环境下，程序的并发执行破坏了程序的封闭性和可再现性，使得程序和计算不再一一对应，程序活动不再处于一个封闭系统中，程序的运行出现了许多新的特征。在这种情况下，程序这个静态概念已经不能如实地反映程序活动的这些特征，为此，人们引入了一个新的概念——进程。

2.2.1 进程的定义

进程的概念是在 20 世纪 60 年代初期，首先由麻省理工学院的 MULTICS 系统和 IBM 公司的 TSS/360 系统引入的。自那之后，有许多人对进程下过各式各样的定义，但迄今为止进程还没有统一的定义和名称，麻省理工学院称进程(process)，IBM 公司称任务(task)，在本书中我们采用进程这一名称。下面给出几种比较容易理解又能反映进程实质的定义。

(1)进程是程序在处理机上的一次执行过程。

(2)进程是可以和其他计算并行执行的计算。

(3)进程是程序在一个数据集合上的运行过程,是系统进行资源分配和调度的一个独立单位。

(4)进程是一个具有一定功能的程序关于某个数据集合的一次运行活动。

上述这些描述从不同的角度对进程进行了定义,尽管各有侧重,但它们在本质上是相同的,即进程是可以并发执行的程序在某个数据集合上的一次计算活动,也是操作系统进行资源分配和保护的基本单位。

2.2.2　进程的特征

在多道程序系统中,多个进程并发执行使得进程具有以下几个基本特征。

(1)动态性。进程是程序在处理机上的一次执行过程,因而是动态的。动态性还表现在它因创建而产生,由调度而执行,因得不到资源而暂停执行,最后由撤销而消亡。

(2)并发性。多个进程实体同时存在于内存中,在一段时间内都得到运行。引入进程的目的就是使程序能与其他程序并发执行,以提高资源利用率。

(3)独立性。进程是能独立运行的基本单位,也是系统进行资源分配和调度的独立单位。

(4)异步性。系统中的各进程以独立、不可预知的速度向前推进。

(5)结构性。为了描述和记录进程的运动变化过程,并使之能正确运行,应为每个进程配置一个进程控制块。这样,从结构上看,每个进程都由程序段、数据段和一个进程控制块组成。

2.2.3　进程和程序的关系

进程和程序是两个密切相关但又不同的概念,它们在以下几个方面存在区别和联系。

(1)进程是动态的,程序是静态的。进程是程序的一次执行过程,程序是一组代码的集合。

(2)进程是暂时的,程序是永久的。进程是一个状态变化的过程,程序可以长久保存。

(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)进程与程序是密切相关的。通过多次执行,一个程序可以对应多个进程;通过调用关系,一个进程可以包括多个程序。

(5)进程可创建其他进程,而程序并不能形成新的程序。

2.2.4　进程控制块

进程控制块(process control block,PCB)是进程实体的一部分,是进程存在的唯一标志。为了准确地描述每个进程,并对进程进行有效的控制与管理,系统为每个进程创建了一个进程控制块。系统通过进程控制块感知进程的存在,通过进程控制块中各项变量的变化了解进程的运行状况,根据进程控制块中各项信息对进程进行调度、控制和管理。因此,当进程创建时,系统为它建立一个 PCB;当进程在运行过程中状态发生变化时,系统将

其运行信息记录在 PCB 中；当进程执行完毕时，系统回收其 PCB。所以 PCB 是进程在其生命期间的管理档案。虽然不同操作系统中进程控制块的结构不同，但通常都包括以下内容。

(1)进程标识符。它是唯一标识进程的一个标识符或整数，以使该进程区别于系统内部的其他进程。在进程创建时，由系统为进程分配唯一的进程标识符。

(2)进程当前状态。说明进程的当前状态，以作为进程调度程序分配处理机的依据。

(3)进程队列指针。用于记录 PCB 队列中下一个 PCB 的地址。系统中的 PCB 可能组织成多个队列，如就绪队列、阻塞队列等。

(4)程序和数据地址。指出进程的程序和数据在内存或外存中的存放地址。

(5)进程优先级。反映进程获得 CPU 的优先级别，优先级高的进程可以优先获得处理机。

(6)CPU 现场保护区。当进程因某种原因释放处理机时，CPU 现场信息被保存在 PCB 的 CPU 现场保护中，以便在进程重新获得处理机后能恢复执行。通常被保护的信息有通用寄存器、程序计数器、程序状态字等内容。

(7)通信信息。记录进程在执行过程中与其他进程所发生的信息交换情况。

(8)家族关系。有的系统允许进程创建子进程，从而形成一个进程家族树。在 PCB 中必须指明本进程与家族的关系，如它的子进程与父进程的标识。

(9)资源清单。列出进程所需资源及当前已分配资源。

系统中通常存在许多进程，它们有的处于就绪状态，有的处于阻塞状态，而且阻塞的原因各不相同，为了方便进程的调度和管理，需要将各进程的进程控制块用适当的方法组织起来。目前常用的组织方式有链接方式和索引方式两种。链接方式是将同一状态的进程控制块链接成一个队列，不同状态对应多个不同的队列，如就绪队列和阻塞队列等。索引方式是将同一状态的进程组织在一个索引表中，索引表的表项指向相应的进程控制块，不同状态对应不同的索引表，如就绪索引表和阻塞索引表等。

2.3 进程的状态与转换

进程是动态的，有生命期的。可以将进程的生命周期划分为一组状态，在不同的阶段有不同的状态，而且各种状态之间是可以转换的。下面介绍进程的基本状态及状态间的转换。

2.3.1 进程的基本状态

在多道程序系统中，同时存在多个进程，这些进程并发运行并共享资源，因而彼此之间相互制约，这使得进程的状态不断发生变化。一个进程从创建而产生至撤销而消亡的整个生命期间，有时占有处理机执行，有时虽可运行但分不到处理机，有时虽有空闲处理机但因等待某个事件的发生而无法执行，这一切都说明进程和程序不同，它是活动的且有状态变化的，这可以用一组状态加以刻画。通常，一个进程至少应有以下 3 种基本状态。

1)就绪状态

进程已获得了除处理机以外的所有资源,一旦获得处理机就可以立即执行,此时进程所处的状态为就绪状态。处于就绪状态的进程已经具备了运行条件,但由于其他进程正占用处理机,它暂时不能运行而处于等待分配处理机的状态。在操作系统中处于就绪状态的进程可以有多个。

2)执行状态

执行状态又称运行状态。当一个进程获得必要的资源并正在处理机上执行时,该进程所处的状态为执行状态。处于执行状态的进程数目不能大于处理机数目,在单处理机系统中处于执行状态的进程最多只有一个。

3)阻塞状态

阻塞状态又称等待状态、睡眠状态。正在执行的进程,由于发生某事件而暂时无法执行下去(如等待输入/输出完成)时,进程所处的状态为阻塞状态。处于阻塞状态的进程尚不具备运行条件,这时即使处理机空闲,它也无法使用。系统中处于这种状态的进程可以有多个。

进程并非固定处于某一个状态,其状态会随着自身的推进和外界条件的变化而发生变化。图 2-4 给出了进程的 3 种基本状态以及引起状态转换的典型原因。

从图 2-4 中可以看出,处于就绪状态的进程,当进程调度程序为之分配了处理机后,该进程便由就绪状态转变为执行状态;正在执行的进程因等待某事件发生时,如进程提出输入/输出请求并等待输入/输出操作完成,则进程由执行状态变为阻塞状态;处于阻塞状态的进程,当其等待的事件已经发生时,如输入/输出操作完成,则进程由阻塞状态转变为就绪状态;正在执行的进程,如果因时间片用完而暂停执行,则该进程便由执行状态转变为就绪状态。

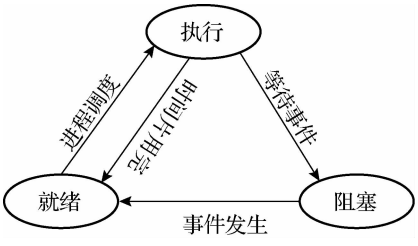

图 2-4　进程状态转换图

2.3.2　进程的创建状态和退出状态

在不少系统中,除了上述 3 种基本状态之外,又增加了两种状态,即创建状态和退出状态,如图 2-5 所示。

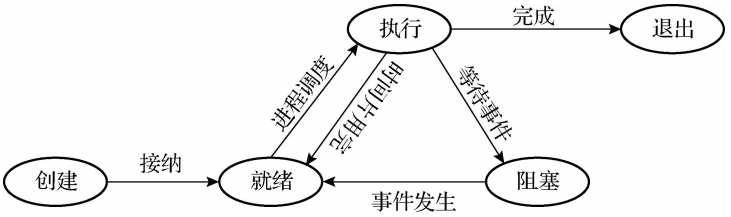

图 2-5　具有 5 种状态的进程状态转换图

(1)创建状态。进程刚被创建,尚未被放入就绪队列,此时进程处于创建状态。

(2)退出状态。进程已结束运行,释放了除进程控制块之外的其他资源,此时进程退出状态。

从图 2-5 中可以看出,当系统有足够的资源能够接纳新进程时,将处于创建状态的进程移入就绪队列;当一个进程执行完成或因失败而终止时,进入退出状态。对于任何一个进程来说,它只能处于创建状态和退出状态一次,但可以在执行状态、就绪状态和阻塞状态之间多次转换。

2.3.3　进程的挂起状态

在某些系统中,为了更好地管理和调度进程及适应系统的功能目标,引入了挂起状态。引入挂起状态可能基于以下原因。

(1)系统出故障或某些功能受到破坏时,就需要暂时将系统中的进程挂起,以便系统故障消除后,再将这些进程恢复到原来的状态。

(2)用户检查自己作业的中间执行情况和中间结果时,因与预期想法不符而产生怀疑,这时用户要求挂起其进程,以便进行某些检查和改正。

(3)系统中有时负荷过重(进程数过多),资源数相对不足,从而造成系统效率下降。此时需要挂起一部分进程以调整系统负荷,等系统中负荷减轻后再恢复被挂起进程的运行。

(4)在操作系统中引入了虚拟存储管理技术后,需要区分进程是驻留在内存还是外存时,可以用挂起表示驻留在外存。

图 2-6 给出了具有挂起状态的进程状态转换图。与图 2-4 相比,图 2-6 对就绪状态和阻塞状态进行了细分,增加了两个新的状态:挂起就绪和挂起阻塞。为了便于区分,将原来的就绪状态称为活动就绪,将原来的阻塞状态称为活动阻塞。

从图 2-6 中可以看出,如果一个进程原来处于执行状态或活动就绪状态,可因挂起命令而由原来状态变为挂起就绪状态,处于挂起就绪状态的进程不能参与争夺处理机,即进程调度程序不会把处于挂起就绪状态的进程挑选来运行;当处于挂起就绪状态的进程接到激活命令后,它就由原状态变为活动就绪状态;如果一个进程原来处于活动阻塞状态,它可因挂起命令而变为挂起阻塞状态,直到激活命令才能把它重新变为活动阻塞状态;处于挂起阻塞状态的进程,其所等待的事件发生后,该进程就由原来的挂起阻塞状态变为挂起就绪状态。

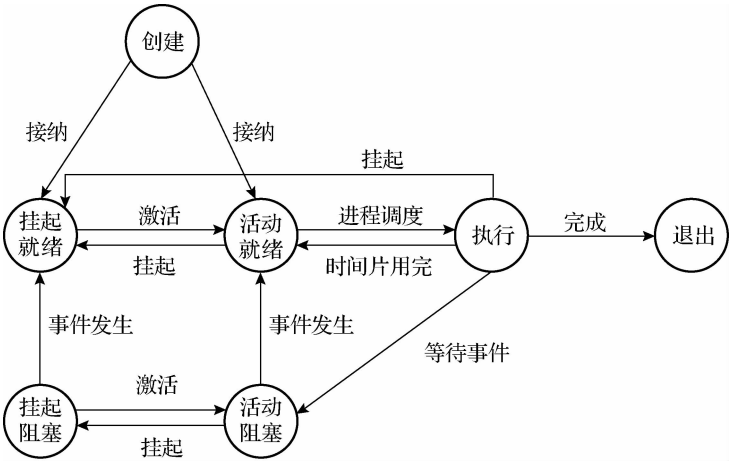

图 2-6　具有挂起状态的进程状态转换图

2.4　进 程 控 制

进程控制的职责是对系统中的所有进程实施有效的管理，其功能包括进程创建与撤销、进程阻塞与唤醒等。这些功能一般由操作系统内核实现。

2.4.1　操作系统内核

在现代操作系统设计中，往往把一些与硬件紧密相关的模块或运行频率较高的模块以及为许多模块所公用的一些基本操作安排在靠近硬件的软件层次中，并使它们常驻内存，以提高操作系统的运行效率，通常把这部分软件称为操作系统内核。操作系统内核是基于硬件的第一次软件扩充，它为系统控制和进程管理提供了良好的环境。操作系统内核的主要功能包括中断、时钟管理、进程管理、存储器管理、设备管理等。

操作系统使用原语对进程进行控制。原语被认为是机器语言的延伸，是完成特定任务的一段基本程序，它具有原子操作性。原子操作是不可分的操作，要么全做，要么全不做，它是通过中断屏蔽来实现的。原语对用户是透明的，一般不允许直接使用，但允许程序员作为一种特殊的系统调用来使用。

任何一个计算机系统中都有两种运行状态，即核心态和用户态。当操作系统内核程序执行时处于核心态，当用户程序运行时处于用户态。

(1)核心态又称管态、系统态，是操作系统管理程序执行时机器所处的状态。这种状态具有较高的特权，能执行一切指令，访问所有的寄存器和存储区。

(2)用户态又称目态，是用户程序执行时机器所处的状态。这种状态具有较低特权，只能执行规定的指令，访问指定的寄存器和存储区。

区分两种运行状态是为了给操作系统内核某些特权。例如，改变状态寄存器的内容的

特权是通过执行特权指令实现的，仅当在核心态下才能执行特权指令，在用户态下执行特权指令是非法的。

2.4.2 进程创建

1)进程图

一个进程可以创建若干新进程，新创建的进程又可以创建子进程，为了描述进程之间的创建关系，引入了图 2-7 所示的进程图。

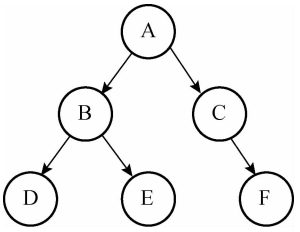

图 2-7 进程图

进程图又称为进程树或进程家族树，是描述进程家族关系的一棵有向树。图中的节点表示进程，若进程 A 创建了进程 B，则从节点 A 有一条有向边指向节点 B，说明进程 A 是进程 B 的父进程，进程 B 是进程 A 的子进程。创建父进程的进程称为祖父进程，从而形成了一棵进程家族树，把树的根节点称为进程家族的祖先。例如，若进程 A 创建了子进程 B、C，子进程 B 又创建了自己的子进程 D、E，子进程 C 创建了子进程 F，则构成了一棵图 2-7 所示的进程家族树，其中进程 A 是该家族的祖先。

2)进程创建原语

在多道程序环境中，只有进程才可以在系统中运行。要使一个程序能运行，就必须为它创建进程。引起进程创建的事件大致有以下几类。

(1)调度新作业。在批处理系统中，提交给操作系统的作业通常存放在磁带或磁盘上。当作业调度程序选中某个作业时，便为该作业创建进程，分配必要的资源并插入就绪队列中。

(2)用户登录。在交互式系统中，当用户登录进入系统时，操作系统要建立新进程(如命令解释进程)，负责接收并解释用户输入的命令。

(3)操作系统提供服务。当运行中的用户程序向操作系统提出某种请求时，操作系统也会创建进程来完成用户程序所需要的服务功能。例如，用户程序请求打印一个文件，操作系统将建立一个打印进程，负责管理用户程序所需要的打印工作。

(4)应用请求。前三种情况都是由操作系统根据需要为用户创建进程的，事实上应用程序也可以根据需要来创建一个新进程，使之与父进程并发执行，以完成特定的任务。例如，一个应用进程可以创建另一个进程，后者接收前者生成的数据，并把它们重新组织以满足后面分析的需要。

创建进程的方式有两种：由系统程序模块统一创建和由父进程创建。它们都需要调用进程创建原语来实现。创建原语的主要工作是，首先扫描系统的 PCB 表，查询有无空的

PCB 表项，如有，则申请一个，并对其进程初始化，初始化的项目有进程标识符(PID)、进程状态和执行程序的起始地址；如果申请不成功，则创建失败。进程创建原语的主要操作过程如下。

(1)向系统申请一个空闲 PCB。从系统的 PCB 表中找出一个空闲的 PCB 表项，并指定唯一的进程标识符(PID)。

(2)为新进程分配资源。根据新进程提出的资源需求为其分配资源，例如，为新进程分配内存空间以存放其程序和数据。

(3)初始化新进程的 PCB。在新进程的 PCB 中填入进程名、家族信息、程序和数据地址、进程优先级、资源清单及进程状态等信息。一般新进程的状态为就绪状态。

(4)将新进程的 PCB 插入就绪队列。

2.4.3　进程撤销

一个进程在完成其任务后应予以撤销，以便及时释放它所占用的各类资源。引起进程撤销的事件大致有以下几类。

(1)进程正常结束。当一个进程完成其任务后，应该调用撤销原语来释放该进程所占用的各种资源和 PCB 结构本身，以利于有效地使用资源。

(2)进程异常结束。在进程运行期间，如果出现了错误或故障，则进程被迫结束运行。导致进程异常结束的事件较多，如运行超时、内存不足、越界错误、I/O 故障、算术运算错误等。

(3)外界干预。进程因外界的干预而被迫结束运行。外界干预包括操作人员或操作系统的干预，如为了解除死锁，操作人员或操作系统要求撤销进程；父进程终止，当父进程终止时操作系统会终止其子孙进程；父进程请求，父进程有权请求系统终止其子孙进程。

撤销原语可以采用两种撤销策略：一种策略是只撤销指定标识符的进程，另一种策略是撤销指定进程及其所有子孙进程。下面给出后一种撤销策略的功能描述。

(1)根据撤销进程标识符，从相应队列中找到它的 PCB。

(2)检查被撤销进程的状态是否为执行状态，若是，则立即停止该进程的执行，设置重新调度标志，以便在该进程撤销后将处理机分配给其他进程。

(3)检查被撤销进程是否有子孙进程，若有子孙进程还应撤销该进程的子孙进程。

(4)回收该进程占有的全部资源并回收其 PCB。

2.4.4　进程阻塞与唤醒

当进程在执行过程中等待某事件(如读写磁盘、接受其他进程的数据)的发生而暂停执行时，通过调用阻塞原语将自己阻塞起来，并主动让出处理机。阻塞原语在执行时，必须先中断处理机并同时保存该进程的 CPU 现场，然后将阻塞进程插入等待队列中，再转由调度程序从就绪队列中选择一个进程投入运行。

当进程等待的条件或事件产生后，等待该事件的所有进程都被唤醒。唤醒进程的方法有两种：系统进程唤醒和由事件发生进程唤醒。它们都需要调用唤醒原语来唤醒另一进程。

唤醒原语首先将被唤醒进程从相应的等待队列取下，并将其状态设置为就绪态后，送入就绪队列。此时，唤醒原语既可以从调用程序处直接返回，也可以转向进程调度程序，让调度程序选择一个合适的进程去执行。需要注意的是，一个进程由执行状态转变为阻塞状态是该进程自己调用阻塞原语完成的；而进程由阻塞状态到就绪状态是另一个发现者进程调用唤醒原语实现的，一般这个发现者进程与被唤醒进程是合作的并发进程。

引起进程阻塞和唤醒的事件大致有以下几类。

(1)请求系统服务。当正在执行的进程向系统请求某种服务时，由于某种原因其要求无法立即满足，进程便暂停执行而变为阻塞状态。例如，当进程在执行中请求打印服务时，由于打印机已被其他进程占用，请求者只能进入阻塞状态等待。当进程请求的系统服务完成时，应将阻塞进程唤醒。

(2)启动某种操作并等待操作完成。如果进程执行时启动了某操作，且进程只有在该操作完成后才能继续执行，那么进程也将暂停执行并变为阻塞状态。例如，进程启动了某 I/O 设备进行 I/O 操作，但由于设备速度较慢而不能立刻完成指定的 I/O 任务，所以进程进入阻塞状态等待。当进程启动的操作完成时，应将阻塞进程唤醒。

(3)等待合作进程的协同配合。相互合作的进程，有时需要等待合作进程提供新的数据或等待合作进程做出某种配合而暂停执行，此时进程暂停执行并变为阻塞状态。例如，计算过程不断地计算结果并将其存入缓冲区中，而打印进程不断地从缓冲区中取出数据进行打印。如果计算进程尚未将数据送到缓冲区中，则打印进程只能变为阻塞状态等待。当合作进程完成协同任务时，应将阻塞进程唤醒。

(4)系统进程无新工作可做。系统中往往设置了一些具有特定功能的系统进程，每当它们的任务完成后便将自己阻塞起来，以等待新任务的到来。例如，系统中设置的发送进程，当发送请求全部完成且尚无新的发送请求时，发送进程阻塞等待。当系统进程收到新的任务请求时，应将阻塞进程唤醒。

阻塞原语的功能是将进程由执行状态转变为阻塞状态，其主要操作过程如下。

(1)停止当前进程的执行。进程阻塞时，由于该进程正处于执行状态，故应停止该进程的执行。

(2)保存该进程的 CPU 现场信息。为了使进程以后能够重新调度运行，应将该进程的现场信息送入其 PCB 现场保护区中保存起来。

(3)将进程状态改为阻塞，并插入相应事件的等待队列中。

(4)转到进程调度程序，从就绪队列中选择一个新的进程投入运行。

唤醒原语的功能是将进程由阻塞状态转变为就绪状态，其主要操作过程如下。

(1)将被唤醒进程从相应的等待队列中移出。

(2)将进程状态改为就绪，并将该进程插入就绪队列。

(3)在某些系统中，如果被唤醒进程比当前运行进程的优先级更高，可能需要设置调度标志。

2.4.5 进程的挂起与激活

当用户或创建者需要了解进程的活动情况或干预进程活动时，可以把某进程置于挂起就绪状态或挂起阻塞状态，这时要调用挂起原语。需要注意的是，调用挂起原语的进程只能

挂起它自己或它的子孙进程。挂起原语的实现方式有多种:把发出挂起原语的进程自身挂起、挂起具有指定标识符的进程、把某进程及其全部或部分子孙进程挂起。下面以挂起具有指定标识符的进程为例,说明挂起原语的主要操作过程。

(1)以进程标识符为索引,到 PCB 表中查找该进程的 PCB。

(2)检查该进程的状态。

(3)若状态为执行,则停止执行该进程并保护 CPU 现场信息,将该进程状态改为挂起就绪。

(4)若状态为活动阻塞,则将该进程状态改为挂起阻塞。

(5)若状态为活动就绪,则将该进程状态改为挂起就绪。

(6)若进程挂起前为执行状态,则转进程调度,从就绪队列中选择一个进程投入运行。

激活原语使处于静止状态的进程变为活动状态,即将挂起就绪状态变成活动就绪状态,或将挂起阻塞状态变成活动阻塞状态。激活方式有多种,如激活一个指定标识符的进程,或激活某进程及其所有子进程等。

一旦被激活的进程处于活动就绪状态,由于其优先级可能已发生改变,便可能引起处理机的重新调度。下面以激活具有指定标识符的进程为例,说明激活原语的主要操作过程。

(1)以进程标识符为索引,到 PCB 表中查找该进程的 PCB。

(2)检查该进程的状态。

(3)若状态为挂起阻塞,则将该进程状态改为活动阻塞。

(4)若状态为挂起就绪,则将该进程状态改为活动就绪。

(5)若进程激活后为活动就绪状态,可能需要转进程调度。

下面再通过一个实例来介绍进程的创建、进程和程序的区别,以及进程间的并发执行。在 Linux 系统中,可用 fork()系统调用来创建一个新进程。系统调用格式为

```
pid=fork()
```

fork()返回值的含义如下。

pid=0:在子进程中,表示当前进程是子进程。

pid >0:在父进程中,返回值为子进程的 ID 值(唯一标识号)。

pid =-1:创建失败。

如果 fork()调用成功,它向父进程返回子进程的 pid,并向子进程返回 0,即 fork()被调用了一次,但返回了两次。此时操作系统在内存中建立一个新进程,所建的新进程是调用 fork()父进程的副本,称为子进程。子进程继承了父进程的许多特性,并具有与父进程完全相同的用户级上下文。父进程与子进程并发执行。

该实例在 Linux 环境下运行的 C 语言程序如下。

```
#include <stdio.h>
int main()
{
      int pid,i;
      pid=fork();       //系统调用,创建进程
      if(pid<0){        //创建不成功,出错
          printf("Fork failed.");
          exit(1);          //系统调用
```

```
    }
    else if(pid==0){     //子进程执行
          for(i=1;i<10;i++)
            printf("BBB\n");
    }
    else{                //父进程执行
          for(i=1;i<10;i++)
            printf("AAA\n");
    }
    exit(0);
}
```

从程序的角度看，3 条分支中，只有一条会被执行，但实际运行结果表明 pid=0 的分支和 pid>0 的分支都被执行了。这是因为执行 fork()系统调用成功后，创建了一个子进程。这说明进程可以创建进程。当子进程运行时，执行 pid=0 的分支；当父进程运行时，执行 pid>0 的分支。所以两条分支都被执行的现象是两个进程分别执行的结果。当多次运行该实例时，会发现输出的次序是不相同的，这种现象表明进程间的并发执行是异步的。

2.5 进程的组织

进程控制块(PCB)是保存进程的状态和控制进程转换的标识，也是进程存在的唯一标识。创建进程时产生 PCB，撤销进程时回收 PCB。进程的组织主要是指对 PCB 的组织方式。进程的组织方式通常有 3 种：线性方式、链接方式和索引方式。

(1)线性方式：将 PCB 顺序地存放在一片连续内存中，如图 2-8 所示。

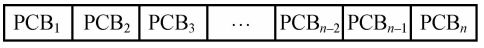

图 2-8 线性方式

(2)链接方式：将进程按状态组织成若干链表，同一状态的 PCB 组成一个链表，如图 2-9 所示。

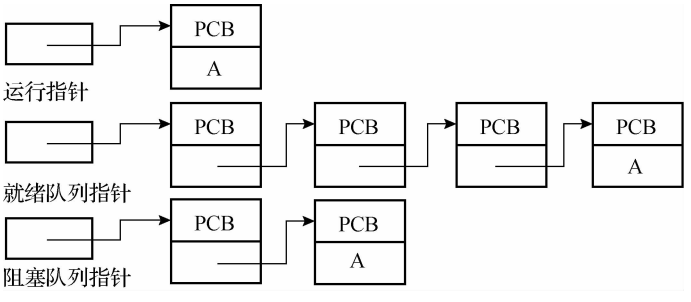

图 2-9 链接方式

(3)索引方式:将同一状态的进程归入一个索引表,再由索引指向相应的 PCB,如图 2-10 所示。

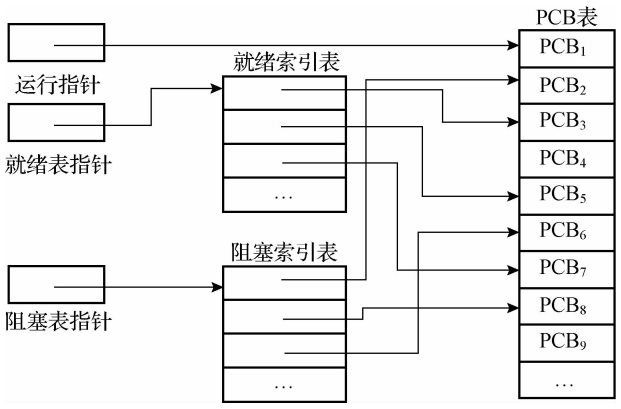

图 2-10　索引方式

2.6 线　　程

线程是近年来操作系统领域出现的一项非常重要的技术,其重要程度一点也不亚于进程。本节将重点介绍线程的概念、作用、实现方式及线程与进程的区别。线程的引入是为了减少进程切换的开销,增加进程并发执行的程度,从而进一步提高系统的吞吐量。我们把包含多个线程的进程称为多线程进程。

在传统的操作系统中,进程是系统进行资源分配的基本单位,以进程为单位分配所需要的虚地址空间、执行所需要的主存空间、完成任务需要的其他各类外围设备资源和文件。同时,进程也是处理器调度的基本单位,进程在任一时刻只有一个执行控制流,通常将这种结构的进程称为单线程进程。

2.6.1　线程的概念

在传统操作系统的单线程进程中,进程和线程的概念可以不加区别。如果说在操作系统中引入进程的目的是使多个程序并发执行,以改善资源利用率及提高系统吞吐量,那么,在操作系统中再引入线程,则是为了减少程序并发执行时所付出的时空开销,使操作系统具有更好的并发性。

1)线程的引入

图 2-11 给出了单线程进程的内存布局,它由进程控制块和用户地址空间,以及管理进程执行的调用/返回行为的系统堆栈或用户堆栈构成。进程的结构可以划分为两个部分:对资源的管理和实际的指令执行序列。并发进程之间的切换和通信均要借助于操作系统的进

程管理和进程通信机制，因而实现代价较大，而较大的进程切换和进程通信代价，又进一步影响了并发的粒度。

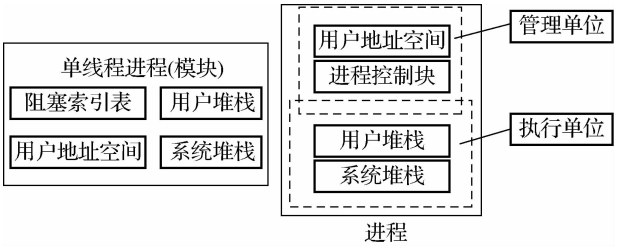

图 2-11　单线程进程的内存布局

进程有以下两个基本属性。

(1)进程是一个拥有资源的独立单元。

(2)进程同时又是一个可以被处理机独立调度和分派的基本单元。

上述两个属性构成了程序并发执行的基础。然而为了使进程能并发执行，操作系统还必须进行一系列操作，如创建进程、撤销进程和切换进程。在进行这些操作时，操作系统要为进程分配资源及回收资源，为运行进程保存现场信息，这些工作都需要较多的时间及空间开销。正因为如此，在系统中不宜设置过多的进程，进程切换的频率也不能太高，否则限制系统并发程度的进一步提高。

为使多个程序更好地并发执行，并尽量减少操作系统的开销，设想是否可以把进程的管理和执行任务分离，让进程成为操作系统中进行保护和资源分配的单位，允许一个进程中包含多个可并发执行的控制流，这些控制流切换时不必通过进程调度，通信时可以直接借助于共享内存区，每个控制流称为一个线程，这就是并发多线程程序设计。

多线程进程的内存布局如图 2-12 所示，在多线程环境中，仍然与进程相关的内容是 PCB 和用户地址空间，而每个线程除了有独立堆栈，以及包含现场信息和其他状态信息外，还要设置线程控制块(thread control block，TCB)。线程间的关系较为密切，一个进程中的所有线程共享其所属进程拥有的资源，它们驻留在相同的地址空间，可以存取相同的数据。例如，当一个线程改变了主存中的一个数据项时，如果这时其他线程也存取这个数据项，它便能看到相同的结果。

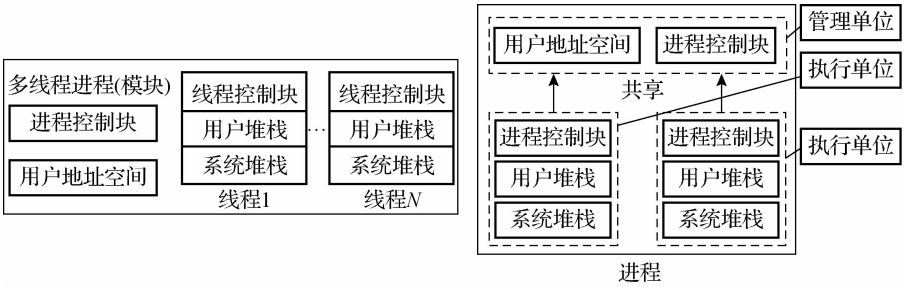

图 2-12　多线程进程的内存布局

2)线程的定义

线程的定义情况与进程类似,存在多种不同的定义方法。这些方法可以相互补充对线程的理解,下面列出了一些较权威的定义。

(1)线程是进程内的一个执行单元。

(2)线程是进程内的一个可调度实体。

(3)线程是程序(或进程)中相对独立的一个控制流序列。

(4)线程是执行的上下文,其含义是执行的现场数据和其他调度所需的信息(这种观点来自 Linux 系统)。

综上所述,我们不妨将线程定义为:线程是进程内一个相对独立的、可调度的执行单元。线程自己基本上不拥有资源,而只拥有一些在运行时必不可少的资源(如程序计数器、一组寄存器和栈),但它可以与同属于一个进程的其他线程共享进程拥有的全部资源。一个线程可以创建和撤销另一个线程;同一个进程的多个线程之间可以并发执行。线程也有就绪、阻塞和执行 3 种基本状态。一个进程至少有一个线程。

多线程是指一个进程中有多个线程,这些线程共享该进程资源,这些线程驻留在相同的地址空间中,共享数据和文件。如果一个线程修改了一个数据项,其他线程可以了解和使用此结果数据。一个线程打开并读一个文件时,同一进程中的其他线程也可以同时读此文件。

引入线程后,线程作为 CPU 调度单位,而进程只作为其他资源的分配单位。线程减小了并发执行的时间和空间开销,因此容许在系统中建立更多的线程来提高并发程度。线程的创建时间、终止时间比进程短;同一进程内的线程切换时间比进程短;由于同一进程内线程间共享内存和文件资源,可直接进行不通过内核的通信。

3)线程的实现

在操作系统中有多种方式可实现对线程的支持,最自然的方法是由操作系统内核提供线程的控制机制。在只有进程概念的操作系统中,可以由用户程序利用函数库提供线程的控制机制。还有一种做法是同时在操作系统内核和用户程序两个层次上提供线程控制机制。

线程可分为内核级线程和用户级线程。

内核级线程是指依赖于内核,由操作系统内核完成创建、撤销和切换的线程。在支持内核级线程的操作系统中,内核维护进程和线程的上下文信息并完成线程切换工作。一个内核级线程由于 I/O 操作而阻塞时,不会影响其他线程的运行。这时,处理机时间分配的对象是线程,所以有多个线程的进程将获得更多处理机时间。

用户级线程是指不依赖于操作系统核心,由应用进程利用线程库提供创建、同步、调度和管理线程的函数来控制的线程。由于用户级线程的维护由应用进程完成,不需要操作系统内核了解用户级线程的存在,因此可以用于不支持内核级线程的多用户操作系统,甚至是单用户操作系统。用户级线程切换不需要内核特权,用户级线程调度算法可针对应用优化,许多应用软件都有自己的用户级线程。由于用户级线程的调度在应用进程内部进行,通常采用非抢占式和更简单的规则,也无须用户态/核心态切换,因此速度特别快。当然,由于操作系统内核不了解用户级线程的存在,当一个线程阻塞时,整个进程都必须等待,这时处理机时间是分配给进程的,进程内有多个线程时,每个线程的执行时间相对就少一些。

有些操作系统提供了上述两种方法的组合实现。在这种系统中,内核支持多线程的建立、调度与管理;同时,系统中又提供使用线程库的便利,允许用户应用程序建立、调度和管理用户级线程。由于同时提供内核线程控制机制和用户线程库,因此可以很好地将内核级线程和用户级线程的优点结合起来。由此产生了不同的多线程模型。

(1)多对一模型。将多个用户级线程映射到一个内核级线程。线程管理在用户空间完成,因此它的效率比较高。但是,如果一个线程调用了导致阻塞的系统调用,那么将阻塞整个进程。而且,因为一次只有一个线程可以访问内核,所以在多处理机环境中多个线程不能并发执行。

(2)一对一模型。将每个用户线程映射到一个内核线程。它允许在一个线程调用导致阻塞的系统调用的情况下持续运行其他线程,从而提供了比多对一模型更好的并发性;它也允许多个线程在多处理机环境中并行执行。这种模型的唯一缺点在于创建一个用户线程就需要创建一个相应的内核线程。因为创建内核线程的开销会加重应用程序的负担,所以这种模型的实现大多数都要限制系统支持的线程数量。

(3)多对多模型。将用户级线程多路复用到与之数量相等或少一些的内核级线程。内核级线程的数量由具体的应用程序或具体的机器确定。

2.6.2 线程与进程的比较

由于进程与线程密切相关,因此有必要了解进程与线程的异同。可以从以下几个方面对它们进行比较。

(1)调度。在传统的操作系统中,拥有资源和独立调度的基本单位都是进程。在引入线程的操作系统中,线程是独立调度的基本单位,进程是资源拥有的基本单位。在同一进程中,线程的切换不会引起进程切换。在不同进程中进行线程切换,如从一个进程中的线程切换到另一个进程中的线程时,会引起进程切换。

(2)拥有资源。不论是传统操作系统还是设有线程的操作系统,进程都是拥有资源的基本单位,而线程几乎不拥有系统资源(只有一些必不可少的资源),但线程可以访问其隶属进程的系统资源。

(3)并发性。在引入线程的操作系统中,不仅进程之间可以并发执行,而且同一进程内的多个线程之间也可以并发执行,从而使操作系统具有更好的并发性,大大提高了系统的吞吐量。

(4)系统开销。由于创建进程或撤销进程时,系统都要为之分配或回收资源,如内存空间、I/O 设备等,操作系统所需的开销远大于创建或撤销线程时的开销。类似的,在进行进程切换时,涉及当前执行进程 CPU 环境的保存及新调度到进程 CPU 环境的设置,而线程切换时只需保存和设置少量寄存器内容,因此开销很小。另外,由于同一进程内的多个线程共享进程的地址空间,因此,这些线程之间的同步与通信非常容易实现,甚至无须操作系统的干预。

习　　题

1)选择题

(1)分配到必要的资源并获得处理机时的进程状态是(　　)。

A. 就绪状态　　B. 执行状态

C. 阻塞状态　　D. 撤销状态

(2)对进程的管理和控制使用(　　)。

A. 指令　　B. 信号量　　C. 信箱　　D. 原语

(3)程序的顺序执行通常在(　　)的工作环境中,具有(　　)特征;程序的并发执行在(　　)的工作环境中,具有(　　)特征。

A. 单道程序　　B. 多道程序

C. 程序的可再现性　　D. 资源共享

(4)下列进程状态变化中,(　　)变化是不可能发生的。

A. 运行→就绪　　B. 运行→等待　　C. 等待→运行　　D. 等待→就绪

(5)当(　　)时,进程从执行状态转变为就绪状态。

A. 进程被调度程序选中　　B. 等待的事件发生

C. 等待某一事件　　D. 时间片到

(6)下面对进程的描述中,错误的是(　　)。

A. 进程是动态的概念　　B. 进程执行需要处理机

C. 进程是有生命周期的　　D. 进程是指令的集合

(7)操作系统通过(　　)对进程进行管理。

A. JCB　　B. DCT　　C. PCB　　D. CHCT

(8)下面所述步骤中,(　　)不是创建进程所必需的。

A. 由调度程序为进程分配 CPU　　B. 建立一个进程控制块

C. 将进程控制块放入就绪队列　　D. 为进程分配内存

(9)多道程序环境下,操作系统分配资源以(　　)为基本单位。

A. 程序　　B. 指令　　C. 进程　　D. 作业

(10)如果系统中有 n 个进程,则就绪队列中进程的个数最多为(　　)。

A. $n+1$　　B. $n-1$　　C. 1　　D. n

(11)原语的主要特点是(　　)。

A. 并发性　　B. 异步性　　C. 共享性　　D. 不可分割性

(12)下面对父进程和子进程的叙述不正确的是(　　)。

A. 父进程创建了子进程,因此父进程执行完了子进程才能执行

B. 父进程和子进程之间可以并发执行

C. 父进程可以等待所有子进程结束后再执行

D. 撤销父进程时,可以同时撤销其子进程

(13)下列关于进程的叙述中,最不符合操作系统对进程理解的是(　　)。

A. 进程是在多程序并行环境中的完整的程序

B. 进程可以由程序、数据和进程控制块描述

C. 线程是一种特殊的进程

D. 进程是程序在一个数据集合上运行的过程,是系统进行资源分配和调度的一个独立单位

(14)当一个进程处于(　　)的状态时,称其处于等待状态。

Ⅰ. 正等待输入一批数据　　Ⅱ. 正等待协作进程的一个消息

Ⅲ. 正等待分给它一个时间片　　Ⅳ. 正等待进入内存

A. 仅Ⅰ　　B. 仅Ⅱ　　C. Ⅰ和Ⅱ　　D. Ⅰ、Ⅱ和Ⅲ

2)填空题

(1)进程的基本状态有执行、_____和_____。

(2)进程的基本特征是_____、_____、_____、_____和_____。

(3)进程由_____、_____、_____3 部分组成,其中_____是进程存在的唯一标志,而_____部分也可以为其他进程共享。

(4)进程是一个程序对某个数据集的_____。

(5)程序并发执行与顺序执行相比产生了一些新特征,分别是_____、_____和_____。

(6)设系统中有 $n(n>2)$个进程,且当前不在执行进程调度程序,试考虑下述 4 种情况。

①没有运行进程,有 2 个就绪进程,n 个进程处于等待状态。

②有 1 个运行进程,没有就绪进程,$n-1$ 个进程处于等待状态。

③有 1 个运行进程,有 1 个就绪进程,$n-2$ 个进程处于等待状态。

④有 1 个运行进程,$n-1$ 个就绪进程,没有进程处于等待状态。

上述情况中,不可能发生的情况是_____。

(7)在一个单处理机系统中,若有 5 个用户进程,且假设当前时刻为用户态,则处于就绪状态的用户进程最多有_____个,最少有_____个。

(8)下面关于进程的叙述中,不正确的有_____。

①进程申请 CPU 得不到满足时,其状态变为等待状态

②在单 CPU 系统中,任一时刻都有一个进程处于运行状态

③优先级是进行进程调度的重要依据,一旦确定不能改变

④进程获得处理机而运行是通过调度实现的

(9)程序顺序执行时的 3 个特征是_____、_____和_____。

(10)如果系统中有 n 个进程,则在等待队列中进程最多为_____个。

3)解答题

(1)进程的定义是什么? 它最少有哪几种状态?

(2)什么是管态? 什么是目态?

(3)试画出下面 4 条语句的前趋图。

S_1:a=x+2;　　S_2:b=y+4;

S_3：c＝a＋b；　　S_4：d＝c＋6；

(4)试利用 Bernstein 条件证明解答题(3)中的语句 S_1 和 S_2 可以并发执行，而语句 S_3 和 S_4 不能并发执行。

(5)进程与线程的主要区别是什么？

(6)进程控制块何时产生？何时消除？它有什么作用？

(7)已知一个求值公式$(A^2+3B)/(B+5A)$，若 A、B 已赋值，试画出该公式求值过程的前趋图。

(8)在一个分时操作系统中，进程可能出现图 2-13 所示的变化，请把产生每一种变化的具体原因填在表 2-1 的相应框中。

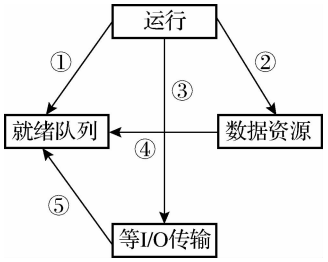

图 2-13　进程状态变化图

表 2-1　进程状态变化原因

变　化	原　因
①	
②	
③	
④	
⑤	

第 3 章 处理机调度

在计算机系统中，可能有许多批处理作业同时存放在磁盘的作业队列中，或者有许多终端与主机相连接。如何从这些作业中挑选作业进入主存运行、如何在进程之间分配处理器时间，便是处理机调度需要解决的问题。

处理机调度是多道程序系统的基础。在多道程序环境下，一个作业从提交到完成通常要经历多级调度，如高级调度、中级调度、低级调度等。处理机调度算法的优劣直接影响到整个计算机系统的性能。本章主要介绍调度算法、周转时间及相关计算、调度目标、策略及评价方法等。

3.1 分 级 调 度

在不同操作系统中所用的调度层次不完全相同。在有的系统中仅采用一级调度，而在有的系统中则可能采用两级或三级调度，在执行调度时所采用的调度算法也可能不同。图 3-1给出了调度层次的示意图，从图中可以看出，一个作业从提交开始直到完成，往往要经历三级调度：作业调度、中级调度和进程调度。

3.1.1 作业调度

作业调度又称宏观调度、高级调度或长程调度，其主要任务是按一定的原则从外存上处于后备状态的作业中选择一个或多个，给它们分配内存、输入/输出设备等必要的资源，并建立相应的进程，以使该作业具有获得竞争处理机的权利。作业调度的运行频率较低，通常为几分钟一次。

在批处理系统中或者通用操作系统中的批处理部分，新提交的作业先存放在磁盘上，因此需要作业调度，将它们分批装入内存。而在其他类型的操作系统中，通常不需要配置作业调度。

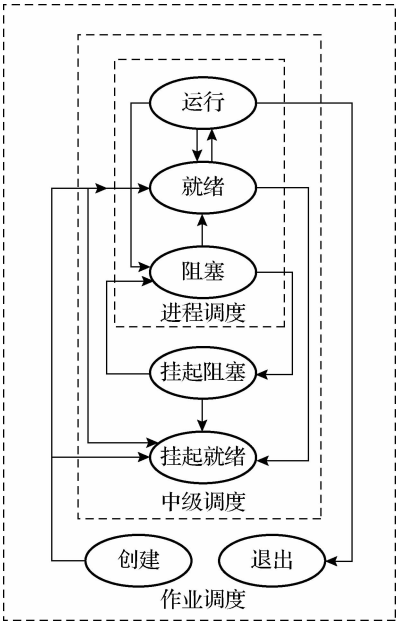

图 3-1　调度层次

3.1.2　中级调度

中级调度又称交换调度或中程调度，其主要任务是按照给定的原则和策略，将处于外存对换区中，且又具备运行条件的进程调入内存，或将处于内存的暂时不能运行的进程交换到外存对换区。中级调度的运行频率介于作业调度与进程调度之间。

引入中级调度的主要目的是提高内存的利用率和系统吞吐量，它实际起着存储器管理中的交换功能，因此中级调度的内容将在内存管理中介绍。

3.1.3　进程调度

进程调度又称微观调度、低级调度或短程调度，其主要任务是按照某种策略和方法从就绪队列中选取一个进程，将处理机分配给它。进程调度的运行频率很高，一般几十毫秒一次。

进程调度是操作系统中最基本的一种调度，在一般操作系统中都必须配置进程调度。

3.1.4　调度性能的评价

在计算机系统中，如何确定调度策略和算法受到多种因素影响，因而对调度性能的评价很复杂，但一般是抓主要矛盾并兼顾其他。

1）调度算法应达到的目标

调度算法的好坏直接影响到系统的效率。影响调度算法的因素很多，同时这些因素之

间常常互相矛盾，所以实际采用的调度算法往往依赖于系统的设计目标，系统设计目标有以下 3 种。

(1)系统的处理能力高。使系统每天运行尽可能多的作业。

(2)系统资源利用充分。使处理机保持忙碌状态，以达到充分利用系统资源的目的。

(3)算法对所有的作业公平合理。使所有用户感到满意。

由于这些目标往往相互冲突，任何一个调度算法想要同时满足上述目标是不可能的。例如，要想执行尽可能多的作业，调度算法就应选择那些估计执行时间短的作业，而这对于那些估计执行时间长的作业不公平。由此看出，要设计一个理想的调度算法是一件很困难的事。因此，实际采用的调度算法往往是根据需要而兼顾某些目标。

2)确定调度算法时应考虑的因素

(1)设计目标。系统选择的调度算法应与系统的总体设计目标一致。例如，批处理系统应尽量提高系统的平均吞吐量；分时系统应保证用户所能忍受的响应时间；而实时系统则应在保证及时响应和处理有关事件的前提下，再去考虑系统资源的利用率。

(2)资源使用的均衡性。注意系统资源的均衡使用，使输入/输出繁忙的作业与 CPU 繁忙的作业搭配运行。

(3)平衡系统和用户的要求。由于系统和用户的要求往往是矛盾的对立双方，确定算法时要尽量缓和双方的矛盾。例如，任何用户都希望作业一进入系统就立即执行，从而很快得到计算结果，然而系统却往往不能满足用户的这一愿望和要求。因此，应保证进入系统的作业在规定的截止时间内完成，而系统应设法缩短作业的平均周转时间。

应该指出，对于一个特定的系统来说，如果考虑的因素过多，势必会使调度算法变得很复杂，从而增加系统开销，对提高系统资源利用率反而不利。因此，大多数操作系统往往采用比较简单且行之有效的调度算法。

3)调度性能的评价准则

不同调度算法有不同的特性。一种算法可能有利于某类作业或进程的运行，而不利于其他类作业或进程的运行。在选择调度算法时，必须考虑各种算法所具有的特性。为了比较处理机调度算法的性能，人们提出很多评价准则，下面介绍几种主要的评价准则。

(1)CPU 利用率。CPU 是计算机系统中最重要最昂贵的资源之一，其利用率是评价调度算法的重要指标。在实际系统中，一般 CPU 的利用率在 40%～90%。

(2)系统吞吐量。系统吞吐量表示单位时间内 CPU 完成作业的数量。对于长作业来说，由于它们要消耗较长的处理机时间，因此会造成系统的吞吐量下降。而对于短作业来说，它们所需消耗的处理机时间较短，因此系统的吞吐量会提高。但调度算法和方式的不同，也会对系统的吞吐量产生较大影响。

(3)周转时间。从一个特定作业的观点出发，最重要的准则就是完成这个作业要花费多长时间，通常用周转时间或带权周转时间来衡量。

①作业的周转时间是指从作业提交到作业完成之间的时间间隔。作业 i 的周转时间 T_i 可用公式表示如下。

$$T_i = T_{ei} - T_{si}$$

其中 T_{ei} 为作业 i 的完成时间，T_{si} 为作业 i 的提交时间。

②平均周转时间是指多个作业周转时间的平均值。n 个作业的平均周转时间 T 可用如下公式表示。

$$T=(T_1+T_2+\cdots+T_n)/n$$

③带权周转时间是指作业周转时间与作业实际运行时间的比。作业 i 的带权周转时间 W_i 可用如下公式表示。

$$W_i=T_i/T_{r_i}$$

其中 T_i 为作业 i 的周转时间，T_{r_i} 为作业 i 的实际运行时间。

④平均带权周转时间是指多个作业带权周转时间的平均值。n 个作业的平均带权周转时间 W 可用公式表示如下。

$$W=(W_1+W_2+\cdots+W_n)/n$$

(4)响应时间。在交互式系统中，周转时间不可能是最好的评价准则，一般采用响应时间作为衡量调度算法的重要准则之一。响应时间是指从用户提交请求到系统首次产生响应所用的时间。从用户角度看，调度策略应尽量降低响应时间，使响应时间控制在用户能接受的范围之内。

3.2 作 业 调 度

作业是用户在一次解题或一个事务处理过程中要求计算机系统所做工作的集合，包括用户程序、所需的数据及命令等。计算机系统在完成一个作业的过程中所做的一项相对独立的工作称为一个作业步，因此，也可以说一个作业是由一系列有序的作业步组成的。例如，我们在编制程序的过程中，通常要进行编辑输入、编译、链接、运行等几个步骤，其中的每一个步骤都可以看做一个作业步。

3.2.1 作业的状态及转换

一个作业从进入系统到运行结束，一般需要经历提交、收容、运行和完成 4 个阶段。与这 4 个阶段相对应的作业处于提交、后备、运行和完成 4 种状态。作业的状态及其转换可用图 3-2 表示。

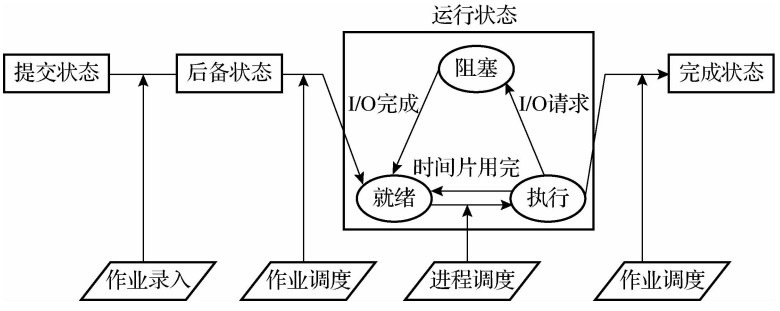

图 3-2　作业状态转换图

1)提交状态

用户为了上机解题或进行某项事务处理,必须事先准备好自己的作业,然后将作业通过键盘等输入设备提交给计算机系统。用户作业由输入设备向系统外存输入时作业所处的状态称为提交状态。

2)后备状态

当一个作业通过输入设备送入计算机,并由操作系统将其存放在磁盘中后,系统为这个作业建立一个作业控制块,并把它插入后备作业队列中等待调度运行。此时,这个作业所处的状态称为后备状态。从作业输入开始到放入后备作业队列这一过程称为收容阶段,也称为作业注册。

3)运行状态

当作业调度程序选中一个作业,为它分配了必要的资源并建立相应的进程之后,这个作业就由后备状态转变为运行状态。

处于运行状态的作业在系统中并不一定真正占有处理机,它可能被进程调度程序选中而得到处理机,正在处理机上执行;也可能在等待某事件的发生;还有可能在等待进程调度程序为其分配处理机。因此,从宏观上看,作业一旦由作业调度程序选中进入内存就开始了运行,但从微观上讲,内存中的作业并不一定正在处理机上执行。为了便于对处于运行状态的作业进行管理,根据进程的活动情况又把运行状态分为 3 种:就绪状态、执行状态和阻塞状态。

刚建立的进程进入就绪状态;从就绪状态向执行状态的转换由进程调度程序实现;处于执行状态的进程,当它使用完分配给它的时间或被更高优先级的进程剥夺处理机后,又回到就绪状态,等待下次被调度;进程在执行中若发生了某事件而暂时无法执行下去,如有输入/输出请求并等待输入/输出完成,则进入阻塞状态;当引起进程阻塞的事件消失时,如输入/输出完成,进程由阻塞状态变为就绪状态,重新获得被调度的资格。

4)完成状态

当作业正常运行结束或因发生错误而终止运行时,作业就处于完成状态。此时,由操作系统将作业控制块从当前作业队列中删除,并收回其所占用的资源,将作业运行结果存入输出文件并调用有关设备进行输出。在 Spooling 系统中,作业将被插入完成作业队列中,将运行结果送入输出井,再由 Spooling 系统完成输出。

3.2.2 作业调度程序

作业调度的主要功能是按照某种原则从后备作业队列中选取作业进入内存,并为作业做好运行前的准备工作和作业完成后的善后处理工作,完成这种功能的程序称为作业调度程序。

1)作业调度程序的功能

作业调度程序主要完成以下工作。

(1)记录进入系统的各个作业情况。为了挑选作业投入执行并在执行过程中对作业进行管理,作业调度程序必须掌握进入系统的作业情况,并随时记录作业在运行阶段的变化情况。为此,系统应为每个作业建立相应的数据结构。

(2)从后备作业中挑选一些作业投入执行。一般来说,系统中处于后备状态的作业较多,有几十甚至几百个。后备作业的多少取决于存储后备作业的空间大小。但是处于运行状态(不是真正在 CPU 上执行)的作业一般只有有限的几个,如 4 个、10 个。因此,作业调度程序的一个重要职能就是,适时按确定的调度策略从后备作业中选取若干作业进入运行状态。作业调度的调度策略和调度时机通常与系统的设计目标有关,并由许多因素决定。为此,在设计作业调度程序时,必须综合平衡各种因素,确定符合系统设计目标的调度算法。

(3)为选中的作业做好执行前的准备工作。作业调度程序在让一个作业从后备状态进入运行状态之前,必须为该作业建立相应的进程,为其分配运行所需要的资源,分配的资源包括内存、磁盘空间和外设等。

(4)在作业运行结束或运行过程中因某种原因需要撤离时,作业调度程序还要完成作业的善后处理工作。例如,作业调度程序要把相应作业的一些信息(如运行时间、作业执行情况等)进行必要的输出,然后收回该作业所占用的一切资源,撤销与该作业有关的全部进程和该作业的作业控制块。

2)作业控制块

在外存中往往有许多作业,为了管理和调度这些作业,就必须记录已进入系统中的各作业的情况。与进程管理一样,系统为每个作业设置一个作业控制块(JCB),其中记录了作业的有关信息。不同系统的 JCB 所包含的信息有所不同,这取决于系统对作业调度的要求。通常作业控制块中包括的主要内容如下。

(1)资源要求。资源要求是指作业运行所需要的资源情况,包括估计运行时间、最迟完成时间、需要的内存容量、外设类型及数量等。

(2)资源使用情况。资源使用情况包括作业进入系统的时间、开始运行时间、已运行时间、内存地址、外设台号等。

(3)作业的控制方式、类型和优先级等。作业的控制方式有联机作业控制(又称为直接控制)和脱机作业控制(又称为自动控制两种)。从不同角度出发可以对作业进行不同的分类,如根据用户是否直接与系统交互可以将作业分为终端型和批量型,根据作业需要 CPU 和设备时间量的情况可以将作业分为 I/O 繁忙型和 CPU 繁忙型。作业的优先级是指作业进入系统运行的优先级,优先级高的作业可以优先进入系统运行。

(4)作业名、作业状态。记录作业的标识信息及作业的当前状态。

通常,系统为每个作业建立一个作业控制块,它是作业存在的唯一标志。系统通过 JCB 感知作业的存在。系统在作业进入后备状态时为作业建立 JCB,从而使该作业可以被作业调度程序所感知。当作业运行完毕进入完成状态之后,系统撤销其 JCB,释放有关资源并撤销该作业。

3.3　进 程 调 度

在多道程序系统中,用户进程数目往往多于处理机的个数,这使得进程为了运行而相互争夺处理机。此外,系统进程也同样需要使用处理机。因此,操作系统需要按一定的策略动

态地把处理机分配给就绪队列中的某个进程，以便让它执行。处理机分配的任务由进程调度程序完成。

3.3.1 进程调度程序

进程调度程序主要完成下面几种功能。

1)记录系统中所有进程的有关情况及状态特征

为了实现进程调度，必须将系统中各进程的执行情况和状态特征记录在各进程的 PCB 中，同时还应根据各进程的状态特征和资源需求等信息将进程的 PCB 组织成相应的队列，并依据运行情况将进程的 PCB 在不同状态队列之间进行转换。进程调度程序通过 PCB 的变化来掌握系统中所有进程的执行情况和状态特征。

2)选择获得处理机的进程

按照一定的策略选择一个处于就绪状态的进程，使其获得处理机执行。根据不同的系统设计目标，有多种选择策略，如先来先服务、时间片轮转等，这些选择策略决定了调度算法的性能。

3)处理机分配

当正在执行的进程由于某种原因要放弃处理机时，进程调度程序应保护当前执行进程的 CPU 现场，将其状态由执行变成就绪或阻塞，并插入相应队列中；同时，进程调度程序还应根据一定原则从就绪队列中挑选出一个进程，将该进程从就绪队列中移出，恢复其 CPU 现场，并将其状态改为执行。

引起进程调度的原因有以下几种。

(1)当前运行进程运行结束。因任务完成而正常结束，或者因出现错误而异常结束。

(2)当前运行进程因某种原因，如 I/O 请求，从运行状态进入阻塞状态。

(3)当前运行进程执行某种原语操作，如 P 操作、阻塞原语等，进入阻塞状态。

(4)执行完系统调用等系统程序后返回用户进程，这时可以看做是系统进程执行完毕，从而可以调度一个新的用户进程。

(5)在采用剥夺调度方式的系统中，一个具有更高优先级的进程要求使用处理机，则使当前运行进程进入就绪队列(这与调度方式有关)。

(6)在分时系统中，分配给该进程的时间片已用完(这与系统类型有关，多用于分时系统中)。

3.3.2 进程调度方式

进程调度方式是指当某一个进程正在处理机上执行时，有某个更为重要或紧迫的进程需要进行处理(即有更高优先级的进程进入就绪队列)，此时应如何分配处理机。通常有以下两种进程调度方式。

1)抢占方式

抢占方式又称剥夺方式、可剥夺方式、可抢占方式，这种进程调度方式是指当一个进程

正在处理机上执行时，若有某个更为重要或紧迫的进程需要使用处理机，则立即暂停正在执行的进程，将处理机分配给更为重要或紧迫的进程。

2)非抢占方式

非抢占方式又称非剥夺方式、不可剥夺方式、不可抢占方式。这种进程调度方式是指当某一个进程正在处理机上执行时，即使有某个更为重要或紧迫的进程进入就绪队列，仍然让正在执行的进程继续执行，直到该进程完成或发生某种事件而进入阻塞状态时，才把处理机分配给更为重要或紧迫的进程。

3.4 调 度 算 法

通常系统的设计目标不同，所采用的调度算法也不相同。在操作系统中存在多种调度算法，其中有的调度算法适用于作业调度，有的调度算法适用于进程调度，有的调度算法两者都适用。下面介绍几种常用的调度算法。

3.4.1 先来先服务调度算法

先来先服务调度算法是最简单的一种调度算法，该调度算法既可以用于作业调度，也可以用于进程调度。

在作业调度中，先来先服务调度算法每次从后备作业队列中选择最先进入该队列的一个或几个作业，将它们调入内存，为其分配必要的资源，创建进程并放入就绪队列。

在进程调度中，先来先服务调度算法每次从就绪队列中选择最先进入该队列的进程，将处理机分配给它，使之投入运行，该进程一直运行下去，直到完成或因某种原因而阻塞时才释放处理机。

下面通过一个例子来说明先来先服务调度算法的性能。假设系统中有 4 个作业，它们的提交时间、运行时间、开始时间、完成时间、周转时间和带权周转时间如表 3-1 所示。

表 3-1　先来先服务调度算法示例

时间 / 作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	8	2	8	10	2	1
2	8.4	1	10	11	2.6	2.6
3	8.8	0.5	11	11.5	2.7	5.4
4	9	0.2	11.5	11.7	2.7	13.5

系统采用先来先服务调度算法，其平均周转时间和平均带权周转时间如下。

平均周转时间 $T=2.5$。

平均带权周转时间 $W=5.625$。

从表面上看，先来先服务调度算法对所有作业都是公平的，即按照作业到来的先后次序进行服务。但若一个长作业先到达系统，就会使许多短作业等待很长时间，从而引起许多短作业用户的不满。目前，先来先服务调度算法已很少用做主要的调度策略，尤其是不能作为分时系统和实时系统的主要调度策略，但它常被结合在其他调度策略中使用。例如，在使用优先级作为调度策略的系统中，往往对多个具有相同优先级的进程按先来先服务原则处理。该算法优先考虑在系统中等待时间最长的作业，而不考虑作业运行时间的长短。

先来先服务调度算法的特点是算法简单，但效率较低；有利于长作业但对短作业不利；有利于 CPU 繁忙型作业而不利于 I/O 繁忙型作业。

3.4.2 短作业优先调度算法

短作业优先调度算法用于进程调度时称为短进程优先调度算法，该调度算法既可以用于作业调度，也可以用于进程调度。

在作业调度中，短作业优先调度算法每次从后备作业队列中选择估计运行时间最短的一个或几个作业，将它们调入内存，分配必要的资源，创建进程并放入就绪队列。

在进程调度中，短进程优先调度算法每次从就绪队列中选择估计运行时间最短的进程，将处理机分配给它，使之投入运行，该进程一直运行下去，直到完成或因某种原因而阻塞时才释放处理机。

对表 3-2 给出的一组作业采用短作业优先调度算法。

表 3-2 短作业优先调度算法示例

时间 / 作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	8	2	8	10	2	1
2	8.4	1	10.7	11.7	3.3	3.3
3	8.8	0.5	10.2	10.7	1.9	3.8
4	9	0.2	10	10.2	1.2	6

其平均周转时间和平均带权周转时间如下。

平均周转时间 $T=2.1$。

平均带权周转时间 $W=3.525$。

与先来先服务调度算法相比，短作业优先调度算法具有较短的平均周转时间和平均带权周转时间，具有较好的调度性能，但该算法对长作业不利。

短作业优先调度算法可以是非抢占式的，也可以是抢占式的。若无特别说明，通常是指非抢占式的算法。抢占式的短作业优先调度算法也称为最短剩余时间优先调度算法，即当一个新进程进入就绪队列时，若它需要的运行时间比当前运行进程的剩余时间短，则将抢占 CPU。

对表 3-3 所示的一组作业采用最短剩余时间优先调度算法。

表 3-3　最短剩余时间优先调度算法示例

时间 / 作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	0	8	0	17	17	2.125
2	1	4	1	5	4	1
3	2	9	17	26	24	2.67
4	3	5	5	10	7	1.4

其平均周转时间和平均带权周转时间如下。

平均周转时间 $T=13$。

平均带权周转时间 $W=1.8$。

需要指出的是，由于用户比较难以准确地估计出作业或进程的运行时间，所以该算法较难真正做到短作业（短进程）优先调度。

可以证明，只有当一批作业同时到达时，最短作业优先调度算法才能获得最短平均周转时间。

3.4.3　优先级调度算法

优先级调度算法也称为优先权调度算法，该算法既可用于作业调度，也可用于进程调度，该算法中的优先级用于描述作业运行的紧迫程度。

1）优先级调度算法的实现

在作业调度中，优先级调度算法每次从后备作业队列中选择优先级最高的一个或几个作业，将它们调入内存，分配必要的资源，创建进程并放入就绪队列。

在进程调度中，优先级调度算法每次从就绪队列中选择优先级最高的进程，将处理机分配给它，使之投入运行。根据进程调度方式的不同，又可以将优先级调度算法分为非抢占式优先级调度算法和抢占式优先级调度算法。

非抢占式优先级调度算法的实现思想是系统一旦将处理机分配给就绪队列中优先级最高的进程，该进程便一直运行下去，直到由于其自身的原因（任务完成或等待事件）主动让出处理机时，才将处理机分配给另一个优先级高的进程。

抢占式优先级调度算法的实现思想是将处理机分配给优先级最高的进程，使之运行。在进程运行过程中，一旦出现了另一个优先级更高的进程（如一个处于阻塞状态的高优先级进程因事件的到来而变为就绪状态），进程调度程序就停止当前进程的运行，而将处理机分配给新出现的高优先级进程。

2）进程的优先级

进程的优先级用于表示进程的重要性及运行的优先性，一般用优先数来衡量优先级。在一些系统中，优先数越大代表优先级越高；而在另一些系统中，优先数越小代表优先级越高。根据进程创建后其优先级是否可以改变，将进程优先级分为静态优先级和动态优先级两种。

静态优先级是在创建进程时确定的，确定之后在整个进程运行期间不再改变。确定静

态优先级的主要依据有以下几种。

(1)进程类型。通常系统中有系统进程和用户进程两类进程。各进程运行速度以及系统资源的利用率在很大程度上依赖于系统进程。例如,若系统中某种共享输入/输出设备由一系统进程管理,那么使用这种设备的所有进程的运行速度都依赖于这一系统进程。因此,系统进程的优先级应高于用户进程。在批处理与分时相结合的系统中,为了保证分时用户的响应时间,前台作业的进程优先级应高于后台作业的进程。

(2)进程对资源的要求。根据作业要求系统提供的处理机时间、内存大小、I/O 设备的类型及数量来确定作业的优先级。由于作业的执行时间事先难以确定,所以只能根据用户提出的估计时间来确定。进程所申请的资源越多,估计的运行时间越长,进程的优先级就越低。

(3)用户要求。系统可以按用户提出的要求设置进程优先级,为防止用户将自己的进程都设置为高优先级,可以采用高优先级高收费的策略。

动态优先级是指在创建进程时,根据进程的特点及相关情况确定一个优先级,在进程运行过程中再根据情况的变化调整优先级。确定动态优先级的主要依据有以下几种。

(1)进程占用 CPU 的时间。一个进程占用 CPU 的时间越长,则优先级越低,再次获得调度的可能性就越小;反之,一个进程占用 CPU 的时间越短,则优先级越高,获得调度的可能性就越大。

(2)进程等待 CPU 的时间。一个进程在就绪队列中等待的时间越长,则优先级越高,获得调度的可能性就越大;反之,一个进程在就绪队列中等待的时间越短,则优先级越低,获得调度的可能性就越小。

3.4.4 时间片轮转调度算法

时间片轮转调度算法主要用于分时系统中的进程调度。在时间片轮转调度算法中,系统将所有就绪进程按到达时间的先后次序排成一个队列,进程调度程序总是选择就绪队列中的第一个进程执行,并规定执行一定时间,如 100 ms,该时间称为时间片。当该进程用完这一时间片时(即使进程并未完成其运行),系统将它送至就绪队列队末尾,再把处理机分配给就绪队列的队首进程。这样,处于就绪队列中的进程就可以依次轮流地获得一个时间片的处理时间,然后回到队列尾部,如此不断循环,直至完成为止。

例如,表 3-4 所示的 A、B、C、D、E 5 个进程,到达时间分别为 0、1、2、3、4,要求运行时间依次为 3、6、4、5、2,采用时间片轮转调度算法,时间片大小 q 为 1。

表 3-4 时间片轮转调度算法示例($q=1$)

进程名	到达时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3	0	7	7	2.33
B	1	6	1	19	18	3
C	2	4	3	16	14	3.5
D	3	5	5	20	17	3.4
E	4	2	7	12	8	4

其平均周转时间和平均带权周转时间如下。

平均周转时间 $T=12.8$。

平均带权周转时间 $W=3.25$。

对于表 3-5 所示的 A、B、C、D、E 5 个进程，到达时间分别为 0、1、2、3、4，要求运行时间依次为 3、6、4、5、2，采用时间片轮转调度算法，时间片大小 q 为 4。

表 3-5　时间片轮转调度算法示例($q=4$)

进程名	到达时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3	0	3	3	1
B	1	6	3	19	18	3
C	2	4	7	11	9	2.25
D	3	5	11	20	17	3.4
E	4	2	15	17	13	6.5

其平均周转时间和平均带权周转时间如下。

平均周转时间 $T=12$。

平均带权周转时间 $W=3.23$。

从表 3-4 和表 3-5 中可以看出，时间片轮转调度算法本质上是一种可剥夺式调度算法，时间片的大小对系统性能的影响很大。如果时间片足够大，以至于所有进程都能在一个时间片内执行完，则时间片轮转调度算法就退化成先来先服务调度算法。如果时间片很小，那么处理机将在进程之间频繁切换，处理机真正用于运行用户进程的时间将减少。因此，时间片的大小应选择适当。

时间片的长短通常由以下因素确定。

(1)系统的响应时间。分时系统必须满足系统对响应时间的要求，系统响应时间与时间片的关系可以表示为 $T=Nq$，其中，T 为系统的响应时间，q 为时间片的大小，N 为就绪队列中的进程数。若系统中的进程数目一定，时间片的大小与系统响应时间成正比。

(2)就绪队列中的进程数目。在响应时间固定的情况下，就绪队列中的进程数目与时间片成反比。

(3)系统的处理能力。通常要求用户输入的常用命令能够在一个时间片内处理完毕。因此，计算机的速度越快，时间片就越短。

3.4.5　高响应比优先调度算法

先来先服务调度算法只考虑作业的等待时间而未考虑作业运行时间，短作业优先调度算法只考虑作业运行时间而忽略了作业的等待时间，这两种调度算法都有不足之处，为此引入了高响应比优先调度算法。

高响应比优先调度算法主要用于作业调度，该算法是对先来先服务调度算法和短作业优先调度算法的一种综合平衡，该算法既考虑作业运行时间，也考虑作业的等待时间。高响应比优先调度算法的实现思想是在每次进行作业调度时，先计算后备作业队列中每个作业

的响应比，然后挑选响应比最高的作业投入运行。响应比的定义如下。

响应比＝作业响应时间/估计运行时间

由于响应时间为作业进入系统后的等待时间加上估计运行时间。因此：

响应比＝1＋作业等待时间/估计运行时间

对表 3-6 所示的进程采用高响应比优先调度算法。

表 3-6　高响应比优先调度算法示例

进程名	到达时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3	0	3	3	1
B	1	6	3	9	8	1.33
C	2	4	11	15	13	3.25
D	3	5	15	20	17	3.4
E	4	2	9	11	7	3.5

其平均周转时间和平均带权周转时间如下。

平均周转时间 $T=9.6$。

平均带权周转时间 $W=2.496$。

从响应比的计算公式中可以看出，该算法有利于短作业，同时适当考虑长作业。也就是说，在相同等待时间的情况下，短作业优先；而在相同运行时间的情况下，等待时间长的作业优先，这样只要系统中的某个作业等待了足够长的时间，它就会成为响应比最高者而获得运行的机会。该算法的不足之处在于作业调度前需要计算后备队列中每个作业的响应比，从而增加了系统开销。

3.4.6　多级队列调度算法

多级队列调度算法用于进程调度，其实现思想是根据进程的性质或类型，将就绪队列划分为若干子队列，每个进程固定属于一个就绪队列，每个就绪队列采用一种调度算法，不同的队列可以采用不同的调度算法。例如，为交互型作业设置一个就绪队列，该队列采用时间片轮转调度算法；为批处理作业设置另一个就绪队列，该队列采用先来先服务调度算法。

3.4.7　多级反馈队列调度算法

多级反馈队列调度算法是时间片轮转调度算法和优先级调度算法的综合和发展。通过动态调整进程优先级和时间片大小，多级反馈队列调度算法可以兼顾多方面的系统目标。例如，为提高系统吞吐量和缩短平均周转时间而照顾短进程；为获得较好的 I/O 设备利用率和缩短响应时间而照顾 I/O 型进程；同时，也不必事先估计进程的执行时间。

多级反馈队列调度算法的示意图如图 3-3 所示。

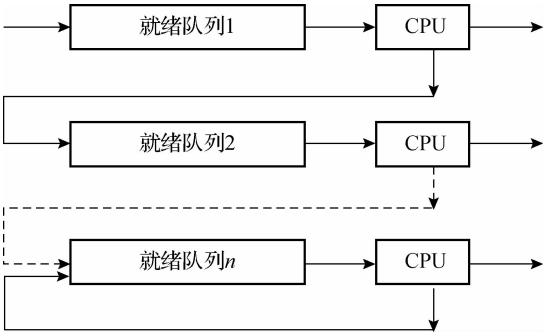

图 3-3　多级反馈队列调度算法的示意图

多级反馈队列调度算法实现思想如下。

(1)系统中应设置多个就绪队列,每个就绪队列对应一个优先级,第一个队列的优先级最高,第二个队列次之,其余队列的优先级逐次降低。

(2)每个队列中进程的时间片大小各不相同,进程所在队列的优先级越高,其相应的时间片就越短。

(3)当一个新进程进入系统时,首先将它放入第一个队列的末尾,按先来先服务的原则排队等待调度。当轮到该进程执行时,如能在此时间片内完成,便可准备撤离系统;如果它在一个时间片结束时尚未完成,调度程序便将该进程转入第二个队列的末尾,再同样地按先来先服务原则等待调度执行;如果它在第二个队列中运行一个时间片后仍未完成,再以同样方法转入第三个队列。如此下去,最后一个队列中使用时间片轮转调度算法。

(4)仅当第一个队列为空时,调度程序才从第二个队列中选择进程运行;仅当第一个队列至第(i−1)个队列均为空时,才会调度第 i 个队列中的进程运行。当处理机正在执行第 i 个队列中的某进程时,若又有新进程进入优先级较高的队列中,则此时新进程将抢占正在运行进程的处理机,即由调度程序把正在执行的进程放回第 i 个队列末尾,重新将处理机分配给优先级更高的新进程。

对表 3-7 所示的进程采用多级反馈队列调度算法。设系统中建立了 3 个队列,第一个队列的时间片 q 为 1,第二个队列的时间片 q 为 2,第三个队列的时间片 q 为 4。

表 3-7　多级反馈队列调度算法示例

进程名	到达时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
A	0	3	0	9	9	3
B	1	8	1	27	26	3.25
C	3	4	3	20	17	4.25
D	4	5	4	22	18	3.6
E	5	7	5	26	21	3

其平均周转时间和平均带权周转时间如下。

平均周转时间 T=18.25。

平均带权周转时间 W=3.42。

在实际系统中使用多级反馈队列调度算法时，还可以使用更复杂的优先级调整策略，以适应特定系统的需要。一般而言，多级反馈队列调度算法具有较好的性能，如对于终端型作业，由于这类作业需要的处理机时间较短，因而能够在前一两个队列中完成，从而保证了终端型作业具有较快的响应时间；基于相同的原因，短作业能在前几个队列中完成，因而其周转时间较短；而长作业可以依次在各队列中得到服务。当然，随着长作业的运行，长作业会逐渐降到较低优先级的进程队列中，若在此之后系统中进入的都是运行时间较短的作业，并且形成了稳定的短作业流，则前述的长作业就可能一直就绪等待，从而出现进程饥饿问题。解决饥饿问题的一种方法是，提高低优先级队列中等待了很长时间的进程的优先级，从而使它们得到再次运行的机会。

3.4.8 公平分享调度算法

公平分享调度算法基于进程组来分配 CPU 时间，其实现思想是对系统中的每个用户赋予某种权值，根据用户权值大小，按比例分配处理机时间。

公平分享调度有许多实现方案，这里讲述 UNIX 系统中的一种实现方案。

UNIX 系统基于优先权调度，优先数越大优先权越低。对进程组 k 中进程 j 的优先数计算公式如下。

$$\mathrm{CPU}_j(i)=\mathrm{CPU}_j(i-1)/2$$

$$\mathrm{GCPU}_k(i)=\mathrm{GCPU}_k(i-1)/2$$

$$\mathrm{P}_j(i)=\mathrm{Base}_j+\mathrm{CPU}_j(i)/2+\mathrm{GCPU}_k(i)/4W_k$$

其中，$\mathrm{CPU}_j(i)$为进程 j 在时间段 i 使用的 CPU 时间；$\mathrm{GCPU}_k(i)$为进程组 k 在时间段 i 使用的 CPU 时间；$\mathrm{P}_j(i)$为进程 j 在时间段 i 的优先数，其值越小表示其优先权越高，Base_j 为进程 j 的基本优先数，W_k 为进程组 k 的权值，且满足下面两个约束条件。

(1) $0\leqslant W_k\leqslant 1$

(2) $\sum_k W_k = 1$

下面用表 3-8 所示的示例介绍公平分享调度算法的思想。系统中有两组进程，第一组中只有一个进程 A，第二组中有进程 B 和 C，每组的 W_k 均为 0.5，Base 为 60。假设所有进程都只使用 CPU，并已处于就绪状态。CPU 的使用情况按下述规则进行：处理机每秒钟中断 60 次；每次中断时，当前正在运行的进程的 CPU 计数和相应的 GCPU 计数都加 1，每秒结束时重新计算优先权。

表 3-8 公平分享调度算法示例

	进程 A			进程 B			进程 C		
时间/s	优先数	CPU 计数	GCPU 计数	优先数	CPU 计数	GCPU 计数	优先数	CPU 计数	GCPU 计数
0	60	0 1 2 … 60	0 1 2 … 60	60	0	0	60	0	0

续表

	进程 A			进程 B			进程 C		
时间/s	优先数	CPU 计数	GCPU 计数	优先数	CPU 计数	GCPU 计数	优先数	CPU 计数	GCPU 计数
1	90	30	30	60	0 1 2 … 60	0 1 2 … 60	60	0	0 1 2 … 60
2	74	15 16 17 … 75	15 16 17 … 75	90	30	30	75	0	30
3	96	37	37	74	15	15 16 17 … 75	67	0 1 2 … 60	15 16 17 … 75
4	78	18 19 20 … 78	18 19 20 … 78	81	7	37	93	30	37
5	98	39	39	70	3	18	76	15	18

从表 3-8 中可以看出内核的调度顺序为 A、B、A、C、A、B、A、C，如此下去，实现了进程组之间、同组内的进程之间公平分享 CPU 时间的目的。该算法的优点是系统中的所有进程都是平等的，可满足不同性质的进程对运行时间的需要。

习　　题

1)选择题

(1)在分时操作系统中，进程调度经常采用(　　)调度算法。

A. 先来先服务　　B. 优先级

C. 短作业优先　　D. 时间片轮转

(2)(　　)优先权是在创建进程时确定的，确定之后在整个进程运行期间不再改变。

A. 作业　　B. 静态　　C. 动态　　D. 资源

(3)(　　)是作业存在的唯一标志。

A. 作业控制块　　B. 作业名　　C. 进程控制块　　D. 进程名

(4)设有 4 个作业同时到达,每个作业的运行时间均为 2 h,它们在一台处理机上按单道方式运行,则平均周转时间为(　　)。

A. 1 h　　B. 5 h　　C. 2.5 h　　D. 8 h

(5)现有 3 个同时到达的作业 J_1、J_2 和 J_3,它们的运行时间分别是 T_1、T_2 和 T_3,且 $T_1<T_2<T_3$。系统按单道方式运行且采用短作业优先调试算法,则平均周转时间是(　　)。

A. $T_1+T_2+T_3$　　B. $(T_1+T_2+T_3)/3$

C. $(3T_1+2T_2+T_3)/3$　　D. $(T_1+2T_2+3T_3)/3$

(6)(　　)是指从作业提交给系统到作业完成的时间间隔。

A. 运行时间　　B. 响应时间　　C. 等待时间　　D. 周转时间

(7)下述作业调度算法中,(　　)调度算法与作业的估计运行时间有关。

A. 先来先服务　　B. 多级队列　　C. 短作业优先　　D. 时间片轮转

2)填空题

(1)进程的调度方式有两种,一种是_____,另一种是_____。

(2)在_____调度算法中,按照进程进入就绪队列的先后次序来分配处理机。

(3)采用时间片轮转调度算法时,时间片过大,就会使时间片轮转法转化为_____调度算法。

(4)一个作业可以分成若干顺序处理的加工步骤,每个加工步骤称为一个_____。

(5)作业生存期共经历 4 个状态,分别是_____、_____、_____和_____。

(6)既考虑作业等待时间,又考虑作业执行时间的调度算法是_____。

3)解答题

(1)单道批处理系统中有 4 个作业,其有关情况如表 3-9 所示。在采用高响应比优先调度算法时分别计算其平均周转时间 T 和平均带权周转时间 W(运行时间为小时,按十进制计算)。

表 3-9　作业的提交时间和运行时间

作业	J_1	J_2	J_3	J_4
提交时间	8.0	8.6	8.8	9.0
运行时间	2.0	0.6	0.2	0.5

(2)什么是 JCB? 其作用是什么? JCB 至少包括哪些内容?

(3)在单 CPU 和两台输入/输出设备(I_1,I_2)多道程序设计环境下,同时有 3 个作业 J_1、J_2 和 J_3 运行。这 3 个作业使用 CPU 和输入/输出设备的顺序和时间如下。

J_1:I_2(30 ms),CPU(10 ms),I_1(30 ms),CPU(10 ms),I_2(20 ms)

J_2:I_1(20 ms),CPU(20 ms),I_2(40 ms)

J_3:CPU(30 ms),I_1(20 ms),CPU(10 ms),I_1(10 ms)

假定 CPU、I_1 和 I_2 都能并行工作,J_1 优先级最高,J_2 次之,J_3 最低,优先级高的作业可以

抢占优先级低的作业的 CPU，但不能抢占 I_1、I_2。试求：

①3 个作业从开始到完成分别需要多少时间？

②从开始到完成的 CPU 利用率。

③每种 I/O 设备的利用率。