操作系统

类目：计算机类
书名：操作系统
主编：马 芳　秦 朗　蒲宏伟
出版社：湖南大学出版社
开本：大 16 开
书号：978-7-5667-3593-5
使用层次：通用
出版时间：2024 年 8 月
定价：49.80 元
印刷方式：双色
是否有资源：是

计算机类创新融合精品教材
“互联网+”教育改革新理念教材

操作系统

主编◎马 芳 秦 朗 蒲宏伟

操作系统

主编◎马 芳 秦 朗 蒲宏伟

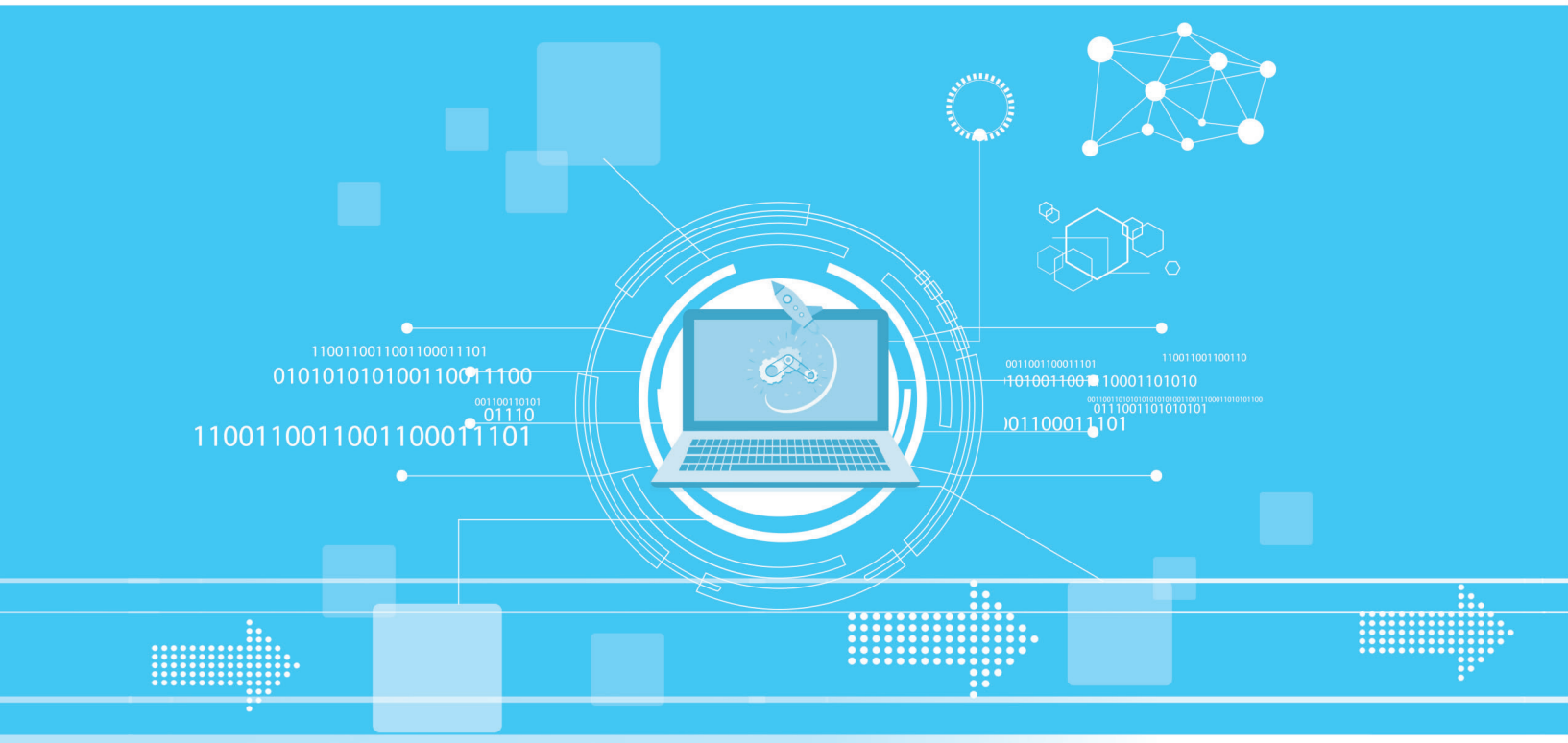
责任编辑：蔡京声 汪斯为
封面设计：旗语书装

ISBN 978-7-5667-3393-5
定价：49.80元

湖南大学出版社

湖南大学出版社

计算机类创新融合精品教材
“互联网+”教育改革新理念教材



操作系统

主　编◎马　芳　秦　朗　蒲宏伟
副主编◎高洪云　李　磊　孔　莹
张　彬

湖南大学出版社
·长　沙·

图书在版编目(CIP)数据

操作系统/马芳，秦朗，蒲宏伟主编. --长沙：湖南大学出版社，2024.8

ISBN 978-7-5667-3593-5

Ⅰ.①操… Ⅱ.①马… ②秦… ③蒲… Ⅲ.①操作系统 Ⅳ.①TP316

中国国家版本馆 CIP 数据核字（2024）第 107823 号

操 作 系 统

CAOZUO XITONG

主　　编：马　芳　秦　朗　蒲宏伟

责任编辑：蔡京声　汪斯为

印　　装：涿州汇美亿浓印刷有限公司

开　　本：889 mm×1194 mm　1/16　　**印　　张**：14　　**字　　数**：444 千字

版　　次：2024 年 8 月第 1 版　　**印　　次**：2024 年 8 月第 1 次印刷

书　　号：ISBN 978-7-5667-3593-5

定　　价：49.80 元

出 版 人：李文邦

出版发行：湖南大学出版社

社　　址：湖南·长沙·岳麓山　　**邮　　编**：410082

电　　话：0731-88822559（营销部）　88821174（编辑室）　88821006（出版部）

传　　真：0731-88822264（总编室）

网　　址：http://press.hnu.edu.cn

电子邮箱：xiaoshulianwenhua@163.com

前言

QIAN YAN

党的二十大报告对建设现代化产生体系作出部署，强调要“推动战略性新兴产业融合集群发展，构建新一代信息技术、人工智能、生物技术、新能源、新材料、高端装备、绿色环保等一批新的增长引擎。”以计算机为基础的新一代信息技术及人工智能技术成为我国建设科技强国的引路标。

在此背景下，各行业纷纷依托大数据技术、第 5 代移动通信技术（5th generation mobile communication technology，5G）、人工智能技术来快速发展高新科技，以推动传统制造业加速向数字化、网络化和智能化转变。作为支撑众多技术得以应用的平台——操作系统，逐步得到了相关部门和越来越多有识之士的关注与重视。

操作系统是计算机核心系统软件，它负责控制和管理整个计算机系统的资源并组织用户以进程为单位高效协调使用这些资源，使计算机各部件极大程度地并行运行。操作系统课程是计算机大类专业的核心课程。随着计算机技术的发展，各类嵌入式系统得到广泛应用，其他相关专业也相继把操作系统作为一门重要的必修或选修课程。

本书共 10 个项目。项目一简要介绍操作系统的构成、发展历史以及主要操作系统等。项目二主要介绍操作系统的运行机制和用户界面。项目三主要介绍进程和处理机管理的有关概念和技术。项目四主要介绍操作系统对进程的支持、同步和互斥的概念、实现同步与互斥的手段、死锁的概念与死锁防止。项目五主要介绍外部设备的分类及 I/O 控制原理等基本知识、设备使用方法、I/O 子系统的层次结构及设备管理子系统的常用技术、辅存及磁盘请求调度技术等。项目六和项目七分别介绍文件系统及设备管理技术。项目八介绍最常用的支持对称多处理机结构的并行操作系统实现技术及网络结构下分布式操作系统的有关技术。项目九介绍计算机安全问题和相关技术。项目十介绍虚拟化的基本原理与关键技术。

本书由锦州师范高等专科学校马芳、秦朗、蒲宏伟担任主编。具体编写分工如下：马芳

负责编写项目四、项目八和项目九的内容；秦朗负责编写项目五至项目七的内容；蒲宏伟负责编写项目一、项目二、项目三和项目十的内容。全书由马芳、秦朗、蒲宏伟负责统稿。

本书可作为高等学校计算机大类专业的教材，对于具有高级程序设计语言初步知识和对计算机有一定了解的专业人士，亦是较全面的参考书。

本书参考了大量发表或出版的文献以及网络资源，由于篇幅所限和其他原因，书末仅列出了部分参考资料，无法一一注明全部参考资料的来源和作者的具体姓名，无论出现何种情况，在此一并向各位作者表示感谢！

由于编者水平有限，虽勤勉谨慎，书中纰漏与不当之处仍在所难免，恳请读者能够理解并给予指正，

编　者

目 录

MU LU

项目一 操作系统引论

操作系统是计算机系统及智能嵌入式设备不可或缺的基础软件，是系统资源的管理、调度、控制中心。一方面，操作系统为用户提供操作使用环境、为用户程序提供系统公共服务；另一方面，操作系统采用合理有效的方法组织多个任务共享计算机的各种资源，最大限度地提高资源的利用率。

本项目将介绍操作系统组成及操作系统在计算机系统中的地位和作用，操作系统服务接口“系统调用”、所管资源的使用方式及为满足资源高效利用所支持的任务并发运行机制，以及操作系统历史的演变过程。

知识目标

1. 掌握操作系统的地位、作用及其在计算机系统中的地位。
2. 操作系统服务接口“系统调用”、所管资源的使用方式及为满足资源高效利用所支持的任务并发运行机制。
3. 理解操作系统的发展历程。
4. 熟悉操作系统的主要分类，如批处理系统、分时系统、实时系统、网络操作系统等。

能力目标

1. 能够对操作系统的基本原理和实践技能有全面而深入的理解和掌握。
2. 培养学生的动手能力和解决实际问题的能力。

素养目标

培养学生的系统思维、科技报国精神、团队合作精神、自主创新能力、国家安全意识、社会责任担当和职业道德素质，培养德智体美劳全面发展的高素质人才。

任务一　什么是操作系统

众所周知，中央处理器（CPU）、主存、磁盘、终端、网卡等硬件资源通过主板连接构成了看得见、摸得着的计算机硬件系统。为了能使这些硬件资源高效地、尽可能并行地被用户程序使用，也为了给用户程序提供易用的访问这些硬件资源及存放于磁盘上的程序或数据等软件资源的方法，必须为计算机配备操作系统。

操作系统的作用是高效管理计算机的处理器、主存、外部设备等硬件资源，及存放于存储设备的文件等软件资源，并组织用户多个任务（以进程或线程的形式）的同时或分时使用这些资源。

资源的有效利用及的便利性是操作系统要实现的两个重要目标。为此，操作系统对资源的管理原则是尽可能共享、减少闲置时间。另外，操作系统提供专门的“系统调用”接口方便上层用户程序调用系统提供的服务，来使用这些资源，提供专门的命令行或窗口用户界面供用户启动各种任务程序。

操作系统要管理的资源主要有处理器、主存、外部设备及外存空间（文件）。操作系统管理着这些资源，对应的程序模块是进程管理、存储管理、设备管理及文件系统，这些模块向上层软件以系统调用的方式提供了资源使用服务。

1.1.1　系统的软件构成

计算机系统的软件层次及构成如图 1-1 所示。

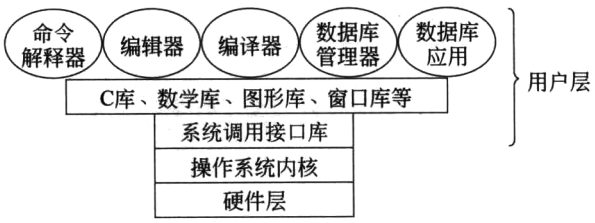


图 1-1　计算机系统的软件层次及构成

当用户在计算机中安装操作系统时，如图 1-1 所示的操作系统内核、命令解释器、编辑器、编译器、各种库程序，数据库管理器、Web 服务器等都从安装介质复制到了计算机系统的磁盘上。

命令解释器是必不可少的一个程序，用户通过它输入命令行来使用计算机系统。注意，在提供窗口界面的系统中，用户通过与命令解释器功能对等的程序管理器，如 Windows 的 explorer.exe，来使用计算机。

在现代操作系统实现中，命令解释器程序不作为操作系统内核的组成部分，但它是不可缺少的，用户在终端输入的命令就是由命令解释器程序接收并解释执行的。其他的操作系统内核层之上的程序则是根据计算机的功能定位而选择安装的。如果将计算机定位成程序开发用的工作站，那么用户必须安装编辑器进行程序编辑，并安装编译器进行程序编译。如果把计算机作为一个网络上的 Web 服务器，那么必须安装 Web 服务器程序。这些在操作系统内核层之上的程序，不管是命令解释器、Web 服务器或用户自编的程序，都是作为任务，通过操作系统提供的进程机制来运行的。

狭义上，操作系统只包含如图 1-1 所示的操作系统内核，它是一个非常重要的系统程序，管理着系统中所有的公共资源，并提供程序运行的进程/线程机制。由于操作系统内核工作的重要性、特殊性，它必须在一种特殊的保护状态下运行，以免受到上层程序的干扰和破坏。它提供一组称为系统调用的接口，供上层程序调用，从而保证操作系统内核在特殊保护状态下运行，并且满足上层程序对系统资源的申请、使用、释放以及进程的创建、结束等诸多服务的调用。

图 1-1 中的各种库程序实际上就是一些可以重用的、公用的子程序，它们提供形形色色的功能。系统提供这些库程序是为了方便用户编程，用户不必为了实现一个通用的功能再重写程序代码，而只要引用库程序中的函数即可。库程序可以看成一些通用的、公共的程序集合，可以是复杂的计算函数，也可以是进一步调用操作系统内核提供的资源服务实现的复合功能的函数，如 C 库中的 printf（）函数，它就

需要进一步调用 write 系统调用完成数据输出。

1.1.2　系统调用

从图 1-1 可以看出，操作系统内核位于计算机硬件之上。操作系统内核为用户层程序提供服务，这些服务以系统调用方式提供。系统调用与普通函数调用有相似性，但又有特殊性。操作系统内核提供了一些可以被任意用户层程序调用的涉及所管资源使用的公共服务，所以用户不需要再编写实现这些服务功能的程序，只要调用操作系统提供相应的“系统调用”函数即可。

要特别注意系统调用的特殊性，即系统调用处理程序运行在一种特殊的保护状态（核心态）下。在这种状态下，程序可以执行一些特权指令，访问用户层程序访问不到的系统存储空间。

系统调用之所以具有这样的特殊性，是因为系统调用服务程序涉及系统资源的操作，不应该被不信任的用户程序以普通函数调用方式调用，以免出现操作系统内核被破坏。系统调用指令中没有要调用的函数地址，在转到一个约定的固定地址后，靠系统调用号来区别转到不同的内核函数运行，这样可以杜绝地址出错引起操作系统内核崩溃。图 1-1 中的系统调用接口库中的函数中就包含系统调用指令，即所谓的“trap”指令，由它转入操作系统内核程序，而上层的用户程序只需要按照普通函数调用来调用系统调用库中的函数即可，这样做的目的是方便高级语言编程者编程。

哪些程序作为普通库函数，哪些程序必须作为系统调用呢？举例来说，求$\sqrt{x}$的值是许多用户程序都要做的工作，可以把它作为一个公共子程序实现。那么它需要作为系统调用在操作系统内核实现吗？回答是否定的。虽然计算$\sqrt{x}$需要许多条机器指令来实现，但因为它只是纯计算，不涉及系统的其他共享资源，因此可以把它作为数学函数库中的子程序来实现。

计算机都使用磁盘来存放系统或用户的程序及数据，存放在磁盘中的程序或数据称为文件。当一个用户程序要访问某个文件中的某段信息的时候，需要知道这段信息存放在磁盘的哪个扇区中，需要向磁盘控制器发送读扇区的请求，需要查看扇区信息是否已经读入主存。如果这些操作都交给用户来编程，不仅复杂，而且重复。因此，操作系统给用户提供一个简单的统一的文件操作界面，即每个文件可以按照读/写方式打开，然后进行读/写操作，最后关闭文件。用户无须知道磁盘是怎么工作的、如何读/写数据，也不需要知道要读/写的数据放在磁盘的哪个扇区中，只需要知道读/写哪个文件的哪一段数据即可，利用这个简单的文件操作界面就可以与磁盘进行数据交换。至于确定文件信息在哪个扇区中、如何读这个扇区，则由操作系统的 read 系统调用处理程序及下层的文件系统模块、磁盘驱动程序来实现。因为磁盘和文件属于共享资源，对其操作的程序属于操作系统内核程序，需要通过对应的系统调用接口来调用并运行。

1.1.3　资源共享

计算机由处理器、主存、辅存、终端设备、网络设备等硬件资源组成。处理器提供程序执行能力；主存、辅存提供程序和数据的存储能力；终端设备提供人机交互能力；网络设备提供机间通信能力。这些硬件资源要能被计算机用户高效地使用，必须有适合每种硬件资源特点的资源分配和使用机制。

为使硬件资源充分利用，必须允许多用户或单用户以多任务方式同时使用计算机，以便让不同的资源由不同的用户任务同时使用，减少资源的闲置时间。例如，当一个用户任务将文件内容从磁盘向主存的缓冲区读出时，另一个用户任务可以让自己的程序在处理器上运行。这样，处理器、主存、磁盘同时工作，也就提高了资源利用率。

要让每种资源被多用户任务充分利用，就需要研究每种资源的特点。对于单处理器来说，它只能执行一个指令流。如果多个用户任务都要使用它，那只有让多个用户任务的程序分时地在处理器上运行，也就是说，处理器交替地运行多个用户任务中的程序。这意味着，操作系统要合理调度多用户任务使用处理器。存储设备为程序和数据提供存放空间，只要多个用户的程序和数据按照规定的位置存放，互不交叉占用，它们是可以共存的，操作系统要做的事就是管理存储空间，把适用的空间分配给用户的程序

和数据使用，当用户任务访问这些程序和数据时要能够找到它们。

针对不同资源特点，资源管理包含两种资源共享使用的方法："时分"和"空分"。

（1）时分就是由多个用户进程分时地使用该资源。多任务分时地在处理器上运行，我们称任务在处理器上并发运行。还有很多其他的资源也必须分时地使用，如设备控制器等，这些控制部件包含控制 I/O 的逻辑，必须分时地使用。

（2）空分是针对存储资源而言，存储资源的空间可以被多个用户进程共同以不同空间各自占用的方式使用。

在时分共享使用的资源当中，有如下两种不同的使用方法。

（1）独占式使用。独占表示某用户任务必须占用资源执行对资源的多个操作，得到一个逻辑完整的结果。例如，如果多用户任务使用打印机，那么对打印机的独占式使用是指多用户任务一定是轮流地使用该打印机的，每个用户任务使用打印机时，执行了多条打印指令，打印了一个完整的对象（如完整的文件）。这里，每个用户任务要执行多条打印指令，为了不让多条打印指令在执行过程中被别的打印任务中断，用户任务需要在执行打印指令前申请独占该打印机资源，执行完所有打印指令后再释放。这种方式的使用，实质上就是互斥使用。

（2）非独占式使用。这种共享使用是指用户任务占用该资源无须得到一个逻辑上的完整的结果，或者说一次使用就是一个逻辑完整的结果。例如，对处理器的使用，用户任务随时都可以被剥夺处理器，只要运行现场保存好了，下次该用户任务再次占用处理器时就可以恢复现场继续运行。

操作系统应针对不同的资源类型，实现不同的资源分配和使用策略，并为资源分配、释放、使用提供相应的系统调用接口。

1.1.4 并发运行机制

用户要实现各种任务，必须执行相应的程序。用处理器来执行程序，用程序驱动外部设备来进行数据交换。用户的意图由程序及程序的输入参数表示出来，为了实现用户意图，必须让实现相应功能的程序执行；为了能让程序执行，需要由操作系统给程序及程序数据安排存放空间；为了能提高资源利用率，增加并发度，还必须让多个用户程序能分时占用处理器；为了能够让一个程序还没运行完就让另一个程序占用处理器运行，就必须保存上一个程序的运行现场。因此，必须要对实现各种用户意图的各个程序的执行过程进行描述和控制。

说明程序执行的状态、现场、标识等各种信息，有选择地调度某个程序占用处理器运行，这些工作必须由操作系统完成，这也是为了实现程序对处理器的分时轮流使用。

操作系统一般用进程（或线程）机制来实现程序的执行。

进程是指程序的执行，即程序针对某个数据集合的处理过程。操作系统的进程调度程序选择进程到处理器上运行。操作系统为用户提供进程创建和结束等的系统调用功能，使用户能够创建新进程以运行新的程序。

系统加电启动，操作系统要进行系统初始化，然后为每个用户创建第一个用户进程，用户的其他进程则可以由先前生成的用户进程通过"进程创建"系统调用陆续创建，以完成用户的各种任务。

在支持交互使用计算机的系统中，用户的第一个进程往往运行命令解释器程序（对于图形窗口终端用户而言，就是具有窗口界面的程序管理器，如 Windows 操作系统的 explorer. exe），这个程序会从终端获得用户输入的命令（或用户单击执行程序图标的信息），再进行相应的处理，通常会调用操作系统的"创建进程"系统调用，创建新进程去运行实现命令功能的程序。例如，在 Linux 操作系统控制的终端上输入

```
$ cp /home/ly/test. c /home/wq/hello. c
```

那么，这一行字符串会由命令解释器程序获得，它会创建一个子进程，由子进程去运行 cp 实用程序，由 cp 实用程序建立一个新文件/home/wq/hello. c，并把/home/ly/test. c 文件的内容读出来，写到/home/

wq/hello. c 中。

有了进程（或线程）机制，就能够实现任务的并发或并行执行。并发与并行的区别：并发是让多个任务以进程或线程轮流在处理器上运行，并行则意味着多个任务以进程或线程在不同处理器上同时运行。

任务二　操作系统的发展历史

操作系统的发展

操作系统并不是与计算机硬件一起诞生的。在计算机诞生初期，计算机系统特别昂贵，提高计算机系统中各种资源的利用率，是操作系统最初发展的推动力。推动操作系统发展的因素很多，主要可归结为两大方面：硬件技术更新和应用需求的变化。

随着硬件技术更新，一方面带来计算机的器件在不断更新，另一方面催动计算机的体系结构不断优化。因此，计算机的运算速度越来越高，数据传输效率越来越高，存储容量越来越大，应用场合也越来越广。这都要求操作系统能以更好的方式去管理和调度硬件设备，使得程序运行更快，数据吞吐量更大，更及时响应用户，设备利用率更高，同时适应从科学计算到商业，工业，管理，办公等不同的应用场合。

应用需求的变化也促进了操作系统的不断更新升级。应用需求从早期单纯的科学计算，推广到业务管理、事务处理等交互式应用，以及分布式数据采集、存储、分析和工业控制等实时应用。这些新的需求也不断推动操作系统的持续发展。

从第一台计算机诞生到现在，计算机无论是在硬件方面还是在软件方面都取得了很大发展，操作系统也经历了从无到有、由弱到强的过程。二十世纪四十年代到五十年代中期，是无操作系统的时代。在二十世纪五十年代中期出现了第一个简单的批处理操作系统。二十世纪六十年代中期产生了多道程序批处理系统，不久之后又出现了以多道程序为基础的分时系统。二十世纪七八十年代，以分时系统为基础，开始出现多种适用于不同应用场景的或者在特定功能上加强了的操作系统。譬如，适合微型计算机的微机操作系统，适合工业控制的实时操作系统，具有网络通信功能的网络操作系统，支持分布式计算的分布式操作系统等。

1.2.1　手动操作阶段

第一代计算机运行速度较慢，外部设备较少，因此，编制和运行一个程序也比较简单。那时候，程序员往往直接使用机器语言来编制一个程序。这种“目标程序”通过“打孔”的方式被记录在卡片（或纸带）上。程序员将已穿孔的纸带（或卡片）装入输入设备，然后启动输入设备把程序和数据输入计算机内存，接着通过控制台开关启动程序开始处理数据。计算完毕后，打印机输出计算结果；用户取走结果并卸下纸带（或卡片）后，才让下一个用户上机。在整个期间，计算机都被一个程序员所占用。因而，不需要专门的操作员，程序员身兼两职，既是操作员，也是程序员。

手动操作方式有两个特点：

（1）程序的启动与结束都是手动处理。

（2）用户预约上机并独占整个计算机。

手动操作方式的缺点主要表现在三个方面：

（1）CPU 运行效率低，在用户占用整个计算机的期间，CPU 实际运行时间极少。

（2）用户独占整个计算机的全部资源，造成资源浪费。

（3）程序的运行过程缺少交互性。

随着计算机速度的提升，手工操作的慢速度和计算机的高速度之间形成了尖锐的矛盾，手工操作方式已严重影响系统资源的利用率。唯一的解决办法就是摆脱人的手工操作，实现作业的自动装载和过渡。这正是作业自动加载和成批处理的思路。

1.2.2 单道批处理系统

单道批处理系统是能够控制计算机自动处理一批作业，逐个加载、运行、撤出其中的每个作业，直到全部作业处理完毕的系统监控程序。

单道批处理系统的目的是提高系统吞吐量和资源的利用率。批处理系统的特点体现在 3 个方面：成批、自动、单道。

成批：在磁盘中预先存放一批作业，构成作业队列，等待处理。

自动：自动识别新的作业，自动完成作业的装入、运行、撤出等一系列操作，不需要人工参与。

单道：作业是逐个被系统处理的，属于串行运行。一个作业没有被处理完之前，其他作业是不能被处理的。因此，批处理系统也称为单道批处理系统。

单道批处理系统的缺点：

（1）平均周转时间长。

（2）无交互能力。用户在批处理系统中把作业交给操作系统之后直到系统将作业完成，用户都不能与自己的作业进行交互，这对修改和调试程序是极为不利的。

单道批处理系统有两种实现形式：联机批处理系统、脱机批处理系统。

联机批处理系统的结构如图 1-2 所示，作业的输入输出过程都由主机来控制，主机与输入设备之间增加一个存储设备——磁带。在运行于主机上的监督程序（所谓的批处理程序）的自动控制下，成批地把输入机上的用户作业读入磁带并形成作业队列，然后依次把磁带上的用户作业逐个装入主机运行并把结果向输出机输出，直到该批作业完成。

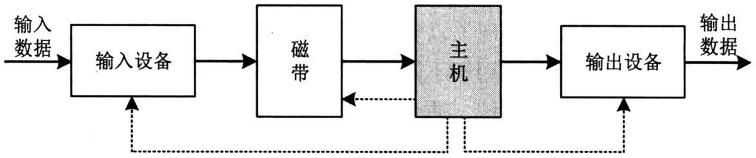


图 1-2 联机批处理系统的结构

监督程序不停地顺次处理各个作业，实现作业到作业的自动转接，减少作业建立时间和手动操作时间，有效解决人机矛盾，提高计算机的利用率。但是，在作业输入和结果输出时，主机的高速 CPU 仍长时间处于空闲状态，等待慢速的输入输出设备完成工作。

脱机批处理系统的结构如图 1-3 所示，增加了一台不与主机直接相连而专门用于与输入/输出设备打交道的卫星机。卫星机的作用是完成输入过程（控制读卡机读取用户作业并放到输入磁带）和输出过程（控制打印机从输出磁带上读取执行结果并打印出来）。主机的作用则是利用上述提到的批处理方式自动按批处理输入磁带上的作业，并将结果存放到输出磁带上。

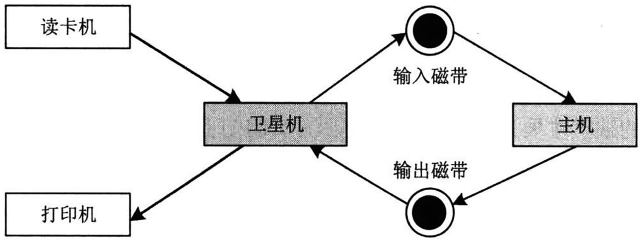


图 1-3 脱机批处理系统的结构

主机不直接与慢速的输入输出设备打交道，而与速度相对较快的磁带机发生联系，这样可有效缓解主机与设备的速度矛盾。主机与卫星机可并行工作，二者分工明确，可以充分发挥主机的高速计算能力。脱机批处理系统在二十世纪六十年代应用十分广泛，它能极大缓解人机矛盾及主机与外设的矛盾。IBM 7090/7094 计算机配备的监督程序就是脱机批处理系统。

单道批处理系统的不足在于，主机内存中仅存放一道作业，每当它在运行期间发出输入输出（I/O）

请求后，高速的 CPU 便处于等待低速 I/O 完成的状态，致使 CPU 空闲。为提高 CPU 的利用率，又引入了后来的多道批处理系统。

随着计算机的发展，一种协助用户高效编写源程序的汇编系统产生了。在汇编系统中，难记的机器操作码被容易理解、容易记忆的指令所代替。源程序按固定的汇编语言格式书写，然后利用预先编制的汇编解释程序，将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。因而，在这样的计算机系统中，一个程序的编写、解释和执行过程分为 2 个阶段、6 个计算步骤，如图 1-4 所示。每步的功能如下：

（1）通过引导程序把汇编解释程序装入到计算机中；

（2）通过汇编解释程序读入源程序，并执行汇编过程；

（3）产生一个目标程序，并输出到卡片或纸带上；

（4）通过引导程序把目标程序装入计算机；

（5）目标程序读入数据卡片上的数据；

（6）产生计算结果，并输出到卡片或打印纸上。

其中（1）～（3）属于程序编写和解释阶段，（4）～（6）属于执行阶段。

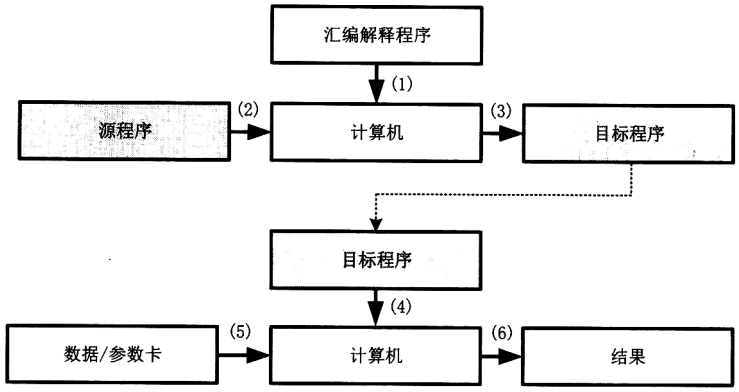


图 1-4　程序编写、解释和执行过程

1.2.3　多道批处理系统

多道批处理系统是指利用多道程序设计技术，在内存中存放多道程序，当某道程序因为某种原因（例如执行 I/O 操作时）不能继续运行时，监控程序便调度另一程序投入运行，这样可以使 CPU 尽量处于工作状态，提高系统效率。多道批处理系统的设计目的是提高系统的资源利用率（或吞吐量），实现 CPU 与外设并行、外设之间并行，从而实现作业处理流程的自动化。

图 1-5 展示了 A、B 两个程序在系统中运行的过程。将 A、B 两个程序同时存放在内存中，它们在系统的控制下，可相互穿插、交替地在 CPU 上运行：当程序 A 因请求 I/O 操作而放弃 CPU 时，程序 B 就可以见缝插针地占用 CPU 运行，使 CPU 尽量不空闲，而正在为程序 A 执行 I/O 操作的 I/O 设备也同时在忙碌。显然，CPU 和 I/O 设备都处于“忙”状态，大大提高了资源的利用率，从而也提高了系统的效率。程序 A、B 全部完成所需时间远比它们各自单独运行所花的时间之和要少。若不考虑系统调度的额外开销，理论上，在内存中存放的程序数量越多，越能充分利用 CPU 和外设。

多道程序设计技术不仅使 CPU 得到充分利用，而且提高了 I/O 设备和内存的利用率，从而提高了整个系统的资源利用率和系统吞吐量，提高了单位时间内处理作业的个数，最终提高了整个系统的效率。

多道批处理系统的缺点：

（1）单个作业处理时间变长。

（2）作业的交互能力差，用户一旦提交作业就失去了对其运行的控制能力。

（3）作业的运行过程不确定。每个作业的实际运行过程受其他作业的运行过程影响。

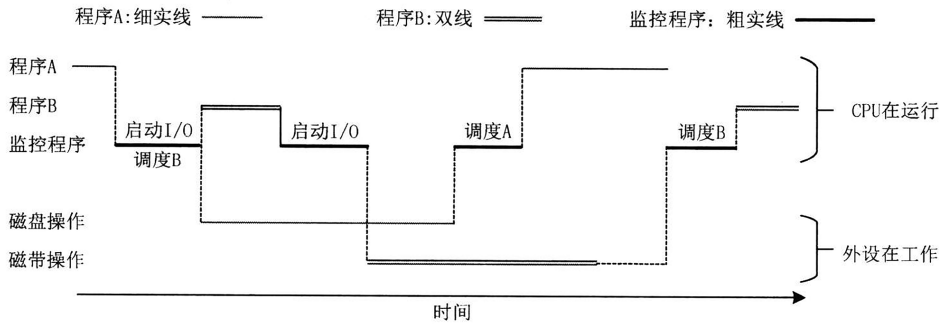


图 1-5　A、B 两个程序在系统中运行的过程

多道批处理系统比单道批处理系统要完善得多，同时也为现代操作系统引入了一些新的重要概念。为了使多个程序能处于就绪状态，必须将它们放入内存中，因此需要设计一些用于存储器管理的数据结构，例如表或数组等。另外，如果有多道作业准备运行，则处理器必须决定先运行哪一个，这就要有调度算法，这些概念将在后面章节讨论。

1.2.4　分时操作系统

随着涌现事务性的任务，用户要求计算机系统能够提供更好的交互性，以支持多用户或多任务的并发执行。此外，随着计算机硬件结构的发展，出现了多终端计算机。图 1-6 展示了多终端计算机的结构，一台计算机主机可同时连接多个用户终端。主机具有强大的 CPU 和存储系统，终端只具备显示设备和键盘。用户可在自己的终端上联机使用计算机，就好像自己独占计算机一样。

在多终端计算机系统中，采用了分时技术来实现“用户似乎独占了主机”的效果。分时技术的思想：把处理机的运行时间分成很短的时间片．以时间片为单位轮流把处理机分配给各联机终端使用。若某个作业在分配给它的时间片内不能完成其计算，也强行把该作业暂时中断，把处理机让给另一个作业使用，等待下一轮时间片时再继续其运行。由于处理机速度很快，在时间片不太大且终端数量不太多的情况下，作业运行轮转得很快，给每个用户的印象是，好像自己独占了一台计算机。每个用户可以通过自己的终端向系统发出各种操作控制命令，并在充分的人机交互情况下，完成其作业的运行。

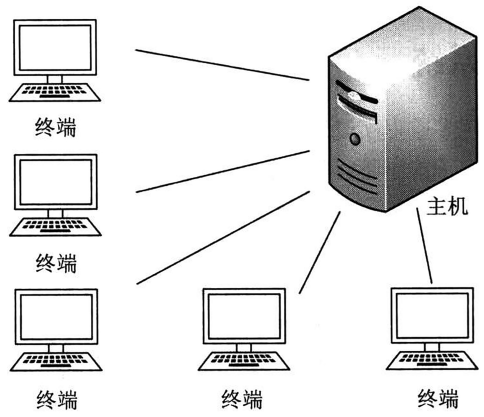


图 1-6　多终端计算机的结构

利用分时技术设计的计算机系统称为分时系统，而对应的操作系统则称为分时操作系统（Time-sharing Operating System）。分时操作系统允许多个用户同时联机使用计算机。

时间片是程序一次运行的最小时间单元。在划分时间片的时候，要根据系统的总体设计框架来考虑。比如说，要考虑用户的响应时间、系统一次容纳的用户数目、CPU 的指令周期、中断处理时间、程序运行现场的保护和恢复时间等。通常来说，在一个时间片内，至少应该能够让程序完成一次完整的输入或输出过程，或者中断处理过程。如果时间片的时间比这还短就会失去实用意义，系统效率将会严重降低。

用户要求的响应时间越短，系统一次容纳的用户数目越多，时间片就必然要设置得更短。例如，用户要求的响应时间为 ΔT，系统可容纳的最大用户数目为 M，则处理机时间片最大为 ΔT/M。

虽然分时系统是多用户系统，但对于每一个用户来说，并不会感觉到单用户机与多用户机的区别，而是各自都感觉在独立地使用着计算机。分时系统具备 3 个特点。

（1）多路性。系统同时支持多路终端的连接。支持多路性的内部机制是处理机分时共享和设备共享。从微观上看，不同用户分享着处理机时间的不同片段，而从宏观上看，所有用户在同时享用着计算机系

统。在现代操作系统中，多路性本质上是多用户性，或多任务性。

（2）独立性。多用户各自独立地使用计算机，相互之间并无影响。独立性的实现主要依赖于存储器的安全保护，由于不同用户占用存储器上的不同区域，因此要求不同区域中的用户程序在执行时不可相互干扰或者破坏，这可以通过一定的存储器保护机制来实现。

（3）交互性。用户可以通过终端直接与计算机进行对话，用户通过键盘向主机输入指令或数据，控制计算机运行，而主机则通过终端给用户返回结果或提示信息、帮助信息等。设计交互性的时候，必须要考虑到交互的及时性。及时性是指用户可以忍受的最大用户响应时间，它与处理机的指令周期和时间片的划分有关。及时性太差可能会使交互失去实际的意义。

1.2.5　分时操作系统衍化

从二十世纪七八十年代开始，以分时系统为基础，开始出现适用于不同应用场景的或者在特定功能上加强的操作系统实例。譬如，适合微型计算机的微机操作系统，适合工业控制的实时操作系统，具有网络通信功能的网络操作系统，支持分布式计算的分布式操作系统等。

（1）微型计算机操作系统。二十世纪七十年代末期，随着超大规模集成电路（Very Large Scale Integration Circuit，VLSI）的发展，微型计算机开始出现。微型计算机的特点是通用性强、体积小、价格便宜，并从早期单纯的科学计算工具发展成为能够处理数字、符号、文字、语言、图形、图像、音频、视频等多种信息的综合计算工具。配置在微型计算机上的操作系统称为微型计算机操作系统，简称微机操作系统。微机操作系统按照处理机的字长可分为 8 位、16 位、32 位或 64 位操作系统。典型的微机操作系统有 DOS、Windows 系列以及近年来引起广泛关注的 Linux 系列操作系统。

（2）实时操作系统。实时操作系统主要应用于工业过程控制、军事实时控制、金融等领域。对实时操作系统的主要要求是响应时间短，系统可靠性高。

实时系统对外部信号必须能及时响应，响应的时间间隔要短至足以控制发出实时信号的环境。实时操作系统的实时性是以控制对象所要求的开始时间和完成时间来确定的，一般为秒级、毫秒级直至微秒级，并在此时间限制内完成特定功能。实时性必须满足从事件发生到系统响应之间的最长时间限制。

在实时操作系统中，软/硬件的任何故障都会给系统带来严重后果。因此，在实时操作系统中，必须采取相应的软/硬件措施，保证系统的绝对安全和高度可靠。

实时操作系统不仅仅是表现在“快”上，而更主要的是实时系统必须在限定时间内对外来事件做出响应。当然这个限定时间的范围是根据实际需要来定的，例如，在普通工业控制系统中反应时间可能较长，而在航空飞行控制系统中这个时间可能就会很短。总之，实时程序必须保证在严格的时间限制内完成响应。

实时操作系统有硬实时和软实时之分，硬实时要求在规定的时间内必须完成操作，任何时间延迟都会导致系统的错误，这是在操作系统设计时被保证的；软实时则只要按照任务的优先级，尽可能快地完成操作即可。

例如，可以为确保生产线上的机器人能实时拾取某个产品而设计一个操作系统。在硬实时操作系统中，如果不能在规定时间内成功拾取产品则视此次任务失败且结束工作。而在软实时操作系统中，生产线仍然能继续工作，但产品的最终输出速度会因产品不能在规定时间内到达而减慢，这使机器人在短时间内出现不生产的现象。

具有通用目的的多数操作系统（也可称为普通的分时操作系统）虽然也具有一定的实时性，也能解决一部分实时应用问题，但是不是严格意义上的实时操作系统。

（3）嵌入式操作系统。嵌入式操作系统（Embedded Operating System，EOS）是嵌入式系统中使用的操作系统。嵌入式系统是把具有计算能力的智能控制模块嵌入到目标系统中，满足目标系统的智能化控制要求的小型、廉价、可靠、专用的计算机系统类型。嵌入式系统有时也称为嵌入式计算机。嵌入式系统广泛使用于工业控制、武器系统、航空航天、机器人、智能仪器仪表、家用电子、消费电子、通信电

子等领域。嵌入式系统通常需要借助电子技术、材料技术、人工智能等其他相关学科的支持才能实现。

嵌入式系统是以计算机技术为核心，面向用户、面向产品、面向应用，软硬件可裁减的，对功能、可靠性、成本、体积、功耗等有严格要求的专用计算机系统。嵌入式系统无论在硬件上，还是在软件上，通常都被设计得非常紧凑，抛弃了普通 PC 上那些冗余无用的功能，以节省空间和成本，提高效率和灵活性。嵌入式系统的典型特点是软硬件可以裁剪，是软硬件一体化的系统。

嵌入式操作系统是嵌入式系统中最底层的核心系统软件，具有通用操作系统的基本功能，譬如 CPU 管理、内存管理等核心功能，负责嵌入式系统的全部软、硬件资源的分配、调度和控制。典型的嵌入式操作系统有 Linux、μcOS-II、VxWorks、RT-Thread、RT-Linux、HarmonyOS 等。

嵌入式 Linux 与标准 Linux 略有不同。嵌入式 Linux 对标准 Linux 操作系统进行了裁剪修改，使之适合在嵌入式计算机系统上运行，具有嵌入式操作系统的特性。嵌入式 Linux 由于具有性能优异、移植性强、源代码开放、应用软件丰富、技术群体庞大、技术生态成熟等特点，使得其在嵌入式系统领域得到了极其广泛的应用。

μcOS-II 是用 C 语言编写的结构小巧、抢占式的多任务实时内核。μcOS-II 能管理 64 个任务，并提供任务调度与管理、内存管理、任务间同步与通信、时间管理和中断服务等功能，具有执行效率高、占用空间小、实时性能优良和扩展性强等特点。

VxWorks 是美国 Wind River Systems 公司于 1983 年设计开发的高性能、可扩展的实时操作系统。VxWorks 支持市场上大多数处理器，因其具有良好的可靠性和卓越的实时性，被广泛地应用在通信、军事、航空、航天等实时性要求极高的高精尖技术领域中。

RT-Thread 操作系统由我国开源社区主导开发。RT-Thread 不仅包含一个实时操作系统内核，也有完整的一整套应用组件，例如 TCP/IP 协议栈、文件系统、Libc 接口、图形用户界面等。RT-Thread 具有相当大的发展潜力，已经被广泛应用于能源、车载、医疗、消费电子等多个行业。

RT-Linux 是美国墨西哥理工学院基于标准 Linux 内核开发的嵌入式实时操作系统。RT-Linux 也是开源开放式自由软件。RT-Linux 使用精巧的内核，并把标准 Linux 核心作为实时核心的一个进程，同用户的实时进程一起调度。这样对 Linux 内核的改动非常小，并且可以充分利用 Linux 下现有的丰富软件资源。

有些嵌入式系统要求具有一定的实时处理能力，这就要求嵌入式操作系统同时具有相应的实时处理能力。前面提到的 VxWorks、RT-Thread、RT-Linux 同时也都是实时操作系统。

（4）网络操作系统。网络操作系统（Network OS）具备计算机网络核心支撑软件。网络操作系统除了具备通用操作系统的基本功能，还具有网络管理功能，即提供网络通信和网络资源共享功能。通过计算机网络，用户可以共享其他计算机上的数据文件、软件应用以及共享硬盘、打印机、扫描仪和传真机等。

在计算机网络中，每台计算机既可以独立工作，作为一个独立自治的计算机系统存在，也可以相互之间通过 IP 地址彼此区分和识别，实现共享资源。所以网络操作系统实质上等同于“普通操作系统+网络通信+网络服务”。

NetWare 操作系统是早期的网络操作系统。NetWare 曾经以对网络硬件的要求较低而十分受欢迎。它支持无盘工作站的组建和管理，兼容 DOS 命令，应用环境与 DOS 相似。一些设备比较老旧的中、小型企业网络，学校实验机房网络、网吧等大量采用 NetWare 操作系统。

目前常见的 UNIX、Linux、Windows 等主流操作系统都具有网络功能，都是网络操作系统。

（5）分布式操作系统。分布式操作系统是指运行在分布式系统中，能直接对系统中各类资源进行动态分配和管理，有效控制和协调任务的并行执行，向用户提供统一接口的操作系统。

在分布式系统中，计算和处理功能分散在构成分布式系统的各个处理单元上。相应地，分布式系统可把大任务划分成可以并行执行的多个子任务，并动态地把这些子任务分配到各处理单元上，使它们能够并行执行。可见分布式系统最基本的特征是处理上的分布，即功能和任务的分布。

分布式系统是由多个处理单元构成的系统。其中，每个处理单元都包含处理机和局部存储器，它们

能独立承担系统分配给它们的任务。各个处理单元通过网络连接在一起，在统一的分布式操作系统的控制和管理下，实现各处理单元间的通信和资源共享，动态地分配任务和对任务进行并行处理。

分布式操作系统具有以下功能。

①资源管理功能：能对并行工作的大量处理单元、存储器和设备等资源进行有效的动态管理。由于分布式系统的构成具有模块性，不仅简化了操作系统对资源的管理，更提高了资源利用率。

②任务分配功能：在分布式系统中，任务的分配不是以一个任务为单位，而是以一组能并行执行的任务集为单位，同时将它们分配到多个处理单元上使之能并行执行。

③分布式进程同步和通信功能：系统采取分布式同步方式来保证不同处理机上的进程严格同步，实现它们之间的通信，以达到高度并行执行的目的。

④各处理单元无主、从之分，都具有执行管理程序的能力。

网络操作系统与分布式操作系统在概念上的主要区别：网络操作系统可以构架于不同的主机操作系统之上，通过网络协议实现网络资源的统一配置，可以在大范围内构成计算机网络。在计算机网络中并不要求对网络资源进行透明地访问，即需要显式地指明资源位置与类型，对本地资源和异地资源的访问区别对待。

分布式操作系统一般具有单点登录特性。在这种操作系统中，网络的概念在应用层被淡化了。所有资源，包括本地资源和网络资源，都用同一方式管理与访问，用户不必关心资源的位置或资源的存储方式。

1.2.6　经典操作系统实例

在办公、管理和各类数据服务应用中，涌现的经典操作系统有 DOS 系列、Windows 系列、UNIX 系列、Linux 系列和 macOS 系列等。

（1）DOS 系列。DOS 是 Disk Operating System（磁盘操作系统）的缩写，是在从 1981 年到 1995 年的微型计算机上使用的一种主要操作系统。DOS 最初由微软公司为 IBM PC 开发，称为 MS-DOS。其他公司后来生产的与 MS-DOS 兼容的操作系统，也沿用了这个称呼，如 PC-DOS、DR-DOS 等。

1981 年，MS-DOS 1.0 发行，作为 IBM PC 的操作系统与之捆绑发售，支持 16KB 内存及 160KB 五寸软盘。

1984 年，MS-DOS 3.0 增加了对新的 IBM AT 的支持，并开始对部分局域网功能提供支持。

1989 年，MS-DOS 4.0 增加了 DOS Shell 操作环境，并且有一些其他增强功能及更新。

1993 年，MS-DOS 6.x 增加了 GUI 功能和磁盘压缩功能，增强了对 Windows 的支持。

1995 年，MS-DOS 7.0 增加了对长文件名的支持以及对 LBA 大硬盘的支持。这个版本的 DOS 并不独立发售，而是内嵌在 Windows 95 中。

（2）Windows 系列。Windows 系列操作系统是由微软公司从 1985 年起开发的一系列窗口操作系统产品，主要包括个人（家用）、商用和嵌入式 3 条产品线。

Windows 1.0 在 1985 年问世，最初目标是在 MS-DOS 的基础上提供一个多任务的图形用户界面。Windows 1.0 是微软公司进入桌面系统视窗化的开端。

Windows 3.1 发布于 1990 年，在界面、交互性、内存管理等多方面有巨大改进。Windows 3.1 及之前的 Windows 版本均为 16 位系统，还不能充分利用硬件的强大功能。同时，由于它们只能运行在 DOS 之上，因而，它们还不能算真正的操作系统。

Windows 95 发布于 1998 年，是一个混合的 16 位/32 位 Windows 操作系统。它改良了硬件标准的支持，例如 MMX 和 AGP，支持 FAT32 文件系统、支持多显示器。Windows 95 重写了操作系统内核，不再基于 DOS；特别是增加了多任务，使得用户可以同时运行多个程序，并在提供强大功能（如网络和多媒体功能等）和简化用户操作（如桌面和资源管理等新特性）这两个方面都取得了突出的成绩。

Windows NT 是工作站和服务器上的 32 位操作系统，能充分利用 32 位微处理器等硬件的新特性和能

力，并使其很容易适应硬件变化，而不影响已有应用程序的兼容性。Windows NT 较好地实现了充分利用硬件新特性、可扩充性、可移植性、兼容性等设计目标。Windows NT 支持对称多处理机结构、内核多线程、多个可装卸文件系统。

WindowsXP 发布于 2001 年，字母 XP 表示体验（experience）之意。与历史版本相比，Windows XP 拥有更加华丽的界面、良好的交互性和系统安全性等特点。Windows XP 是微软历史上最受欢迎的操作系统之一，也是 Windows 系列操作系统中寿命最长的操作系统。

Windows 7 发布于 2009 年，内核版本号为 Windows NT 6.1。Windows 7 是第一个支持 64 位的操作系统。

Windows 10 发布于 2015 年，和之前版本相比，不仅人机交互界面变化很大，而且在易用性和安全性方面也有极大的提升。Windows 10 对云服务、智能移动设备、人机交互等新技术有较好的支持。Windows 10 对固态硬盘、生物识别、高分辨率屏幕等硬件也进行了优化支持。

（3）UNIX 系列。UNIX 操作系统是一种由美国 Bell 实验室研制的多用户、多任务通用操作系统。它从一个实验室的产品发展成为当前使用普遍、影响深远的主流操作系统，经历了一个逐步成长、不断完善的发展过程。

UNIX 的体系结构和源代码是公开的，采用 C 语言编写。UNIX 有两个基本的版本，一个是由 AT&T 的 Bell 实验室研制开发的 System V；另一个是由美国加州大学伯克利分校发布的 BSD UNIX。随着 UNIX 的发展，还产生了很多其他的商业版本，包括 IBM 公司的 AIX、HP 公司的 HP-UX 以及 Sun 公司的 Solaris 等。UNIX 操作系统同时实现了内核程序和核外程序的恰当分离和有机结合。内核程序包括进程管理、存储管理、设备管理及文件管理等四个部分。操作系统的其余功能则从内核中分离出来，以核外程序出现，并在用户环境下运行。UNIX 向用户提供两种界面：一种是用户使用命令，通过终端和系统进行交互；另一种是面向用户程序的界面，称为系统调用。

UNIX 具有树形结构的文件系统。它由基本文件系统和若干个子文件系统组成。UNIX 操作系统的文件系统是可动态装卸的，这样不但可以扩大文件的存储空间，而且有利于文件的安全和保密。在 UNIX 操作系统中，普通文件、文件目录和输入输出设备都是作为文件统一处理的。它们在用户面前具有相同的语法和语义。文件机制既简化了系统的设计，又便于用户使用。

（4）Linux 系列。Linux 是一种类似 UNIX 风格的多用户、多任务操作系统。Linux 最早是由芬兰人托瓦兹（Linus Torvalds）设计的。Linux 的设计借鉴了很多 UNIX 的思想。目前，Linux 操作系统可以运行在 x86、Alpha、MIPS 等多种架构的计算机上。从功能来看，它既可以作为普通的桌面操作系统，也可以作为网络操作系统和服务器操作系统。此外，它还可以作为嵌入式操作系统。

Linux 是可以在 GNU 公共许可权限下免费获得的，是一个符合可移植操作系统接口（Portable Operating System Interface，POSIX）标准的操作系统。

Linux 向用户提供两种界面：操作界面和系统调用。Linux 的传统操作界面是基于文本的命令行界面，即 Shell。Shell 具有很强的程序设计能力，用户可以方便地用它编制脚本程序（Shell Script）。脚本程序为用户扩充系统功能和控制计算机提供了更高级的手段。系统调用给用户提供了编程时可使用的界面，用户可以在编程时直接使用系统调用，系统通过这种界面为用户程序提供基础、高效率的服务。

Linux 是具有设备独立性的操作系统，它的内核具有很强的适应能力。由于 Linux 属于开源内核，因此用户可以修改其内核源代码，以适应新增加的外部设备。Linux 是一种具有良好移植性的操作系统，能够适应从微型计算机到大型计算机，再到定制的嵌入式系统的绝大多数硬件环境。

Linux 主流的发行版本有数十种之多，其中包括 Ubuntu、Red Hat、Slackware、Gentoo、Fedora、Debian、CentOS、KylinOS 等。图 1-7 为 Ubuntu 的工作桌面，具有良好的人机交互方式。

（5）macOS 系列。macOS 是运行于苹果 Macintosh 系列计算机上的操作系统，是首个在商用领域采用图形用户界面的操作系统。

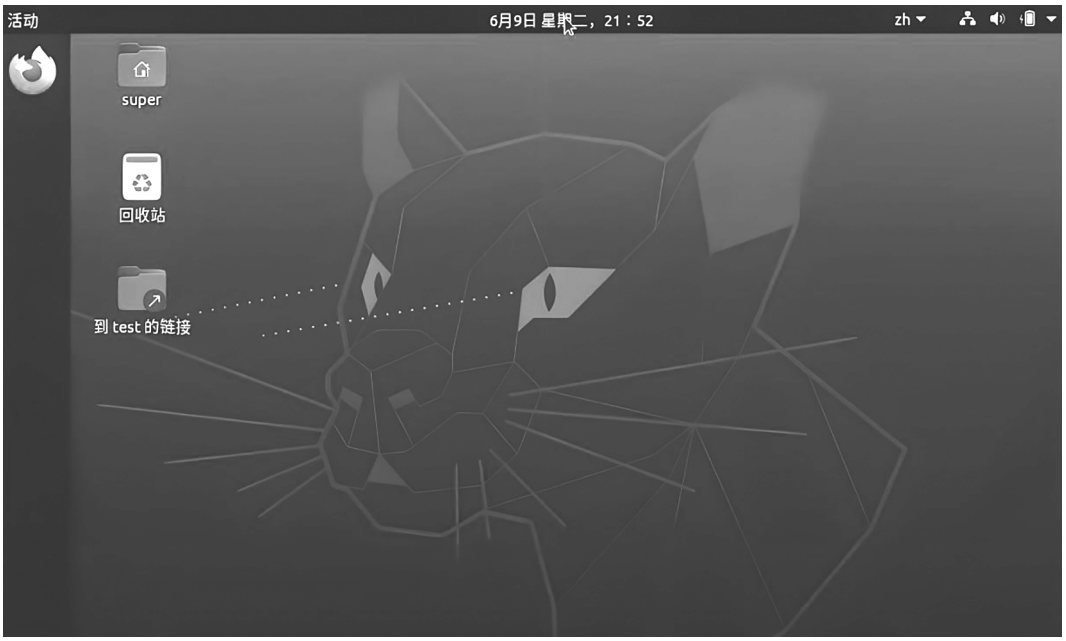


图 1-7　Ubuntu 的工作桌面

早期的 macOS 操作系统以 System x. xx 方式命名，是一个纯粹的图形操作系统，没有命令行模式。System x. xx 系列操作系统在内存管理、协同式多任务、功能扩展等方面具有一定的局限性。

后期的 macOS 以 macOS x 方式命名，并基于 BSD UNIX 的内核做了优化，实现了 UNIX 风格的内存管理和抢占式多任务处理技术。macOS x 系列操作系统允许同时运行更多程序，消除了因为一个程序的崩溃而导致其他程序崩溃的可能性。macOS x 系列操作系统同时支持命令行模式。

（6）国产操作系统。国产操作系统的研发和应用主要分为两个方向：一个是基于 Linux 内核的通用操作系统或服务器操作系统方向。这个方向典型的操作系统有麒麟系列操作系统 Kylin OS、深度 Linux 等。由于应用生态的原因，国产操作系统目前主要面向政府机关、事业单位、国防领域等对信息安全要求较高的领域。另一个是嵌入式操作系统方向。这个方向典型的操作系统有 HarmonyOS、RT-Thread 等。

任务三　主要操作系统介绍

随着计算机技术和软件技术的发展，已形成了各种类型的操作系统，以满足不同的应用要求。操作系统根据使用环境和对作业的处理方式可分为以下 6 个基本类型。

（1）批处理系统。
（2）分时系统。
（3）通用操作系统。
（4）个人计算机操作系统。
（5）网络操作系统。
（6）分布式操作系统。

下面对它们进行概要的说明。

1. 3. 1　批处理系统

批处理系统是一种早期的大型机操作系统。现代操作系统大都具有批处理功能。

图 1-8 给出了多道批处理系统中的作业处理步骤及状态。

多道批处理系统的主要特征如下：

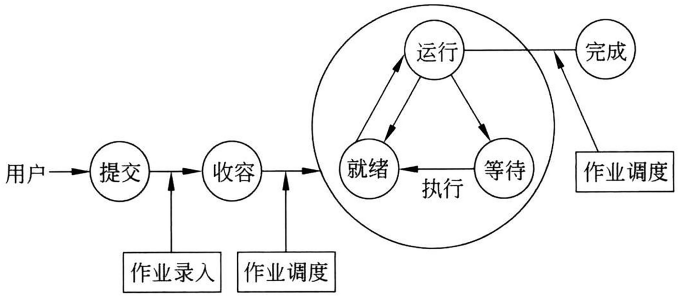


图 1-8 多道批处理系统中的作业处理步骤及状态

（1）用户脱机使用计算机。用户在提交作业之后，直到获得结果之前都不再和计算机打交道。作业可以直接交给计算中心的操作员，也可以通过远程通信线路提交。提交的作业由系统外存收容，成为后备作业。

（2）成批处理。操作员把用户提交的作业分批进行处理，由操作系统或监督程序负责对一批作业进行自动调度执行。

（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。

在多道批处理系统中，由于系统资源为多个作业所共享，其工作方式是作业之间自动调度执行，并且在运行过程中用户不干预自己的作业，因此大大提高了系统资源的利用率和作业吞吐量。其缺点是无交互性，用户一旦提交了作业，就失去了对其运行的控制能力。另外，由于它采用批处理方式，作业周转时间长，用户使用不方便。

注意，不要把多道程序（multiprogramming）系统和多重处理（multiprocessing）系统相混淆。一般讲，多重处理系统配有多个 CPU，因而能真正同时执行多道程序。当然，要想有效地使用多重处理系统，必须采用多道程序设计技术。但是，反之则不然，多道程序设计原则不一定要求多重处理系统的支持。多重处理系统与单处理系统相比，尽管增加了硬件设施，却换来了提高系统吞吐量、可靠性、计算能力和并行处理能力等好处。

1.3.2 分时系统

分时系统一般采用时间片轮转的方式，使一台计算机为多个终端用户服务。它对每个用户能保证足够快的响应，并提供交互会话能力，因此具有下述特点：

（1）交互性。交互会话工作方式给用户带来了许多好处。首先，用户可以在程序动态运行时对其加以控制，从而加快调试过程，有利于软件开发。其次，用户上机提交作业方便。特别是远程终端用户，不必将其作业交给机房，在自己的终端上就可以提交、调试、运行其程序。最后，分时系统还为用户之间进行合作提供了方便。用户可以通过文件系统、电子邮件或其他通信机制交换数据和信息，共同完成某项任务。

（2）多用户同时性。多个用户同时在自己的终端上机，共享 CPU 和其他资源，充分发挥系统的效率。

（3）独立性。由于采用时间片轮转方式使一台计算机同时为多个终端服务，对每个用户的操作命令又能快速响应，因此，用户感觉不到有别人也在使用这台计算机，如同自己独占这台计算机一样。

分时操作系统是一个联机多用户交互式操作系统。UNIX 是当今最流行的一种多用户分时操作系统。CTSS 和 MUTICS 这两个系统也是值得一提的。前者是一个实验性的分时系统，在 1963 年由麻省理工学院研制成功。后者是由麻省理工学院、贝尔实验室和 GE 公司从 1965 年开始联合设计的，尽管它并没有取得最后的成功，但对 UNIX 的研制是有影响的。

1.3.3　通用操作系统

批处理系统、分时系统和实时系统是操作系统的3种基本类型，在此基础上后来又出现了具有多种类型操作系统特征的操作系统，称为通用操作系统。它可以同时兼有批处理、分时处理、实时处理和多重处理的功能（或至少两种功能）。

1.3.4　个人计算机操作系统

个人计算机操作系统是联机单用户交互式操作系统，它提供的联机和交互功能与通用分时系统相似。由于它是个人专用的，因此在多用户和分时所要求的对处理机调度、存储保护等方面简单得多。然而，由于个人计算机的应用普及，对于更方便、友好的用户接口的要求会越来越高。

多媒体技术已迅速进入个人计算机系统，多媒体计算机给办公室、家庭、个人提供声、文、图、数据并重的全面的信息服务。它要求计算机具有高速信号处理、大容量内存和外存、大数据量宽频带传输等能力，能同时处理多个实时事件，也就是一个具有高速数据处理能力的实时多任务操作系统。

目前个人计算机操作系统以Windows和Linux为主。

1.3.5　网络操作系统

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联，实现信息交换、资源共享、可互操作和协作处理的系统。它具有以下特征：

（1）计算机网络是一个互联的计算机系统的群体。这些计算机系统在物理上是分散的，可在一个房间里，在一个单位里，在一个城市或几个城市里，甚至在全国或全球范围内。

（2）这些计算机是自治的，每台计算机都有自己的操作系统，独立工作，并在网络协议控制下协同工作。

（3）系统互联要通过通信设施（硬件、软件）来实现。

（4）系统通过通信设施执行信息交换、资源共享、互操作和协作处理，实现多种应用要求。互操作和协作处理是计算机网络应用中较高层次的要求，需要有能够支持网络中异种计算机系统之间的进程通信，实现协同工作和应用集成的环境。

网络操作系统的研制开发是在原来的计算机操作系统的基础上，按照网络体系结构的各个协议标准开发的，以实现网络管理、通信、资源共享、系统安全和多种网络应用服务。

由于网络计算的出现和发展，现代操作系统的主要特征之一就是具有上网功能。因此，除了20世纪90年代初期Novell公司的NetWare等被称为网络操作系统之外，人们一般不再特指某个操作系统为网络操作系统。

1.3.6　分布式操作系统

粗看起来，分布式系统与计算机网络没有多大区别。分布式系统也可以定义为通过通信网络将物理上分布的具有自治功能的数据处理系统或计算机系统互联，实现信息交换和资源共享，协作完成任务的系统。但是，两者有以下明显的区别：

（1）计算机网络现在已经有了明确的通信网络协议体系结构及一系列协议族，计算机网络的开发都遵循协议；而各种分布式系统并没有标准的协议。当然，计算机网络也可认为是一种分布式系统。

（2）分布式系统要求一个统一的操作系统，实现系统操作的统一性。在分布式系统中，为了把数据处理系统的多个通用部件合并成为一个具有整体功能的系统，必须引入一个高级操作系统。各处理机有自己的专有操作系统，必须有一个策略使整个系统融为一体，这就是高级操作系统的任务，它可以有两种形式：一种形式是在每个处理机的专有操作系统之外独立存在，专有操作系统可以识别和调用它；另一种形式是在各处理机专有操作系统的基础上加以扩展。对于各个物理资源的管理，高级操作系统和各

专有操作系统之间不允许有明显的主从关系。

在计算机网络中，实现全网统一管理的网络管理系统已成为越来越重要的组成部分。

（3）系统的透明性。分布式操作系统负责整个分布式系统的资源分配和调度、任务划分、信息传输控制协调工作，并为用户提供统一的、标准的接口，用户通过这一接口进行操作和使用系统资源，至于在哪一台计算机上执行操作或使用哪一台计算机的资源则是系统的事，用户是不用知道的，也就是说系统对用户是透明的。但是，对于计算机网络，若一台计算机的用户希望使用另一台计算机的资源，则必须指明是哪一台计算机。

（4）分布式系统的基础是计算机网络。它和计算机网络一样具有模块性、并行性、自治性和通用性等特点，但它比计算机网络又有进一步的发展。分布式系统不仅是物理上松散耦合的系统，同时还是逻辑上紧密耦合的系统。分布式系统由于更强调分布式计算和处理，因此对于多机合作和系统重构、鲁棒性和容错能力有更高的要求，系统要有更短的响应时间、更高的吞吐量和更高的可靠性。

（5）分布式系统还处在研究阶段，目前还没有真正实用的系统；而计算机网络已经在各个领域得到广泛应用。

20 世纪 90 年代出现的网络计算已使分布式系统变得越来越现实。特别是 SUN 公司的 Java 语言和运行在各种通用操作系统之上的 Java 虚拟机和 Java OS 的出现，进一步加快了这一趋势。另外，软件构件技术的发展也加快了分布式操作系统的实现。

项目小结

操作系统是计算机系统中的一个系统软件，是一些程序模块的集合，它们能够以尽量有效、合理的方式组织和管理计算机的软硬件资源，合理地组织计算机的工作流程，控制程序的执行，并向用户提供各种服务功能，使用户能灵活，方便，有效地使用计算机，使计算机系统高效运行。

操作系统是配置在计算机硬件上的第一层软件，是对硬件系统的首次扩充。其主要作用是管理好这些设备，提高它们的利用率和系统的吞吐量，并为用户和应用程序提供一个简单的接口，便于用户使用。

随着计算机技术和软件技术的发展，已形成了各种类型的操作系统，以满足不同的应用要求。操作系统根据使用环境和对作业的处理方式可分为以下 6 个基本类型：批处理系统、分时系统、通用操作系统、个人计算机操作系统、网络操作系统、分布式操作系统。

课后练习

1. 操作系统应该包含哪些程序？

2. 操作系统哪个程序支持用户操作计算机？调用系统服务的指令是什么？为什么不带地址信息？操作系统怎么保证程序在 CPU 上轮流运行？

3. 主存和处理器这两个计算机资源应该以何种方式使用？

4. 试述多道程序设计技术的基本思想。为什么采用多道程序设计技术可以提高资源利用率？

5. 举一个现实生活当中的并发与共享的例子，重点说明人（处理器）如何在多个任务（程序）当中切换，多个任务如何使用同一个工具（资源）。

项目二 操作系统运行机制与用户界面

操作系统的主要功能就是管理处理器、主存、外部设备及文件，并提供支持程序并发与并行运行的机制。广义的操作系统还包含一些如文件列表等常用的实用程序，以命令或图符的形式通过命令解释器或程序管理器供用户使用。操作系统内核程序实现了资源管理并向上提供服务。本项目先讨论操作系统内核程序是如何执行的，然后讨论用户与操作系统的系统编程接口及用户操作界面。

知识目标

1. 理解中断和异常的概念、类型、处理流程以及应用场景，掌握中断和异常处理的关键技术和方法。
2. 了解操作系统的运行模式及其相关概念和技术，掌握用户态与内核态的切换机制。
3. 了解系统调用的概念、原理和应用场景，掌握常见系统调用的使用方法和性能优化技巧。
4. 掌握人机界面的相关知识和技能。

能力目标

1. 能够设计简单的用户界面。
2. 能够分析现有操作系统的用户界面，识别其优点和不足，并提出改进的建议。

素养目标

培养学生的创新精神，尝试设计出新颖、独特的用户界面，以提升用户体验，为在计算机领域的发展打下坚实的基础。

任务一　中断和异常

通常，主流操作系统提供的主要功能都是由内核程序实现的，处理器在运行上层程序时进入操作系统内核程序运行的重要途径就是通过中断或异常。当中断或异常发生时，运行用户模式程序的处理器会马上进入内核运行。

中断和异常是操作系统的重要概念。中断是与当前运行程序无关的一个外部事件，而异常是由正在运行的指令所产生的。

当初为了由操作系统前身的监督程序能够发现用户误编的死循环程序，首先引入了时钟中断，由硬件定时器发出，表示一个固定的时间间隔已到，让监督程序能够记录被中断程序的运行时间，看看程序运行累积时间是否还在事先指定时间内，如果超过指定时间，系统判断发生死循环，对用户程序进行结束处理。现代操作系统扩展了时钟中断的功用，利用时钟中断进行各种计时、启动进程调度程序、定时运行系统各种任务等。

I/O 中断的引入是因为外部设备控制器的功能变强，外部设备控制器可以独立控制外部设备进行 I/O，为了开发处理器和外部设备之间的并行操作，当处理器启动外部设备进行 I/O 操作后，外部设备就可以独立于处理器工作了，处理器可以去做与此次 I/O 操作不相关的其他事情，那么外部设备 I/O 操作完成后，还必须告诉处理器，让处理器继续运行 I/O 操作完成后的程序。外部设备控制器告诉操作系统本次 I/O 操作已经完成，必须通过中断机制。

异常表示处理器执行指令时本身出现算术溢出、零作除数、取数时发生奇偶错、访存指令越界，或执行了一条所谓“trap”的自陷指令等情况，这时也可以中断当前的执行流程，转到相应的错误处理程序或自陷处理程序。自陷指令（也称 trap 指令或访管指令）的出现，使得在用户模式下运行的程序可以调用内核程序。用户自编程序或系统实用程序等在处理器上执行时，如欲请求操作系统为其服务，可以安排执行一条自陷指令引起一次特殊的异常。可以说这个异常只是一种特殊的程序调用，特殊在于处理器态从“用户模式”变成了“内核模式”，另一个特殊点在于普通程序调用指令包含被调程序地址，而自陷指令无须给出地址，为了内核的安全，也只允许转到“固定陷入点”的程序执行。

最初，中断和异常并没有区分开，统称为中断。随着它们的发生原因和处理方式的差别越来越明显，才有了以后的中断和异常，因此必须注意中断这个词，在不同的历史阶段有不同的含义。

2.1.1　中断和异常的区别

下面来看看计算机系统会有一些什么样的中断和异常。通过对中断和异常进行如下分类，主要是想解释进入内核运行的可能情形，以及不同类型的中断和异常在处理方式上的差别。

目前流行的分类方法是，根据打断当前程序运行的原因把历史上混为一体的中断分为两类。

（1）中断（Interruption），也称“外中断”。指来自处理器正执行指令以外的事件发生，如外部设备发出的各种 I/O 中断，表示设备 I/O 处理已完成，希望处理器能够运行中断处理程序向设备发出下一个 I/O 请求，同时让完成 I/O 后的程序能后续运行；时钟中断，表示一个固定的时间间隔已到，让处理器运行处理计时程序、启动定时运行的任务等。这一类中断通常是与当前程序（进程）运行无关的事件。每个不同中断具有不同的中断优先级，表示事件的紧急程度。在处理高级中断时，低级中断可以被临时屏蔽。

（2）异常（Exception），也称“内中断”、例外或自陷。指源自处理器正执行指令产生的事件，如程序的非法操作码、地址越界、算术溢出、虚存系统的缺页及专门的自陷指令等。异常不能被屏蔽，一旦出现应立即处理。

下面列举的是更详细地打断处理器当前指令正常执行顺序的各种中断和异常事件。

（1）I/O 中断。这是用于反映外部设备工作状态（如打印机输出结束、磁盘传输结束）的中断。

（2）时钟中断。固定间隔由时钟部件发生的中断，这也是系统发生最多的中断，操作系统利用时钟中断处理各种计时，如进程时间片是否用完判定、I/O 超时判定等都是由时钟中断处理程序激活的。

（3）机器故障。它是机器发生错误（如机器校验错、电源故障、主存读数错等）时产生的异常，用于反映硬件故障，以便进入诊断程序。

（4）程序性异常。指程序执行错误，错误使用指令或错误访问数据引起的，如非法操作码、无效地址、算术溢出等。

（5）自陷指令。程序由于执行了“trap”指令（系统调用指令）而产生异常。这表示用户程序欲请求操作系统为其完成某项工作（如创建进程、读/写文件等）。

2.1.2　中断分级

中断的具体过程

在计算机系统中，不同的中断源可能在同一时刻发出中断信号，也可能前一中断还未处理完，紧接着又发生了新的中断。为了区分和不丢失每个中断信号，通常用一些固定的触发器来寄存它们，并规定其值为 1 时表示有中断信号，其值为 0 时表示无中断信号。这些寄存中断的触发器组成的寄存器称为“中断寄存器”。其中每个触发器称为一个“中断位”。为了控制方便，一般对中断寄存器的各位顺序编号（如从左至右顺序编号），并称此编号为“中断序号”。

外部中断信号是由不同外部设备产生的，可能在同一时刻由不同外部设备发出多个中断信号，所以存在谁先被响应和处理的优先次序问题。即级别不同的两个以上中断信号同时出现时，首先响应级别高的中断，而且级别高的中断可以打断级别低的中断的处理过程。通常，把中断享有的高、低不同的响应权利称为中断优先级。

而当前实际系统中，硬件提供可编程中断控制器。有多少中断级，每个中断应划在哪一级，可由操作系统设计者来设定。一般来说，高速设备的中断优先级高，慢速设备的中断优先级低。因为高速设备的中断被处理器优先响应，可以让处理器尽快地向高速设备发出下一个 I/O 请求，提高高速设备的利用率。但是也要考虑到特殊用户需求，如实时控制系统数据输入设备，处理速度不一定是最快的，但是为了实时控制却要将其中断级别设置为高。如某操作系统把中断级别分为如下几级。

（1）时钟中断的中断优先级为 6 级。

（2）磁盘中断的中断优先级为 5 级。

（3）终端等其他外部设备中断的中断优先级为 4 级。

对同一中断优先级内的若干中断源，按照该中断寄存器中从左至右的顺序来决定处理的先后次序。在多级中断系统中，处理器处理中断的轨迹会复杂些。当多级中断同时产生时，处理器按照优先级由高到低的顺序响应，如图 2-1 所示。当正在处理低级中断时，若出现了高级中断，则高级中断的处理立即打断低级中断的处理，如图 2-2 所示。

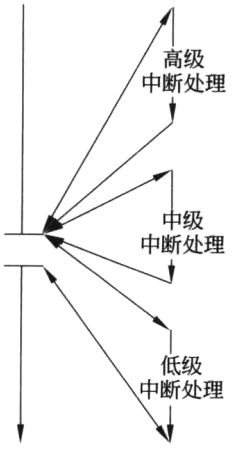


图 2-1　多级中断同时产生的处理器响应轨迹

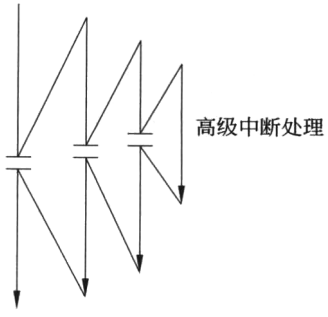


图 2-2　高级中断打断低级中断的处理器响应轨迹

所谓“中断屏蔽”，指禁止响应中断。现代计算机提供可以由程序设置中断屏蔽的方法。例如，在计算机所用的可编程中断控制器中，处理器可以执行特权指令来设置可编程中断控制器的屏蔽码，即使外设将已置上相应屏蔽码的中断位置上，处理器也不会响应该中断。硬件只是保存此次中断，以便将来在屏蔽解除时再处理。通常的设备中断、时钟中断等外部中断可以被暂时禁止响应。

由于操作系统可以通过特权指令来设置中断屏蔽寄存器，从而可由操作系统来决定中断的响应次序，达到由操作系统设计者决定中断级别的目的。如图 2-3 所示，说明了虽然有两个中断发生了，但屏蔽寄存器对应位为 1 的中断先被响应。

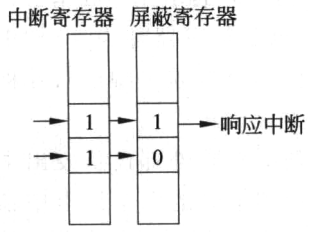


图 2-3　设置中断屏蔽位后影响中断响应的情形

在每次响应某个中断时的首要工作，就是重新设置屏蔽寄存器，确保在处理该中断时比该中断优先级低的及同等优先级的中断对应屏蔽位置上，而比该中断优先级高的中断则可以响应。

处理器优先级，指处理器当前正运行程序的中断响应级别。当处理器处于某个优先级时，只允许处理器响应比该优先级高的中断，而屏蔽低于或等于该优先级的中断。设置处理器优先级，即是设置屏蔽寄存器，确保处理器不会响应优先级小于或等于处理器优先级的中断。以前述某系统中断优先级设置为例，当处理器优先级为 5 时，除时钟中断外其他中断全部被屏蔽。

处理器执行指令产生异常，如执行非法指令、trap 指令时不能被屏蔽，必须马上响应处理。在异常处理程序运行时，是否屏蔽外部中断或屏蔽哪些外部中断会根据异常处理的需要设置，一般不需要对外部中断进行屏蔽。

任务二　中断/异常响应和处理

操作系统内核程序是不能让用户态程序以转移指令的方式转入的，只能通过中断/异常机制转入。处理器如何响应中断和异常，又如何转到中断和异常处理程序执行呢？

2.2.1　中断/异常响应

中断信号是外部设备或时钟部件发出的，在处理器的控制部件中需增设一个能检测中断的逻辑。该逻辑能够在每条机器指令执行周期内的最后时刻扫描中断寄存器，“查看”是否有中断信号。若无中断信号或中断被屏蔽，处理器继续执行程序的后续指令，否则处理器停止执行当前程序的后续指令，无条件地转入操作系统内核的对应中断处理程序，这一过程称为中断响应。

异常是在执行指令的时候，由指令本身的原因发生的，指令的实现逻辑发现异常发生则转入操作系统内核的对应异常处理程序。

在响应中断/异常时，一定会涉及如何保护中断点运行现场，如何找到中断/异常处理程序，中断/异常处理程序在什么处理器模式下运行的问题。下面详细讨论这些问题。

（1）断点和恢复点。处理器一旦响应中断，立即开始执行中断处理程序，当中断处理结束后，重新返回中断点的后续指令执行，如图 2-4 所示。故当中断发生时，处理器刚执行完的那条指令地址称为“断点”。一般地，断点应为中断的那一瞬间程序计数器（PC 寄存器）所指指令的前一条指令地址，即中断发生时正在执行的那一条指令的地址。中断时程序计数器所指的地址（即断点的逻辑后续指令）称为“恢复点”。

中断处理是一项短暂性的工作，逻辑上处理完后还要回到被中断的程序，从其恢复点继续运行。为了能实现正确的返回，在中断处理前后必须保存和恢复被中断的程序现场。

所谓现场信息，是指中断那一时刻确保程序继续运行的有关信息，如程序计数器、通用寄存器及一些与程序运行相关的特殊寄存器中的内容。现场的保护和恢复可由硬件、软件共同配合完成，现场信息通常保存在与被中断程序（进程）相关的栈中或保存在中断处理专用栈中。

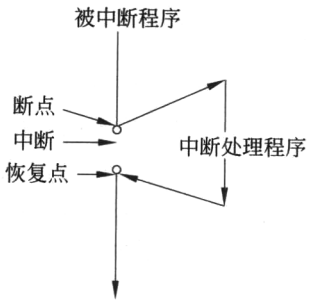


图 2-4　中断处理的处理器响应轨迹图

在异常发生时，返回点会因为不同的异常而有所区别。对于大部分由用户程序指令执行出错而引起的异常，操作系统的处理方式是结束发生异常的程序，因此也不会回到用户程序了。如果通过 trap 指令进行系统调用，则处理完成后返回 trap 指令的下一条指令。

（2）核心态和用户态。中断和异常的处理程序不是一般的程序，必须在一种特权状态下运行，因为这些程序需要执行一些特权指令，另外这些程序及涉及资源管理表格所占用的空间也不能被一般的用户程序直接访问。

处理器通常执行两类不同性质的程序：一类是用户自编程序或系统外层的应用程序或实用程序，另一类是内核程序。这两类程序的作用不同，后者是前者的服务提供者和控制者。显然，如果对两类程序给予同等“待遇”，则对系统的安全极为不利，内核程序享有外层程序所不能享有的某些特权。因此，将处理器的程序运行状态分为核心态和用户态，内核程序在核心态下运行，核心态下处理器允许执行所有的指令（包括特权指令）；而外层所有程序在用户态下运行，特权指令不允许在用户态下执行，用户态下执行的程序也不能直接用访存指令来访问内核程序空间。在程序状态字（也可称处理器状态字，简写为 PS 或 PSW）寄存器内设置一个标志位，根据其当前值为 1 或 0 来分别表示处理器所运行的程序处在核心态或用户态。特权指令或涉及系统空间的访存指令执行时需要判断这个标志位。

划分了核心态和用户态后，就严格区分了两类不同性质的程序，它们各自有严格区别的存储空间。用户程序中许多涉及共享资源的工作是通过“调用”内核程序代为完成的，当用户程序需要操作系统为之服务时，绝对不能通过通常的程序调用方式来调用操作系统的相应程序，而必须设法通过执行 trap 指令（系统调用）引起一次异常而转入内核程序。

核心态也称为内核态、管态、系统状态、监督方式，用户态也称为目态、用户状态或用户方式等。而且，在许多系统中为了进一步增加系统的安全性，进一步将核心态细分为若干不同的状态，同时也将操作系统分为若干层，不同层次的软件运行在不同的状态。例如，Intel x86 处理器运行状态有 1，1，2，3 环之分，不过 Intel x86 上运行的操作系统（如 Linux 和 Windows）只用到 0 环和 3 环，0 环表示核心态，3 环表示用户态。

（3）中断/异常向量及 PS 和 PC 寄存器。一般为系统中每个中断/异常编制一个相应的中断/异常处理程序，并把这些程序的入口地址放在特定的主存单元中。通常将这片存放中断/异常处理程序入口地址的主存单元称为中断/异常向量。

对不同的系统，中断/异常向量中的内容细节也不尽相同。中断/异常向量的每个单元中除存储中断/异常处理程序的入口地址外，还常用来保存处理器状态转换的信息。例如，中断/异常处理程序运行要用到新 PS 值和新 PC 值。

处理器的取指令部件根据 PC（程序计数器）寄存器的值到主存中取指令。

PS（程序状态字，也可称为处理器状态字）寄存器的值描述处理器的执行状态，主要包含处理器当前运行态、处理器优先级、屏蔽外中断等标志位。

PS 中的处理器优先级表示当前处理器所运行的中断处理程序所对应的中断级别，显然处理器优先级与中断优先级有对应关系。如果处理器优先级等于 4，则表示处理器正在处理中断优先级为 4 的中断，这时 4 级及 4 级以下优先级的中断都应该被屏蔽。

在中断/异常向量中，每个中断/异常占用连续的两个单元：一个单元存放中断/异常处理程序的地址

（对应的 PC 值）；另一个单元保存处理中断/异常时处理器应具有的状态（对应的 PS 值）。

中断/异常向量是中断系统中一个非常重要的数据结构。当响应中断/异常时，硬件先把当前 PS 和 PC 寄存器的值作为程序现场保存起来，然后从中断/异常向量的相应单元中取出新的 PS 和 PC 值，并装入 PS 和 PC 寄存器。处理器便根据新装入的 PC 值转去进行中断/异常处理。因为 PS 装入了新值，而且确定处理器状态为核心态，所以中断处理程序总是在 PS 状态域所表示的核心态下执行。

2.2.2 中断/异常处理

整个中断/异常从发现到处理完毕是由硬、软件相互配合协调完成的。大部分系统的中断/异常处理过程均类似。中断/异常处理一般包括保存现场、进入相应的中断/异常处理程序、可能重新选择程序（进程）运行、恢复现场等过程，如图 2-5 所示。

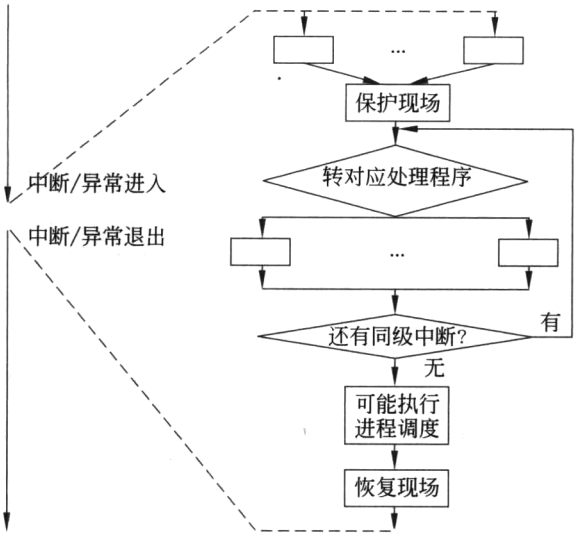


图 2-5 一般的中断/异常处理过程

【例 2.1】 下面以某计算机上的 UNIX 系统为例，介绍中断/异常处理过程中各部分的工作。

UNIX 的中断/异常向量如图 2-6 所示，每个中断/异常对应中断/异常向量中的两个单元，分别保存处理该中断/异常时的程序状态字（PS）和中断/异常处理程序的入口地址。

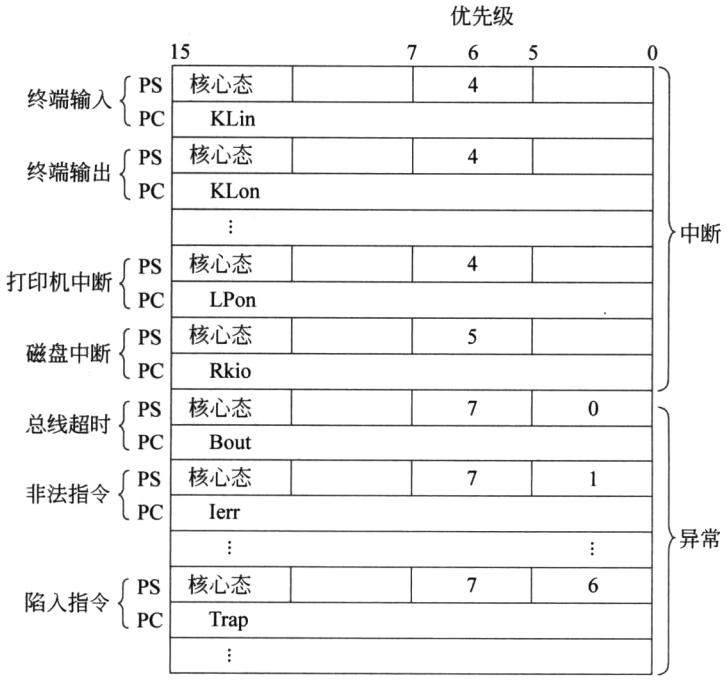


图 2-6 中断/异常向量示例

（1）中断/异常进入。中断/异常发生后，硬件执行如下的逻辑——交换 PS 和 PC 的值，具体步骤如下。

①硬件自动将 PS 和 PC 的值存入处理器中的暂存寄存器。

②根据发生的中断/异常号，硬件从对应的中断/异常向量单元中取出新的 PS 和 PC 值，分别装入 PS 和 PC。

③硬件将保留在内部寄存器中的原 PS 和 PC 值作为现场信息保存到与被中断程序相关的栈中，这里的栈用于保护原来运行程序的现场。

将 PS 和 PC 先存入处理器中的暂存寄存器的目的，是让（2）和（3）能够并行。硬件完成以上三步后，控制便根据中断/异常向量的 PC 值转入相应的处理程序。对于异常，在这个例子中，硬件最初将控制转入一个入口地址为 Trap 的处理程序，如图 2.6 所示。因此，为了使软件能区别不同的异常并给予不同的处理，在中断/异常向量相应的异常单元中，分别在对应的 PS 单元中的低 5 位存放一个不同的编号，该编号被称为异常类型号，如总线超时的类型号为 0。总之，对于不同的中断，将转入不同的中断入口程序。对于所有的异常则首先转入公共的入口程序。

一般的中断/异常处理过程均包括如下三个阶段：

①保存现场。

②根据原因调用相应的处理程序。③恢复现场。

在该例子中，系统专门设置了一个总控程序，负责这三个工作的转入。

对于中断/异常，尽管硬件通过中断/异常向量表项最初转入的处理程序入口各不相同，但在各个处理中，都先记住本次中断/异常号，转入总控程序进行进一步现场保存等公共工作后，再根据中断/异常号去调用各个不同的中断/异常处理程序。

总控程序完成如下工作：

①继续保存断点现场。

②根据中断/异常号转调中断/异常处理程序。

③恢复断点现场，返回。

（2）保存现场。在该例子中采用分散保存现场的方法，操作系统对每个程序（进程）分配一片“现场空间”，每当中断/异常发生时，便将现场保存在当前程序（进程）相关的现场空间内。现场区均组织成“栈”结构。当出现中断/异常时，将现场信息一条条地压进栈，恢复现场时再逐步退栈。

在多级中断系统中，高级中断能够打断低级中断的处理，待高级中断处理结束后，再返回低级中断处理。为了不丢失低级中断的现场信息，显然应该用栈结构保存现场。当然，子程序调用也需要栈来保存返回地址及子程序要使用的工作寄存器原内容。所以说，栈是程序运行不可缺少的数据结构。

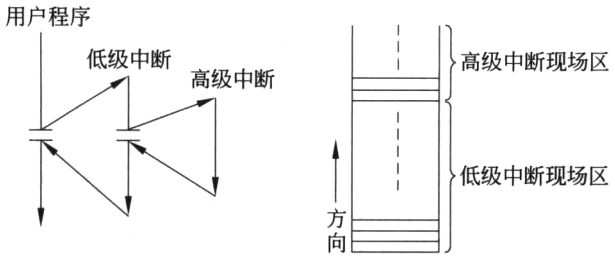


图 2-7　高级中断打断低级中断的处理过程

如图 2-7 所示，它描述了高级中断打断低级中断的处理过程。在最初响应和处理低级中断时，被中断的程序现场保留在栈的底部。当高级中断打断低级中断时，高级中断则将低级中断处理断点现场压入栈。当高级中断结束时，通过恢复现场撤除栈上的相应现场信息。紧接着低级中断便得到处理，当低级中断处理完成返回时，最后撤除栈上的相应现场信息。只要栈空间足够大，便能保证多级中断的嵌套处理。

UNIX 响应、处理一次中断/异常时的现场内容如图 2-8 所示。最初响应中断/异常时硬件将 PS 和 PC 的内容压在栈底，而后转入相应的中断/异常处理入口程序，将 r0 寄存器的内容压进栈，进入总控程序后，有必要继续保存剩余现场到栈中，在总控程序调用的中断/异常处理程序中再进一步保存用到的 r2～r5 寄存器。总而言之，在程序要用某些寄存器之前，寄存器原来的数据作为现场必须先保存到栈中。

（3）中断/异常处理程序。总控程序在统一保存好现场后，因为中断/异常原因不同，调用对应的中断/异常处理程序，同时将返回总控程序的地址保存在栈中。

I/O 中断是表示先前发送给设备控制器的 I/O 请求已经完成，I/O 中断的处理通常会向设备控制器发送下一个 I/O 请求，并使原来等待 I/O 结束的进程变为就绪，可进行下一步处理工作。时钟中断是最频繁

发生的中断，它是定时发生的，一旦发生时钟中断，时钟中断处理程序需要完成系统及刚被中断的进程的时间统计、软定时器计数减时操作等。

异常处理则需要区分是正常发生的操作系统的系统调用处理（这时运行操作系统的系统调用处理程序），还是产生了其他指令执行出错（如发生溢出等其他异常时，对产生异常的程序进行结束处理）。

（4）恢复现场。当中断/异常处理结束后，必须退出中断/异常。在多级中断系统中，当中断处理结束返回时，必须依据原先被中断的程序，完成不同的工作。

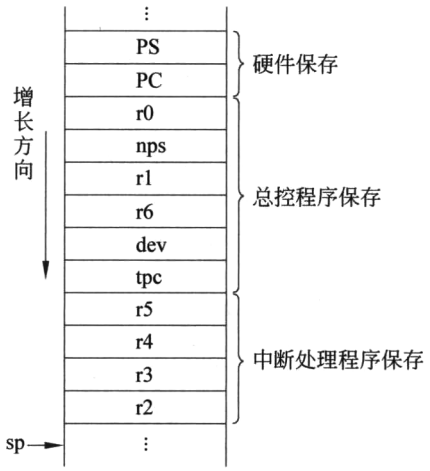


图 2-8　UNIX 响应、处理一次中断/异常时的现场内容

①如果此次是高级中断，并且被中断的程序是一个低级中断处理程序，则此次中断返回应返回到该低级中断处理程序。UNIX 判断处理器的先前状态若是核心态，则根据保存的现场恢复被中断的低级中断处理程序现场，具体恢复现场的动作如下。

a. 从栈中恢复除 PS、PC 之外的寄存器值。

b. 执行 rtt 指令，该指令自动将栈中的 PS 值、PC 值装入 PS 和 PC 寄存器中，从而退出此次中断。

②如果原来被中断的是用户态程序，则退出中断/异常以前应先考虑进行一次调度选择，即运行进程调度程序，以挑选更适合在当前情况下运行的新程序（进程）。这是因为，原来被中断的用户程序在此次中断/异常处理过程中，可能由于其要等待的事件没有发生而不具备继续运行的条件；也可能被降低了运行的优先权；还可能由于此次中断/异常的处理使得其他程序获得了比其更高的运行优先权。为了权衡系统内各个程序（进程）的运行机会，此时有必要进行一次调度选择。在进行了调度选择后，无论是挑选出另外一个新程序（进程），还是仍然选择原程序（进程）继续运行，都必须恢复所选程序的现场。

任务三　操作系统运行模式

操作系统的功能模块有哪些？什么时候、如何运行操作系统的功能程序？下面先介绍操作系统主要由哪些功能模块组成，然后说明典型的操作系统运行模式，弄清操作系统功能模块在什么模式下运行。

2.3.1　操作系统功能模块

操作系统通常都包含进程管理、存储管理、外部设备管理、文件管理等主要功能模块，还需要提供支持用户使用计算机的命令解释程序或窗口界面程序。以 UNIX/Linux 为例，大部分功能模块以内核程序方式实现，即在核心态下运行，但是有些系统功能以用户程序方式实现，如命令解释程序就是在用户态下运行的。

【例 2.2】　以 UNIX/Linux 为例，介绍操作系统的主要功能模块如下。

（1）系统的初始化模块。当系统加电后，计算机 ROM 程序按约定将操作系统内核程序目标码加载入主存，并执行系统初始化程序。系统初始化模块入口只在系统启动时进入一次，在以后系统正常运行期间不再进入。首先，系统初始化程序初始化系统数据区及硬件，如初始化空闲物理页帧、系统的各种缓冲池、中断控制器、中断/异常向量表、外部设备等。在初始化各种系统表格后，还要创建 1 号进程。让 1 号进程运行 INIT 程序，创建 tty 终端进程（tty 终端进程会检验从终端输入的用户登录信息，然后运行命令解释程序，从终端读取并解释执行用户命令）。初始化模块还会创建许多只运行内核程序的线程，运行

一些需要定期运行的系统任务，如内存回收。当初始化完成后，代表用户的 tty 终端进程已经存在，操作系统的其他功能模块可以以中断/异常方式随时被用户程序调用执行。

（2）进程管理模块。进程是操作系统组织用户程序在计算机上运行的机制。进程管理模块包含有关进程的系统调用处理，如进程/线程创建和结束、进程通信等，也包含对处理器的分时使用程序，如进程调度和进程切换。这些程序都涉及对管理进程用的进程控制块等数据结构的操作，但它们会在不同的时机被调用。如进程创建、进程通信及进程同步过程往往由用户程序发出系统调用时执行，而进程调度和进程切换则是要重新分配处理器时运行，如当中断/异常处理完成要返回用户态程序时调用执行。

（3）存储管理模块。存储管理与进程管理关系密切，因为进程运行必须占有主存空间，因此把进程空间分配程序也划分到存储管理模块中，如主存管理模块、进程空间分配、支持虚空间的进程页面内外存之间交换等。进程空间分配在进程创建时被执行，进程页面交换是希望多个进程轮流占用主存并运行（详见存储管理内容），进程页面交换程序通常作为操作系统独立任务被定时地启动运行。

（4）I/O 设备管理模块。I/O 设备管理模块包含设备访问接口程序、数据缓冲管理模块、各种驱动程序用公共程序、各种设备驱动程序和设备中断处理程序等，实现对设备的管理，并为用户提供 I/O 功能。

（5）文件管理模块。文件管理模块包含文件访问接口程序、文件系统目录结构管理程序、文件数据缓冲管理模块、辅存空间管理程序等，实现对外部存储设备的管理，并提供用户对文件的使用功能。

2.3.2　操作系统运行模式

操作系统的这些功能程序如何运行呢？在 UNIX/Linux 实现中，上述模块都是在核心态下运行的，主要是在中断/异常发生时嵌入用户进程中运行。

当前主流操作系统（如 UNIX/Linux、Windows）都采用嵌入用户进程运行模式，但是微内核模式也是值得深入研究的模式。

（1）内核嵌入用户进程运行模式。这种模式可以理解成内核程序当作函数被调用运行。不同于如图 2-9（a）所示的内核作为独立运行单位的模式，如图 2-9（b）所示，操作系统服务程序（各功能模块）、中断处理程序在执行 trap 指令或中断时，利用进程的核心栈空间，运行于进程中。所谓操作系统运行于进程，只是利用了属于该进程的核心栈。

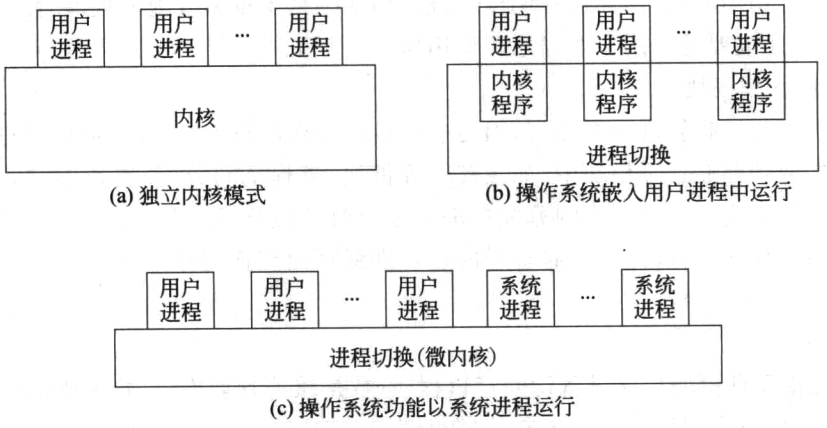


图 2-9　操作系统的运行模式

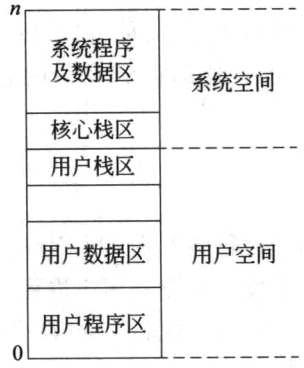


图 2-10　系统空间描述

操作系统空间独立于进程的用户空间，而且操作系统空间地址不与进程用户空间地址重叠，各占一片连续地址空间的高部与低部，如图 2-10 所示，如实用的 Linux 或 Windows 的每个进程都有一个自己的用户空间，逻辑编址都是 0~k，但是系统空间全系统只有一个，编址为 k+1~n。

当运行用户程序的处理器执行 trap 指令或响应中断时，处理器转到核心态下运行，控制转移给内核程序，用户程序的现场被保护起来，启用刚被打断进程的核心栈作为以后程序执行、函数调用的工作用栈。注意，这时只是处理器状态的转换，程序还是认为在当前的用户进程中执行，并没有发生进程切换，

而只是在同一进程内的处理器运行态的切换。所以说这种模式把调用内核程序（如系统调用处理程序、中断处理程序）运行，当作一种特殊的函数调用。特殊在程序状态字 PS 会被重置，也无须在 trap 指令中提供函数地址，而是按约定地址转移，保证了系统安全。

在内核完成它的工作后，如果决定继续原来被中断的用户程序的运行，则又将处理器的状态恢复到用户态，并恢复刚被打断的现场运行。在这种情况下，一个用户进程在运行用户程序的中途可以运行内核程序，运行完后又接着运行原用户程序。在运行完内核程序后如果有高优先级的进程可运行，则可进行进程调度切换。如果决定进行进程切换，则将当前进程置成非运行状态，把选定要运行的进程置成运行状态，再恢复被选进程的现场。逻辑上，进程调度切换程序不属于任一进程。

（2）微内核运行模式。如图 2-9（c）所示，将原来由核心态实现的大部分操作系统功能转由一些用户态运行的进程来实现，系统调用转接代码、进程调度切换代码和中断处理程序还是在核心态下执行。采用该模式，可以使核心态下运行的程序非常少，微内核操作系统 Mach 3.0 就采用这种运行模式。虽然它在模块化、层次化方面有可取之处，但是因为用户程序和操作系统功能程序运行在各自独立的进程中，它们之间在通信、合作时开销大，损失了系统的性能，因此此种结构的操作系统并没有被广泛接纳。华为鸿蒙操作系统采用了微内核结构，但是利用硬件支持改进了内核与进程之间的通信开销。

大部分现代操作系统以如图 2-9（b）所示的模式运行，操作系统内核功能以外的其他系统功能，是由独立的进程实现的。如 UNIX 的 1 号进程，实现对终端用户进程的创建，还有一些如安全检查、FTP、WWW 网络服务等系统功能都采用独立的进程来实现，这些运行系统外围功能的进程又称为系统守护进程。

任务四　系统调用

在计算机系统中运行着两类程序：操作系统和应用程序。为了保证操作系统不被应用程序有意或无意地破坏，CPU 设置了两种状态：核态和用户态。

核态也称为系统态、核心态、管态，操作系统在核态运行。

用户态也称为目态，应用程序只能在用户态运行。

在实际运行过程中，CPU 会在系统态和用户态间切换。一方面，系统提供了保护机制，防止应用程序直接调用操作系统的服务，提高了系统的安全性。另一方面，应用程序又必须取得操作系统所提供的服务，否则，应用程序就几乎无法做任何有价值的事情，甚至无法运行。为此，操作系统提供了系统调用，使应用程序可以通过系统调用的方法，间接调用操作系统的相关函数，取得相应的服务。

2.4.1　系统调用概念

系统调用（System Service Call，System Call）是操作系统内核为应用程序提供的服务，是应用程序与操作系统之间的接口。

系统调用一般涉及核心资源或硬件的操作，运行于核态。每个系统调用具有唯一的编号。调用系统调用的过程会产生中断，这种中断是自愿中断，既是软件中断，也是内部中断。图 2-11 展示了系统调用的基本概念。用户程序调用 sys_ foo（）函数，但是该函数却是在内核中真正实现的。当然，sys_ foo（）函数能从用户空间穿越到内核空间，显然该过程利用了中断机制，产生了中断。

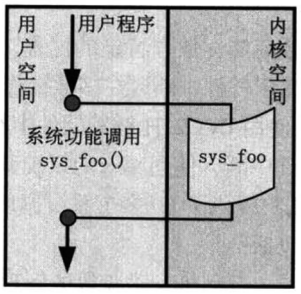


图 2-11　系统调用的基本概念

2.4.2　系统调用工作原理

在内核中预先设计了一系列系统调用，每个系统调用都有唯一的编号，以区别彼此。应用程序通过

形如 SVCN 的访管指令调用第 N 号调用，调用过程中发生了中断。图 2-12 展示了调用第 X 号系统调用产生中断的过程。

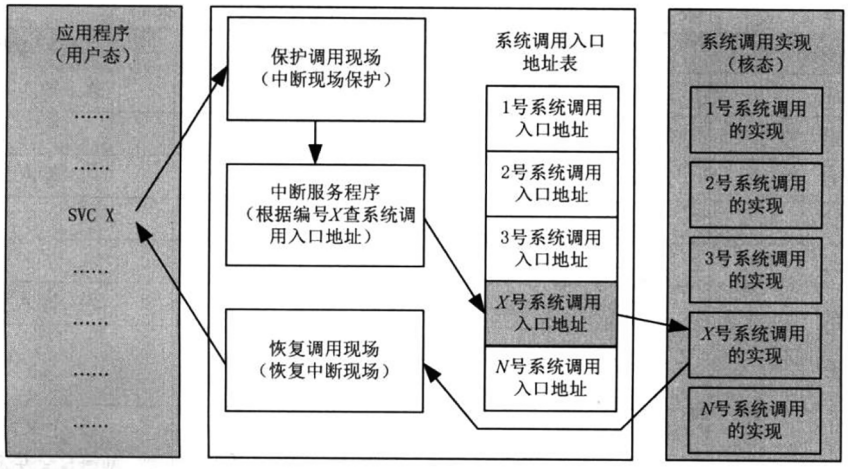


图 2-12　第 X 号系统调用的调用和工作过程

（1）应用程序使用 SVC X 指令准备调用第 X 号系统调用。

（2）内核识别并响应 SVC X 指令引起的中断。

（3）系统执行保护现场。

（4）根据系统调用编号 X 在系统调用入口地址表中查找相应的入口地址。

（5）转入相应的系统调用函数。

（6）恢复现场数据。

（7）返回应用程序。

操作系统维护系统调用的入口地址表，并维护系统调用的编号与入口地址之间的对应关系。

不同类型、不同版本的操作系统，系统调用的命令格式、数目和所完成的功能，以及每条系统调用的编号等都不尽相同。但是系统调用的步骤和基本工作原理是相同的。

在使用系统调用的时候，除了提供系统调用的编号之外，很多时候还需要更多的参数才能实现。比如要在屏幕上显示字符串，至少需要提供以下参数：

（1）系统调用的编号。

（2）待显示字符串的地址。

（3）字符串的长度。

（4）屏幕设备对应的文件句柄。

在使用系统调用之前，这些参数应当存放在特定的位置，以便进入中断服务程序之后，中断服务程序能够获取它们。系统调用属于软件中断，在多数操作系统中都会使用 INT 指令触发软件中断。譬如，在 DOS 操作系统中，使用 INT 21H 触发软件中断，使用相应的系统调用；在 Linux 操作系统中，使用 INT 80H 触发软件中断，使用相应的系统调用。

DOS 系统调用（INT 21H）为用户提供了 80 多个系统调用的服务程序，可在汇编语言程序中直接调用。这些子程序的主要功能包括：

（1）设备管理（如键盘、显示器、打印机、磁盘等的管理）。

（2）文件管理和目录操作。

（3）内存管理。

（4）时间和日期管理。

表 2-1 展示了常用 DOS 系统调用的编号，参数和基本功能。一般在 AH 寄存器中提供系统调用的编号，其余的参数在 AL、BX、CX、DX 等寄存器中存放，具体参数和放置方式与系统调用本身有关，可以

参照技术手册。

表 2-1 常用 DOS 系统调用（INT 21H）一览表

AH	功能	调用参数	返回参数
01	键盘输入且回显		AL=输入字符
02	显示输出	DL-输出字符	
03	异步通信输入		AL=输入数据
04	异步通信输出	DL=输出数据	
05	打印机输出	DL-输出字符	
07	键盘输入无回显		AL=输入字符
09	显示字符串	DS：DX=串地址 ´$ 结束字符串	
0F	打开文件	DS：DX=FCB 首地址	AL=00 文件找到 AL=FF 文件未找到
10	关闭文件	DS：DX=FCB 首地址	AL=00 目录修改成功 AL=FF 目录中未找到文件
13	删除文件	DS：DX=FCB 首地址	AL=00 删除成功 AL=FF 未找到
3D	打开文件	DS：DX=ASCII 串地址 AL=0 读 AL=1 写 AL=3 读/写	成功：AX=文件代号 错误：AX=错误码
3E	关闭文件	BX=文件号	失败：AX=错误码
3F	读文件或设备	DS：DX=数据缓冲区地址 BX=文件号 CX=读取的字节数	读成功： AX=实际读入的字节数 AX=0 已到文件尾 读出错：AX=错误码
40	写文件或设备	DS：DX=数据缓冲区地址 BX=文件号 CX=写入的字节数	写成功： AX=实际写入的字节数 写出错：AX=错误码

DOS 操作系统下调用系统调用的过程如下。

（1）给出系统调用的编号。系统调用编号写入 AH 寄存器中。

（2）给出相关的入口参数。参数写入相关的寄存器中。

（3）TNT 21H。

程序员给出以上三方面信息，DOS 就可根据所给信息自动转入相关系统调用执行。

【例 2.3】 如代码 2-1 所示利用 DOS 的 INT 21H 系统调用向屏幕输出一个字符串。

【例 2.4】 在代码 2-2 的键盘交互式程序中，等待用户按下数字键 1，2，3，程序转入相应的服务子程序，若按其他键则会继续等待。当 AH=1 时，执行 INT 21H 系统调用后，出现提示输入的光标，等待用户从键盘输入一个字符并保存其 ASIIC 码到 AL 寄存器中。

表 2-2 显示了 Linux 中支持的系统调用（INT 80H），在 EAX 寄存器中提供系统调用的编号，其余的参数在 BX、CX、DX 等寄存器中存放，具体的使用方式参照技术手册。

```
DATA SEGMENT
  STR1 DB 'HOW DO YOU DO?',0DH,0AH,'$'
DATA ENDS

CODE SEGMENT
   ASSUME CS:CODE,DS:DATA
START:
       MOV AX,DATA
       MOV DS,AX
       MOV DX,OFFSET STR1 ;字符串首偏移地址放到DX中
       MOV AH,9
       INT 21H;输出字符串

       MOV AH,4CH
       INT 21H
CODE ENDS
   END START
```

代码 2-1　DOS 中断输出字符串的例子

```
InputKey:
       MOV  AH, 1
       INT  21H      ;等待输入一个字符
       CMP  AL, '1'
       JE   ONE      ;如果输入'1'则跳到ONE处执行
       CMP  AL, '2'
       JE   TWO      ;如果输入'2'则跳到TWO处执行
       CMP  AL, '3'
       JE   THREE    ;如果输入'3'则跳到THREE处执行
       JMP  InputKey ;如果不是1,2,3，则继续要求输入
ONE:   ......
TWO:   ......
THREE: ......
```

代码 2-2　DOS 中断键盘交互式程序例子

表 2-2　Linux 部分系统调用

编号	名称	功能	调用参数
00	setup	安装根文件系统	EBX=硬盘参数表地址
01	exit	退出进程	EBX=退出码
02	fork	创建进程	
03	read	读文件	EBX=文件描述符，ECX=缓冲区首址，EDX=字节数
04	write	写文件	EBX=文件描述符，ECX=缓冲区首址，EDX=字节数
05	open	打开文件	EBX=文件名，ECX=打开标志，EDX=属性
06	close	关闭文件	EBX=文件描述符
07	waitpid	等待进程终止	EBX=进程 ID，ECX=返回状态地址，EDX=选项
0b	execve	执行程序	EBX=文件名，ECX=argv 指针，EDX=envp 指针
0c	chdir	更改当前目录	EBX=目录名
0d	time	获取当前时间	EBX=时区
0f	chmod	修改文件属性	EBX=文件名，ECX=文件属性
14	getpid	取进程 ID	
17	setuid	设置进程用户 ID	EBX=用户 ID
18	getuid	获取进程用户 ID	

【例 2.5】　代码 2-3 直接使用 Linux 的 write 系统调用。write 系统调用（内部的名称是 sys_ write）的编号是 4，编号被存在 EAX 寄存器中，其余的各个参数也被存放到相应的寄存器中：

EBX：文件描述符

ECX：指向要写入的字符串的指针

EDX：要写入的字符串长度

需要说明的是，Linux 中 0 表示标准输入，一般是键盘。1 表示标准输出，一般是终端屏幕。

有的系统允许系统调用直接为高级语言程序所用，这时系统调用通常被封装为库函数。高级语言程序在调用库函数时会通过库函数间接地使用汇编指令来请求系统调用，这就是系统调用的隐式调用方式，或者称为间接使用方式。与之对应的，直接使用汇编指令请求系统调用的方式称为显式调用。汇编指令形式的系统调用常常会列在汇编语言的开发手册中。

```
;源文件名: hello.asm
[section .data]
strHello  DB  "Hello World!"
strLen    EQU $ - strHello

[section .text]
global _start

_start:
    MOV EDX, strLen
    MOV ECX, strHello
    MOV EBX, 1
    MOV EAX, 4   ;sys_write
    INT 0x80
    MOV EAX, 1   ;sys_ext
    INT 0x80

;makefile
;All:
;   NASM -f elf -o hello.o hello.asm
;   ld -o hello.out hello.o
```

代码 2-3　使用 write 系统调用显示 Hello World!

2.4.3　Linux 系统调用机制

在 Linux 操作系统下，系统调用（通常称为 syscalls）接口是内核与上层应用程序进行交互通信的唯

一接口。用户程序通过直接或间接（通过库函数）调用中断 INT 0x80，并在 EAX 寄存器中指定系统调用功能号，即可使用内核资源。不过通常应用程序都是使用具有标准接口定义的 C 函数库中的函数间接地使用内核的系统调用。处理系统调用 INT 0x80 中断的是文件 kernel/system_ call. s 中的 system_ call 函数。

在 Linux 内核中，每个系统调用都具有唯一的功能号。这些功能号被定义在文件 include/unistd. h 中。例如，read 系统调用的功能号是 3，定义为符号_ _ NR_ read。这些系统调用功能号实际上对应 include/linux/sys. h 中定义的系统调用处理函数指针数组表 sys_ call_ table [ ] 中各项的索引值。因此，read 系统调用的处理函数指针就位于该数组的第 3 项处。

为了方便使用系统调用，系统开发人员通常会将系统调用封装在标准库中。内核源代码在 include/unistd. h 文件中定义了宏函数_ syscalln ( )，其中 n 代表携带的参数个数，可以是 0~3。封装系统调用的库函数通常使用_ syscalln ( ) 宏展开进行定义。例如对于 read 系统调用，带 3 个参数，其定义是：

```
#define _ _ LIBRARY_ _
#include   <unistd. h>
_ syscall3 (int, read, int, fd, char * , buf, int, n)
```

其中，第 1 个参数对应系统调用返回值的类型，第 2 个参数是系统调用的名称，随后是系统调用参数的类型和名称。这个宏会被扩展成包含内嵌汇编语句的 C 函数，如代码 2-4 所示。

```
1  int read(int fd, char *buf, int n)
2  {
3      long_res;
4      __asm__ volatile (
5          "int$0x80"
6          :"=a" (__res)
7          :"0" (__NR_read), "b" ((long)(fd)), "c" ((long)(buf)),
8          "d" ((long)(n)));
9      if (__res>=0)
10         return int __res;
11     errno=- res;
12     return -1;
13 }
```

代码 2-4　read 函数的展开

可以看出，这个宏经过展开就是一个封装 read 系统调用的库函数的实现。宏中使用了嵌入汇编语句，以功能号_ _ NR_ read3 作为参数执行系统调用对应的 int 0x80 指令。

当进入内核中 0x80 的号中断处理程序 kernel/system_ call. s 后，system_ call 的代码会首先检查 EAX 中的系统调用功能号是否在有效系统调用号范围内，然后根据 sys_ call_ table ( ) 函数指针表调用相应的系统调用处理函数。

```
call sys_ call_ table ( ,%eax, 4)
```

read 系统调用的功能号是 3，即 read 的系统调用处理函数指针在 sys_ call_ table ( ) 中的索引为 3，保存在 EAX 寄存器中。通过执行上面的汇编语句后，内核会跳转执行 read 系统调用的处理程序。

根据以上分析，Linux 系统调用的工作机制可以概括为图 3-10 所示的流程，具体包括 5 步：

（1）应用程序调用封装系统调用的库函数。

（2）库函数展开为内含 INT 0x80 指令和系统调用编号的汇编指令，调用相应的系统调用。

（3）进入 INT 0x80 的中断处理函数，并调用相应的系统调用函数。

（4）系统调用函数完成用户请求的服务。

（5）返回应用程序。

2. 4. 4　Linux 系统调用实现

本任务以 Linux 0. 11 内核为例，分析 Linux 系统调用的代码实现，包括 0x80 中断初始化、系统调用的公共入口、系统调用处理函数指针表和典型的系统调用处理函数。

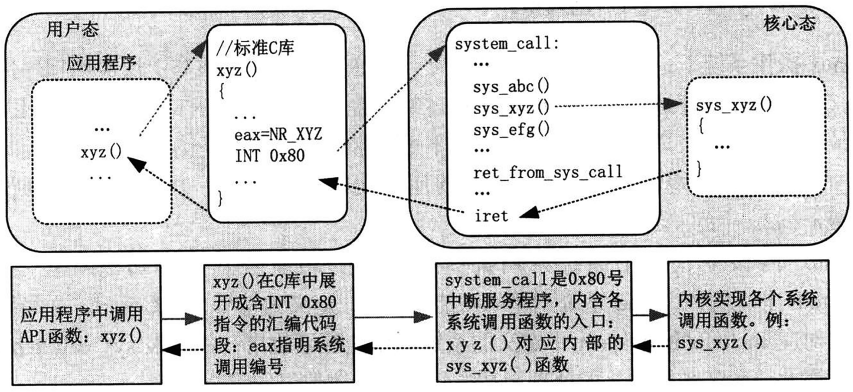


图 2-13　Linux 系统调用的工作机制

（1）0x80 中断初始化。应用程序使用系统调用会触发 0x80 中断，0x80 中断对应的中断服务程序 system_ call（）的入口地址放在系统的中断描述符表（Interrupt Descriptor Table，IDT）中。IDT 是 CPU 保护模式中的重要概念，关于保护模式请参考第 8 章的相关内容。Linux 系统初始化时，IDT 由/kernel/sched. c 中的 sched_ init 函数完成初始化。在初始化函数中，将 system_ call（）设置为 0x80 号中断的处理程序，如代码 2-5 所示。

```
1  void sched_init(void)
2  {
3      ......
4      set_intr_gate(0x20,&timer_interrupt);
5      ......
6      set_system_gate(0x80,&system_call);
7  }
```

代码 2-5　初始化 sched_ init 函数

经过初始化以后，每当执行 INT 0X80 指令时，产生一个中断使系统陷入内核空间并执行 0x80 号中断处理函数，即系统调用处理函数 system_ call（）。可见，应用程序通过使用 INT 0x80，就可以间接使用内核资源。

（2）系统调用公共入口 system_ call（）是中断服务程序，也是所有系统调用的公共入口。system_ call（）函数的主要功能包括保护系统调用的中断现场、进行参数正确性检查、根据系统调用功能号确定正确的系统调用处理函数、执行系统调用处理函数、执行 ret_ from_ sys_ call（）函数完成返回用户空间前的最后检查、从堆栈中恢复现场并执行 iret 指令返回系统调用的断点。代码 2-6 展示了 system_ call（）的关键代码。

```
1  system_call:
2      cmpl $nr_system_calls-1,%eax #系统调用编号若超范围则出错
3      ja bad_sys_call
4      push %ds        # 保存原段寄存器值。
5      push %es
6      push %fs
7      pushl %edx      # ebx,ecx,edx 中存放有C函数的调用参数。
8      pushl %ecx      # push %ebx,%ecx,%edx as parameters
9      pushl %ebx      #       to the system call
10     ......
11     call sys_call_table(,%eax,4)
12     ......
13     #从系统调用的C函数返回
14 ret_from_sys_call:
15     ......
16     popl %eax
17     popl %ebx
18     popl %ecx
19     popl %edx
20     pop %fs
21     pop %es
22     pop %ds
23     iret
```

代码 2-6　system_ call（）的关键代码

第 11 行的语句使用 call 指令，调用指定的系统调用，其中 eax 就是系统调用的编号。call 指令将调用的目标函数地址是：

目标调用地址=sys_ call_ table+%eax * 4

其中，sys_ call_ table［］是上一节中提及的系统调用处理函数指针表，里面按序包含有所有系统调用处理函数的指针，该表按 4 字节对齐。

（3）系统调用处理函数指针表。系统调用处理函数指针表 sys_ call_ table［］定义在/include/linux/sys. h 中。在该文件中，首先使用 extem 关键字声明所有系统调用处理函数在外部的定义，目的是在定义系统调用处理函数指针表时可以引用这些函数名。然后该文件定义了系统调用处理函数指针表 sys_ call_ table［］，数组元素即声明的函数名，并且数组元素是严格按照系统调用功能号排列的，目的是使其可以使用给定的系统调用功能号在 sys_ call_ table［］中找到对应的系统调用处理函数。代码 2-7 展示了 sys_ call_ table

```
1  extern int sys_setup();  // 系统启动初始化设置
2  extern int sys_exit ();  // 程序退出
3  extern int sys_fork ();  // 创建进程
4  extern int sys_read ();  // 读文件
5  extern int sys_write();  // 写文件
6  ......
7  extern int sys_chdir();  // 更改当前目录
8  ......
9  extern int sys_getpid(); // 取进程ID
10 ......
11 extern int sys_kill ();  // 终止一个进程
12 //系统调用的入口地址表
13 fn_ptr sys_call_table[] =
14 {
15  sys_setup,
16  sys_exit,
17  sys_fork,
18  sys_read,
19  sys_write,
20  ...
21  sys_chdir,
22  ...
23  sys_getpid
24  ...
25 };
```

代码 2-7　系统调用处理函数指针表

[ ] 的部分代码。

（4）系统调用处理函数。系统调用入口函数会在系统调用处理函数指针表 sys_ call_ table [ ] 中找到相应的处理程序，并跳转执行该程序以完成用户请求的服务。

以 write 系统调用为例，简单介绍该系统调用处理函数的工作内容，不同系统调用的处理函数各有不同。write 系统调用的功能号是 4，系统调用处理函数为 sys_ write ( )，实现在 read_ write. c 文件中，其关键代码如代码 2-8 所示。

```
int sys_write(unsigned int fd,char * buf,int count)
{
    struct file * file;
    struct m_inode * inode;

    //异常错误处理
    if (fd>=NR_OPEN || count <0 || !(file=current->filp[fd]))
        return -EINVAL;
    if (!count)
        return 0;

    inode=file->f_inode;
    if (inode->i_pipe)
        return (file->f_mode&2)?
    write_pipe(inode,buf,count):-EIO;

    if (S_ISCHR(inode->i_mode))
        return rw_char(WRITE,inode->i_zone[0],buf,count,&file->f_pos);

    if (S_ISBLK(inode->i_mode))
        return block_write(inode->i_zone[0],&file->f_pos,buf,count);

    if (S_ISREG(inode->i_mode))
        return file_write(inode,file,buf,count);
    printk("(Write)inode->i_mode=%06o\n\r",inode->i_mode);
    return -EINVAL;
}
```

代码 2-8　函数 sys_ write 的关键代码

write 系统调用的功能是把用户缓冲区的内容写入指定文件中，其对应的处理函数 sys_ write ( ) 仅仅区分写入的目标文件是什么类型：字符设备文件、块设备文件、常规文件还是管道文件。不同类型的文件操作将交由对应的设备驱动程序去实现，完成用户请求的服务后，将执行结果返回给调用程序。

不同版本的 Linux 内核，其系统调用的具体实现细节和效率是不同的，但是基本流程大致相同。针对具体版本的 Linux，若要增加或修改其系统调用，需要了解这些差异。

任务五　人机界面

除了系统调用这个编程接口，操作系统还需要给用户一个操作界面。要实现用户上机的目的，需要运行一系列相关程序，要让计算机去运行哪个程序（系统已有程序或用户自编程序）必须由用户说明，为此必须借助语言表达出来，这就是作业控制语言或命令语言。

用户使用语言指示计算机要做的事情，对用户的记忆力要求很高，用户要知道语言的语法语义，如格式稍有差错都不能成功。另外，用户要运行系统实用程序，也必须熟知计算机有哪些实用程序，并通过作业控制语句或输入命令告知系统，否则系统不知道用户要运行什么。为减少用户的记忆负担，现代操作系统都提供图形用户界面，用图符提示用户该系统有哪些实用程序，用对话框提示用户输入实用程序所需的参数。

2.5.1　命令语言

命令语言是用于人机通信的工具。该语言的性质很大程度上与操作系统的类型有关。分时操作系统与批处理操作系统相比，工作方式有很大不同。因此这两种系统使用的通信语言也不相同。

随着批处理系统的出现，产生了控制作业运行的作业控制语言。用户用作业控制语言预先写好作业控制说明书，将作业控制说明书和作业程序、数据一起提交给计算机，操作系统按作业控制说明书的控制语句来执行作业，以达到按照用户意图控制作业运行的目的。在批处理作业执行过程中，缺少用户与系统之间动态交互的能力，用户一旦向系统提交一道作业后，就按照已写好的控制说明书对该作业的执行过程进行控制，无法灵活应对。故用户和系统间基本上处于一种脱机的状态。这种通信语言的功能较强，除含有启动系统实用程序（如编译器）和用户自编程序的功能外，还包含控制转移语句，具有可编程能力，如利用条件转移语句，在判断编译器运行出错时，绕过运行装配器而直接结束作业。

批处理系统中设置了一个作业控制语言解释程序，它顺序地读取并解释执行作业控制说明书中的语句。根据语句的含义，或者直接“启动”实用程序。如果是简单实现的语句，则由解释程序直接处理以完成语句所指定的动作。

在分时系统中，用户可以直接使用终端命令频繁地与系统进行交互。分时系统使用的通信语言，既可以实现实用程序或用户自编程序的启动，也可以包含控制转移语句。与批处理系统不同的是，分时系统中提供的命令解释程序除了可以从指定文件读取控制说明书中的语句外，也可以从用户终端读取用户联机输入的命令。

为支持命令语言的解释执行，系统设置了一个命令解释程序（Command Interpreter）来负责解释执行用户当前输入的命令。在 UNIX/Linux 操作系统实现中，命令解释程序属于操作系统内核之外，它运行于用户态下，作为一个进程来运行。UNIX/Linux 的 1 号进程会为每个用户终端创建一个进程，用于运行 shell 命令解释程序，该程序不断地读取它所控制的终端发来的命令。

当用户在终端上输入一条命令时，命令解释程序要做如下的工作。

（1）通过 read 系统调用从终端设备读取命令，判断命令的合法性。

（2）识别命令，如果是简单命令则处理命令（可能向操作系统发出系统调用），然后继续读取下一条命令。

（3）如果是不认识的命令关键字，则在约定目录下查找与命令关键字同名的执行文件，创建子进程去执行“执行文件”程序，等待子进程结束后转而继续读取下一条命令。

当命令执行完成后，命令解释程序在终端上显示提示符，允许用户输入新的命令。终端命令的一般形式如下：

```
Command argl arg2 . . . argn
```

其中，Command 是命令关键字，arg1，arg2，…，argn 是执行该命令的参数。

终端命令一般都是串行执行的，也可以并行执行。即当用户输入的一条命令处理完后，系统发出新的提示符，用户可继续输入下一条命令。若执行一条命令需要较长的处理时间而用户不需要等待它的结果（即与后续命令的执行无关），就可以在该命令的末尾加上一个“开关”（以 UNIX/Linux 为例，是在终端命令末尾加上一个符号“&”），将这条命令作为后台命令处理。在处理后台命令时，用户可以接着输入下一条命令，系统可同时对前后两条命令做并行处理。实现后台执行只需要由命令解释程序创建子进程来执行“执行文件”程序，且命令解释程序不等待子进程结束即可继续读取下一条命令。

当然，命令解释程序也可以从文件中读取用户先前输入的命令脚本程序，且命令脚本程序可以包含执行语句及控制语句。控制语句包含循环语句、条件转移语句等，这一类语句由命令解释程序直接处理，它一般不在交互式输入命令时使用，而在用户预先编写的命令脚本程序文件中使用。

系统为用户提供了大量的实用程序，它们都可以通过输入命令关键字为实用程序名的终端命令而运行。当用户输入命令解释程序不认识的命令关键字时，命令解释程序不是报错，而是去寻找与命令关键字同名的文件。所以，如果要运行某个实用程序，只要输入该实用程序的执行文件名即可。主要的系统实用程序如下。

（1）编辑器。供用户建立和修改源程序文件及其他文件。它会提供一组内部编辑命令由编辑器解析执行。

（2）编译器和装配器。实现编译源程序、连接模块、装配目标程序等功能。

（3）文件及文件系统相关的实用程序。文件的复制、打印、文件系统装卸等实用程序。

（4）显示系统进程、资源状态的实用程序。如进行进程状态显示、文件状态显示、内外存空间显示的相关实用程序。

（5）用户管理。如添加/删除用户、修改口令。

【例 2.6】　下面说明 UNIX/Linux 的一些主要命令及其使用示例，其中像 pwd，cd 等简单命令是由 shell 命令解释程序直接解释执行的，而其他的命令是以实用程序的方式由 shell 命令解释程序创建子进程执行的。

- pwd：查看当前工作目录。
- cd /usr/home/sally：改变当前工作目录。

- ls -l：查看当前工作目录下的文件及其属性，-l 是传给 ls 实用程序的命令参数。
- more /etc/passwd：显示/etc/passwd 文件内容。
- cp /etc/passwd ./passwd：将/etc/passwd 文件复制为当前目录下 passwd 文件。
- mv passwd oldpasswd：将当前目录 passwd 文件改名为 oldpasswd。
- rm oldpasswd：删除当前目录的 oldpasswd 文件。
- mkdir temp：在当前目录下建立一个 temp 子目录。
- cp-r/mnt：将/mnt 下的内容复制到当前目录的 mnt 子目录下。
- find/-name passwd -print：从根目录开始查找名为 passwd 的文件。
- find/-size +250 -print：从根目录开始查找大于 250 个块大小的文件。
- grep “init” *.*：在当前目录的所有文件中查找 init 字符串。
- who：显示使用终端的用户名单。
- su：进入超级用户。
- wall" good bye"：向所有登录用户广播" good bye"。
- ps-elf：显示系统所有当前运行的进程信息。

2.5.2 图形化的用户界面

用户操作计算机的界面随着计算机的发展发生了翻天覆地的变化。在批处理系统中，因采用脱机方式工作，使用的是作业控制语言。而命令语言的另一种形式是大家较熟悉的，它就是在分时系统或个人计算机上使用的键盘命令，如 UNIX/Linux 的 shell 命令、MS DOS 上的键盘命令。

随着计算机的广泛应用，人们逐渐感到这种交互命令方式不太方便，因为这种命令不直观，它是比较难记的一串串字符命令，还带有各种参数和规定的格式。另外，不同的操作系统所提供的命令语言的词法、语法、语义和表达风格也不一样。随着计算机的广泛应用，计算机迅速进入了各行各业、千家万户，其用户来自不同阶层，如何使人机交互方式进一步变革，使人机对话的界面更为方便、友好、易学，是一个十分重要的问题。在这种需求的推动下，出现了菜单驱动方式、图符驱动方式，直至视窗操作环境。

（1）菜单驱动方式。为了解决命令难记的问题，系统将所有有关的命令和系统能完成的操作，用类似餐馆的菜单方式分类、在屏幕上列出。用户根据菜单提示，像点菜一样选择某个命令或某种操作，以控制系统完成指定的工作。

为此，引入了对话框，提示用户需要输入的参数及参数的取值范围或可选项等内容。过去，需要用户一次性输入命令关键字和所有参数来实现一次操作，如今在菜单驱动方式下，可分成多步，先选择菜单，再在对话框的提示下输入参数，最后执行。

在系统主菜单中可以显示系统所提供的实用程序列表，而实用程序菜单又可提供该实用程序支持的子功能列表。可以看出，菜单是一种提示系统功能或实用程序子功能的方法。利用菜单和对话框可提示性地帮助用户指导计算机工作，大大方便了用户。

菜单有多种类型，如下拉式菜单、上推式菜单和随机弹出式菜单等。

（2）图符驱动方式。图符驱动方式是一种面向屏幕的图形菜单选择方式。图符（Icon）也称图标，是一个较小的图符符号，它代表操作系统中的命令、系统功能或者被处理的对象（各种资源，如文件、打印机等）。如用小矩形代表被处理对象文件，用小剪刀代表剪切功能。

所谓图形化的命令驱动方式就是当需要启动某个系统命令或操作功能，或请求编辑或访问某个系统文件时，可以选择代表它的图符，并借助鼠标一类的标记输入设备（也可以采用键盘）的单击和拖曳功能，完成命令和操作的选择与执行。

图符与菜单最大的不同是，图符也可将被处理对象罗列出来，这更符合用户的使用心理。因为用户使用计算机多数是要对某个对象（如数据文件）进行处理，作为被处理对象的文件用图符表示，系统根据文件类型自动启动对应的实用程序，用户可以省去选择实用程序的时间。