操作系统

第3章
CHAPTER 3

购买组装机(裸机)后,首要步骤是安装操作系统。若无操作系统,计算机则无法正常运行。操作系统若崩溃,计算机将失去所有功能。操作系统负责计算机的内存管理、资源分配、设备控制及网络操作等核心任务,并为用户提供交互界面。要了解计算机系统,就需要掌握计算机系统中的操作系统。操作系统是计算机科学里重要的内容之一,本章将从操作系统的基础概述、操作系统对进程的管理、操作系统对内存的管理、操作系统对文件系统的管理,以及操作系统对设备的管理5方面对计算机操作系统进行详细讲述。本章思维导图如图3-1所示。

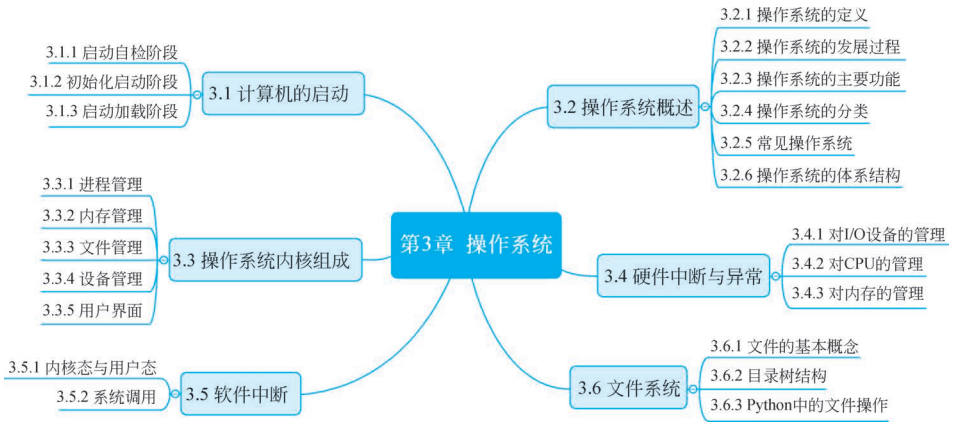

图3-1 第3章思维导图

3.1 计算机的启动

不论是台式机、笔记本的Windows系统或者Linux系统,还是手机的Android系统或者iOS系统,所有设备在开机启动过程中都会经过3个共同的阶段:启动自检阶段、初始化启动阶段和启动加载阶段。

计算机系统的启动过程共分为3个阶段:自检、初始化和加载。这些核心环节主要由

BIOS(Basic Input Output System)来执行。BIOS作为计算机启动的基础软件被固化在主板的ROM芯片中,包含基本输入/输出程序、系统设置信息、自检程序及系统自启动程序。用户可以根据需求,通过特定的按键组合(如Esc、F2或Delete键),进入BIOS配置界面进行个性化设置。需要注意的是,不同品牌和型号的台式机或笔记本电脑进入BIOS时的按键是不一样的。

3.1.1 启动自检阶段

当用户按下电源按钮后,计算机随即进入启动自检阶段。这一阶段,计算机刚接通电源,随后读取并运行BIOS程序。BIOS程序负责硬件的检测,它存放于只读存储器(Read Only Memory,ROM)中,这意味着即便在断电状态下,其存储的内容也不会丢失。这一过程也被称为加电自检(Power On Self Test,POST)。

加电自检的主要功能在于全面检查计算机的整体状态。通常,POST自检流程涵盖对CPU、ROM、主板、串并口、显卡及键盘等重要硬件组件的测试。若自检过程中发现问题,则计算机会给出相应的提示信息或发出警告声。

在启动自检过程中,计算机的屏幕上会显示出详细的自检信息,这对于用户了解计算机状态及可能出现的问题具有重要意义。通过这一流程,用户可以更直观地掌握计算机的硬件状况,并在必要时采取相应的解决措施。

3.1.2 初始化启动阶段

在完成启动自检阶段后,若系统确认无异常,则计算机将进入初始化启动阶段。在这一阶段,计算机将遵循BIOS中预先设定的启动顺序,识别并定位优先启动的设备。这些设备可能包括本地磁盘、CD驱动器或USB设备等。一旦确定了启动设备,计算机便会从该设备中加载并准备启动系统。此外,初始化启动阶段还需要完成寄存器的设置、外部设备的初始化和检测等关键任务。这些步骤共同确保了计算机能够按照预定的配置和设置,安全、稳定地启动和运行。在初始化启动过程中,计算机屏幕处于黑屏状态。

3.1.3 启动加载阶段

在初始化启动阶段顺利完成后,计算机将开始读取准备启动设备上的相关数据。考虑到大多数系统文件存储在硬盘中,BIOS将指定启动设备,并从中读取操作系统核心文件。鉴于不同的操作系统会使用不同的文件系统格式(如FAT32、NTFS、EXT4等),需要一个专门的启动管理程序来处理这些核心文件的加载过程。这个启动管理程序就是所谓的Boot Loader。

Boot Loader在启动过程中发挥着重要作用,主要表现在两方面:一是它为用户提供了选择不同启动项目的菜单,允许用户通过不同的启动项目启动计算机上的不同系统;二是它负责加载核心(Kernel)文件,直接指向可执行的程序段,从而启动操作系统。值得注意的是,在整个启动加载过程中,计算机的屏幕通常处于黑屏状态。

计算机启动的整个过程完成之后，操作系统开始装载进入内存，BIOS 开始将计算机的控制权移交给操作系统。也就是说，接下来计算机的所有操作将由操作系统来完成。

1. 内核装载阶段

在内核装载阶段，操作系统通过内核程序对各外围设备，如存储装置、CPU、网卡、声卡等进行测试与驱动。在此期间，部分操作系统可能会对硬件进行再次检测。这意味着，只有当操作系统开始使用内核程序进行设备测试和驱动时，其核心功能才正式接替 BIOS 的工作。以 Windows 为例，此阶段操作系统需加载各设备的驱动程序。操作系统要确保掌握当前所有外围设备的信息，这样才能准确加载相应的驱动程序。这些信息被详细记录在注册表中，操作系统通过访问注册表，如 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet，来获取已安装的驱动程序信息，并按需加载硬件的驱动。

Windows 操作系统中的注册表是一个重要的核心数据库，自 Windows 95 系统起，它便成为管理系统运行参数的主要工具，取代了原先的 Win32 系统中的.ini 文件。注册表以树状分层的结构组织了各种参数，这些参数包括软件和硬件的配置与状态信息、应用程序和资源管理器外壳的初始条件、首选项和卸载数据、计算机的系统设置和许可、文件扩展名与应用程序的关联、硬件的描述、状态和属性、计算机性能记录及底层的系统状态信息等。控制系统和应用程序的启动与运行是注册表的主要工作。每当 Windows 启动时，注册表会根据关机时创建的一系列文件来构建注册表，并且一旦注册表载入内存，它就会被持续维护，因此，注册表可以被视为一个关系数据库，其中存储了系统参数，它直接控制着启动过程、硬件驱动程序的加载及一些应用程序的运行，从而在整个系统中扮演着核心角色。总体而言，注册表是一个集成了各种系统信息的复杂数据库。全部系统和应用程序的初始化信息均集成在注册表中，从而使 Windows 操作系统能够顺利运行。当然，作为操作系统的核心数据库，如果注册表受到破坏，则无法控制系统的启动和运行，可能会导致系统启动运行异常，甚至使整个系统瘫痪，因此，在使用系统优化软件时，需要特别注意，应慎用优化注册表功能。

在内核装载过程中，计算机屏幕会显示操作系统的图标及进度条等信息，表示系统成功启动。

2. 登录阶段

在登录阶段，计算机主要执行两项关键任务。首先，启动所有在机器上完成了安装并设置为自动启动的 Windows 服务，其次，计算机会展示登录界面，以供用户输入凭据并进行身份验证。这两个步骤是登录过程中不可或缺的环节，确保了计算机系统的正常运行和用户安全。

Windows 服务也称为 Windows 服务程序，是在 Microsoft Windows 操作系统中运行的后台进程，是一种特殊类型的应用程序。这些服务可以在计算机启动时自动启动，并在后台执行特定的功能或任务，而无须用户的交互。Windows 服务通常在操作系统级别上提供某种功能或服务，例如网络连接、打印机管理、文件共享等。

Windows 服务的主要特点包括以下几点。

(1) 长期运行：Windows 服务属于长时间运行的应用程序。在系统启动时 Windows

服务自动启动,并且一直在后台持续运行,直到系统关闭或手动停止。

(2) 无须用户界面:与一般的应用程序不同,Windows 服务在没有用户交互的情况下执行任务,所以通常不显示任何用户界面。

(3) 自动启动:Windows 服务通常应设置为在系统启动时自动加载和运行,确保在系统可用时始终提供服务。

(4) 系统级功能:Windows 服务通常提供操作系统级别的功能和服务,这些功能会直接影响系统的正常运行。

(5) 可管理性:Windows 服务可以通过服务控制管理器(Service Control Manager,SCM)进行管理,用户可以通过 SCM 完成启动、停止、暂停、恢复服务和更改其启动类型等操作。

Windows 服务既能由 Microsoft 予以提供,也能由第三方开发者进行创建。诸多常见的 Windows 服务涵盖 Windows 更新服务、Windows 时间服务、Windows 安全服务等。此类服务提供了很多关键性能,例如自动更新、系统时间同步及安全保护等。

总之,Windows 服务是 Windows 操作系统里用于提供系统级功能与服务的后台进程,它们保障了系统的稳定性与可靠性,令用户能够无缝地使用各类功能和服务。

3.2 操作系统概述

第 1 章详细地介绍了冯·诺依曼体系结构计算机的组成。现代计算机系统由硬件系统和软件系统两部分组成。一台典型计算机的硬件包括运算器、控制器、存储器、输入设备和输出设备五部分,至少包含一个 CPU(可能是多核多线程的)、一定空间的内存(例如 8GB)和数目不等的 I/O 设备;在软件上由各式各样的程序构成,其数目、执行过程、软件需求各不相同,然而,随之而来的是一个现实问题:如何确保这些程序高效、便捷地利用计算机硬件资源满足各自的业务需求?例如,计算机只有一个 CPU,而同时运行的程序可能有数十个甚至数百个,那么如何合理分配和利用 CPU 的计算能力,以确保这些程序的执行符合用户的预期和需求?这正是操作系统需要面对和解决的核心问题。

3.2.1 操作系统的定义

操作系统(Operating System,OS)是管理和控制计算机硬件与资源的系统程序,为其他软件提供运行环境。操作系统是迄今为止最复杂的软件之一,例如常用的 Windows 11、Linux 操作系统等,实际的源代码量高达数千万行,一般由数百甚至数千名顶级程序员花费数年才能开发完成。一般认为,操作系统是计算机硬件和“用户”(人或者上层程序)之间的中间层,它使“用户”可以更方便、更有效地实现对计算机硬件和软件资源的利用和访问。操作系统给用户带来的主要益处有两个。一是高效地利用硬件资源,从而提升用户的工作效率。二是更便捷地利用计算机的软硬件资源,即降低用户使用这些资源的门槛和难度。

任何现代操作系统均需要很好地解决如下问题:

(1) 提供良好的机制，从而使上层程序或者程序集被高效地运行，能够充分地利用CPU 和内存执行程序。

(2) 作为通用管理程序管理着计算机系统中每个部件的活动，确保系统中的软硬件资源被合理有效地利用，并且及时处理出现的冲突问题。

(3) 计算机的主要功能是处理和存储数据，上述(1)和(2)主要是关于处理数据方面的工作，操作系统的另外一个重要的工作是需要提供一种通用、统一、高效的机制，实现数据访问和持久化存储。

(4) 操作系统还需要为用户提供一种方便的接口。

上述问题均是操作系统需要解决的核心问题。在现代计算机的使用过程中，操作系统是必不可缺的系统软件，下面将从计算机的发展过程来引出操作系统的核心功能。

3.2.2　操作系统的发展过程

操作系统并非与计算机硬件同期诞生，而是运用于计算机的发展进程之中，为了提升资源利用率、强化计算机系统性能，伴随计算机技术本身及其应用的逐渐发展，进而逐步发展起来的。操作系统涵盖的范围包含个人计算机端操作系统、工业应用操作系统及移动端操作系统。以下是操作系统发展的 7 个主要阶段。

1. 手工阶段

在计算机刚刚出现时，并没有操作系统。计算机的使用主要处于手工操作阶段。程序员直接与硬件打交道，直接编写程序。这造成输入/输出缓慢，相对来讲计算机处理速度过快，用户独占全机，人机速度矛盾突出。这一阶段的操作方式主要具有以下特点。

(1) 用户交互界面设计：在早期的手工操作阶段，用户与计算机的交互方式相对基础，主要通过控制台或纸带等方式向计算机发出指令。这一时期的用户接口设计相对简单，没有现代操作系统中图形用户界面(Graphical User Interface，GUI)的直观与便捷。

(2) 任务调配：在那个时代，作业的调度主要依赖于用户的手动操作。用户需将编写完成的程序提交给计算机，然后等待其按序处理。由于没有自动化的作业调度系统，所以作业的执行顺序完全取决于用户提交的顺序。

(3) 进程管理机制：在手工操作阶段，进程的概念尚未明确。每个用户提交的程序被视为一个独立的单元，由计算机按顺序执行。由于缺乏现代操作系统中的进程管理功能，如进程切换和同步等，所以导致程序的执行效率相对较低。

(4) 内存管理方法：在手工操作阶段，内存的管理同样相对基础。用户需要手动为编写的程序分配和管理内存空间。由于缺少现代操作系统中的内存管理功能，如内存分配、保护和碎片整理等，所以使内存的使用效率较低，并容易出现内存泄漏和冲突等问题。

(5) 文件管理：在手工操作阶段，文件的创建、保存、备份和删除等管理任务通常由用户亲自完成。没有现代操作系统的文件系统支持，文件管理工作相对复杂，并且容易发生数据丢失或损坏等问题。

(6) 设备管理：这一时期的设备配置与管理同样需要用户手动进行。用户需要负责配

置和管理计算机的输入/输出设备,如打印机和磁盘等。由于缺少现代操作系统的设备管理功能,如设备驱动程序和设备抽象层,所以设备的使用效率较低,容易出现设备冲突和错误。

(7) 系统维护:在手工操作阶段,系统的正常运行主要依赖于用户的专业知识和经验。用户需要定期地检查和维护计算机的硬件和软件。由于缺少现代操作系统中的系统维护工具和功能,如系统备份、恢复和监控等,使系统维护变得相对困难,容易出现因维护不当而导致的问题。

综上所述,手工操作阶段下的操作系统特性体现为简单性、烦琐性和低效性。随着计算机技术的不断革新,现代操作系统的诞生使计算机的使用变得更高效、便捷和稳定。

2. 批处理操作系统

随着 20 世纪 50 年代第一台计算机 ENIAC 的诞生,人们开始探索提高计算机利用率的方法,进而催生了批处理操作系统的概念。批处理操作系统是一种早期的操作系统类型,其核心功能是实现批处理作业的自动化处理。在这种操作系统中,用户将需要完成的任务和相关数据整合成作业,并提交给操作系统。操作系统会根据预设的规则和顺序自动执行这些作业,全程无须用户的直接干预,其主要特点有以下几点。

(1) 自动化的作业调度:操作系统能够基于作业的具体要求和特性,智能地安排作业的执行次序和资源分配,从而极大地提升了系统的效率和吞吐量。

(2) 批量作业处理:作业以集合的形式被提交和处理,减少了用户与计算机的直接交互次数,特别适用于处理大量且相似的任务。

(3) 全面的资源管理:操作系统负责协调并管理计算机的核心资源,如 CPU 时间、内存和磁盘空间等,确保每个作业都可以得到公平且高效的资源分配。

(4) 缺乏交互性:在批处理操作系统中,用户与计算机之间的直接互动被大大限制,用户通常只能通过提交作业和查看结果来与系统进行沟通。

批处理操作系统在早期的计算机应用中发挥了核心作用,尤其是在需要大规模数据处理和批处理的场景下。然而,随着用户对于交互式操作和实时反馈的需求增加,批处理操作系统逐渐被更灵活和交互性更强的分时操作系统和实时操作系统所取代。

现代操作系统融合了批处理、分时和实时处理等多种特点,以满足多样化的应用场景需求。例如,在现代的服务器环境中,可能同时存在批处理任务、实时处理请求和交互式用户操作,操作系统需要具备强大的任务管理和调度能力,以确保各种类型的工作负载都可以得到高效处理。

3. 分时操作系统时期

20 世纪 60 年代,分时系统(Time-Sharing Operating System)的概念应运而生。分时系统是一种多用户操作系统,允许多个用户通过终端或其他设备共享单一的计算机资源,并同时进行各自的操作。在这种系统中,计算机的 CPU 时间被切分成多个微小的时间片,每个用户在分配到的时间片内享有对计算机的控制权,从而进行各种交互操作。系统在这些用户之间快速切换,使每个用户都感觉自己仿佛在独占计算机资源。分时操作系统的核心特点体现在以下几方面。

（1）多用户共享：多个用户可以同时利用计算机资源，从而显著地提高了资源的利用率和系统效率。

（2）出色的交互性：用户能够通过终端与计算机进行实时对话和交互，以及时获得系统的响应和反馈。

（3）独立性保障：每个用户在各自独立的会话环境中工作，就像在使用一个专属于自己的计算机系统一样，用户间的操作互不干扰。

分时操作系统对计算机应用产生了深远影响，特别是在多用户交互的环境中。它通过划分 CPU 时间片并在用户间快速切换，不仅显著地提高了资源利用效率，还为用户提供了直观友好的操作体验。随着技术的持续进步，分时操作系统的概念得以拓展，并与网络通信、分布式计算等先进技术相结合，催生出现代复杂而高效的操作系统，其中，一个值得关注的特性是系统的及时响应能力，即系统能在短时间内响应用户需求，确保流畅的用户体验。

分时操作系统被广泛地应用于多用户环境，如服务器、终端系统或网络环境，为用户提供了便捷的交互方式。这种方式使多个用户能够并行工作，极大地提升了计算机的使用效率。在实际应用中，UNIX、Linux 和 Windows 等操作系统均具备分时处理能力，支持多用户同时登录和系统资源的共享。这些系统的广泛应用进一步证明了分时操作系统的重要性和价值。尽管分时操作系统为计算机应用带来了显著变革，尤其是在多用户交互环境中，但在处理大量并发用户时，也面临着资源竞争和调度方面的挑战。为确保系统的稳定性和高效性，操作系统必须采取有效的资源管理和调度策略。首先，资源管理对于确保分时操作系统的稳定性和性能至关重要。操作系统需要确保各个用户进程在访问共享资源时不会发生冲突，并且要合理地分配资源，以满足不同用户的需求，其次，设计高效的调度算法也是分时操作系统的关键任务。通过合理的调度算法，操作系统可以确保每个用户在其分配的时间片内获得足够的 CPU 时间，从而实现系统的公平性和响应性。此外，随着计算机技术的不断演进，其他类型的操作系统（如实时操作系统和分布式操作系统）也在不同领域得到了广泛应用。实时操作系统强调对时间的精确控制，适用于需要实时响应的应用场景，如工业自动化、航空航天等；而分布式操作系统则注重将多个计算机资源连接起来，形成一个整体，以提供更高的性能和可扩展性，适用于大规模计算和数据处理等场景。

分时操作系统在为用户提供便捷交互和资源共享方面具有显著优势，但在处理大量并发用户时仍面临资源竞争和调度问题。为了应对这些挑战，操作系统需要采取有效的资源管理和调度策略，并确保与其他类型的操作系统相互补充，以满足不同应用场景的需求。

4. 多程序环境操作系统时期

20 世纪 70 年代，多程序环境操作系统崭露头角，这一创新将内存划分为多个独立区域，允许不同的程序分别存放于其中。这种设计不仅简化了分布式计算机系统资源的管理和利用，还提高了整体效率。与早期的单任务操作系统相比，多程序环境操作系统实现了多个程序的并行运行，显著地提升了 CPU 的利用率和系统效能。

在这一多程序环境中，操作系统扮演着至关重要的角色，负责以下核心功能。

（1）进程管理：操作系统负责创建、调度和监管多个进程。它通过精心安排，确保各个

进程在 CPU 上能够公平地分享资源和执行时间。

(2) 内存管理：系统高效地为每个进程分配内存空间，并实施了内存保护机制，有效地预防了进程间的相互干扰和潜在错误。

(3) 资源分配：操作系统协调各个进程对 CPU、输入/输出设备、文件等资源的访问，从而避免了资源冲突和死锁现象的发生。

(4) 进程通信机制：这一机制为进程间的信息交流和同步提供了可能，促进了进程间的协作和数据共享。

(5) 灵活的调度策略：操作系统根据多种调度算法判断哪个进程应优先获得 CPU 使用权，从而实现系统性能的最大化。

相比于以往的操作系统，多程序环境操作系统的主要优势如下。

(1) 优化的资源利用：通过并行运行多个程序，CPU 和其他系统资源得以高效利用，大幅减少了资源的闲置时间。

(2) 增强的并发性：多个程序的同时执行不仅提高了系统的响应速度，还增强了整体执行效率。

(3) 提升的用户交互体验：允许用户在后台运行一个程序的同时，前台进行其他操作，极大地方便了多任务处理。

(4) 稳固的系统稳定性：当一个进程出现问题或崩溃时，其他进程仍可正常运行，这增强了系统的整体可靠性。

现代操作系统，诸如 Windows、Linux 和 macOS，均支持多程序环境。它们通过高效的进程管理和资源分配策略，为用户提供了同时执行多个程序的能力，使用户能够更加充分地利用计算机资源。

5. 分布式操作系统时期

20 世纪 80 年代，分布式系统概念的出现改变了计算机系统的局面。分布式操作系统是指计算机系统的资源不仅限于一台主机，而是由多台计算机组成，能够简单高效地管理和利用分布式计算机系统的资源。分布式操作系统(Distributed Operating System)是一种用于管理分布式系统中多个节点(计算机)的操作系统。它将计算任务分布在多个节点上，通过网络进行通信和协作，以实现资源共享和协同工作。分布式操作系统的主要特点和功能包括以下几点。

(1) 分布式处理：任务可以在不同的节点上并行执行，从而提高系统的性能和处理能力。

(2) 资源共享：各个节点的资源，如内存、磁盘、处理器等，可以通过分布式操作系统进行共享，从而提高资源利用率。

(3) 容错性：系统具有容错能力，即使部分节点出现故障，也能保证整个系统的正常运行。

(4) 透明性：对用户来讲，分布式操作系统提供了一种透明的方式来使用分布式系统，用户无须关心任务具体在哪个节点上执行。

（5）通信与协调：操作系统负责节点之间的通信和协调，确保各个节点能够协同工作，完成共同的任务。

（6）安全性：操作系统提供了安全机制，保证数据的完整性和保密性，以及对资源的访问控制。

在分布式系统中，如分布式数据库、计算集群和云计算平台，分布式操作系统发挥着至关重要的作用。它显著地提升了系统的可扩展性、可靠性和性能，然而，随着系统的复杂性和规模的增加，分布式操作系统也面临着网络时延、节点故障、数据一致性和同步等一系列挑战。为了克服这些挑战，需要精心设计和实施有效的分布式协议、容错机制和资源管理策略。这些策略和机制不仅提高了系统的稳定性，还确保了数据的完整性和一致性。

在实际应用中，像 Hadoop 和 Kubernetes 这样的分布式操作系统已经得到了广泛认可和应用。Hadoop 在大数据处理领域具有卓越的表现，而 Kubernetes 则在容器编排和云原生应用中表现出色。

6. 云操作系统时期

21 世纪初，伴随信息技术的疾速进步，以及云计算概念的提出，云操作系统逐渐变为计算机操作系统的发展走向。云操作系统以云计算技术为基石，把计算与存储资源安设在云端，用户借由互联网及云端服务进行通信，达成数据处理及存储。云操作系统（Cloud Operating System）是一类用于管控和掌控云计算环境中资源的软件。它承担分配与管理云计算资源，涵盖计算、存储、网络等，以保障云环境的高效运作和资源的最优利用。云操作系统的主要功能包括以下几种。

（1）资源管理：对云中的计算、存储和网络资源进行管理和分配，确保资源的合理利用和高效分配。

（2）虚拟化支持：支持虚拟化技术，使多个虚拟机可以在同一物理服务器上运行，提高资源利用率。

（3）自动化和弹性：提供自动化的资源部署和配置功能，根据需求快速扩展或收缩资源，实现弹性伸缩。

（4）监控和计费：监控资源使用情况，提供计费和成本管理功能，帮助用户了解和控制云计算成本。

（5）安全性和访问控制：确保云环境的安全性，包括用户认证、授权和数据保护等。

（6）服务管理层：管理和提供各种云服务，如云数据库、云存储、云应用等。

云操作系统可以提高云计算的效率和灵活性，使企业和开发者能够更快地部署应用、更好地管理资源，并降低成本。它们在云计算基础设施中起着关键的作用，为云计算的发展和应用提供了重要的支持。

常见的云操作系统有 AWS Elastic Compute Cloud（EC2）、Microsoft Azure、Google Cloud Platform 和 VMware vSphere 等。这些操作系统提供了一系列的工具和服务，使用户能够轻松地构建、部署和管理云应用和服务。

7. 未来的操作系统

随着科技的不断发展和进步,未来的操作系统将会更加先进和智能。未来的操作系统可能会在以下几方面发展和演进。

(1) 智能化和自动化:操作系统可能会更加智能化,能够自动识别用户的需求和偏好,提供个性化的服务和功能。可以通过机器学习和人工智能技术来学习和适应用户的行为,提供更加智能的交互体验。

(2) 跨设备和平台的一致性:随着物联网的发展,未来的操作系统可能会更好地支持各种设备和平台,实现跨设备的无缝连接和数据同步。用户可以在不同设备上获得一致的操作体验和功能。

(3) 增强的安全性:随着网络安全威胁的不断增加,未来的操作系统可能会加强安全性功能,包括更好的身份验证、数据加密和恶意软件防护。

(4) 虚拟现实和混合现实支持:随着虚拟现实(VR)和混合现实(MR)技术的发展,操作系统可能会更好地支持这些新兴技术,提供更好的沉浸式体验和交互方式。

(5) 云原生和边缘计算:随着云计算和边缘计算的发展,操作系统可能会更加注重云原生应用的支持,以及在边缘设备上的高效运行。

(6) 更加开放和可定制:未来的操作系统可能会更加开放,允许用户和开发者更容易地定制和扩展系统功能,以满足不同的需求和应用场景。

(7) 对量子计算的支持:随着量子计算技术的逐渐成熟,未来的操作系统可能需要进行相应改进,以支持量子计算设备和应用。

(8) 更强大的语音和手势交互:操作系统可能会更加注重语音和手势交互,提供更加自然和便捷的用户界面。

(9) 更好的能源管理:随着对能源效率的关注增加,操作系统可能会更加注重能源管理,优化设备的能耗,延长电池寿命。

(10) 更强的隐私保护:用户对隐私保护的需求不断增加,未来的操作系统可能会提供更加强大的隐私保护功能,控制数据的收集和使用。

以上只是一些可能的发展方向,实际的发展取决于技术的进步和市场的需求。操作系统的发展是一个不断演进的过程,随着新技术的出现和用户需求的变化,未来的操作系统将不断创新和改进。

3.2.3 操作系统的主要功能

操作系统是管理计算机硬件和软件资源的系统软件,它的主要功能包括以下几种。

(1) 进程管理:负责管理计算机系统中同时运行的多个进程,包括进程的创建、调度和终止等。

(2) 内存管理:有效地管理内存空间,包括内存分配、内存保护和内存回收等。

(3) 设备管理:管理计算机系统中的各种硬件设备,包括设备驱动程序的安装、设备的分配和使用等。

(4) 文件管理：提供文件和目录的管理功能，包括文件的创建、打开、读写、删除等操作。

(5) 用户接口：为用户提供与计算机系统交互的接口，包括命令行界面、图形用户界面等。

(6) 系统安全：确保操作系统的安全性和稳定性，防止未经授权的访问和操作。

(7) 资源共享：实现计算机系统中资源的共享，包括硬件资源和软件资源的共享。

(8) 网络通信：支持计算机之间的网络通信，实现网络资源的共享和信息传递。

这些功能使操作系统能够协调计算机系统的各部分，提供一个稳定、高效和安全的运行环境，方便用户使用计算机系统并充分发挥其性能。不同类型的操作系统可能会在功能上有所侧重，但以上是操作系统的一些主要功能。

3.2.4　操作系统的分类

自 20 世纪电子计算机问世起始，操作系统始终是计算机科学的关键研究领域，其发展与演化历经了很长一段时期，并且仍在持续地改良。伴随计算机软硬件技术在材料、工业、方法上的不停进步，还有通信与互联网技术的高速发展，现代操作系统多数在功能达成上趋于“同质化”，也无法脱离网络环境，主要分类如下。

1. 批处理操作系统

批处理操作系统是 20 世纪早期的操作系统，其观念为将每名运行的程序视作一个作业(Job)，每名用户把自行编制的程序当作一个 Job 提交给操作系统，批处理操作系统会依照作业的顺序由一个作业转移至另一作业。倘若当前作业运行成功，便输出结果数据，不然报错并转至下一道程序。目前，批处理操作系统在一些极为简单的应用系统中仍然在运作，此外，目前在众多的操作系统中仍保留了批处理作业执行模式。

2. 分时操作系统

分时操作系统在现代操作系统中依然占有举足轻重的地位，例如，现在流行的 Windows 11、macOS、Linux 等操作系统本质上还是分时操作系统。为高效地利用 CPU 等计算资源，分时操作系统引入了多道程序的概念，其核心思想是把多道程序装载到内存，然后通过时间片轮转等手段来共享 CPU 的执行时间，使这些程序轮流使用 CPU(例如 20ms 轮流一次)。由于 CPU 执行指令的速度非常快，基本上每个用户程序都能够得到及时响应，从而使每名用户有种整个系统都在为自己单独服务的错觉。UNIX 是分时操作系统的典范，它提出了一个现代操作系统历史上具有划时代意义的概念——进程。在后面的内容中会对进程更详细地进行介绍。

3. 实时操作系统

实时操作系统和分时操作系统在很多方面类似，它们间的显著不同是：实时操作系统必须在特定的时间段内完成特定的任务，否则可能造成灾难性的后果。例如，运行在飞机上的很多控制程序。实时操作系统通常被应用于实时应用当中，例如交通、医疗、军事、航空航天、工业控制、汽车控制系统等，而且，实时操作系统和其上层应用软件经常作为固件被嵌入

计算机硬件中,这类系统常被称为嵌入式系统。常见的实时操作系统有很多,例如VxWorks、RT-Linux、μClinux等,其实,在现代操作系统中,分时操作系统和实时操作系统的分界不是那么明显,很多实时操作系统是由分时操作系统增加实时调度机制改造而成的,例如RT-Linux、μClinux、嵌入式Windows操作系统等。此外,大部分现代分时操作系统具备一定的实时调度功能,这些在Windows 11、Linux中均有体现。

4. 并行操作系统

在一个计算机中安装多个CPU,这在现代计算机系统中非常常见,小到普通的多CPU多核的微机、PC服务器,大到类似天河Ⅱ号这样的超级计算机。这些CPU之间通过高速内部总线连接在一起,每个CPU可以处理一个程序或者一个程序的一部分,这意味着很多计算任务可以被真实地并行处理,而不再是单CPU单核情况下的串行处理。能支持这类并行计算机架构的操作系统被称为并行操作系统。当前,大部分并行操作系统还是源于分时操作系统,例如Linux、Windows Server等均可以被认为是并行操作系统。

5. 分布式操作系统

随着网络化,尤其是互联网的发展,使计算可以被延展到整个网络中的计算机节点,这是一个被充分研究的领域,叫作分布式计算。能够很好地支持分布式计算的操作系统就是分布式操作系统,其所需要管辖和协调的资源包括诸多的计算资源、存储资源和网络资源。与并行操作系统一样,现代分布式操作系统通常是在主流分时操作系统中植入网络和分布式处理功能之后得以体现的,例如Linux操作系统、Windows操作系统、macOS操作系统,甚至iOS、Android等都可以被认为是分布式操作系统。

3.2.5 常见操作系统

1. UNIX操作系统

UNIX操作系统是1969年由AT&T贝尔实验室的Thomson和Ritchie研发出来的。自其推出以来,经历了很多版本的演化,直到今天依然具有顽强的生命力,已然成为操作系统界的伟大丰碑,其诸多开创性的光辉思想依然照耀着现代操作系统的发展之路。UNIX是一个典型的多用户分时操作系统,具有可移植、多进程、基于抽象文件概念的设备无关性等特性,其设计理念非常简洁优美,操作系统内核由几百个简单、功能单一的函数构成,这些函数可以组合起来完成任何复杂的处理任务,所以其灵活性和可扩展性非常好。在结构上,主要包含内核、命令解释器(Shell),一组标准工具(例如著名的文本编辑器vi、emacs等)和其他应用程序。基于UNIX的设计理念和部分的开放源码,出现了大量类似UNIX的操作系统,例如AT & T的System V、FreeBSD、HP-UX,IBM的AIX,SUN的Solaris,以及现在流行的macOS X、各类Linux发行版本等。

特别要说明Linux。1991年,芬兰学生Linus Torvalds根据类UNIX系统Minix编写并发布了Linux操作系统内核,其后在Richard Stallman的建议下以GNU通用公共许可证发布,成为自由软件UNIX变种。如今,各类Linux发行版本在桌面和服务器市场均大放异彩,例如,面向桌面的Ubuntu Desktop版本、面向服务器的RedHat和CentOS等,尤

其在服务器市场，Linux 的市场占有率处于绝对领先地位。

2. Windows 操作系统

Microsoft Windows 操作系统是微软公司在给 IBM 设计 IBM-DOS 操作系统的基础上发展而来的图形化操作系统。在 20 世纪 80 年代后期，微软公司开始开发替代 MS-DOS(微软脱离 IBM-DOS 之后的操作系统版本)的新的图形化单用户操作系统，即 Windows 操作系统。Windows 操作系统主要提供图形化的人机交互接口，便于桌面用户方便、简易地操控计算机，其核心概念就是窗口(Window)。从此，拉开了 Windows 操作系统的演进之路，在桌面版本上，其发展历经了 Windows 3.1、Windows 95、Windows Me、Windows 98、Windows XP、Windows Vista、Windows 7 直至 Windows 11 等。服务器版本也从较早的 Windows NT，经历了 Windows Server 2000、Windows Server 2003、Window Server 2008、Windows Server 2016 等不同版本。

3. iOS 和 Android

进入 21 世纪以来，随着通信技术、移动通信网络和互联网等的飞速发展，以及移动终端的小型化，很快，移动终用户端远远超过桌面用户，出现了众多面向移动终端的操作系统，如早期的 PalmOS、Symbian、Windows Phone 等，现在占主导地位的是苹果公司的 iOS 和谷歌公司的 Android 操作系统。

iOS 被看成 iPhone 版本的 macOS X，OS X 的内核 Darwin 是类 UNIX 操作系统核心，所以，iOS 可以被认为是移动版的类 UNIX 操作系统。

3.2.6　操作系统的体系结构

操作系统的体系结构是操作系统的组织和设计方式，它定义了操作系统的各个组件之间的关系、交互方式及功能的划分。操作系统的体系结构描述了如何将系统的核心功能、模块和层次组织起来，以实现有效的资源管理、进程调度、内存管理、设备管理等操作系统的基本任务。它决定了操作系统的整体架构和运行机制。

常见的操作系统体系结构包括单核体系结构、分层体系结构、微内核体系结构、客户-服务器体系结构等。每种体系结构都有其特点和优势，适用于不同的场景和需求。了解操作系统的体系结构有助于理解操作系统的工作原理、组件之间的交互及如何进行系统设计和优化。它是操作系统研究和开发中的重要概念。操作系统的体系结构可以有多种不同的形式，以下是一些常见的操作系统体系结构。

(1) 单核体系结构：在这种体系结构中，所有的操作系统功能都集中在一个核心模块中。核心负责管理系统资源、进程调度、内存管理等关键任务。

(2) 分层体系结构：将操作系统划分为多个层次，每层负责不同的功能。这种分层结构有助于提高系统的模块化和可扩展性。

(3) 微内核体系结构：采用微内核的设计，将核心功能最小化，只提供基本的服务，如进程间通信、内存管理等，其他功能则通过外部模块实现。

(4) 客户-服务器体系结构：将操作系统的功能分为客户和服务器两部分。客户请求服

务，服务器提供相应的功能。

(5) 分布式体系结构：适用于分布式系统，操作系统的功能分布在多个节点上，通过网络进行协作和通信。

(6) 实时体系结构：专门为实时应用设计的体系结构，强调实时性和确定性。这种体系结构通常具有优先级调度和硬实时响应能力。

这些体系结构的选择取决于操作系统的设计目标、应用场景和性能要求。不同的体系结构有其各自的优势和适用范围，操作系统开发者会根据具体需求选择合适的体系结构来构建操作系统。同时，现代操作系统可能会结合多种体系结构的特点，以实现更高效、灵活和可靠的系统。

3.3 操作系统内核组成

操作系统的主要工作是对计算机的主要资源，如 CPU、内存、输入/输出设备等计算存储资源，提供有效的管理机制，使上层用户(指应用程序和计算机操作人员)可以高效、便利地利用计算机开展工作。此外，计算机的另外一个核心功能是提供对数据的持久化存储。这些内容是操作系统必须支持的核心功能，通常组成了操作系统内核的组成部分，被称为进程管理、内存管理、文件管理和设备管理。操作系统的核心功能组成如图 3-2 所示。

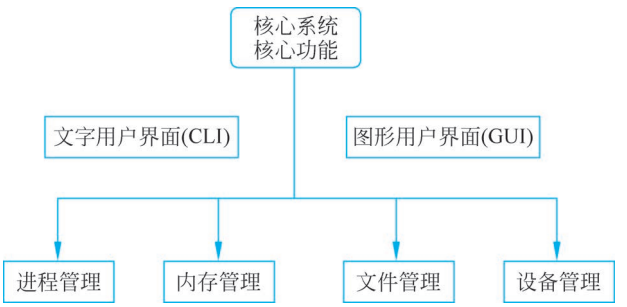

图 3-2 操作系统的核心功能组成

3.3.1 进程管理

操作系统中最重要的资源是 CPU 资源，主要实现算术运算、逻辑运算等关键功能。高效利用 CPU 的计算能力是操作系统内核设计最关键的问题之一。进程管理是现代操作系统高效使用 CPU 等计算资源的最重要的技术。在介绍进程管理之前，先介绍几个重要术语，即程序、进程和线程，接下来再重点介绍进程管理的核心组件——进程调度器。

1. 程序、进程和线程

(1)程序(Program)：程序指由程序员编制的指令的集合，一般存储在外部存储器中，如普通机械硬盘、固态硬盘、光盘等，例如，微信安装到硬盘上的可执行程序对应的各种文件。

(2) 进程(Process)：进程指一个程序被加载到内存，正在运行，但尚未结束。换而言

之，进程是一个驻留在内存中正在运行的程序，例如，当双击 Windows 11 桌面上的微信图标时，Windows 操作系统的装载器（Loader 程序）将为微信程序在内存中配置各种相关资源，即其对应的执行环境，然后把微信程序镜像装载到内存，并启动，使其执行，这时硬盘上的微信程序被转换为内存中正在执行的进程。

（3）线程（Thread）：从进程定义可知，进程可以被分成两部分，即执行环境等相关资源、可执行的指令集合。较早的操作系统，如 UNIX 等只支持进程概念，后面一些操作系统，例如 Windows，为了进一步有效地使用 CPU，以满足进程内部的不同执行子过程间的并发能力，把进程的执行部分分割为更小的执行线索，这些执行线索被称为线程。如今，线程已经成为现代操作系统任务调度的一个标志性的概念，例如，Linux、macOS X、iOS 等都提供了线程管理机制。简而言之，进程是正在运行中的程序，线程是进程的执行部分，可以用一个简单等式来描述这种关系：进程＝公共数据资源＋线程集，进程和线程的关系如图 3-3 所示。

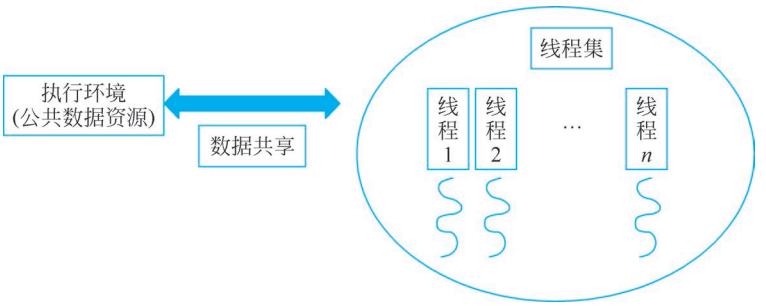

图 3-3　进程和线程

2. 进程调度器

为了有效地利用 CPU，现代操作系统都支持多进程/多线程的并发执行，操作系统中负责进程或线程调度的部件称为进程调度器或者任务调度器，即 schedule 程序，其作用是有效地调度多道程序，使之实现并发（Concurrency）调度的目的。从第 1 章中关于计算机架构组成中可知，单核计算机中只有一套 ALU 和 CU（单核 CPU），同一时刻只能执行一条指令，所以，单核 CPU 在一个时刻不可能同时执行两道程序，即不具备并行计算（Parallel Computing）的能力。现实中，使用一台计算机，可以同时运行几道甚至几十道程序，例如同时听歌、玩游戏、处理 Word 文字、浏览新闻等。根据上述说明，单核 CPU 是没有办法同时执行多道程序的，但是在感觉上，用户的确同时使用计算机做多件事情。这是如何实现的呢？

人类对任务同时执行的理解是秒级的，而计算指令的执行是纳秒级的，只要进程调度造成程序执行的响应满足用户的时间要求即可体现用户级的同时性。现代操作系统在进程调度时采用的是基于时间片轮转的分时调度策略，即把 CPU 执行指令的过程按照时间片轮询的方式对多个任务进行交替执行，如图 3-4 所示。

一般而言，现代操作系统以一定的时间为单位（例如 20ms），轮流使用 CPU 执行不同的

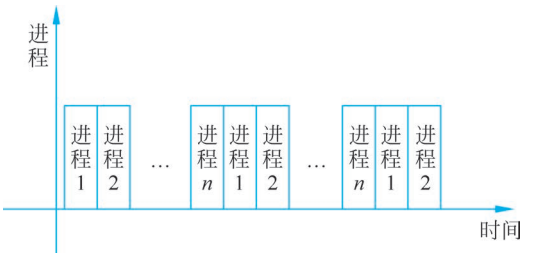

图 3-4 时间片轮转的分时调度策略

进程，例如，在图 3-4 中，如果时间片轮询的单位为 20ms，则进程 1、进程 2，一直到进程 n 以 20ms 为轮询单位，交替执行，即第 1 个 20ms 给进程 1 执行指令，第 2 个 20ms 给进程 2 执行指令，第 n 个 20ms 给进程 n 执行指令；第 $n+1$ 个 20ms 给进程 1 执行指令，第 $n+2$ 个 20ms 给进程 2 执行指令，第 $2n$ 个 20ms 给进程 n 执行指令；以此类推。接下来再来考虑一个实际案例，例如，考虑两个程序：音乐播放程序、Word 文字处理程序。音乐播放过程可以被分解为音乐数据传输、数据缓存、音乐播放 3 个阶段，其中音乐数据传输指 CPU 把一段音乐数据(假如是 1min 音乐数据)传输到声卡(假设需要 1ms)，数据缓存即声卡把 CPU 传输过来的数据缓存在声卡内部的缓存器中(假设需要 1ms)，音乐播放指声卡播放电路把缓存器中的音乐数据实时连续播放(1min)。Word 文字处理程序可以设想为一个等待用户输入字符的循环程序，可以被分解为等待接收用户字符输入(从键盘缓冲区读取 1 个字符，假设为 1ms)、保存输入字符数据(假设为 1ms)、将字符输出到显示器(假设为 1ms)。上述两个程序真正需要使用 CPU 的只有音乐播放程序中的音乐数据传输、Word 文字处理中的所有过程。

根据上述策略来分析这两个程序的并发执行过程，以及它如何做到用户体验的同时性：假定进程调度器的时间片轮转周期为 20ms，则其在第 1 个时间片花费了 1ms 就把 1min 的音乐数据传送给声卡(这时声卡会连续播放音乐，其可以在 1min 以内不需要新的音乐数据)，然后选择执行 Word 程序；Word 程序一旦感知键盘敲击，CPU 迅速把键盘扫描码数据读出、保存并显示到显示器，总共花费 3ms；由于敲击键盘的速度至少是以秒为间隔的，那么只要没有敲击键盘动作，CPU 完全可以在 Word 程序时间片用完时去处理音乐程序的数据传送；一般而言，只要进程不太多，CPU 便有足够的时间在音乐播放器把当前缓冲区的音乐数据播放完毕之前，把新的音乐数据填充过来，以保证人们感觉上的连续播放的认知。

3. 进程调度状态

现代操作系统，进程调度器经常会将一个进程从一种状态转换到另外一种状态，一般而言进程有 3 种典型状态：就绪、运行和等待状态，如图 3-5 所示。

(1) 就绪状态：指进程具备所有执行条件，只是在时间上还没有轮到该进程。

(2) 运行状态：指该进程是操作系统正在运行的进程。

(3) 等待状态：指一个进程必须等待某个特定事件发生才能继续执行的状态。

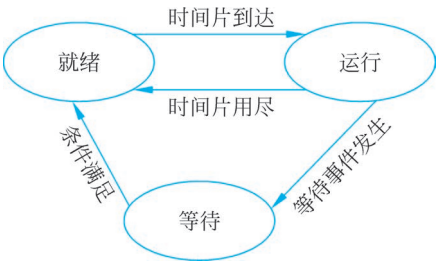

图 3-5　进程调度的 3 种状态转换关系

当进程满足除了时间片以外的所有运行条件时，它会被添加到进程管理的就绪队列中等待执行，即进入就绪状态。一旦时间片轮转到该进程，进程调度器就会将其从就绪队列中选取出来并执行，同时将其状态设置为运行状态。在运行的过程中，如果该进程需要等待某个外部事件，例如键盘输入，但此事件尚未发生，则该进程将被放入等待队列，并将其状态设置为等待状态。一旦所等待的事件发生，中断程序就会触发进程调度程序将该进程从等待队列中移出并放入就绪队列（可能还有其他进程也在等待执行），并将其状态重新设置为就绪状态，等待下一次的时间片轮转。

3.3.2　内存管理

内存管理作为现代操作系统的核心功能之一，负责高效地管理程序的内存分配、使用及释放。基于冯·诺依曼体系计算机的程序执行理念，即“存储程序，顺序执行”，程序在执行前需将其指令和数据加载至内存中。根据内存中加载的程序数量，操作系统的内存管理可被划分为单道程序和多道程序管理。当物理内存不足以满足需求时会采用虚拟内存技术来扩展内存空间，确保程序的顺畅运行。

1. 单道程序的内存管理

单道程序的内存管理出现在比较早的操作系统中，例如 MS-DOS。在单道程序中，内存除了装载操作系统之外，只支持装载一道程序。当这道程序被执行完毕后，它将被全部移出内存，继续装载下一道程序并运行。单道程序的内存模型如图 3-6 所示。单道程序在执行时需要注意以下几点。

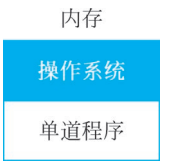

图 3-6　单道程序的内存模型

（1）如果当前需要装载和运行的程序大小超过了可用的内存，则装载失败，程序无法被执行。

（2）当一个程序正在运行时，其他程序无法运行。不幸的是，假如这道程序以 I/O 操作为主，绝大部分时间是处于等待外部设备的输入和输出，真正使用 CPU 的时间很少，也就是说 CPU 大部分时间处于空闲，CPU 也没有办法为其他程序提供服务，所以，在这种情况下，CPU 和内存的使用效率非常低。为了缓解上述问题，出现了多道程序。

2. 多道程序的内存管理

多道程序是指操作系统支持把多道程序装载到内存当中，在进程调度器的控制下并发

图 3-7　多道程序的内存模型

执行多道程序，多道程序的内存模型如图 3-7 所示。

多道程序的内存管理，从 20 世纪 60 年代被提出来以后，经过了多年的改进，出现了分区调度、分页调度、请求分页调度、请求分段调度、请求分页和分段调度等多种策略。简单来讲，可以被划分为两类：非交换式多道程序和交换式多道程序。所谓非交换式多道程序是指程序在执行前被装载到内存，在执行过程中一直常驻内存，不会被交换到外部存储器。这种模式其实就是单道程序的简单扩展，只是支持多道程序的交替执行罢了。单道程序存在的装载失败问题一样会发生，而交换式多道程序则把程序分割成更小单位的“段”和“页”，根据当前内存的实际空闲情况，一次装载程序少量的段或者页进行执行，一旦所需要访问的指令或数据不在内存中，就会发生缺段或者缺页异常，引发换段和换页操作，即把内存中的段或页和外部存储器中的段或页的数据进行交换。

3. 虚拟内存

在交换式多道程序的内存管理中，程序的一部分内容驻留在内存中，另一部分则放置在外部存储器(例如硬盘、SSD 等)。假如实际的物理内存是 1000MB，运行 20 道程序，每道程序大小为 200MB，总共需要 4000MB 的内存空间。在交换式多道程序模式下这 20 道程序在段或页交换的机制下，可以顺利执行，实际上相当于系统只有 1000MB 的物理内存，而另外 3000MB 的内存为虚拟内存。当前，绝大多数的主流操作系统，例如 Windows、Linux 等支持虚拟内存。Windows 11 操作系统下虚拟内存的设置对话框如图 3-8 所示。特别需要注意的是：当程序在执行过程中发生了请求换段或者换页操作时，需要把硬盘上的段或者页换入内存，此时执行速度会大幅下降，因为硬盘的读写速度远远小于内存(差几个数量级)，因此，为了提升执行效率，一般的举措是增大物理内存配置或者提升外部存储器速度，例如把普通硬盘换成高速的 SSD 固态硬盘。

3.3.3　文件管理

操作系统的另外一大职能是实现对数据的有效持久化存储。计算机中的内存数据，一旦断电，所有数据就会消失，无法实现数据的持久化。普通硬盘、SSD 固态硬盘、光盘等媒介是常见的持久化存储介质。为有效地对这些数据进行组织和存储，现代操作系统通过文件管理的核心组件来实现。

在 UNIX 操作系统时代，就提出了抽象的文件和文件系统的概念。文件是指具有符号名(文件名)的一组相关元素的有序序列，是一段程序或数据的集合，例如以 doc、ppt、exe、c、java 作为扩展名命名的文件。文件系统是操作系统统一管理信息资源的软件组件，管理文件的存储、检索、更新，提供安全可靠的共享和保护手段，并且方便用户使用。文件系统包含文件管理程序(文件与目录的集合)和所管理的全部文件，是用户与外存的接口，系统软件为用户提供统一方法，访问存储在物理介质上的信息。Windows 系统下的 FAT32、NTFS 和 Linux 下的 ext3、ext4 等都是文件系统的代表。

图 3-8　虚拟内存设置

文件管理的一般功能如下。

(1) 控制文件的读写访问权限：UNIX、Windows、Linux 等操作系统都可以针对不同的用户和进程设置相应文件的读、写、执行等权限，允许或者禁止这些用户、进程对文件数据的相应访问操作。

(2) 管理文件的创建、修改和删除：给操作用户提供创建、修改和删除文件的能力。

(3) 修改文件名称：给操作用户提供修改文件名称的能力。

(4) 提供一系列系统调用给上层应用程序使用。

现代操作系统为上层应用程序提供了系列系统调用以支持丰富的文件操作，常见的有 open(打开文件)、read(读文件)、write(写文件)、close(关闭文件)等。对一般程序而言，使用系统调用即可满足对文件的所有操作。

3.3.4　设备管理

计算机的外围设备种类繁多，以操作系统的观点来看，设备使用特性、数据传输速率、数据的传输单位、设备共享属性等都是重要的性能指标。可以按照不同角度对它们进行分类。

(1) 按设备的使用特性分类，可把设备分为两类。第一类是存储设备，也称为外存、后备存储器、辅助存储器，是计算机系统用于存储信息的主要设备。该设备速度慢、容量大、价

格便宜。第二类是输入/输出设备,可分为输入设备、输出设备和交互式设备,如键盘、鼠标、扫描仪、打印机、显示器等。

(2) 按传输速率分类,可将 I/O 设备分为 3 类。第一类是低速设备,其传输速率仅为每秒几字节至几百字节的设备,如键盘、鼠标等。第二类是中速设备,其传输速率为每秒数千字节至十万字节的设备,如行式打印机、激光打印机等。第三类是高速设备,其传输速率在数十兆千字节至数百吉字节的设备,如磁带机、磁盘机、光盘机等。

(3) 按信息交换的单位分类,可把 I/O 设备分为两类。第一类为块设备,这类设备用于存储信息,信息以数据块为单位。如磁盘,每个盘块 512B～4KB,传输速率较高,通常每秒几兆位。块设备的特征是可寻址,即对它可随机地读写任一块,磁盘设备的 I/O 常采用 DMA 方式。第二类是字符设备,用于数据的输入和输出,其基本单位是字符,属于无结构类型,如打印机等,其传输速率较低,通常为几字节至数千字节。字符设备的特征是不可寻址,即输入和输出时不能指定数据的输入源地址及输出的目标地址。此外,字符设备常采用中断驱动方式。

(4) 按设备的共享属性分类,可以分为 3 类。第一类为独占设备,在一段时间内只允许一个用户(进程)访问的设备,即临界资源。第二类为共享设备,在一段时间内允许多个进程同时访问的设备。当然,每时刻仍然只允许一个进程访问,如磁盘(可寻址和可随机访问)。第三类为虚拟设备,通过虚拟技术将一台设备变换为若干台逻辑设备,供若干用户(进程)同时使用。

操作系统的设备管理功能主要体现在设备处理程序(又称为驱动程序)的机制设计上,它是 I/O 系统的高层与设备控制器之间的通信程序,其主要任务是接收上层软件发来的抽象 I/O 要求,如 read 或 write 命令,把它转换为具体要求后,发送给设备控制器,启动设备去执行;反之,它将由设备控制器发来的信号传送给上层软件。由于驱动程序与硬件密切相关,故通常应为每类设备配置一种驱动程序。设备驱动程序的主要功能如下:

(1) 接收与设备无关的软件发来的命令和参数(例如文件的 read 或者 write 命令),并将命令中的抽象要求转换为与设备相关的低层操作序列。

(2) 检查用户 I/O 请求的合法性,了解 I/O 设备的工作状态,传递与 I/O 设备操作有关的参数,设置设备的工作方式。

(3) 发出 I/O 命令,如果设备空闲,就立即启动 I/O 设备,完成指定的 I/O 操作;如果设备忙碌,则将请求者的请求块挂在设备队列上等待。

(4) 及时响应由设备控制器发来的中断请求,并根据其中断类型,调用相应的中断处理程序进行处理。

设备驱动程序的处理过程如下:

(1) 将请求抽象要求转换为具体要求。

(2) 检查 I/O 请求的合法性。

(3) 读出和检查设备的状态。

(4) 传送必要的参数(磁盘在读写前,要将参数传递至控制器的寄存器中)。

(5) 启动 I/O 设备。

3.3.5　用户界面

根据定义，操作系统除了应高效利用硬件资源，还有一项重要功能，即方便用户操作。每个操作系统都通过用户界面（User Interface）接收用户的输入并解释和执行这些请求。用户界面一般分为两种：一种是命令行界面（Command Line Interface，CLI），例如 DOS 命令行界面；另一种是图形用户界面（Graphic User Interface，GUI），例如 Windows 的桌面图形化接口等。DOS 的命令行界面如图 3-9 所示。

图 3-9　DOS 的命令行界面

Windows 11 图形用户界面如图 3-10 所示。

图 3-10　Windows 11 图形用户界面

3.4 硬件中断与异常

操作系统管理的硬件资源主要包括各种各样的 I/O 设备、计算资源和存储资源。键盘、显示器、U 盘等常用的设备均为 I/O 设备,操作系统需要统一对这些硬件进行管理。计算资源主要指 CPU;存储资源通常包括内存和外存,内存是 CPU 直接通过系统总线来访问的,而外存是通过标准的 I/O 来管理的,CPU 和内存都是计算机内部很多程序所共享的资源。

操作系统有条不紊地对这 3 种中断进行处理,以管理系统资源。本节将首先介绍操作系统对 I/O 设备的管理,然后分别介绍 CPU 与内存这两类共享资源。

3.4.1 对 I/O 设备的管理

除了计算资源和内存资源外,操作系统对其他资源都通过 I/O 来管理,如键盘、鼠标等输入设备,显示器、打印机等输出设备,以及磁盘、闪存(U 盘)等外存设备。随着计算机相关领域的发展,I/O 设备的种类繁多。诸如显卡、磁盘、网卡、U 盘、智能手机等都是外接 I/O 设备,并且持续不断地有新的 I/O 设备出现。面对层出不穷的 I/O 设备,操作系统如何识别它们呢?事实上,操作系统定义了一个框架来容纳各种各样的 I/O 设备。除了一些专用操作系统以外,现代通用操作系统(如 Windows、Linux 等)都会提供一个 I/O 模型,允许设备厂商按照此模型编写设备驱动程序(Device Driver),并加载到操作系统中。I/O 模型通常具有广泛的适用性,能够支持各种类型的设备,包括对硬件设备的控制能力,以及对数据传输的支持。简单来讲,I/O 模型对计算机下层硬件设备提供了控制的能力,同时对上层应用程序访问硬件提供了一个标准接口。

CPU 通常使用轮询和硬件中断两种方式检测设备的工作状态。

所谓轮询指的是 CPU 通过不停地查询设备的状态寄存器来获知其工作状态,如图 3-11 所示,CPU 向设备 1 发出询问,如果设备 1 有 I/O 请求,则将 I/O 请求信息反馈给 CPU,否则询问设备 2。这种轮询的方式在实现中存在 3 个弊端:①检测中断速度慢。每次需要依次询问各台设备,以获知发出中断的设备。②可能存在设备一直处于“饥饿”状态。某一设备有中断请求却一直得不到 CPU 的响应,例如,在图 3-11 中,用户正在编辑文档,设备 1 一直处于忙碌状态,CPU 依照轮询策略,每次都优先满足设备 1 的请求,那么打印机的中断就得不到响应。③系统处理中断事务不灵活,如图 3-11 中,各台设备的优先级是固定的。设备 1 的优先级大于设备 2,也就是说设备 1 与设备 2 同时产生中断时,设备 2 不会被响应,因此这种中断检测方式不适应现代操作系统。

相比于轮询方式,另外一种更有效的做法是使用硬件中断类型码来分辨是哪个硬件发起中断。当某台设备状态发生变化时,该设备能主动地通知 CPU 并反映其当前的状态,从而让操作系统采取相应的措施。在硬件中断发生时,每台设备都有一个中断类型码(Interrupt Type Code),如图 3-12 所示,作为设备的标识符,使操作系统能区分来自不同设

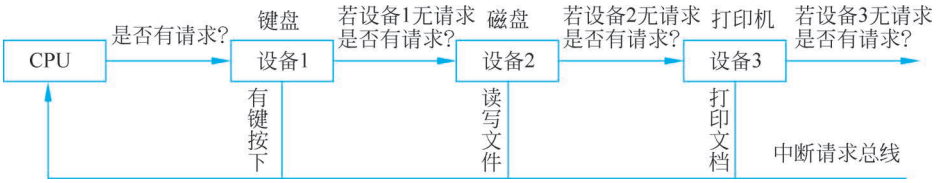

图 3-11　轮询响应流程

备的中断请求，以提供不同的服务。操作系统利用一个重要的表格就可以完成中断类型码与需执行的服务程序的连接。这个重要的表格就是中断向量表(Interrupt Vector Table)。中断类型码是中断向量表的索引，所以 n 种中断类型码就代表在中断向量表有 n 行。每行存储指向相关服务程序的起始位置，这个服务程序叫作中断服务程序(Interrupt Service Routine)，每个中断类型码都有一个自己的中断服务程序。当 CPU 收到了中断类型码时，例如当前收到的中断类型码是 9，就会自动到中断向量表的第 9 行找到它的中断服务程序的起始位置，然后跳到此程序去执行。

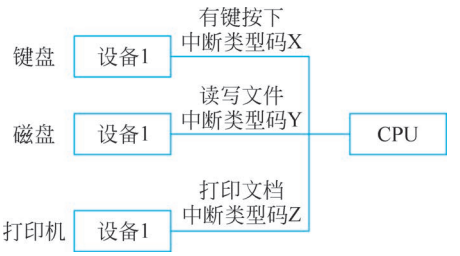

图 3-12　硬件中断流程

以键盘输入产生的中断为例，当用户在键盘按下一个键时会产生一个键盘扫描码，此扫描码被送入主板上的相关接口芯片的寄存器中。输入到达后，键盘将会发出中断类型码为 9 的中断信息。CPU 检测到中断信息后唤醒操作系统，并查找中断向量表的 9 号向量，进而转到中断服务程序入口(函数调用)，执行中断服务程序。这个过程如图 3-13 所示。中断向量表和相关的中断服务程序是极其重要的，需要特别保护起来，一般用户是不可以改变它们的，这些都是放在操作系统的内核(Kernel)中保护起来的。

3.4.2　对 CPU 的管理

计算机的多核时代已经到来。为了满足系统的性能要求，提高任务处理的效率，现在主流的计算机通常配置有一个或多个 CPU，每个 CPU 中又有多个核(Core)，然而核的数量远远小于需要执行的程序的数量。一个计算机系统一般有几十个程序(或叫任务，Task)在等待执行，并且都抢着占用 CPU，所以，操作系统需要合理地安排和调度任务，使计算资源得以充分利用。在此假设系统只有一个 CPU 核。

在现代操作系统中，任务的数量远超过 Core 的数量，为了使多个任务可以较公平地在系统中运行，避免出现死循环而导致整个系统崩溃的情况，就需要一种有效的机制唤醒操作系统，然后让操作系统在不同的任务间进行切换。注意操作系统的运行是需要 CPU 的，而 CPU 正在被进程的程序给占据着，操作系统怎样能抢到 CPU 呢？

这就需要 CPU 之外的硬件来将中断发给 CPU。计算机通过 Timer(硬件)将中断发给 CPU，从而让其从当前运行的进程中释放出来。操作系统为每个任务分配一个定长的时间

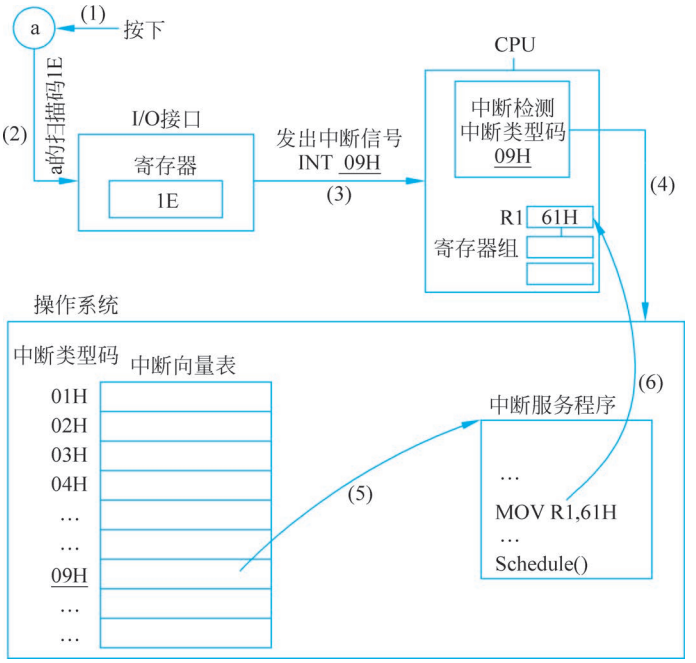

图 3-13 中断响应流程

片,在此时间内,CPU 由获得该时间片的任务所占据,然而每当当前时间片被用完时 Timer 硬件便会自动将中断发给 CPU,经过 3.4.1 节所讲述的硬件中断过程,CPU 会跳到 Timer 的中断服务程序去执行,在此中断服务程序里会调用操作系统的一个叫作调度器(Scheduler)的核心程序。调度器根据当前的任务执行情况,将 CPU 合理地分配给任务使用。当前可能有多个就绪(Ready to Run)任务在等待 CPU 的执行。操作系统维护了一个就绪任务队列(Ready to Run Queue),用于存放这些就绪的任务。这个队列中的所有任务都在等待 CPU 资源。选择哪一个任务去使用 CPU 是调度器的工作,如图 3-14 所示,在 Timer 发出中断后,现在执行的任务就会被放到就绪队列中,调度器会从就绪队列中选择一个任务来使用 CPU。

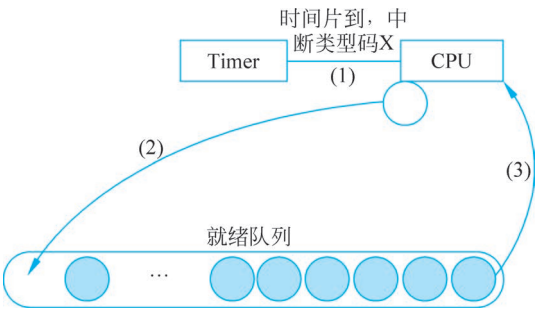

图 3-14 Timer 中断

每个任务在执行过程中，一旦时间片消耗完，该任务就有可能会被调度器切换出 CPU，但是当该任务被再次执行时，要如何恢复运行呢？为此需要解决如下问题：①程序从哪里开始执行？②假设程序能恢复从切换出时的语句开始执行，之前运行的结果怎么恢复？如果解决不了这两个问题，则造成的后果是：程序一旦换出 CPU，再次被换回 CPU 中准备执行时，就不能恢复换出时的状态，这样系统根本无法继续执行任务。为了解决这些问题，操作系统为每个执行中的程序（任务）创建了一个进程（Process），用以保存每个任务执行时的所有环境信息。进程中保存了程序计数器（Program Counter，PC）、所有的寄存器和程序运行时所涉及的变量、堆、栈等。进程保存程序被切换出 CPU 时所执行到的步骤及在运行的过程中产生的数据变量和当时的堆栈等一切信息。每当进程切换出 CPU 时，这些信息随着进程一起保存到内存，等到该进程重新调入 CPU 时，能够根据保存的信息恢复到换出时的运行环境，程序得以继续执行。这个一出一进的过程是比较浪费时间和资源的。应尽量减少它的次数，所以调度的好坏就关乎整个系统的性能。

3.4.3　对内存的管理

任务执行时需要内存，内存和 CPU 一样都是珍贵的资源。操作系统管理内存，使多个任务能共享内存资源。在一个任务结束执行时，操作系统会对所分配的内存资源进行回收，为其他任务所使用。由于内存资源有限，所以操作系统还需要对任务存储在内存中的数据进行换入换出的管理，以应对内存不足的情形。换入的数据从硬盘加载进内存，而换出的数据将从内存保存到硬盘。变量被换出后，就不在内存中了。假如这个程序需要再次使用已被换出的变量，CPU 在读取数据时，发觉这个变量不在内存，就会产生异常。此时，操作系统就要被唤醒，以便处理这个异常中断。操作系统会把此变量所在的一块数据（叫作“页”，Page）从硬盘载入内存中。

试想在执行某任务的某条语句时，一个变量还没有加载进内存，如果这时对该变量进行访问，则会产生一个异常，抛出异常中断（这类异常叫作页错误，Page Fault），操作系统就会被唤醒，跳到页错误处理程序（Page Fault Handler），将该变量所在的部分（一般是一页数据，4K）加载进内存。因为这个页错误处理程序牵扯到硬盘的 I/O 操作，所以整个过程需要花时间。另外，假如内存已经没有空间来存放这一页，操作系统将采用不同的页替换算法（Page Replacement Algorithms）来决定将内存中的哪页换出以腾出空间。最常用的页替换算法是 LRU（Least Recently Used）算法。简单来讲，LRU 算法就是将内存中最长时间未使用的那一页换出去。

3.5　软件中断

设计出任何的新硬件如果要连接到计算机或手机，则必须提供驱动程序（Device Driver），这些驱动程序都要通过安全检测，假如有病毒是要负法律责任的。用户在使用新硬件前都必须先安装驱动程序，这些驱动程序就变成了操作系统的一部分。所有的程序要

使用这个硬件时都必须经过操作系统来实现,这样可以保证硬件不被有意或无意地破坏,也可以经由操作系统来保证特权(Privilege)的维护,例如部分用户只有权力执行读操作,而不能执行写操作等。在计算机系统使用时,绝对要禁止用户程序跳过操作系统而直接使用硬件,但是要如何禁止呢?

3.5.1 内核态与用户态

一个用户程序可以直接读写某个硬件吗?例如,小明和小华在一台计算机上分别有各自的用户账号,小华将期末考试试卷存放在自己的文件夹下,并且不允许其他用户访问,听起来好像很安全,然而如果小明能直接读写硬盘,并且知道试卷文件在硬盘的物理位置,他写了一段汇编语言程序,跳过操作系统,直接去读取硬盘所存的二进制数据,操作系统所保证的安全性就荡然无存了,所以,绝对不允许用户程序直接访问硬件设备。怎样防范用户小明的程序直接读写硬盘数据?操作系统是软件,在这个问题上软件是没有办法防护的,必须CPU 提供硬件支持。

基本思想是:CPU 将指令集分为需要特权的指令(Privileged)和一般的指令(NonPrevileged),而所有的 I/O 指令都属于需要特权的指令,一般用户不能执行这类 Privileged 指令,必须是系统内核才能执行这类 Privileged 指令,所以小明是没有办法直接读写硬盘的。

然而 CPU 是怎么知道现在执行的程序是操作系统内核还是普通用户进程呢?在程序运行时,CPU 会显示出现在的运行状态:内核态(Kernel Mode)还是用户态(User Mode)CPU 有个特殊的寄存器叫作状态寄存器(Status Register),其中显示当前的 CPU 处于内核态还是用户态。假如 CPU 处于用户态,那么任何的 Privileged 指令 CPU 都不可以执行,一旦执行,CPU 就会发生异常错误。当用户程序直接执行 Privileged 指令时,CPU 会检测当前状态是否为内核态,假如当前状态为用户态,CPU 就不会执行该指令,发生异常错误。这些检测过程不是由软件完成的,而是 CPU 硬件执行每条指令时自动检测的。

至于 CPU 如何从用户态转换成内核态,这是现代操作系统的一项重要技术。那就是使用中断方式,只有这种方式 CPU 才会进入内核态。不管是哪种中断发生,CPU 都会自动进入内核态模式。由于软件中断更需要注意安全问题,所以在此特别讨论软件中断。当一个用户程序要得到操作系统的服务时,它执行软件中断。在最底层通过执行一个特殊的叫作 int 的指令来实现(每种 CPU 都有类似的指令,只是名字不一样,Intel x86 指令集叫作 int 指令,ARM 指令集叫作 swi 指令,在一些操作系统教科书中叫作 trap 指令)。用户程序通过执行该指令获取操作系统提供的服务。重点是在执行这条指令时,CPU 会自动地将状态置为内核态。操作系统会保存一个中断向量表,每行存储着中断服务程序的起始位置。int 指令有一个参数 #n,即 int #n。当 CPU 执行到 int #n 指令时会自动将模式转换为内核态,读取中断向量表的第 n 条记录,跳到相对应的中断服务程序去执行。至于要操作系统执行哪一项具体服务,一般是用暂存器来传递的。

现在以 Linux 的软件中断为例。首先将想要执行的系统调用编号放入暂存器 EAX

中，例如 read 的编号是 3，write 的编号是 4，open 的编号是 5，close 的编号是 6 等，然后执行软件中断 int 80h(80h 是十六进制)指令就行了。MS-DOS、Windows 等系统也用相似的方式，只是 int ＃n 中的＃n 用不同的编号，例如 MS-DOS 用 int 21h 为软件中断，写底层驱动程序时，需要知道操作系统的中断编号，一般的软件设计者不需要知道这些细节，操作系统都有较方便的接口来给软件调用。很多高阶语言(例如 Python、Java 等)再包装操作系统的接口，提供多种更高阶、更方便的函数接口供软件设计者来使用。

使用 int 指令后，用户程序就可以获得操作系统提供的服务了，状态也自动变成内核态，从而可以执行需要特权的指令。注意此时的程序是操作系统，而结束中断服务程序后，当调度器选择一个用户进程执行时，CPU 会将状态转换为用户态。注意此时的程序是用户的程序。用这种方式就是为了保证没有用户能跳过操作系统而直接使用 I/O。

所以，小明想要跳过操作系统而直接读取硬盘存储的二进制数据是不可行的。综上所述，如图 3-15 所示，经过内核态和用户态方式的保护后，操作系统运行于内核态，除了操作系统以外的任何软件都运行于用户态，也就是说，应用软件处于用户态，但是，应用软件有时也会使用硬件设备，这时就需要唤醒操作系统来为应用软件做事了，而唤醒操作系统的方式就是软件中断。为了获得操作系统所提供的服务，用户程序需要进行系统调用(System Call)。在系统调用时就一定会用到 int 指令，在 int 指令执行中系统将自动进入内核态，然后执行中断服务程序。当服务结束时，控制权经由调度器程序转交给用户程序，返回用户模式。

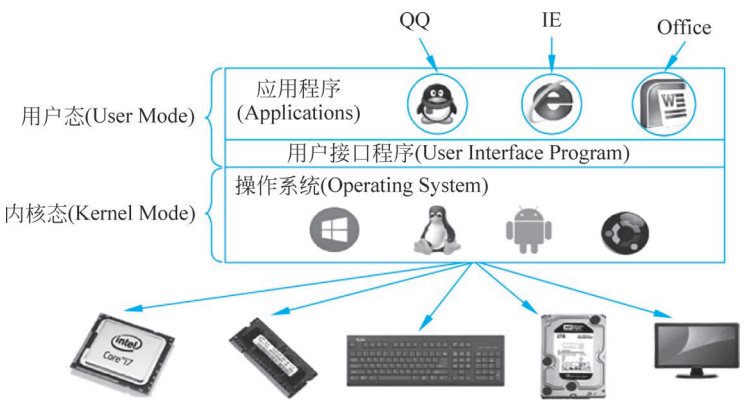

图 3-15　用户态与内核态

3.5.2　系统调用

操作系统中设置了一组用于实现系统功能的子程序，称为系统调用函数。系统调用函数和普通函数调用非常相似。不同之处在于系统调用函数的操作一定运行于内核态，而普通的函数调用由函数库或用户自己提供，运行于用户态。当程序需要使用操作系统的服务来完成某项功能时，就需要使用系统调用函数。CPU 运行到系统在调用函数时，将会执行 int ＃n 指令，CPU 会产生软件中断，唤醒操作系统，接下来再运行操作系统提供的服务。

注意,int ＃n 指令的目的是唤醒操作系统来提供服务,转换成内核态是隐藏在 int 指令里自动做的事情,所以,用户的程序只能在调用系统后由系统调用函数时,CPU 才会转换成内核态,以正确地执行操作系统里的内核程序。系统调用结束后将返回用户模式,CPU 寄存器的状态位改为 User Mode,继续执行用户程序,也就是说用户程序是不可能在内核态执行的。

3.6 文件系统

在现代计算机系统中,需要用到大量的程序和数据。内存的速度虽然远远快于外存,但其容量有限,并且不能长期保存程序和数据信息,因此,系统将这些程序和数据组织成文件,存储在外存设备(硬盘、光盘、U 盘等)中,例如,在本书之前章节中编写的 Python 代码都会存储到一个文件中。平时生活中听的音乐、拍的照片,也都会以二进制信息的形式存储于一个文件中。

存储在外存设备的文件,使用时需要先调入内存。如果由用户直接管理这些文件,则不仅要求用户熟悉外存特性,了解各个需要使用的文件属性,还要知道这些文件在外存中存储的位置。显然,这些繁杂的工作不能交付给用户完成。于是,对文件的管理顺理成章地交付给了操作系统。操作系统中有一个文件系统,专门负责管理外存上的文件。这不仅方便了用户对文件的操作,同时保证了系统中文件的安全性。

本节将介绍文件的基本概念,文件系统最常用的目录树结构,以及 Python 中如何对文件进行简单读写操作等内容。

3.6.1 文件的基本概念

1. 文件的命名

各个操作系统的文件命名规则有所不同,文件名的格式和长度因系统而异。常见的文件名由两部分构成,格式为文件名.扩展名。文件名与扩展名都是由字母或数字组成的字符串,通常文件的文件名可以由用户自定义,而文件的扩展名则代表不同的文件类型,例如在 Windows 系统下,可执行文件为“文件名.exe”,Python 文件多以“.py”结尾,常见的音视频文件如“文件名.mp3”“文件名.mp4”“文件名.avi”等。

2. 文件的类型

在 Linux 操作系统中将显示器、打印机等外设也看作一个文件,而系统根据文件所具有的不同类型,能够区分普通文件、外设文件及各个不同种类的外设。具体来讲,Linux 中支持如下几种文件类型。①普通文件:指存储于外存设备上通常意义上的文件,包括用户建立的源程序(Python、C、C++)文件、数据(照片、音视频等)文件、库(提供系统调用)文件、可执行程序文件等。②目录文件:统一管理普通文件等(类似于 Windows 文件夹)。一个目录文件既可以包含多个普通文件,也可以包含目录文件,它为文件系统形成了一个逻辑上的结构。这部分内容将在 3.6.2 节中进行介绍。③块设备文件:用于管理磁盘、光盘等块设

备，并提供相应的 I/O 操作。④字符设备文件：用于管理打印机等字符设备，并提供相应的 I/O 操作。除了以上类型的文件之外，Linux 文件类型还包括套接字文件（用于网络通信）、命名管道文件（用于进程间通信）。

3.6.2　目录树结构

回顾 3.6.1 节的问题，如何实现多个文件具有相同的文件名？目录树结构解决了这个问题。在文件系统目录树中，最顶层的节点为根目录，从根目录向下，每个有分支的节点都是一个子目录，而树叶节点（没有分支）就是一个文件，如图 3-16 所示，“/”所示节点为根节点，该节点为一个目录文件，其下有 dev、bin、usr 共 3 个目录文件，在 usr 目录下，又有目录 Zhen、Ming 及其他文件。这样，即便有同名的文件，但两个文件所在路径是不同的，也就可以区分这两个文件了。

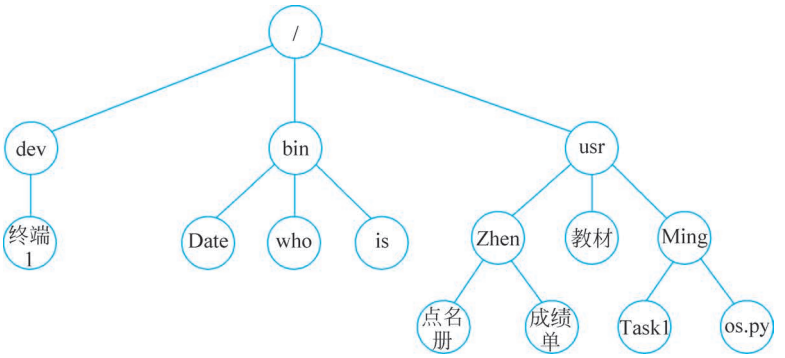

图 3-16　目录树

目录查找是文件系统的一项很重要的工作，每当需要使用系统调用 open 打开文件时，必须给出路径名及文件名，例如，如果要打开名为 os.py 的 Python 文件，则需要使用 fd = open("/usr/Ming/os.py") 打开，有了文件描述符 fd 后，就可以对该文件进行一系列操作了。

路径通常可以分为两类：绝对路径与相对路径。绝对路径是指从根向下直到具体文件的完整路径，如上述例子/usr/Ming/os.py 就是一个绝对路径，但是，随着文件系统层次的增加，使用绝对路径变得十分烦琐。更糟的情况是，在某文件系统下写的程序要到另一个文件系统下去运行，如果使用绝对路径，则要求两个文件系统有相同的目录树结构，这是不灵活的。为了解决这一系列问题，在程序中除了使用绝对路径外，还可以使用相对路径。相对路径就是指目标文件的位置与当前所在目录的路径关系。相对路径中包含两个符号“.”和“..”，其中“.”表示当前目录，而“..”表示父节点目录，例如，现在有目录文件/usr/Ming/os.py，可以用“./”表示/usr/Ming 目录，而用“../”表示/usr 目录。

一般读写一个文件的顺序是：第 1 步，打开（open）这个文件，参数包含路径；第 2 步，用一个循环来读/写（read/write）文件里的数据。为什么要先执行 open 指令，而不是在每次读写操作时去寻找路径呢？执行 open 指令的目的是什么？

仔细研究 open 函数,就会发现 open 是个很花时间的操作,尤其是当路径要经过多重目录时。每层目录都要执行硬盘 I/O 操作,寻找下一个子目录,一层层地找下去,由于 open 操作包含了这么多 I/O 操作,所以需要花时间。为节约时间,希望花时间的操作在循环之前只执行一次,而不要在每次循环中都执行一次,所以在循环前执行 open 操作是有利的,而 open 操作的目的是:第一,最终获得此文件数据在硬盘中的位置;第二,在路径遍历过程中,检查用户是否有权限来执行对此文件的操作;第三,当有多个进程要读写相同的文件时,有时需要利用 open 操作在读写前锁住文件,以取得文件的一致性,所以 open 操作具有多样性的功能。Python 语言给编程者提供了一系列方便的文件读写操作函数,而在这些文件操作函数的具体实现中会调用 3.5.1 节所介绍的系统调用(软件中断)来要求操作系统提供服务,也就是执行 int 指令。这些调用操作系统的细节较为复杂,一般用户不需要知道这些细节,用户只要会使用 Python 所提供的文件读写函数就可以了,所以,当要在 os.py 文件中打开 Task1 文件时,只要了解 Python 为编程者提供了哪些文件操作函数即可。

3.6.3 Python 中的文件操作

文件操作是每个操作系统、每台计算机都需要完成的事情。在此,简单分析 Python 中的文件操作:分清要操作的对象是什么,该对象提供了哪些方法,以及系统提供了哪些内置函数。Python 提供了文件对象,并内置了 open 函数来获取一个文件对象。open 函数的使用方法如 file_obiect = open(path,mode),其中,file_object 是调用 open 函数后得到的文件对象;path 是一个字符串,代表要打开文件的路径,而 mode 是打开文件的模式,常用的模式如表 3-1 所示。

表 3-1 打开文件时的常用模式

文件模式	解释
r	以只读模式打开:只允许对文件进行读操作,不允许写操作(默认方式)
w	以写模式打开:当文件不为空时清空文件,当文件不存在时新建文件
a	追加模式:如果文件存在,则在写入时将内容添加到末尾
r+	以读写模式打开:打开的文件既可读又可写

回到 3.6.2 节的例子,在 os.py 文件中要打开 Task1 文件进行读写,需要使用 r+模式,实现如下:f=open('./Task1','r+')。简单一条语句便实现了打开文件操作,之后对该文件的操作只需对新得到的文件对象 f 使用文件对象提供的方法。

3.7 本章小结

本章详细地介绍了操作系统的基础知识。首先,从计算机的启动开始,讲解计算机在启动过程中完成的各项工作,从而引出操作系统,其次,对操作系统的定义、发展过程、主要功能与分类进行了简单介绍。重点对几种常见的操作系统及其体系结构进行了讲解。最后,结合操作系统的内核组成对进程管理、内存管理、文件管理、设备管理及用户界面进行详细

讲解。进程与中断这两个概念比较重要，需要重点关注。最后，结合 Python 中对文件的操作，对操作系统的文件系统进行了阐述。操作系统是计算机类专业中较为重要的知识。不管是在专业知识的学习中，还是在以后的工作中，操作系统的使用方法、实现思想、计算思维均会伴随着读者。

3.8　习题

一、单项选择题

1. 在下列选择中，(　　)不是操作系统关心的主要问题。

A. 管理计算机裸机　　B. 设计、提供用户程序计算机硬件系统的界面

C. 管理计算机系统资源　　D. 高级程序设计语言的编译器

2. 批处理系统的主要缺点是(　　)。

A. CPU 利用率低　　B. 不能并发执行

C. 缺少交互性　　D. 以上都不是

3. 多道程序设计是指(　　)。

A. 在实时系统中并发运行多个程序

B. 在分布式系统中同一时刻运行多个程序

C. 在一台处理器上同一时刻运行多个程序

D. 在一台处理器上并发运行多个程序

4. 以下最早的 OS 是(　　)。

A. 分布式系统　　B. 实时系统

C. 分时系统　　D. 批处理系统

5. 批处理 OS 提高了计算机系统的工作效率，但(　　)。

A. 不能自动选择作业执行　　B. 无法协调资源分配

C. 不能缩短作业执行时间　　D. 在作业执行时用户不能直接干预

6. 分时 OS 追求的目标是(　　)。

A. 高吞吐量　　B. 充分利用内存

C. 快速响应　　D. 减少系统开销

7. 多道批处理系统提高了计算机系统的资源利用率，同时(　　)。

A. 减少了各作业的执行时间　　B. 增加了作业吞吐量

C. 减少了作业的吞吐量　　D. 减少了部分作业的执行时间

8. 设计实时 OS 时，(　　)不是主要追求目标。

A. 安全可靠　　B. 资源利用率

C. 及时响应　　D. 快速处理

9. 现代 OS 的两个基本特征是(　　)和资源共享。

A. 多道程序设计　　B. 中断处理

C. 程序的并发执行　　D. 实现分时与实时处理

10. OS 中采用多道程序设计技术提高了 CPU 和外部设备的(　　)。

A. 利用率　　B. 可靠性　　C. 稳定性　　D. 兼容性

11. OS 的基本类型有(　　)。

A. 批处理系统、分时系统及多任务系统

B. 实时 OS、批处理 OS 及分时 OS

C. 单用户系统、多用户系统及批处理系统

D. 实时系统、分时系统和多用户系统

12. 为了使系统中所有的用户都可以得到及时响应,该 OS 应该是(　　)。

A. 多道批处理系统　　B. 分时系统

C. 实时系统　　D. 网络系统

13. 下列叙述中正确的是(　　)。

A. OS 的不确定性是指在 OS 控制下的多个作业执行顺序和每个作业的执行时间是不确定的

B. 在分时系统中,响应时间≈时间片×用户数,因此为了改善响应时间,常用的原则是使时间片越小越好

C. 数据库管理程序需要调用系统程序,OS 程序的实现也需要数据库系统的支持

D. 用户程序通常可以直接访问系统缓冲区中的数据

14. 在(　　)OS 控制下,计算机系统能及时处理由过程控制反馈的数据并作出响应。

A. 实时　　B. 分时　　C. 分布式　　D. 单用户

15. 分时系统的响应时间是根据(　　)确定的,而实时系统的响应时间则是由控制对象所能接受的时延确定的。

A. 时间片大小　　B. 用户数目

C. 计算机运行速度　　D. 用户所能接受的等待时间

二、填空题

1. 采用多道程序设计技术能充分发挥(　　)与(　　)并行工作的能力。

2. OS 的基本功能包括(　　)。

3. 分时 OS 的主要特征是(　　)。

4. 在主机控制下进行的输入/输出操作称为(　　)操作。

5. (　　)系统不允许用户随时干预自己程序的运行。

6. 为了赋予 OS 某些特权,使 OS 更加安全可靠地工作,实际 OS 中区分程序执行的两种不同的运行状态是(　　)和(　　),(　　)态下执行的程序不能执行特权指令。

7. 批处理系统是在解决(　　)和(　　)的矛盾中发展起来的。

8. 所谓虚拟是指把一个(　　)变为若干个(　　)。

9. 在分时系统中,响应时间与(　　)有关。

10. 进程管理的基本功能是(　　)。

三、判断题

1. 操作系统控制作业运行的方式主要有批处理方式、分时方式、实时方式。（　　）
2. 操作系统中的控制程序一定具有分时处理能力。（　　）
3. 系统初启引导不属于 OS。（　　）
4. 批处理系统不允许用户随时干预自己程序的运行。（　　）
5. 操作环境不是 OS。（　　）
6. 多道批处理 OS 适合于终端作业。（　　）
7. 在多道程序设计的系统中，系统的效率与并行的道数成正比。（　　）
8. OS 本身的所有功能都与硬件相关。（　　）
9. 实时 OS 强调系统的实时性和高可靠性，其次才考虑资源的利用率。（　　）
10. 进程的等待状态是指等待占用处理机时的进程状态。（　　）

四、简答题

1. 什么是操作系统？它的主要功能是什么？
2. 简述单道操作系统与多道操作系统的区别。
3. 解释进程和线程的区别。
4. 简述中断工作过程。
5. 简述内核态与用户态的区别。
6. 什么是文件系统？它的作用是什么？