计算机科学1999Vol.26№.3

现代操作系统概论*)

The Outline of Modern Operating Systmes

耿 技 周明天

(电子科技大学计算机学院 成都 610054)

Abstract The paper introduces fundamental knowledge of modern operating systems, including interfaces, functions, structures, and securities. The knowledge is useful to learn, use and investigate operating systems.

Keywords Operating system, Kernel, Subsystem, API, User interface, System security

一、引言

给现代操作系统一个明确的界定是一件很难的事情,因为操作系统的发展历史是一个连续的过程,不可能割裂开来。好在本文的目的并不在此。我们只是用现代操作系统来统称一些目前被广泛使用的、成熟的操作系统。这些操作系统绝大多数采用多道程序设计技术。所谓多道程序设计技术是指,系统允许多道程序准备运行;当正在运行的那道程序因为某种原因(比如等待输入或输出数据)暂时不能继续运行时,系统将自动地启动另一道程序运行;一旦原因消除(比如数据已经到达或数据已经输出完毕),暂时停止运行的那道程序在将来某个时刻还可以被系统重新启动。

多道程序设计技术使得现代操作系统呈现出两个基本特征:一是程序共行,二是资源共享。程序共行有两层含义:从宏观上看(即从操作系统外部看),程序共行是指系统中有多道程序同时运行;从微观上看(即从操作系统内部看),程序共行是指单处理机系统中的程序并发(即多道程序在单处理机上交替运行)或多处理机系统中的程序并行(即多道程序在多个处理机上同时运行)。资源共享也有两层含义:从宏观上看,资源共享是指多道程序可以同时使用系统中的软硬件资源;从微观上看,资源共享是指多道程序可以交替地或互斥地访问系统中的某个资源。

本文将给出认识现代操作系统的一些基本观点,并在此基础上介绍现代操作系统的基础知识,包括系统界面、内部功能、组成结构以及安全措施。

二、关于现代操作系统的基本观点

对于现代操作系统,人们常用下面四种观点来描述:

·用户环境观点:认为操作系统是计算机用户使用计算机系统的接口,它为计算机用户提供了方便的工作环境。

·虚拟机器观点:认为操作系统是建立在计算机硬件平台上的虚拟机器,它为应用软件提供了许多比计算机硬件功能更强或计算机硬件所没有的功能。

·作业组织观点:认为操作系统是计算机系统工作流程的组织者,它负责协调在系统中运行的各个应用软件的运行次序。

·资源管理观点:认为操作系统是计算机系统各类资源的管理者,它负责分配、回收以及控制计算机系统的各类软硬件资源。

上述四种观点是不同的人从不同的角度观察现代操作系统时所形成的不同看法。其中,用户环境观点和虚拟机器观点分别是计算机用户和程序设计人员从外部观察现代操作系统时所形成的看法,作业组织观点和资源管理观点则是操作系统设计者从内部分别就系统的动态和静态功能观察现代操作系统时所形成的看法。显然,要想较为全面地认识现代操作系统,必须综合使用上述四种观点。

*)本文为四川省新型计算机应用技术重点实验室基金课题"柔性操作系统研究"系列研究报告之一。

三、现代操作系统的系统界面

一般地，我们将由操作系统实现的、供计算机用户以及应用软件与操作系统进行通信和相互作用的通道称为操作系统的系统界面或系统接口；其中、供计算机用户与操作系统进行通信和相互作用的系统接口叫做用户接口，供应用软件与操作系统进行通信和相互作用的系统接口叫做程序接口。下面我们分别讨论这两种类型的系统接口。

3.1 用户接口

用户接口由一组命令组成，因此也叫做命令接口。所谓命令是指、计算机用户要求计算机系统为其工作的指示。一般地、不同的操作系统其用户接口有所不同。这不仅可能体现在命令的种类、数量以及功能方面，也可能体现在命令的用法和形式方面。

从用法上看、命令分为批处理和交互式两种类型。批处理命令又叫做脱机命令，主要用来描述脱机作业。交互式命令又叫做联机命令，主要用来描述联机作业。所谓作业是指，计算机用户为某种特定目的要求计算机系统所做工作的集合。作业中每一项相对独立的工作都是用命令来指定的。脱机作业和联机作业的区别在于：用户不能控制脱机作业的处理过程，但能够随时调整联机作业的处理步骤。现代操作系统的许多命令既可以作为批处理命令也可以作为交互式命令来使用。

从形式上看，命令分为字符式、图形式以及菜单式三种类型。字符式命令是传统的命令形式。虽然对缺乏经验的用户来说，字符式命令十分繁琐、难以记忆，但对有经验的用户而言、字符式命令使用起来十分灵活、所以至今仍有许多操作系统支持这种命令形式。图形式命令是目前最流行的命令形式。它非常直观而且易于使用，因而受到用户的普遍欢迎，得到现代操作系统的广泛支持。菜单式命令是在图形显示器被广泛使用之前操作系统为用户提供的一种过渡性质的命令形式。由于与字符式命令相比它缺乏灵活性，与图形式命令相比它缺少直观性，因此菜单式命令很少单独被现代操作系统采用。

随着近些年来个人计算机系统的广泛流行，计算机系统的应用范围得到了进一步扩大，缺乏计算机专业知识的用户也随之增多。如何不断更新技术，为这些用户提供直观、易于掌握、便于使用、功能强大的用户接口，便成为操作系统领域的一个热门课题，受到操作系统研究人员的普遍重视。研究人员努力的结果使得用户接口成为近些年来操作系统领域发展最快的一个分支。

3.2 程序接口

程序接口也叫做应用编程接口，常用英文缩写API来表示、它由一组系统调用组成。所谓系统调用是指、由操作系统实现的、供应用软件引用的系统服务。尽管人们一直试图为不同的操作系统定义一个统一的、标准的程序接口、并已经推出了可以实施的国际标准POSIX.1，但实际情况是、不同的操作系统其程序接口仍然存在着差异，这种差异既可能体现在系统调用的种类、数量以及功能方面、也可能体现在系统调用的引用机制上。

系统调用的引用机制涉及两个方面：一是、应用软件与系统调用之间的参数传递方式；二是、应用软件与系统调用之间的控制转移方式。通常，应用软件与系统调用之间可以用寄存器来传递参数，这是一种既简单又快捷的方式，但转移的参数不多，如果要传递许多参数，一种可供选择的方案就是利用堆栈。当然，也可以使用专门设计的参数表。在现代操作系统中、软中断和过程调用是两种常用的应用软件到系统调用的控制转移方式。

程序接口事实上定义了一台虚拟机器。该虚拟机器包含一组抽象概念以及与这组抽象概念相关的系统服务。其中，抽象概念定义了虚拟机器的基础设施，相关的系统服务规定了虚拟机器的使用方式。对程序设计人员来说，在这种虚拟机器上开发应用软件要比直接在硬件裸机上开发应用软件简单方便得多，这种虚拟机器提供了更多更强的功能。

进程是现代操作系统中最常见的、也是最重要的抽象概念之一。所谓进程是指应用软件一次相对独立的运行过程。在传统操作系统中，进程既是系统中独立运行的最小单位，也是系统中资源分配的基本对象，作为一个重要的传统设施、被用来支持应用软件之间的并行性。在现代操作系统中、进程不再是系统中独立运行的最小单位。为了支持应用软件的内部并行性，绝大多数现代操作系统在进程内部引入了线程这一概念，线程是现代操作系统中独立运行的最小单位。同一进程中的多个线程共享进程中的资源、在现代操作系统中，进程仍然是资源分配的基本对象。

虚拟存储器也是现代操作系统中一个重要的抽象概念。所谓虚拟存储器，简单地说，就是进程的逻辑地址空间。它是现代操作系统对计算机系统中多级物理存储体系进行高度抽象的结果。由于传统操作系统不允许应用软件管理虚拟存储器，因而几乎

没有提供与虚拟存储器相关的系统服务。在传统操作系统中，虚拟存储器及其相关的系统服务隐藏在程序接口之后，不为程序设计人员所见。与传统操作系统不同，现代操作系统允许应用软件管理虚拟存储器，它们提供了许多与虚拟存储器相关的系统服务[5,6]。从发展趋势上看，虚拟存储器将成为现代操作系统中越来越常见、越来越重要的一个抽象概念。

文件是现代操作系统中另一个最常见、最重要的抽象概念。所谓文件是指命名了的字节流。它是现代操作系统对计算机系统中种类繁多的外部设备进行高度抽象的结果。它屏蔽了各种外部设备千差万别的构造细节，用几种简洁的相关操作取代了各种外部设备多种多样的使用方式，为应用软件提供了一种统一的、规范的抽象设施。文件及其相关操作的引入极大地方便了应用软件开发人员，但对某些应用软件来说，这种统一的、规范的抽象设施未必合乎其意。

随着计算机系统的应用范围不断扩大，个性化应用软件越来越多，它们从功能和性能等方面对操作系统提供的基础设施以及相关的使用方式提出了各种各样的要求。为了满足这些要求，许多年来人们不断引入新的操作系统抽象概念，在操作系统中实现新的系统服务。这样做的结果是，操作系统越来越大、越来越复杂，操作系统的可靠性和性能也随之变得越来越差。这一现象近些年来引起了操作系统研究人员的关注。如何精减操作系统的功能，使其灵活、可靠、高效，已经成为操作系统领域一个重要的研究课题。

四、现代操作系统的内部功能

不管各个现代操作系统在具体功能上有多大差别，它们的功能总是可以分为两类：一是启动、终止以及控制应用软件的运行，二是分配、回收以及控制系统中的各类资源。前者称为进程管理，包括进程控制和进程通信。后者称为资源管理，包括软件资源管理和硬件资源管理。

现代操作系统的进程控制功能总是围绕进程状态转换模型进行设计。一般地，进程状态转换模型包含三个基本的状态：运行状态、就绪状态以及阻塞状态。有些系统基于某种需要（比如调试程序、调节负荷等）还增加了挂起状态。除运行状态外，就绪状态、阻塞状态以及挂起状态在操作系统具体实现过程中常常被进一步分解为若干个子状态。显然，进程状态转换模型不同，进程控制功能也就不同。

现代操作系统常常提供两种类型的通信机制：一是低级的同步机制，二是高级的通信机制。低级的同步机制只允许进程之间交换少量的、种类固定的状态信息，高级的通信机制则允许进程之间交换大量的、种类多样的数据信息。最经典的同步机制是信号量机制。最常用的通信机制是消息传递机制、管道机制以及信箱机制。通常，高级的通信机制在实现当中需要引用低级的同步机制。

现代操作系统的软件资源管理功能建立在两种抽象模型基础上，即虚拟存储器模型和文件模型。前者适用于进程可以"直接控制"的软件资源的管理，后者适用于进程只能"间接访问"的软件资源的管理。进程、虚拟存储器以及文件之间的关系非常类似于物理处理机、物理存储器以及物理设备之间的关系：物理处理机可以直接寻址物理存储器，但只能间接存取物理设备。

现代操作系统的硬件资源管理功能通常被进一步分解为处理机管理、存储器管理以及设备管理。尽管这几种硬件资源管理功能管理的对象各不相同，操作系统在其中扮演的角色却是一样的。现代操作系统硬件资源管理功能的实现遵循"申请—分配—使用—释放—回收"这一模式，在这种模式中，操作系统扮演着资源分配者和回收者的角色。

在现代操作系统中，与硬件资源管理功能密切相关的一个内部功能是异常与中断管理功能。该功能主要用来处理来自于处理机运算、存储器地址变换、设备运转过程中所发生的各种异常与中断事件。由于异常与中断管理功能几乎从不通过程序接口提供给程序引用，因而从某种意义上说，该功能是操作系统中唯一的、真正的"内部"功能。

进程是操作系统中的动态实体，资源是操作系统中的静态实体。因此，进程管理体现了操作系统内部功能模型的动态面，而资源管理则体现了操作系统内部功能模型的静态面。只有将二者结合起来，才能比较完整地刻画出操作系统内部功能模型。事实上，在操作系统内部，进程的状态转换总是伴随资源的分配与回收，这一事实充分说明，进程管理和资源管理是同一操作系统内部功能模型的两个不同侧面。

顺便指出，有些现代操作系统将网络通信功能——诸如 TCP/IP 协议——也作为自己的内部功能。这种操作系统常常被称为网络操作系统。在网络操作系统的支持下，人们可以使用网络应用软件，比如电子邮件软件、文件传输软件、WWW 浏览软

件等。

五、现代操作系统的组成结构

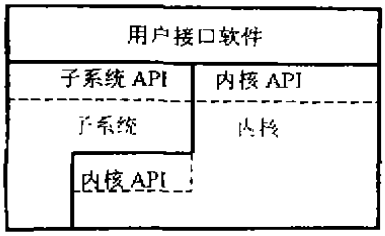

图1 现代操作系统基本结构模型

传统操作系统所采用的单体结构因其不便于修改、扩展、移植操作系统而被现代操作系统放弃，绝大多数现代操作系统采用模块结构。图1给出了现代操作系统基本结构模型，一般地，现代操作系统可以分为四个构件：一是用户接口软件，用于实现用户接口功能；二是程序接口模块，用于实现程序接口功能；三是子系统，用于实现操作系统的扩展功能；四是内核，用于实现操作系统的基本功能。

在现代操作系统中，用户接口软件是独立性最强的构件，它往往就是一个应用软件。对用户输入的每一个命令，该软件或者直接实现其功能，或者调用其它应用软件来完成对该命令的处理。无论是在内部实现命令功能，还是在外部完成对命令的处理，绝大多数情况下该软件需要引用操作系统提供的系统服务。与普通的应用软件一样，该软件对操作系统提供的系统服务的引用必须通过程序接口来完成。许多现代操作系统同时提供了多个用户接口软件供用户选用。事实上，用户完全可以用操作系统提供的系统服务实现自己的用户接口软件。

人们常常忽视程序接口模块的研究，其实它在现代操作系统中的地位是至关重要的，因为它是应用软件取得现代操作系统提供的系统服务的唯一通道。程序接口模块主要用来实现系统调用的引用机制，包括参数类型以及访问权限的检查、保护域转换、动态链接等功能；经常地也实现信号检测和处理等功能。

由于采用单体结构，传统操作系统没有提供子系统这一构件，操作系统的内部功能全部在内核中实现。为了增加操作系统的灵活性，现代操作系统引入了子系统。在现代操作系统中，操作系统的内部功能被分解为扩展功能和基本功能两部分，并分别在子系统和内核中实现。当然，哪些内部功能是扩展功能，哪些内部功能是基本功能，对现代操作系统来说并没有一个统一标准，而是由各个现代操作系统自己规定。对应用软件来说，子系统所提供的系统服务只是可供选择的方案，应用软件可以重新实现这些系统服务，并且可以实现子系统没有提供的功能。通常，子系统以一组动态链接库的形式出现在系统中。在采用"微核技术"的系统中，子系统则是一组服务进程，应用软件与子系统采用客户/服务器模式相互作用(图2)。由于动态链接库中的各个过程直接在应用软件地址空间中运行，因此系统效率较高，但安全性能较差。相反，服务进程与应用软件拥有不同的地址空间，因而系统安全性能较好，但由于应用软件访问服务进程时需要跨越地址空间，从而导致系统效率下降。目前，人们正在研究保护共享库(PSLs)技术，希望该技术既能保留动态链接库和服务进程各自的长处、又能避免它们的不足[5]。顺便指出，在新一代面向对象操作系统中，子系统表现为一组类库或一组对象[19]。

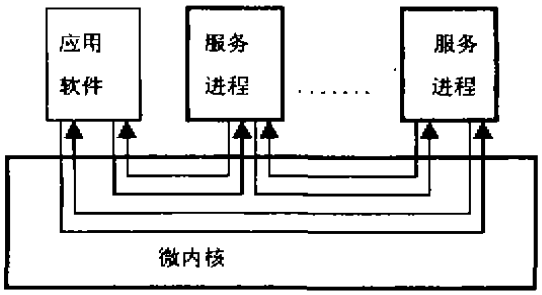

图2 基于微核的操作系统结构模型

内核是最能体现操作系统本质特征的唯一构件，因而它的功能、性能、稳定性历来是操作系统研究人员关注的焦点。经过多年的努力，人们取得了"微核技术(Microkernel)"这一重要研究成果[11]。该技术的基本思想是：尽最大努力剔除内核里的多余成份，将它们移到核外实现，内核只实现一组简单的抽象，提供一些基本的机制，以保持内核短小精悍。随着对微核技术的深入研究，人们又提出了"纳核技术(Nanokernel)"[12,13]。其基本思想是：在微核技术的基础上进一步精减内核的功能。由 MIT 提出的"外核技术(Exokernel)"是近些年来内核技术的最新发展[14]。该技术主张：内核的主要职责是保证应用软件和子系统能够安全地直接使用硬件平台提供的主要硬件设施，所有的抽象设施都应当移到核外实现。必须指出，尽管大多数现代操作系统的组成结构并未采用这些新的内核技术，但可以肯定这些新的内核技术将在新一代操作系统组成结构中发挥主

导作用。

六、现代操作系统的安全措施

在现代操作系统中有多个应用软件在同时运行。这些应用软件或者由于相互合作或者因为共享资源而存在着种种制约关系。因此，它们的任何一个病态行为对整个系统的安全性都是一种威胁。大多数现代操作系统采用以下三种措施来防范这种威胁，

● 空间隔离。现代操作系统常常为不同的应用软件规定不同的地址变换函数，使得不同应用软件其逻辑地址空间被映射到物理存储体系的不同区域，从而实现不同应用软件逻辑地址空间的彼此隔离。这样做将避免应用软件之间的相互干扰，

● 访问控制，如果多个应用软件共享某些代码或数据，那么空间隔离技术将无法避免这些应用软件之间相互干扰。此时，需要使用访问控制技术。访问控制技术分别就应用软件的各个部分规定了在什么情况下它们可以被访问以及可以被施加那些访问操作。

● 特权保护。在现代操作系统中，访问控制并不限于简单的读写控制，还包括特权控制。在提供特权控制的系统中，所有的代码和数据均被赋予某种特权级别，对它们的访问均按照某种特权规则进行。

特权控制并不仅仅是访问控制的一种。计算机系统中的某些指令——诸如改变地址变换函数的指令、改变访问控制权限的指令、改变代码和数据特权级别的指令等——如果不加以控制使用，对那些蓄意制造麻烦的应用软件来说，空间隔离技术以及访问控制技术都将失去效用。这些指令常常被人们称为特权指令，特权控制也意味着对特权指令的使用加以控制。

尽管一些计算机系统支持多级别特权，但许多现代操作系统只使用两个特权级别。其中，一个用于应用软件，另一个用于操作系统。

一般地，为了保护操作系统，我们可以为操作系统规定一个专用的地址变换函数，使得操作系统的逻辑地址空间与各个应用软件的逻辑地址空间隔离开来。然而，由于应用软件经常要引用操作系统提供的系统服务，因此这样做将导致在应用软件与操作系统之间进行频繁的跨越地址空间的访问，从而增加系统开销。所以绝大多数现代操作系统将其代码和数据作为应用软件逻辑地址空间的一部分，为系统中所有应用软件共享。在这种情况下，操作系统的保护只能利用访问控制技术和特权保护技术来完成。

结束语 本文概述了现代操作系统的基础知识，这些知识是人们进一步学习、使用以及研究操作系统的起点，必须指出，由于都建立在冯·诺依曼硬件体系结构上，因此现代操作系统不是对传统操作系统的简单否定，相反它继承了传统操作系统的许多优秀成果。同样，如果硬件体系结构不发生本质变化，下一代操作系统也不会对现代操作系统进行简单否定，它一定会继承现代操作系统的优秀成果，

参 考 文 献

1 Bach M J. UNIX 操作系统设计. 见：陈葆钰等译. 北京大学出版社，1989

2 ISO 标准. 操作系统标准之一·计算机环境的可移植操作系统界面 POSIX. 1. 见：中软总公司第二开发部译. 电子工业出版社，1991

3 IBM 资料. AIX Version 3. 2 for RISC System/6000. Technical Reference：Kernel and Subsystem，Volume 4. 1992

4 IBM 资料. AIX Version 3. 2 for RISC System/6000. Assembly Language Reference. 1992

5 Helen Custer. Windows NT 技术内幕. 见：程渝荣译. 清华大学出版社，1993

6 Richter J. Windows 95，Windows NT3. 5高级编程技术. 见：郑全水等译. 清华大学出版社，1996

7 Deitel H M. Kogan M S. The Design of OS/2，New York：Addison Wesley，1992

8 Iacobucci E. OS/2 Programmer's Guide. Osborne McGraw-Hill，U. S. A.，1988

9 Banerji A，Cohn D L. Protected Shared Libraries：A New Approach to Application-Extensible Operating Systems：[Technical Report TR-94-36]. Department of Computer Science and Engineering，University of Notre Dame，Notre Dame，IN，1994

10 Almes G T，et al. The Eden System：A Technical Review. IEEE Trans. On SE，1985，11(1)

11 Acetta M et al. Mach：A New Kernel Foundation for UNIX Development. In：Proceedings of the Summer 1986 USENIX Conference (Atlanta，GA，July). The USENIX Association，Berkeley，CA，1986. 93～112

12 Tan S M，et al. A Case for Nano-Kernels. Department of Computer Science，University of Illinois at Urbana-Champaign. Available at：http. //Choices. cs. uiuc. edu/Choices/uChoices. html

13 Alan B，et al. The KeyKOS Nanokernel Architecture. In：Proceedings of the USENIX Workshop on Micro-Kernel and Other Kernel Architectures. April 1992. 95～112

14 Engler D R，et al. Exokernel，An Operating System Architecture for Application-Level Resource Management. In：Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles. Copper Mountain，CO，December 1995