第 1 章　操作系统及 Linux 简介

操作系统(operating system,OS)是计算机的重要系统软件,它的功能是实现计算机软、硬件资源管理和隐藏计算机硬件操作细节,并以应用程序接口(application programming interface, API)的方式,向应用程序提供通用底层服务,为应用程序搭建一个友好的开发和运行平台。

本章主要内容:

(1) 什么是计算机操作系统;
(2) 操作系统的地位与功能;
(3) 操作系统的引导和装入;
(4) 操作系统与应用程序之间的关系;
(5) 宏内核和微内核的基本概念;
(6) 操作系统分类;
(7) Linux 基础知识。

1.1　概　　述

1.1.1　什么是计算机操作系统

众所周知,计算机系统由硬件和软件两部分组成。硬件(诸如中央微处理器、存储器、接口及外部设备等)是计算机的物质基础,它以指令集的形式向软件提供了操作接口;软件则利用这个接口,通过指令序列(程序)操作硬件来完成特定的计算任务。

计算机硬件系统、指令系统和软件系统之间的关系如图 1-1 所示。

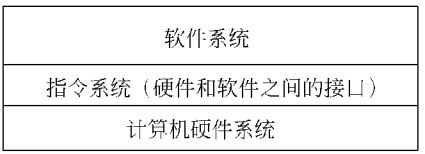


图 1-1　计算机硬件系统、指令系统和软件系统之间的关系

目前,计算机软件大致可分为两大类:一类是直接与计算机硬件打交道,提供通用功能的软件;另一类则是在通用软件基础上开发并运行的实际应用软件。前者通常就是操作系统,而后者则为应用软件。

那么什么是操作系统呢?

在长期工程实践中人们发现,计算机程序中,凡是与硬件相关的功能软件都具有通用性,例如,硬件的初始化、存储空间的分配和管理、外设的分配和管理等。显然,对于一般用户来讲,这种通用性软件最好是现成的。于是,人们就专门组织一些专家来编写这些通用软件,并作为一种软件产品提供给人们使用。这样,以应用为目的的软件开发人员就可以在这些通用软件基础上来开发和运行自己的应用软件,而没有必要再为实现那些通用功能去耗时费力了。

如果把应用软件看成计算机的使用者，那么它所面对的就是上述通用软件。或者说，从应用程序的角度来看，上述通用软件就是一种用来操作计算机硬件的工具，所以这种通用软件被叫做“操作系统”。

例如，Windows 就是一种操作系统，而那些以它为工作基础，以特定应用为目的的 Microsoft Word 和 Microsoft Excel 之类的软件就是应用软件。也就是说，Microsoft Word 和 Microsoft Excel 是通过 Windows 来使计算机完成相应的文字处理和电子表格处理任务的。所以，人们常说，计算机只有安装了某种操作系统之后，才能称得上是一个完整的、真正的计算机系统，否则就只能被称为“裸机”。

计算机硬件系统、操作系统和应用软件之间的关系如图 1-2 所示。

图 1-2　计算机硬件系统、操作系统和应用软件之间的关系

从图 1-2 中可知，计算机的硬件系统是操作系统的平台，而操作系统又是应用程序的运行平台。计算机硬件以指令集作为接口向操作系统提供服务，而操作系统则以函数作为接口向应用程序提供服务，所以操作系统提供的这些函数也叫做应用程序接口（API）。

除了操作系统和应用软件之外，计算机系统中还有一种通用但不以提供底层服务为目标的软件——工具软件，例如编译器、汇编器、连接器等，这种软件常常与操作系统归为一类，称为系统软件。

1.1.2　操作系统的作用

可以从两方面来认识操作系统的功能：①它是对计算机硬件的封装和功能的扩充；②它是计算机各种资源的管理者和分配者。

1. 操作系统是计算机硬件的封装和功能的扩充

为了说明问题，首先看一看当一个用户面对一台裸机时，会出现什么问题和困难。首先，用户不得不使用低级语言来编写程序。例如，需要从磁盘读取一批数据，那么凡是涉及读磁盘程序的每一个工作步骤和细节（诸如给出磁头号、启动驱动步进电机并命令磁头移动到给定的磁道位置、给出扇区号、等待磁头和扇区移动到合适位置、读出数据等）都需要用户自己来编写。不仅如此，类似的工作还有键盘、鼠标数据的读取、向打印机输出数据、向磁盘输出数据、向显示器输出数据、计算机硬件异常的处理、存储器的分配等。显然，上述这些低级语言程序的设计对于普通计算机用户是极其困难和艰巨的，因为这要求用户既要通晓计算机硬件的所有技术细节，又要精于汇编语言程序设计。

但是，人们发现这些汇编语言程序模块都有一个共同特点：它们都具有很强的通用性，这种软件一旦编写出来就可以供其他人使用。于是，人们认识到，请一些通晓计算机硬件工作机理，并精于汇编语言的人来编写这些程序功能模块，再把这些模块封装成高级语言形

式，然后把它们作为通用软件提供给用户是一个好办法。例如，如果系统中已经安装了三个具有接口的汇编语言程序模块：磁头移动并定位模块、读磁盘数据模块和写入磁盘模块，那么，用户应用程序从磁盘读取一批数据的工作就变得简单多了，应用程序只要在需要的地方调用相应的模块就行了。这种提供了一些例程接口，从而使应用程序可以通过这些接口对计算机硬件进行操作的软件，叫做计算机硬件的抽象层 HAL。它作为操作系统的最底层，是对计算机硬件的第一次软件封装。

更进一步，如果在硬件抽象层与应用程序之间再添一层，或者说，在硬件抽象层的基础上再加一层，即使用高级语言再编写一些程序模块，并通过调用诸如磁头移动、定位、读磁盘数据和写入磁盘等这些底层程序模块，把这些繁杂的硬件操作根据需要进行适当的组合并封装起来形成一些有通常意义的模块，那么用户就更方便了。例如，可以在函数 read( )中调用两个汇编例程：磁头移动并定位和读磁盘数据。那么，用户应用程序需要读取磁盘数据时，就可以简单地提出类似如下请求：

```
read(fp,buf,count);
```

并提供必要的参数(fp 是要读取的文件句柄，buf 是内存存放所读取数据的缓冲区，count 是要读取数据的字节数)就可以了。于是，应用程序设计人员在编写程序时就可以使用类似 read( )、write( ) 等功能更为综合、强大的函数，来编写程序了，如图 1-3 所示。

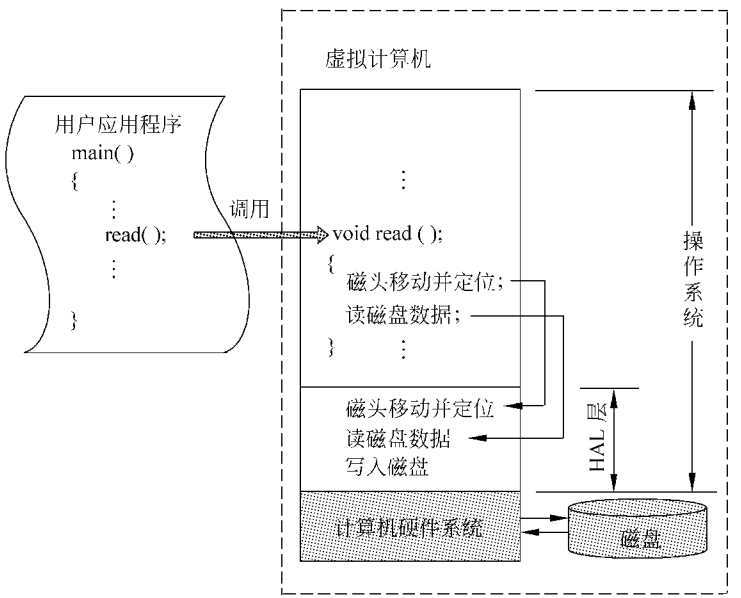


图 1-3　在读磁盘数据时，用户应用程序、操作系统与计算机硬件之间的关系

可见，有了上述的封装，应用程序设计人员所面对的就再也不是那些陌生的硬件装置及语义晦涩、功能单一的指令接口，而是一些倍感亲切、功能强大的高级语言接口了。

除此之外，为了方便用户，操作系统还提供了一些与硬件操作并无紧密关系的通用程序模块，并把它们与上述的 read( )、write( )等模块组织成函数库供应用程序来调用，即 API 函数。这样，从用户的角度看，现在的计算机能够接受更为抽象的高级语言指令，使原来必

须用几十条甚至几百条汇编语言指令完成的任务，只用一条或很少几条高级语言指令就可以完成，于是计算机的功能被大大地扩充了。

综上所述，可以知道，操作系统就是对计算机硬件的一个软件包装，这个包装不但使得用户感觉不到硬件的存在，而且看到的也不是硬件的模样。所以有人说，操作系统的功能就是为应用程序设计人员提供了一个更便于使用的虚拟计算机，这就是所谓的虚拟机观点。

2. 操作系统是计算机资源的管理者

其实，对于现代操作系统来说，除了上述的封装与抽象的作用之外，更主要和更大量的工作是对系统资源的管理。

通常，人们把由计算机提供的，应用程序运行所需要消耗、占用的物质条件，叫做计算机资源。这些资源包括：处理器时间、内存空间、外部设备和一些应用程序可使用的软件等。所以为了使用户应用程序可以方便、合理、高效地利用这些资源完成它的任务，作为以服务为目标的操作系统，责无旁贷地要承担起这些资源的管理任务。所以人们说，操作系统是计算机资源的管理者。

1.1.3 操作系统的管理功能

尽管人们从不同的角度对操作系统的作用和功能有各种不同的提法，但从应用程序的角度来看，操作系统就是应用程序运行环境的提供者，这个运行环境是否良好，直接取决于操作系统对资源的管理水平。这就像一家饭店一样，对于顾客来说，其硬件条件固然不可忽视，但其软件条件(管理及服务水平)则更为重要。

1. 处理器的管理

程序是由处理器来运行的，它是应用程序必须要使用的重要资源。所以操作系统的首要任务就是处理器的管理。

在单任务系统中，处理器的管理并不是什么大事，因为系统中只有一个程序在运行，那么处理器就归它用，没争没抢。但随着计算机应用要求的提高，在同一台计算机上通常都要同时运行多个程序。例如，一边使用文本编辑软件，一边用杀毒软件杀毒，甚至还可能在打印文档期间玩玩游戏放松一下。那么在处理器资源有限的情况下，怎么同时运行多个程序呢？很简单，就像人们一边喝茶一边谈生意的情况一样，同一张嘴，喝一口茶，说一段话，喝茶、说话两不误，“同时”进行。在单处理器计算机系统中也可以这样做。例如，有 3 个程序要运行，那么就可以把处理器的运行时间分成段，然后规定每个程序占用一个时间段轮流使用处理器来运行。如此这般，循环往复地让这些程序一段一段交叉地占用处理器，那么这些程序就可以“同时”运行了。这就像学校的公共教室一样，在同一个学期内，多个班级的不同课程要在同一个教室上课，该怎么办？轮流用就是了。当然，为了防止冲突，得为这个教室配备一个管理者。在计算机系统中，这个管理者就是操作系统。

上述做法还带来了另一个好处，就是可以提高处理器的效率。设想有 3 个应用程序，假如它们都需要从某一个输入设备读取数据，并在数据处理之后还要把结果在同一个输出设备上输出。如果这 3 个应用程序按照图 1-4 所示的方式一个接着一个顺序地运行，那么就会使输入设备、处理器、输出设备经常处在空闲状态，从而导致这些设备的利用率极为低下。

显然，如果把这 3 个应用程序的运行改为图 1-5 所示的方式，那么就可以使处理器、输入设备和输出设备都处于满负荷状态，从而大大地提高计算机资源的利用率。

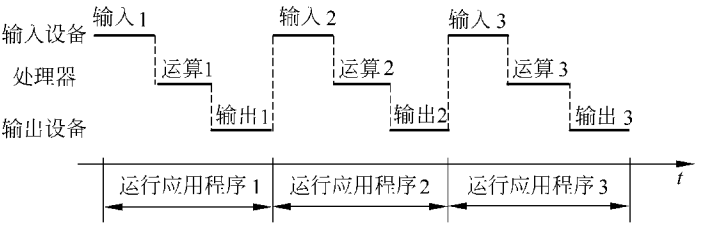


图 1-4　顺序运行 3 个应用程序的示意图

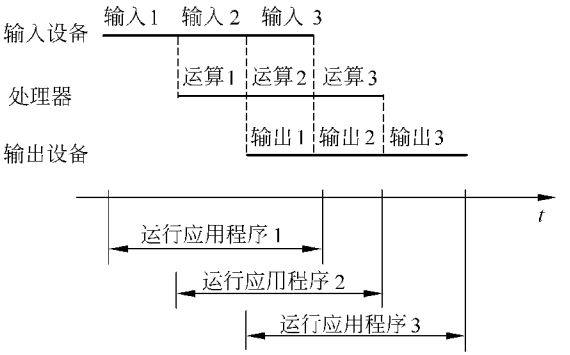


图 1-5　并发运行 3 个应用程序的示意图

仔细观察图 1-5 所示的运行方式，可以看到，处理器的这种工作方式还带来了另外一个好处，就是这三个应用程序的总运行时间也大大地缩短了，或者说计算机的工作速度好像更快了。

为了讨论问题方便，在计算机技术中，人们把上述这种看似同时，实际并不同时的程序运行方式叫做“并发”方式。显然，为了实现这种并发方式，操作系统中必须制定一个合理的处理器使用规则，并创建一个专门的管理者，按照规则来为程序分配处理器。也就是说，处理器管理是操作系统的重要工作之一。

附带说一句，计算机技术中，人们把程序在系统中运行的过程叫做“进程”，这就像饭店把正在吃饭（使用这个饭店的资源）的人叫做“客人”一样（如果你不吃饭，他就不把你叫做客人了）。

2. 内存管理

毋庸置疑，内存是应用程序需要使用的重要资源，如何合理地为并发程序分配内存，更是计算机操作系统责无旁贷的责任。存储管理的主要功能包括：

（1）存储分配。存储管理将根据用户程序的需要分配给它内存资源，这是多道程序能并发执行的首要条件。

（2）存储共享。存储管理应该能使多个用户程序方便实现某些存储资源的共同使用（共享），以提高内存的利用率或达到一些特殊目的。

（3）存储保护。存储管理必须要保证各个用户程序相互隔离，互不干扰，以保护系统和用户程序存放在内存中的信息不被破坏。

（4）存储扩充。存储管理应该能从逻辑上来扩充内存，为用户提供一个比内存实际容

量大得多的逻辑空间，以方便用户的编程和使用。

3. 文件管理

计算机中，程序和数据通常都以文件的形式存储于硬盘等外存储器中。由于文件量巨大，如果没有良好的管理方法，则会导致严重的后果。

文件管理的主要任务是实现按名存取、文件共享、保护和保密，并向用户提供一整套方便的操作和命令。具体来说，文件管理要完成以下任务：

(1) 提供文件逻辑组织方法；

(2) 提供文件物理组织方法；

(3) 提供文件存取方法；

(4) 提供文件使用方法；

(5) 实现文件的目录管理；

(6) 实现文件的共享和存取控制；

(7) 实现文件的存储空间管理。

4. 设备管理

计算机系统一般都配有外部设备，因此计算机操作系统还必须承担起外部设备管理的任务，以便完成用户提出的 I/O 请求，提高 I/O 设备的利用率，提供方便简单的设备使用方法。通常，设备管理应该具有以下功能：

(1) 提供设备的控制与处理；

(2) 提供缓冲区的管理；

(3) 提供设备独立性；

(4) 外围设备的分配和去配；

(5) 实现共享型设备的调度；

(6) 实现虚拟设备。

5. 网络和通信管理

近年来，计算机网络的应用已十分广泛。使用网络的计算机除了需要配备联网硬件之外，操作系统还必须具有管理网上资源、通过网络进行通信、故障处理、安全管理、性能管理等网络功能。

联网操作系统至少应具有以下管理功能：

(1) 网上资源管理功能。计算机网络的主要目的之一是资源共享，网络操作系统应实现网上资源的共享，管理用户应用程序对资源的访问，保证信息资源的安全性和完整性。

(2) 数据通信管理功能。按照通信协议的规定，完成网络上计算机之间的信息传送。

(3) 网络管理功能。包括：故障管理、安全管理、性能管理、记账管理和配置管理等。

6. 提供用户接口

为了服务于应用程序，计算机操作系统都配备了大量的服务例程，为使用户可以有效地使用这些例程，操作系统必须提供良好的用户接口，从而使程序设计人员能有效地组织应用程序。

1.1.4 操作系统管理用表

与人们进行事务管理的方法相同，操作系统也要用一些表格来记录软硬件资源的现状、

使用情况、保密等级等信息，所以为了对系统中的资源进行有效管理，现代操作系统都建立了大量管理用表。当然，不同的操作系统，所使用的管理用表也各具特色，不尽相同，但大体上都有如图 1-6 所示的几个主要部分。

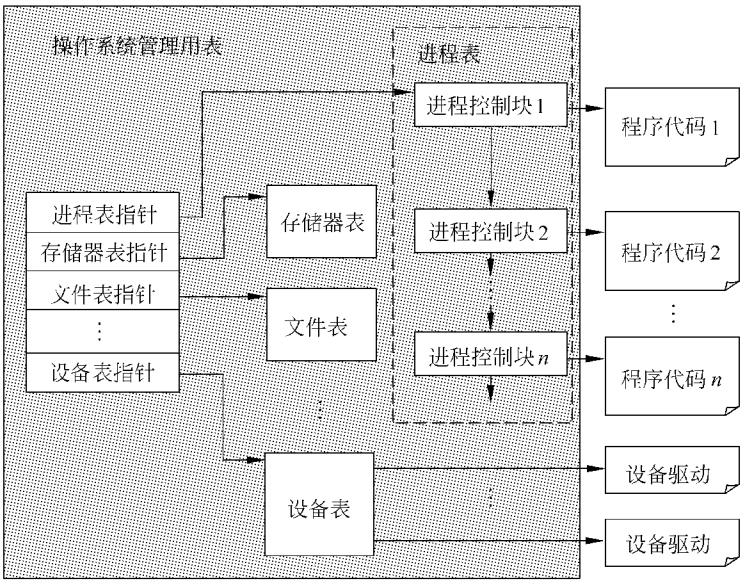


图 1-6　操作系统的主要管理用表

1.1.5　操作系统的应用程序接口

粗略地看，操作系统就是底层程序功能模块的集合。应用程序可以用两种方式来调用这些模块：在程序中调用 API 函数和在操作系统界面上使用操作命令。这两种用户接口与操作系统之间的关系如图 1-7 所示。

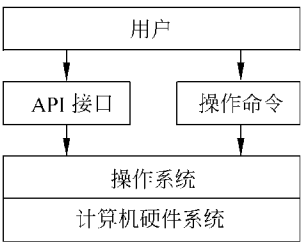


图 1-7　操作系统的用户接口

操作命令接口又叫做操作接口或作业接口，它是用户在操作系统界面上，以命令方式来操作和控制计算机的手段。在一般情况下，一个完整的操作系统在启动之后就会提供一个供用户对计算机进行操作的界面。例如，DOS 操作系统会在显示器上显示一个字符操作界面；Windows 操作系统会显示一个图形界面。这样，用户就可以以输入命令的方式来使用操作系统的某种功能。

API 又叫做编程接口，从使用的方式来看，它是在应用程序中以函数调用的方式来享用系统服务的。

在比较完善的操作系统中，还提供了一些诸如汇编、编译、编辑等通用的系统软件来供用户使用。这些程序虽然像应用程序一样是用来完成特定任务的，但由于这些特定任务具有某种程度的通用性(例如，C 编译器)，所以它们还是属于系统软件范畴。为了与应用程序在名称上区分开来，就把它们叫做实用程序。

于是，操作系统的一种可能的层次关系如图 1-8 所示。

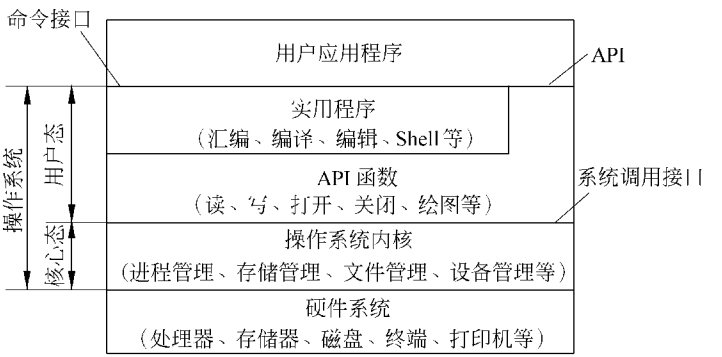


图 1-8　一种操作系统的层次关系

1.1.6　操作系统的特性

1. 并发性

现代操作系统都是并发系统，并发性是它的重要特征。由于在同一个时期内有多个程序交替、穿插地运行，所以它应该具有处理和调度多个程序同时执行的能力。

并发系统又称为多任务系统或多道程序系统。

2. 共享性

从应用程序的角度来看，除了处理器、存储器、外部设备等，操作系统也是程序运行所需的资源。出于经济上的考虑，一次性向每个用户程序分别提供它所需的全部资源不但浪费，也不可能，现实的方法是让操作系统和多个用户程序共享同一套计算机资源，即现代操作系统具有共享性。

3. 异步性

操作系统的第三个特性是异步性，或称随机性。在多道程序环境中，允许多个进程并发执行。由于资源有限而进程众多，因此一个进程的执行并不是连续执行到底，而是“走走停停”地执行。例如，一个进程在处理器上运行一段时间后，可能会因不能获得某个必要的资源而暂停执行，从而必须进入等待状态，以待以后某个时机再次运行。这就是说，系统中的进程何时执行，何时暂停，以什么样的速度向前推进，进程总共要花多少时间执行才能完成，这些都是在编程序时不可预知的，其后果就是程序执行结果可能不惟一。这种不可预知的异步性执行会给系统带来危险。因此，一个操作系统能否正确处理可能发生的随机事件，是衡量它是否稳定的一个重要指标。

4. 虚拟性

由于用户面对的是操作系统提供的软件接口，换句话说，用户看到的不是硬件设备而是软件这个中间层，因此系统就可以利用这个软件中间层来对硬件进行一些变换，而使其虚拟化。所谓虚拟化是指操作系统中的一种管理技术，这种技术能通过软件把一个物理实体变成逻辑上的多个对应物，或把多个物理实体变成逻辑上的一个对应物。这就像在饭店中一样，用户用餐时面对的是服务员，而不是厨房和饭店的售货台，那么这个服务员就是厨房和售货台的虚拟，是这两个硬件设备的对应物，用户可以利用这个逻辑对应物来点菜或者来购买一些香烟之类的小商品。

显然，采用虚拟技术的目的是为用户提供更方便、更高效的操作环境。例如，在多道程序系统中，物理处理器可以只有一个，每次也仅能执行一道程序，但通过多道程序和分时使用技术，宏观上可以有多个程序在执行，就好像有多个处理器在为各道程序工作一样，物理上的一个处理器变成了逻辑上的多个处理器。再例如，窗口技术可把一个物理屏幕变成逻辑上的多个虚拟屏幕等。

1.2 操作系统与应用程序的关系

1. 操作系统和应用程序都是程序

本质上来讲，操作系统与应用程序没什么区别，它们都是程序，只不过分工不同。

操作系统提供了包括资源管理在内的底层服务，而应用程序则以操作系统为基础来完成自己的任务。它们之间是服务和被服务的关系，这多少有些像一些研讨会的情况，主持人（操作系统）负责会务工作的底层管理工作，例如，准备会场、分配坐席、安排发言次序和在会议期间对会议进程进行控制等服务工作，而参会人员（应用程序）则在主持人的服务下完成自己的发言。参会人员与主持人没什么实质性的区别，他们都要占用讲台（相当于处理器），只不过角色不同。

与研讨会主持人需要最先开始工作一样，操作系统是系统最先运行的程序。

2. 操作系统的引导和装载及工作流程

由于操作系统是系统最先运行的程序，所以必须得先把它装入内存。但操作系统作为一种比较大的软件，通常以映像文件的形式存放在磁盘中，因此系统启动之后的一个主要工作就是把它装入主存，即操作系统的装载。

早期的做法是，在计算机配备的非挥发性存储器（CMOS）里存放一个专门负责装载操作系统工作的小型程序模块，在系统启动后，计算机会自动先执行这个模块把操作系统装入主存，然后再运行它。

后来由于操作系统的规模越来越大，操作系统的装载工作也越来越复杂，单靠 CMOS 中的小型装载模块已不能完成整个操作系统的装载任务，于是人们就又设计了一个功能较强、规模较大的装载模块，并将其与操作系统一起存放在磁盘中而 CMOS 中的那个小型装载模块则负责装载磁盘上的那个大型装载模块，待这个大型装载模块装到内存后，再启动这个大型模块来装载操作系统。于是，为了区别，CMOS 上的那个模块就改称为“引导模块”。也就是说，操作系统的引导和装载过程就像用火柴来引爆成吨的炸药那样，先点燃导火索，导火索再点燃雷管，最后再由雷管来引爆成吨的炸药。

操作系统被装载之后，处理器即开始运行操作系统，这时它的首要工作就是先为自己的管理工作做好准备，例如，划分内存空间，建立管理所需的各类数据结构等。待一切准备就绪后，操作系统立即就为自己创建一个进程（一个基本空转的程序），并运行它，等待用户命令。通常，这个空转的程序就是在显示器上提供一个字符形式的界面（例如，操作系统 DOS 的界面）或者图形形式的界面（例如，操作系统 Windows 的界面），由于这个程序似乎是操作系统的一个可见的“壳”，所以常常把它叫做操作系统的 Shell。这个 Shell 的作用主要就是接收用户命令，并对命令进行解释执行，所以 Shell 也叫做命令解释程序或命令解释器。这就像一家饭店一样，早上做完开店准备（初始化）工作之后，如果没有客人，那就只能做一些

没有事情的事情(等待客人),也可以派一个服务人员到店门口去做“Shell”。

当然,在有些不需要界面的操作系统中(例如,某些嵌入式系统),这个程序可以是一个不做任何事情的死循环。

操作系统的一般工作流程如图 1-9 所示。

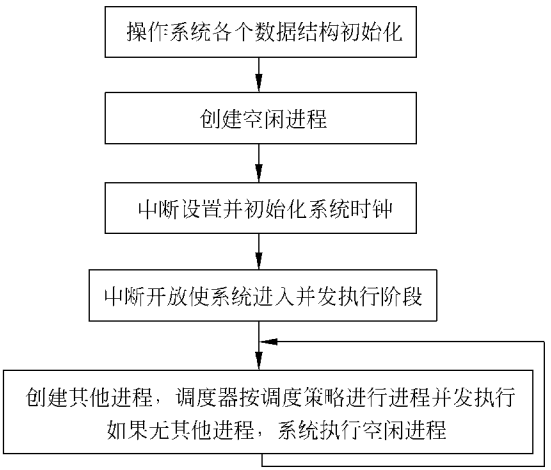


图 1-9　操作系统的工作流程

3. 应用程序的装载和运行

由于用户应用程序是以文件形式存放在磁盘上的,所以当有用户用命令要求操作系统运行一个应用程序时,操作系统会在建立相应的管理表格之后打开这个程序文件,然后根据具体情况为该程序分配内存空间和处理器来运行它。

命令方式下操作系统的工作过程如图 1-10 所示。

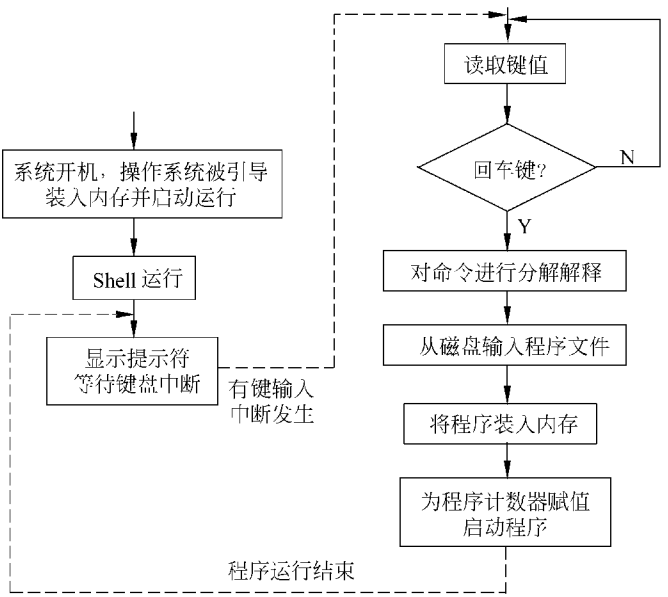


图 1-10　在命令方式下操作系统的工作过程

在图形界面的操作系统中(例如,Windows),用户可以使用鼠标双击程序图标(或使用快捷菜单中的【打开】命令)来装载启动一个应用程序,如图 1-11 所示。

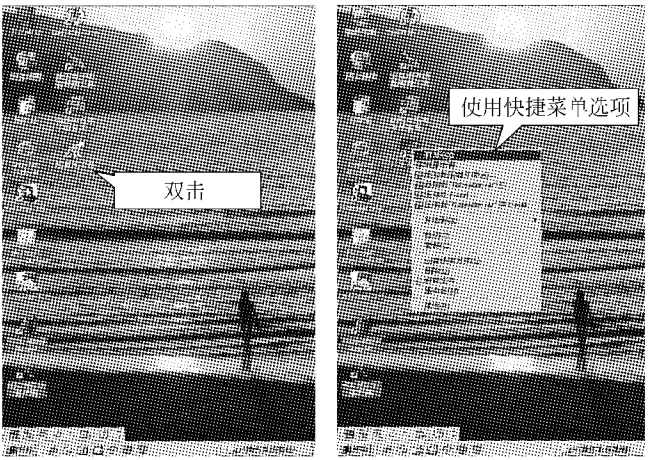


图 1-11　在 Windows 系统中启动应用程序

与命令行 Shell 的方式相仿,图形界面操作系统的 Shell 把鼠标双击图标解释成命令然后再启动并运行程序。但有一点重要的区别,就是对于具有图形界面的应用程序,其结束命令则是由用户通过单击程序窗口右上角的结束按钮来完成的。

除了上述两种应用程序的启动运行方式之外,还有其他多种启动方式。比较常见的还有,用应用程序启动另一个应用程序的方法(例如,在编译器程序中启动正在开发的程序)和程序自启动方法(例如,一些游戏程序)等。

4. 系统调用

上述的应用程序启动运行是操作系统的最基本服务,除此之外,操作系统还要以函数的方式向应用程序提供其他服务。从应用程序的角度看,操作系统就是一个函数库。应用程序调用操作系统服务函数的方法与普通的调用没有什么区别,但调用系统函数时,操作系统必须得把应用程序从用户态转入内核态,所以为了区别于普通函数调用,人们把应用程序对系统函数的调用叫做系统调用。

5. 回调

前面一再讲过,操作系统也是程序,所以必要时操作系统还需要调用应用程序中的一些函数,这种反向的调用叫做回调。

系统调用和回调的示意图如图 1-12 所示。

其实,C 程序的 main( )函数就是一个回调函数。所以教科书上讲:“C 程序是由函数组成的,其中的 main( )函数是系统首先调用的函数,即 C 程序是从 main( )函数开始运行的……”。再就是 Windows 应用程序中的窗口函数也是一种典型的回调函数。

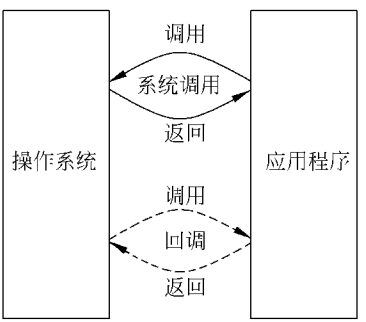


图 1-12　系统调用和回调的示意图

6. 运行时操作系统和应用程序的关系

当操作系统被装载并启动了用户程序后,操作系统与各个进程是什么关系呢?简单地说,在系统的整个运

行过程中，处理器有时运行操作系统，有时运行应用程序。就像上面所说的研讨会那样，有时是主持人发言，而有时是与会者发言。

图 1-13 表示了操作系统与进程之间的关系。从图中可以知道，处理器究竟是运行操作系统还是运行应用程序取决于开关 K 的接通方向。使得开关 K 向上接通运行操作系统的有 4 种情况：一是在系统启动、引导、装载结束后，系统开始运行操作系统时(对应图 1-13 中的①)；二是外部设备发生中断请求时(对应图 1-13 中的②)；三是系统时钟定时时间到，操作系统调度器对进程进行切换时(对应图 1-13 中的③)；四是当进程调用操作系统 API 函数时(对应图 1-13 中的④)。使得开关 K 向下接通运行程序(进程)则有两种情况：一是调度器选中某一进程并要运行它时(对应图 1-13 中的⑤)；二是进程调用了操作系统函数，函数运行完毕返回时(对应图 1-13 中的⑥)。

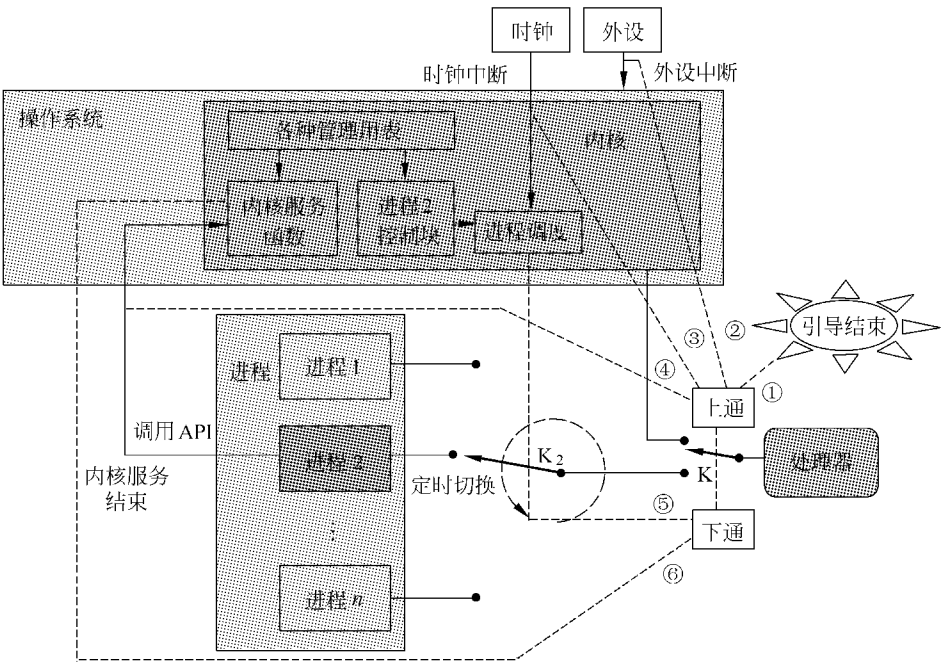


图 1-13　操作系统内核、进程与处理器之间的一般关系

1.3　操作系统结构

1.3.1　内核

从上面的叙述中已经知道，进程需要依赖系统资源来运行，如果把进程看做是资源的需求方，则资源就是供应方，那么由谁来负责这两者之间的协调和管理工作呢？当然，不能是进程，更不能是资源，只能是另外一组软件功能模块，这组软件功能模块就是操作系统的内核。于是，在操作系统设计时，人们就把一些与硬件紧密相关、运行频率较高的公共基本操作程序模块，及一些关键性的数据结构安排在内核，并使之常驻内存。一般来说，操作系统的内核应该提供中断处理、处理器调度、内存管理、文件管理、设备管理、进程通信等基本功能。

在上述功能模块中，有一些关键的操作程序是不可中断的，这种程序段叫做“原语”，意思是说，这种操作具有“原子性”，不可分割。

另外，作为管理者，内核模块的另一个重要特点就是它可以使用特权指令。某些现代计算机系统提供了用户状态和系统状态等多种机器工作状态。在系统状态下，用户可以使用计算机的所有指令，而在常态下，用户只能使用计算机指令系统的一部分。也就是说，现代计算机把指令系统分成了两部分：一部分是整个指令系统，另一部分只是指令系统的一个子集。这样，高级用户可以把计算机置于系统状态下来使用计算机的全部指令，低级用户则只能在常态下，使用那些不会给计算机造成致命异常的一部分指令，这样就可以大大提高计算机的安全性。

操作系统内核的一种工作流程图如图 1-14 所示。

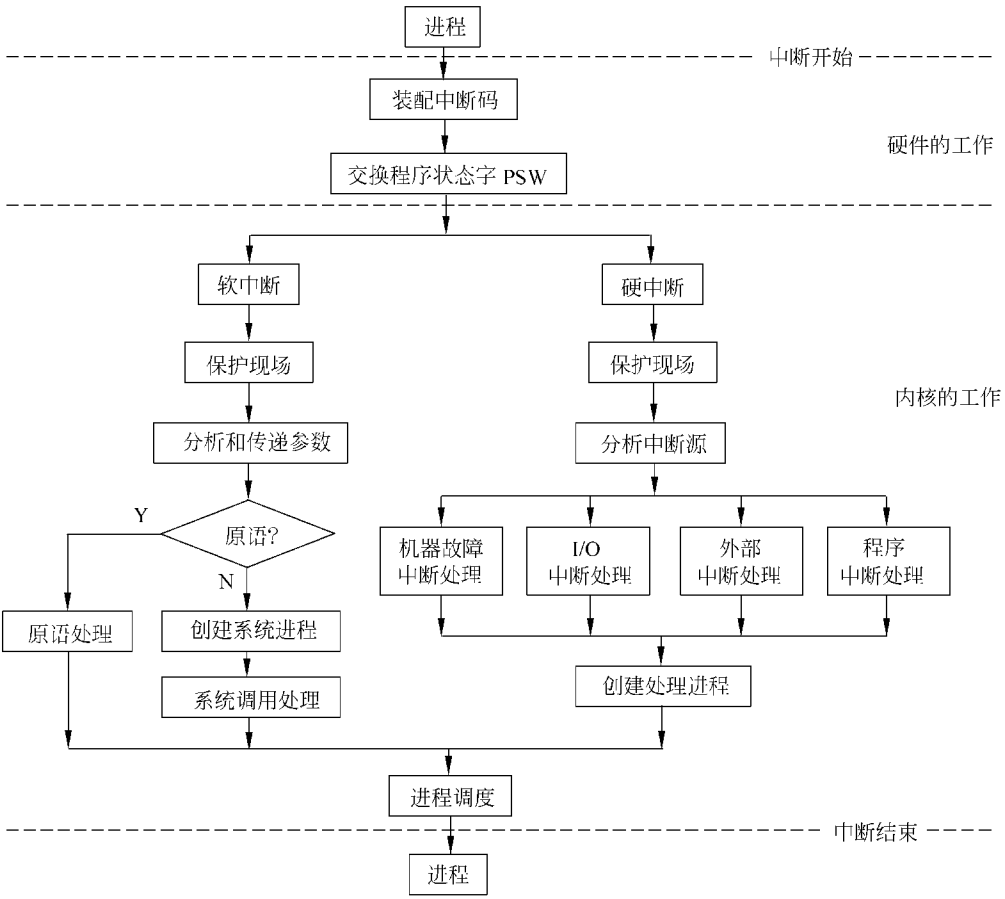


图 1-14　操作系统内核的工作流程

从图 1-14 中可以看到，操作系统的运行主要是靠中断来驱动的。操作系统的系统调用、命令输入、进程调度、设备驱动、文件操作等重要活动都是由中断来激活的。所以对于中断的管理和处理是操作系统赖以活动的基础。

内核是操作系统对裸机的第一次软封装，内核和裸机组成了一台虚拟机，进程就在这个虚拟机上运行。虚拟机没有中断的概念，用户进程无需处理中断，且进程在执行中感觉不到

其他进程存在，似乎处理器就是自己私有的。与裸机相比，虚拟机为进程提供了功能更为强大的新的指令系统，从而使用户可以在较高级的层次上进行程序设计。

1.3.2 内核结构

1. 整体式结构

整体式结构又叫做模块组合结构，这是早期的一种结构化程序设计方法。具体做法为：按功能需求把整个程序分解成若干个模块，每个模块具有一个相对独立的功能，功能模块之间可以自由调用，通常用若干个相关联的模块来完成一个较大的系统功能。

这种方法的优点是组合灵活、结构紧密、执行效率高。但也有致命的弱点，正因其结合紧密，也就导致了模块的独立性差、模块之间牵扯多、调用关系复杂，因此当系统较大时，结构不清晰，正确性难以保证，系统维护困难。所以这种整体式结构已不再为现代大型操作系统所采用，只出现在一些小型操作系统中，例如，嵌入式操作系统。

2. 层次式结构

层次式结构的基本组成部分仍然是模块，但这些模块按其功能的抽象程度分布在不同的层次中。

系统的层次式结构如图 1-15 所示。

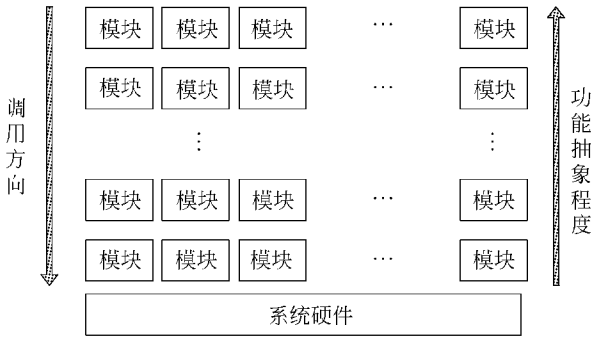


图 1-15 系统的层次式结构

从图 1-15 中可见，模块只能在层次间调用，而且只能向下调用，即低层模块只为其上层模块服务，横向及跨层之间不允许调用，也不允许循环调用。所以这种结构层次清楚、明晰，便于维护及扩展和移植。但这种结构也有一个缺点，就是灵活性较差，所以在实际应用中很多系统采用了更为灵活的半序层次结构。相对应地，图 1-15 所示的是全序层次结构。

所谓的半序层次结构就是同层模块也可以相互调用的层次结构，显然这种半序层次结构更为灵活。

3. 微内核

在设计操作系统内核时，把凡是为进程服务的程序模块都集中地放到内核里是一个自然的做法。例如，内核要包括文件管理模块、设备驱动模块、进程管理模块等。但是，随着服务需求的不断增加，操作系统的内核也就越来越大，随之也出现了一系列问题。首先，内核如果太大，则会因其占用的存储空间太大而不适合用在存储器资源比较紧张的系统中。其次，对于一个模块之间存在着各种耦合关系的大内核的维护也相当困难。因为内核只要有改动，就得重新编译整个内核。第三，大内核会使得处理器在内核运行的时间变长，从而不

适合在速度要求较高的场合下应用。总之，操作系统内核大到一定程度之后，就像人得了肥胖症一样，会出现各种各样的问题。

为了解决上述这些问题，人们想了一系列办法，试图在满足应用程序需要的前提下把内核做小。其中一个有效的办法是把内核各个服务程序模块中的部分内容移到内核的外边，作为一个进程来看待，在内核中只保留内核服务与用户进程的接口，或者说只保留一个“壳”。在用户进程需要该服务时，由这个“壳”通过发送消息的方法与服务进程进行联系，当与这个服务相关的服务进程接收到这个信息时就马上启动这个服务。这样，内核中保留的只是一些服务模块的“壳”，或者说是消息的转送站，于是内核就可以大大地变小了。这种内核就叫做“微内核”，具有微内核的操作系统叫做微内核操作系统。

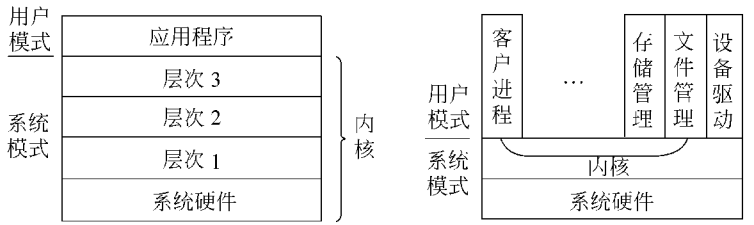
因此，目前在操作系统内核的设计上有两种结构：一种是宏内核结构，另一种是微内核结构。

宏内核虽然被分为若干个功能模块(或者若干个层次)，但在运行时，它是一个独立的二进制大映像，模块之间的协作是通过直接调用其他模块函数来实现的。也就是说，模块之间函数的直接调用使整个内核形成了一个整体，所以这种内核也叫做一体化内核。

而微内核则不然，它的系统调用功能模块不再负责完成整个服务任务，而只起一个接待作用。当有用户服务请求时，内核模块只对这个请求进行简短的接待处理，然后就通过消息传递，把实质性的服务工作交给另外的一个进程来处理。在典型情况下，每个系统调用程序模块都有一个与之对应的进程。

显然，微内核结构有助于实现模块间的隔离，便于维护、扩展和移植。另外，微内核的这种分立结构为系统的部分装入创造了条件，即允许系统运行时只装入当前需要使用的那一部分，而暂时不需要的可不加载到内存，因此可以更有效地利用内存。所以有人说，宏内核是一个什么事情都自己做的大而全的公司，而微内核则是一个只承揽业务而把事情交给别人干的贸易公司。

宏内核与微内核的比较如图 1-16 所示。



(a) 宏内核　　(b) 微内核

图 1-16　宏内核和微内核比较

1.4　操作系统的演变与发展

1.4.1　人工操作阶段

从计算机诞生到 20 世纪 50 年代中期，计算机速度慢、规模小、外设少，完全由程序员采用手工方式直接控制和使用计算机硬件。程序员使用机器语言编程，并将事先准备好的程

序和数据穿孔在纸带或卡片上，利用纸带或卡片输入机将程序和数据输入计算机。然后，启动计算机运行程序，程序员通过控制台上的按钮、开关来操纵和控制程序，通过氖灯来显示计算机的运行状态。

后来，二进制形式的数字操作码被字符形式的助记符语言——汇编语言所代替，预先编制的汇编程序可以把汇编语言的“源程序”解释成计算机能直接执行的机器语言格式的目标程序。在 20 世纪 50 年代中后期，FORTRAN、ALGOL 和 COBOL 之类的高级程序设计语言出现，进一步方便了编程。

在这一阶段，尽管在程序设计方面有了比较大的发展，但计算机的操作方式仍然还是以人工为主，并没有多大改变。人工操作方式下的用户独占资源，计算机系统效率低下，程序员任务繁重，手工操作过多，极易发生差错。这一切都严重地阻碍了计算机技术的发展，成为当时急需解决的主要问题。

1.4.2 管理程序阶段

为了摆脱通过开关和按钮来控制计算机执行的人工方式，人们发展了作业控制语言，使用这种语言通过作业控制卡来描述作业的加工控制步骤。同时专门设计了一种可以对作业进行自动控制和成批处理的执行程序，并使之驻留在计算机的内存中。程序设计人员除了编写程序之外，还需编制一个作业卡，然后把作业卡连同程序、数据一起提交给计算机的操作员，操作员收集到一批作业后一起把它们放到卡片机上输入计算机，计算机则在执行程序的控制下，自动进行作业转换以减少系统空闲和手工操作的时间。其工作流程如下：执行程序将一批作业从纸带或卡片机输入到磁带上，每当一批作业输入完成后，执行程序自动把磁带上的第一个作业装入内存，并把控制权交给作业。当该作业执行完成后，执行程序收回控制权并再调入磁带上的第二个作业到内存执行。计算机在执行程序的控制下，就这样一个作业接着一个作业连续地执行，直至磁带上的作业全部完成。由于系统能实现作业到作业的自动转换，缩短了作业的准备和建立时间，减少了人工操作和干预，从而使计算机可以尽可能快地连续运转。

后来，经过不断地改进、扩充，上述的执行程序就形成了所谓的作业管理程序，简称管理程序。

1.4.3 多道程序设计与操作系统的形成

1. 多道程序设计

在早期的单道批处理系统中，内存中仅有单个作业在运行，致使系统中仍有许多资源空闲，设备利用率低，系统性能较差。

直至 20 世纪 60 年代初，中断和通道这两项技术的发展，为实现 CPU 和 I/O 设备并行工作提供了技术支持，由此使早期人们期盼的多道程序运行方式变成了现实。多道程序设计(multiprogramming)是指允许多个程序(作业)同时进入一个计算机系统的内存储器并启动进行交替计算的方法。也就是说，计算机内存中同时存放了多道(两个以上相互独立的)程序，它们均处于开始点和结束点之间，从宏观上看是并行的，从微观上看是串行的，各道程序轮流占用 CPU，交替地执行。引入多道程序设计技术的根本目的是提高 CPU 的利用率，充分发挥计算机系统部件的并行性。现代计算机系统都采用了多道程序设计技术。

2. 操作系统的形成

第三代计算机的性能有了更大提高，机器速度更快，内外存容量增大，I/O 设备数量和种类增多，为软件的发展提供了有力支持。为更好地发挥硬件功效，更好地满足各种应用的需要，迫切要求扩充管理程序的功能。

但是，从半自动的管理程序方式过渡到能够自动控制程序执行的操作系统方式，对辅助存储器性能的要求极高。这个阶段虽然有个别的磁带操作系统出现，但操作系统的真正形成还期待着大容量高速辅助存储器的出现。大约到 20 世纪 60 年代中期以后，随着磁盘的问世，才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，直到这个时候才标志着操作系统的正式形成。

计算机配置操作系统后，其资源管理水平和操作自动化程度有了进一步提高，具体表现在以下几个方面。

(1) 操作系统实现了计算机操作过程的自动化。批处理方式更为完善和方便，作业控制语言有了进一步发展，为优化调度和管理控制提供了新手段。

(2) 资源管理水平有了提高，实现了外围设备的联机同时操作，进一步提高了计算机资源的利用率。

(3) 提供虚存管理功能。由于多个用户作业同时在内存中运行，在硬件设施的支持下，操作系统为多个用户作业提供了存储分配、共享、保护和扩充的功能，导致操作系统步入实用化。

(4) 支持分时操作，多个用户通过终端可以同时联机地与一个计算机系统交互。

(5) 文件管理功能有改进，数据库系统开始出现。

(6) 多道程序设计趋于完善，采用复杂的调度算法，充分利用各类资源，最大限度地提高计算机系统的效率。

1.4.4 操作系统的发展

促使操作系统不断发展的主要动力有以下 5 个方面。

(1) 器件快速更新换代。微电子技术是推动计算机技术飞速发展的“引擎”，从而推动着微机的快速更新换代，使它由 8 位机、16 位机、32 位机，快速发展到了 64 位机，相应的操作系统也就由 8 位机操作系统发展到 16 位、32 位、64 位微机操作系统。

(2) 计算机体系结构不断发展。硬件的改进导致操作系统发展的例子很多，内存管理支持硬件由分页或分段设施代替了寄存器以后，操作系统中便增加了分页或分段存储管理功能。图形界面代替了字符界面后，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。计算机体系结构的不断发展有力地推动着操作系统的发展。

(3) 提高计算机系统资源利用率的需要。多用户共享一套计算机系统的资源，必须千方百计地提高计算机系统中各种资源的利用率。各种调度算法和分配策略相继被研究和采用，这也成为操作系统发展的一个动力。

(4) 让用户使用计算机越来越方便的需要。从批处理到交互型分时操作系统的出现，大大改变了用户上机、调试程序的环境；从命令行交互进化到 GUI 用户界面，操作系统的界面变得更加友善。

(5) 满足用户的新要求，提供给用户新服务。当用户要求解决实时性应用时，便出现了实时操作系统。当发现现有的工具和功能不能满足用户需要时，操作系统往往要进行升级换代，开发新工具，加入新功能。

目前，操作系统的主要发展方向如下所述。

1. 并行操作系统

增加计算机的处理能力是人们无尽的要求。为了达到极高的性能，除提高元器件的速度外，更重要的是在计算机系统结构上不断进行改进，并行处理(parallel processing)技术就是一个重要的方向。目前，已经开发出的并行计算机有阵列处理机、流水线处理机和多处理机。毋庸置疑，为了发挥并行计算机的性能，除了硬件结构的保证之外，在软件方面还需要有并行算法、并行语言等许多软件的配合，而其中的并行操作系统则是并行计算机发挥高性能的重要基础和保证。所以，近年来，并行操作系统的研究和开发已成为计算机界的热门研究课题。

2. 网络操作系统

计算机网络是通过通信设施将地理上分散的并具有自治功能的多个计算机系统互连起来的系统。网络操作系统(network operating system)是能够控制计算机在网络中方便地传送信息和共享资源，并能为网络用户提供各种所需服务的操作系统。网络操作系统主要有两种工作模式：客户机/服务器(client/server)模式和对等(peer-to-peer)模式。

在客户机/服务器模式中有两类站点，一类是作为网络控制中心或数据中心的服务器，提供文件打印、通信传输、数据库等各种服务；另一类是本地处理和访问服务器的客户机。这是目前较为流行的工作模式。

对等模式中，网络中的站点都是对等的，每一个站点既可作为服务器，又可作为客户机。

网络操作系统应该具有以下几项功能：

(1) 网络通信。其任务是在源计算机和目标计算机之间，实现无差错的数据传输。具体来说是完成建立/拆除通信链路、传输控制、差错控制、流量控制、路由选择等功能。

(2) 资源管理。对网络中的所有硬、软件资源实施有效管理，协调诸用户对共享资源的使用，保证数据的一致性、完整性。典型的网络资源有硬盘、打印机、文件和数据。

(3) 网络管理。包括安全控制、性能监视、维护功能等。

(4) 网络服务。如电子邮件、文件传输、共享设备服务、远程作业录入服务等。目前，计算机网络操作系统有三大主流：UNIX、Netware 和 Windows NT。UNIX 是惟一能跨多种平台的操作系统；Windows NT 工作在微机和工作站上；Netware 则主要面向微机。支持 C/S 结构的微机网络操作系统则主要有 Netware、UNIXware、Windows NT、LAN Manager 和 LAN Server 等。

下一代网络操作系统应能支持以下功能：

(1) 位置透明性。支持客户机、服务器和系统资源不停地在网络中装入卸出，且不固定确切位置的工作方式。

(2) 名字空间透明性。网络中的任何实体都必须从属于同一个名字空间。

(3) 管理维护透明性。如果一个目录在多台机器上有映像，应负责对其同步维护；应能将用户和网络故障相隔离；同步多台地域上分散的机器的时钟。

(4) 安全权限透明性。用户仅需使用一个注册名及口令，就可在任何地点对任何服务

器的资源进行存取，请求的合法性由操作系统验证，数据的安全性由操作系统保证。

(5) 通信透明性。提供对多种通信协议的支持，缩短通信的延时。

3. 分布式操作系统

以往的计算机系统，其处理和控制功能都高度地集中在一台计算机上，所有的任务都由它完成，这种系统称集中式计算机系统。而分布式计算机系统是指由多台分散的计算机，经互连网络连接而成的系统。每台计算机高度自治，又相互协同，能在系统范围内实现资源管理、任务分配，能并行地运行分布式程序。

用于管理分布式计算机系统的操作系统称为分布式操作系统。它与单机集中式操作系统的主要区别在于资源管理、进程通信和系统结构三个方面。

集中式操作系统的资源管理比较简单，一类资源由一个资源管理程序来管。这种管理方式不适合于分布式系统，例如，一台机器上的文件系统来管理其他计算机上的文件是有困难的。所以，分布式系统中，对于一类资源往往有多个资源管理程序，这些管理者必须协调一致的工作，显然这种管理比单个资源管理程序方式要复杂得多。

在结构上，分布式操作系统也与集中式操作系统有重大区别，它往往有若干相对独立的部分，各部分分布于各台计算机上，每一部分在另外的计算机上往往有一个副本，当一台机器发生故障时，由于操作系统的每个部分在他机上有副本，因而仍可维持原有功能。

分布式系统研究和开发的主要方向有：

(1) 分布式系统结构。研究非共享通路结构和共享通路结构。

(2) 分布式操作系统。研究资源管理方法、同步控制机制、死锁的检测与解除、进程通信模型及手段等。

(3) 分布式程序设计。扩充顺序程序设计语言，使其具有分布式程序设计能力；开发新的分布式程序设计语言。

(4) 分布式数据库。设计开发新的分布式数据库。

(5) 分布式应用。研究各种分布式并行算法，研究在办公自动化、自动控制、管理信息系统等各个领域的应用。

4. 嵌入式操作系统

在 20 世纪 70 年代，由于微处理器的出现，使计算机的应用出现了历史性的变化。以微处理器作为核心的微型计算机，以其小型、价廉、高可靠性及具有高速数值计算能力的特点，迅速引起了自动控制领域专业人员的极大兴趣和注意。因为把微型机嵌入到一个对象体系(如汽车、火箭等)中，可以很方便地实现这个对象体系的智能化控制。例如，将微型计算机经电气加固、机械加固，并配置了各种外围接口电路之后，安装到飞机中就可以构成智能化程度很高的自动驾驶仪或发动机状态监测系统；安装到洗衣机中就可以使洗衣机根据所要洗涤衣物的具体状况而自动采取不同的洗涤模式，从而提高洗衣的效率和效果；安装到照相机中可以自动对摄像的各个技术参数进行设置，从而使人们可以获得质量更好的照片；安装到音像设备中可以获得高保真的音响和影像等。

显而易见，这种用途的计算机系统在一定程度上改变了通用计算机系统的形态与功能。为了区别于原有的通用计算机系统，人们把嵌入到对象体系中，为实现对象体系智能化控制的计算机系统，称作嵌入式计算机系统，简称嵌入式系统。

应用在上述系统中的操作系统就是嵌入式操作系统。

嵌入式操作系统和其他嵌入式软件都有下列特点：

(1) 微型化。由于嵌入式系统硬件平台的局限性(内存少、不配置外存，微处理器字长短且运算速度有限等)，在保证应用功能的前提下，微型化是设计嵌入式操作系统的主要出发点。

(2) 可定制。嵌入式操作系统运行的平台多种多样，应用更是五花八门，因而表现出专业化的特点。从而要求嵌入式操作系统的各个模块，例如，文件系统、图形接口、网络通信与TCP/IP、多媒体处理等各种功能模块都作为可选件，以供用户根据实际需要选择。

(3) 实时性。嵌入式操作系统广泛应用于过程控制、数据采集、传输通信、多媒体信息(语音、视频影像处理)及关键要害领域等要求迅速响应的场合，实时响应要求严格，因而，实时性是其主要特点之一。针对应用的响应时间要求，可设计成硬实时、软实时和非实时不同级别的系统。对于应用在军事武器、航空航天、交通运输等特殊领域有硬实时要求的系统，其中断响应、处理器调度等机制必须严格符合时间要求。

(4) 可靠性。系统构件、模块和体系结构必须达到应有的可靠性，对关键要害应用还要提供容错和防故障措施，进一步改进可靠性。

(5) 易移植性。为了提高系统的易移植性，通常采用硬件抽象层(hardware abstraction level，HAL)和板级支持包(board support package，BSP)的底层设计技术。HAL 提供了与设备无关的特性，屏蔽硬件平台的细节和差异，向操作系统上层提供统一接口，保证了系统的可移植性。

(6) 开发环境。嵌入式操作系统的开发环境，除了通常的代码编辑器、编译器和链接器、程序调试器、系统配置器之外，还有系统仿真器等一些普通应用程序开发用不到的特殊工具。

1.5 Linux 基础知识

从诞生以来，只经过十几年的光景，Linux 就风靡了全世界并得到了广泛的应用，这足以说明它的魅力和强大。本节在简单介绍 Linux 发展历史的基础上，重点介绍 Linux 在嵌入式应用的各种修改版的特点。与此同时还介绍了 Linux 的 C 语言与汇编语言特点及内核常用链表。

1.5.1 Linux 的发展

20 世纪 80 年代，Andrew S. Tanenbaum 教授为了满足教学的需要，编写了一个小型操作系统——MINIX。几年后，芬兰赫尔辛基大学的学生 Linus Tovalds 在 MINIX 的基础上编写了一个实用的操作系统，并将其命名为 Linux 发表在互联网上。从此，Linux 传遍世界，并在 GNU 自由软件组织和众多软件专家及业余爱好者的努力下，蓬勃地发展起来，成为一个优秀的操作系统。

Linux 之所以能够发表与发展，GNU 自由软件组织功不可没。该组织为打破垄断，使软件业健康发展，拟定了一份通用公共许可证(General Public License，GPL)，以保证任何人都有发表、共享和修改软件的自由。这个可以充分发挥个人想象力和创造力的组织一出现，就吸引了众多软件专家及业余爱好者。Linux 一经在这个组织中发表，便立即引起了人