希望
Beijing Hope Electronic Press
www.bhp.com.cn

经典范例50讲系列

北京希望电子出版社 总策划
任志宏 程 超 编 著

# C语言经典范例50讲

中国物资出版社
北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

# 前　言

操作系统是计算机科学中一门重要的专业基础课，同时又是一门实践性很强的技术课程。掌握操作系统原理，熟悉操作系统的使用是各层次计算机软硬件开发人员必不可少的基本技能。

目前国内相当多学校的操作系统课程教学偏重于讲授原理，对实践环节重视不够。其结果是很多学生只知其然，不知其所以然，不能深刻地理解操作系统的本质，因而也不能在工作中充分应用操作系统课程所学知识和操作系统提供的功能来解决实际问题。造成这种局面的原因之一是完整的、实用的实验教材和与之相配的实验系统尚不多见。基于上述原因，我们以A. S. Tanenbaum 设计的教学操作系统 MINIX 为背景，编写了本书，并开发了集实验、操作、调试及演示为一体的实验系统，为学生提供了一个自己动手学习、编制和修改操作系统的实验环境。通过几年来在教学实践中的试用，受到了学生的欢迎和同行的一致好评。

本书分为上下两篇。上篇主要介绍 MINIX 操作系统的基本结构和组成原理；MINIX 的进程管理、I/O 管理、存储管理、文件管理等几大部分的实现原理和方法；并详细地介绍了 MINIX 的各种系统调用的实现方法等。下篇主要介绍如何使用 MINIX 系统，指导学生自行分析、设计、修改和重建新的“MINIX 系统”。共分为六个实验专题，每个专题都有实验的目的、任务和要求，并提供了 MINIX 各部分主要过程流程图，以供学生分析参考。另外本书还附有实验软件。

本书可与操作系统原理课程配合使用，根据课程进度，选择有关章节内容作为实例分析和实验环节，即课程与实验同步进展。也可待操作系统课程全部讲完之后，再讲授 MINIX 系统和上机实验。两种方法各有其优点，前者与课程结合较为紧密，具有较好的连续性。后者待学生对操作系统有了比较完整的理解之后，再来设计、分析 MINIX 系统，这样就会比较得心应手，有更深的认识，收到更好的效果。

本书讲授时可按 MINIX 基本原理与专题实验同步的方式进行；或先讲完基本原理后，再进行专题实验。课堂教学约需 15 学时，上机实验约需 20 学时，另有 15 课外学时供学生进行分析和习题练习，部分带“＊”号的习题作为学生的选做题。

学生在进行实验时，可以根据个人兴趣、设备配置、机时多少等因素，独立完成或与他人合作完成。多数实验完成后，都可通过运行最新建立的 MINIX 系统得到验证。

本教材适用于计算机专业和相关专业的本科、专科的实验教学，也可供教师和专业技术人员参考。

本书是在铁道部计算机、自动控制、工业电气化自动化教学指导委员会支持下完成的。在本书的编写过程中，北方交通大学须德教授给予了积极的支持和帮助，于百忙之中审阅了全书，并提出了许多有益的意见。专业指导委员会主任、西南交通大学靳蕃教授、北方交通大学丁嘉种教授、西南交通大学朱怀芳教授、刘太平老师、张雪老师对本书的出版给予了大力的支持。在此一并表示诚挚的谢意。

由于编者水平有限，时间仓促，书中难免有错误和不妥之处，恳请广大读者批评指正。

编者

1994 年于成都

# 目　录

## 上篇　MINIX 基本原理

## 下篇　实验专题

# 上篇 MINIX 基本原理

# 第一章 MINIX 操作系统概述

## §1.1 操作系统基本概念

计算机系统通常有硬软件两大类资源。硬件资源包括中央处理机、存贮器和输入/输出(I/O)设备等物理资源。软件资源是程序和数据等信息资源。

没有软件的计算机仅由硬件构成，称之为裸机，裸机是构成计算机系统的物质基础。然而只有裸机没有软件的计算机不能有效地工作，无法发挥其潜在能力；用户使用这种裸机进行工作也是十分困难的。

计算机的软件大致可以分为两大类：一类是对计算机本身进行管理；另一类是提供用户使用的程序。操作系统是计算机系统的一种系统软件，它用于管理和控制计算机系统的硬件和软件资源，为用户提供良好、有效的使用环境。因此，操作系统是计算机系统中最重要又最基本的系统软件之一。

一个计算机系统是由硬件和软件按层次方式构成。图 1－1－1 是计算机系统的一个层次结构。最底层是硬件层，它由物理器件、微程序、机器语言等构成。操作系统对硬件层作第一次功能扩充，以便为开发系统进行有效的服务。

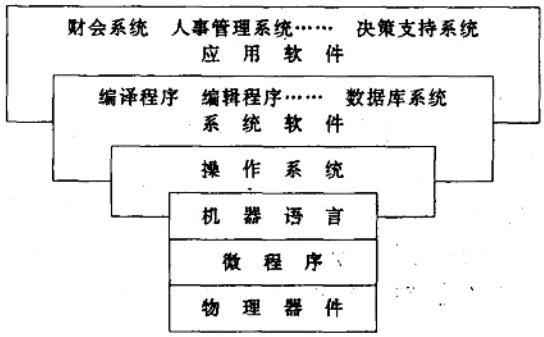


图 1－1－1 系统结构图

## §1.2 操作系统的功能

操作系统的作用是充分发挥系统资源的效率和方便用户。因此，它的主要功能是对系统资源实施有效管理，为用户提供操作使用的环境。从资源管理和用户接口的角度来看，操作系统的功能应有：处理机管理、存贮管理、设备管理、文件管理和作业管理。

### 1.2.1 进程与处理机管理

1. 进程概念

一个复杂的程序通常可以分为若干个程序模块，并按照某种次序来执行。在早期的计算机

中，对程序模块的执行是按照先后次序来执行的。为增强系统能力和提高机器利用率，现代计算机中广泛采用并行操作技术，使多种硬件设备并行工作，从而提高系统能力和机器效率。如何有效地利用系统的并行能力，很重要的一方面是取决于程序的编制。在一个程序的若干程序模块中，有些模块的执行必须有先后次序，另外很多模块是可以共行执行的。所谓共行执行是指这些程序模块的执行在时间上是可以重叠的，即使这种重叠只有很小的一部分。把一个程序划分为若干个可共行执行的程序模块的优点是：简化了程序设计；在多处理机系统中，可以加快程序的执行速度；在多道程序环境下，可以并行操作，有效利用系统资源。这种把一个程序分为若干个可同时执行的程序模块的方法称为并发程序设计。也可以说，每个程序模块和它执行时所需的数据就构成一个进程。

更进一步地说，进程与程序是两个既有联系又有区别的不同概念，它们的主要区别是：

(1) 程序是指令的有序集合，是静止的。而进程是程序的一次执行过程，因而是动态的，它由“创建”而产生，由调度而执行，因得不到资源而暂停，以至最后由“撤销”而消亡。进程有一定的生命周期，动态地产生和消亡。

(2) 进程是一个可独立运行的单位，能与其它进程共行执行。而程序则不然。

(3) 一个程序可以对应多个进程，它可以根据给定的活动空间和不同的初始条件形成若干个不同的进程。反之，一个进程至少对应一个程序。

因此，可以简单地说，所谓进程，就是一个程序在给定空间和初始环境下在一个处理机上的执行过程。

2. 进程的状态及其转换

一般来说按照进程在执行过程中的不同时刻的不同状态，我们可以定义进程的三个基本状态：

阻塞状态：在执行中的进程，由于某事件使之暂时无法执行(例如等待 I/O 操作完成)而处于暂停状态。

就绪状态：进程已分配到必要的资源，而正在等待得到处理机。一旦分配到处理机就可以立即执行。

运行状态：进程已获得必要的资源，并占有一台处理机正在执行。

每个进程在执行过程中，任一时刻处于且仅处于上述三种状态之一。同时，每个进程在执行过程中，它的状态也会发生变化。在一个实际的系统中，进程的状态及其转换会更复杂一些。

3. 进程控制块

为了记录进程执行时的情况，每个进程都有一个进程控制块，根据它对进程进行控制和管理。当进程完成后，系统收回进程控制块，进程也随之消亡。一般来说，进程控制块应包含四类信息：

第一类：“标识信息”，用于标识一个进程，如进程名等。

第二类：“说明信息”，用于说明一个进程，如进程状态、等待原因等。

第三类：“现场信息”，用于保留进程在运行状态时存放在处理器中的各种信息，如通用寄存器、控制寄存器的内容、程序状态字等。

第四类：“管理信息”，用于调度进程，如进程的优先级、调度用的队列指针等。

4. 处理机调度

处理机是计算机系统中最重要的资源之一。在处理机管理中，最重要的是它的运行时间。

在多道程序或多用户的情况下，要组织多个作业同时运行，就必须解决处理机的分配策略、时间片和调度等问题。

处理机调度负责动态地把处理机分配给每个进程。又称之为低级调度。其主要功能是：

(1) 记录进程的状态，一般记录在进程控制块中。

(2) 决定某个进程何时可以获得处理机，以及占用多长时间。

(3) 把处理机分配给进程。即把选中的进程的进程控制块内有关现场信息送入处理机相应的寄存器中，并让它占有处理机运行。

(4) 收回处理机。将处理机有关寄存器内容送入该进程的进程控制块中，并使该进程让出处理机。

一般情况下，处于同一状态的所有进程控制块是链接在一起的，称之为进程队列。当发生某些事件使一个进程的状态发生变化时，这个进程就要退出原来所在的队列，而排入到另一个队列中去，称之为出队与入队。处理机调度中负责入队和出队工作的功能模块称之为队列管理模块。

处理机调度的策略较多，常用的有如下几种：

(1) 优先数法。 对每个进程给定一个优先数，处理机调度每次选择就绪进程中优先数最大者，让它占用处理机运行。它的关键问题是如何确定优先数，通常优先数分为静态优先数和动态优先数两种。一般优先数需要考虑进程类型、作业对资源的要求情况、作业到达时间等因素。

(2) 时间片轮转法。 轮转法是为每个进程规定一个时间片，每个进程轮流地运行一个时间片。当时间片结束时，就强迫运行的进程让出处理机。最简单的方法是分给每个进程大小相同的时间片。更复杂一些的轮转法则是对不同的进程分配不同大小的时间片，并根据情况动态地修改时间片，这样可以提高运行效率。

(3) 多级队列法。 多级队列法的主要思想是将就绪进程分为两级或多级，系统建立相应的两个或多个就绪进程队列。一般根据进程的某些特性，如系统进程或用户进程，把各个进程分别链入其中某一队列中，每个队列都有自己的调度算法。此外，在各队列之间也要进行调度，通常采用静态优先数法。即处理机每次先从高优先数的就绪进程队列中选取可占用处理机的进程，只有选不到时，才从较低一级优先数的就绪队列中选取。

### 1.2.2 存贮管理

在计算机系统中，另一类重要的资源是主存贮器。主存贮器的管理应与处理机调度相结合，即只有当进程在主存时，它才有可能得到处理机执行；反之，只有它可以得到处理机执行时才将其调入主存。在现代计算机系统中，通常采用多道程序设计技术，要求存贮管理具有以下功能：

1. 主存贮器空间的分配与回收

当需要把作业装入主存时，必须按照规定的方式向操作系统提出申请，由存贮管理进行分配。存贮管理必须记录存贮空间的分配情况，根据申请者的要求按一定的策略找出足够的自由空间分配给申请者，并修改有关记录。若没有足够的空间满足申请者，则让申请者处于等待主存资源状态，直到有足够的存贮空间为其分配。

当主存贮器中某个作业撤销或主动归还主存资源时，存贮管理应收回它占用的部分或全

部主存资源，使其成为自由空间，并修改有关记录项。

2. 主存贮器空间的共享

共享主存贮器是为了提高主存空间的利用率，所谓共享主存贮器的含义是：

(1) 共享主存贮器资源。采用多道程序设计技术使若干个程序同时进入主存空间，各自占用一定数量的存贮空间，共同使用一个主存。

(2) 共享主存贮器的某些区域。若干个作业拥有共同的程序段或数据段时，可将这些共同的程序段或数据段放到某个存贮区内，各作业执行时都可以访问它们。

3. 存贮保护

主存贮器中不仅有系统程序，而且还有若干道用户作业的程序和数据。为了避免主存中的若干道程序相互干扰、破坏，必须对主存中的程序和数据进行保护。通常硬件提供保护功能，软件配合实现。具体保护方法有：基址、界限寄存器法，存贮键和锁等方法。

4. 主存贮器扩充

主存贮器是计算机系统中较为缺乏的资源之一，尤其在多道程序运行环境中，主存资源变得更为紧张。在计算机硬件的支持下，软硬件协作，可以把磁盘、磁鼓等辅助存贮器作为主存器的扩充部分来使用，实现这种功能的软件技术称为“虚拟存贮技术”。当一个大型程序要装入主存时，可先把其中的一部分装入主存贮器，其余部分放在磁盘上，如果在执行中需要存取不在主存中的信息时，由操作系统再将其调入主存。这样，用户在编制程序时就不需要考虑实际主存空间的容量。就好像他在使用一个足够大的主存贮器一样。

5. 存贮与存贮分配无关性

当用户的程序和数据存贮到主存贮器中时，用户和程序员无法预知这些程序和数据将安置到主存的什么位置，占用多少存贮空间，而且他们也希望摆脱存贮地址、存贮空间大小这类繁琐的问题。为此，存贮管理应提供“地址重定位”、“重定位装配程序”或“地址映象机构”等。

### 1.2.3 设备管理

在现代计算机系统中配置了大量的外部设备。特别是在计算机应用中，信息管理占了很大的比重，外部设备的作用显得越来越重要，种类越来越多。计算机的外部设备主要是I/O型设备，如终端显示器、打印机等。它们把外界信息输入计算机，把处理结果和有关信息从计算机输出。

设备管理是操作系统中最为繁杂的工作，其原因是：

(1) 设备管理与硬件密切相关，而且各种外部设备物理特性各不相同，品种繁多，用法各异；

(2) 我们希望各种外部设备都应能与主机并行工作，但是设备的所属关系较为复杂，有的是系统设备，有的是用户设备；

(3) 主机与外部设备、各外部设备之间的速度极不匹配，差别很大。

在一个完善的系统中，不仅具有中断机构，而且还引入了通道和缓冲技术。虽然它们能显著地提高处理机与通道之间的并行操作能力，但也给管理工作带来了很大的困难。因此，设备管理的基本任务是，按照用户的要求来控制外部设备工作，完成用户所希望的I/O操作，以减轻用户编制程序的负担。在现代操作系统中允许多个进程共行执行，但由于进程数多于外部设备数，必将引起进程对资源的争夺。因此，设备管理的另一个重要任务是按照一定的策略把

设备分配给提出请求的进程，以保证系统有条不紊地工作。此外，在现代计算机系统中，各种外部设备所占投资比重很大，如何充分而有效地使用这些设备，尽可能地发挥它们的并行操作能力也是设备管理的重要任务。为实现这些任务，设备管理应具有以下功能：

(1) 进行设备分配。按照设备的类型和一定的分配策略把设备分配给请求该设备的进程。在进行设备分配的同时还应分配相应的控制器和通道，以保证 I/O 设备与处理机之间的信息传递。未分到设备的进程应排成一个等待队列。

(2) 实现真正的 I/O 操作。为此，设备管理应具有下面三个功能：首先，在设置有通道的系统中，I/O 操作应由通道执行通道程序来完成，因此设备管理应能构成相应的通道程序，提供给通道去执行；其次，启动指定的设备进行 I/O 操作；最后，能够对通道发来的中断请求作出相应的反应和处理。

(3) 缓冲技术。在大多数系统中都配置有硬件缓冲器，以缓和处理机与 I/O 设备之间的速度不匹配。当进行 I/O 时，可以先将信息送到缓冲器中，然后再由处理机或外设到缓冲器中取出信息。因此，设备管理应具有对缓冲器进行管理的功能。

(4) 设备与设备分配无关性。用户向系统申请和使用的设备应与实际操作的设备无关。这一特征不仅为用户使用设备提供了方便，提高了设备的利用率，而且也改善了系统的适应性和可扩展性。

### 1.2.4 文件管理

在操作系统中，处理机管理、存贮管理和设备管理是对计算机系统硬件资源的管理。现代计算机系统中还有另一类重要的资源，即软件资源。软件资源是指各种程序和数据的集合。现代计算机操作系统提供文件管理来管理用户和系统信息的存贮、检索、更新、共享和保护，并为用户提供一套方便有效的文件使用和操作方式。

文件是逻辑上具有完整意义的信息集合，每个文件用一个名字(文件名)来识别。文件名一般是以字母开头的字母数字串。文件这一术语不但反映了用户概念中的逻辑结构，而且也和存放文件的存贮介质(如磁带、磁盘等)的存贮结构紧密相关。所以，同一个文件必须从逻辑文件和物理文件两个侧面来看待它。组成文件的信息可以是各种各样的：一个源程序、一批数据、各类语言的编译程序等都可以各自组成一个文件。

操作系统把信息组织成文件形式加以管理和控制是计算机信息管理的重大进展。其主要的优点是：

(1) 方便用户。使用者无需考虑信息在存贮设备上的物理位置，也无需考虑具体的存贮技术，只要知道文件名，按照有关的操作要求，就可以存取信息，实现了“按名存取”。

(2) 安全可靠。由于用户要通过文件系统才能实现对文件的访问、操作，因而文件系统可以提供各种安全、保密和保护措施，可以防止有意或无意地破坏或窃用文件信息。

(3) 文件共享。文件系统还能够提供文件的共享功能，既节省了文件存贮空间，又减少了文件传输的时间，进一步提高了文件和文件存贮空间的利用率。

文件系统的主要功能和特性是：

(1) 有效地分配和回收文件的存贮空间；

(2) 实现文件的按名存取，使文件的存取方便、多样；

(3) 对用户尽可能透明；

(4) 实现对存贮文件的安全、保密和保护；

(5) 提供文件的共享；

(6) 有效地实现文件的各种操作命令。

### 1.2.5 作业管理

一个作业，就是用户在一次信息处理过程中要求计算机系统所做的工作的集合。在一个多道程序的共行系统中，一个作业是独立于其它作业的计算工作的一个单位。一般，一个作业需要经过若干个加工步骤才能完成，每一项相对独立的工作叫做一个作业步。一个作业的源程序、数据以及控制要求等初始信息可以组织在某种输入设备上，由输入设备将这些信息送入计算机系统。

操作系统的一个重要目的是为了方便用户使用计算机系统。在操作系统的协助下，用户能够方便、灵活、安全可靠、经济有效地使用计算机系统的资源来处理问题。操作系统为用户提供两种类型的接口，以便于用户与操作系统建立联系。一种接口是操作系统提供一组广义指令(也称系统调用)，用户在程序中可直接用这些广义指令向操作系统提出使用各种外部设备的要求，申请分配和回收主存资源，以及其它各种控制要求等。操作系统则按用户的要求启动外部设备，分配或回收内存资源，调度、显示信息、暂停运行、解除干预等。另一种接口是操作系统提供的作业控制语言或操作控制命令，以便用户描述对作业进行的加工步骤，操作系统按用户的作业控制语句或操作控制命令来控制作业的执行。

操作系统的作业管理有两个功能：

(1) 作业调度。在多道程序设计系统中，系统可以同时处理多个作业，系统要在许多作业中按一定的策略选取若干个作业，为它们分配必要的资源，建立相应的用户作业进程和为其服务的系统进程。最后，把它们的程序段调入主存等待进程调度进行调度，这就是作业调度，作业调度又称为高级调度。

(2) 作业控制。作业是在操作系统控制下执行的。这种控制包括把作业输入到系统中、当作业被选中后控制作业的执行、作业执行中出现故障后的处理，以及控制计算结果的输出等等。

## §1.3 MINIX 系统简介

自从 70 年代初贝尔实验室研制出 UNIX 操作系统后，其发展极其迅速。目前，UNIX 已被广泛地应用于从微型机到大型机的各种类型计算机中，成为当今世界最流行的多用户操作系统。同时，也成为操作系统教学的实际例子之一。

但是，UNIX 作为一种实用的商品软件，其目标旨在实用性和系统的高效。由于 UNIX 涉及的细节较多，给教学和分析增加了一定难度，至于要想在实验中设计、修改和重建一个系统就非常困难了。MINIX 的诞生恰好纠正了这种偏向。它采用了 UNIX 的设计思想，用户界面与 UNIX 相似，但内部结构不尽相同。它小巧而功能齐全，是一个专门用于教学的操作系统。MINIX 的主要特点有：

(1) 提供了与 UNIX 第七版相同的系统调用和外壳。

(2) 在内部构造上与 UNIX 完全不同，力求简洁与精巧。如存贮管理仅采用简单的分区分配。

(3) 该系统主要利用 C 语言书写，并配有 3000 多处注释，使之易读、易懂、易修改、易移植，且十分紧凑。

(4) 在 MINIX 中使用了树型结构的文件系统，与 UNIX 一样具有良好的安全性、保密性和可维护性。

(5) 系统提供了消息会合通信机制，以满足各模块之间消息的传递。

## §1.4 MINIX 的内部结构

MINIX 构造为四层：最底层是进程管理，其次是 I/O 任务、服务员进程和用户进程。如图 1－1－2 所示。

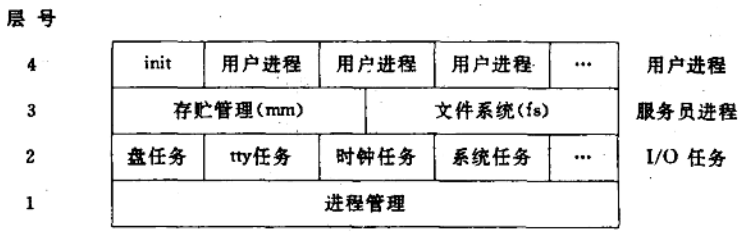


图 1－1－2 MINIX 的层次结构

进程管理层的主要任务是：捕获一切中断和陷阱；保护与恢复寄存器；将中断信号转变成实际消息，并在物理存贮中定位发送和接收缓冲区，从发送者向接收者复制字节等。该层涉及最低级的中断处理部分是用汇编语言编写的。

第二层包含 I/O 进程。为了与普通的用户进程相区别，这里也叫做 I/O 任务 (task)，又称为设备驱动程序。每种类型的设备都由其对应的 I/O 任务驱动，包括盘任务、打印机任务、终端任务和时钟任务等。此外还有一个系统任务，它稍特殊一点，不对应于任何 I/O 设备，将在 §3.6 专门讨论。第二层和第一层中所有代码连接成单独的一个程序，称作内核(kernel)，也即整个系统的核心。

第三层包含两个服务员进程：存贮管理程序 mm 和文件系统 fs。它们各自作为一个独立的程序在内核外运行。mm 实现涉及存贮管理的全部系统调用，如 FORK、EXEC 和 BRK；fs 实现所有的文件系统调用，如 OPEN、READ 和 MOUNT。

最顶层包含所有的用户进程：shell 壳、编辑程序、编译程序，以及用户编写的 a.out 程序等。

一般来说，操作系统主要做两件事：资源管理和提供系统调用。MINIX 中资源管理多半在内核中，系统调用的实现在第三层。进程管理、I/O 任务、存贮管理和文件系统四大部分组成了一个完整的操作系统。下面几章将予以详细讨论。

## §1.5 MINIX 的系统调用

系统调用是运行程序和操作系统之间的接口。用户在程序中通过系统调用能调用操作系统提供的一些子功能，处理更基层的服务。MINIX 共配置了四十一个系统调用，与 UNIX 第七版完全一样，如表 1－1－1 所示。

这些系统调用分别在程序模块 mm 和 fs 里实现。实验五和实验六列出了其中大部分流程图。

MINIX 的 系 统 调 用　　表 1－1－1

| 分属 | 系统调用名称 | 功　　能 | 编号 |
|---|---|---|---|
| 进程管理 | pid=fork | 创建一个与父进程相同的进程作为子进程 | 2 |
| | s=wait(&status) | 等待子进程终止，并获取其出口状态 | 7 |
| | s=execve(name,argv,envp) | 进程内存映象的更换 | 59 |
| | exit(status) | 终止进程的执行，返回出口状态 | 1 |
| | size=brk(addr) | 将数据段大小调整为"addr" | 17 |
| | pid=getpid( ) | 返回调用者进程的 id | 20 |
| 信号 | oldfunc=signal(sig,func) | 规定对软中断信号的响应方式 | 48 |
| | s=kill(pid,sig) | 向进程发送信号 | 37 |
| | residual=alarm(seconds) | 在 seconds 秒后发一个 SIGALARM 信号报警 | 27 |
| | s=pause( ) | 挂起调用者，直至下一个信号到来 | 29 |
| 文件管理 | fd=creat(name,mode) | 创建一个新文件或截取一个现存的文件 | 8 |
| | fd=mknod(name,mode,addr) | 创建一个正规的、特别的或目录的 i 节点 | 14 |
| | fd=open(file,how) | 打开文件供读、供写或兼而有之 | 5 |
| | s=close(fd) | 关闭一个已打开的文件 | 6 |
| | n=read(fd,buffer,nbytes) | 把文件中的数据读进缓冲区 | 3 |
| | n=write(fd,buffer,nbytes) | 把缓冲区中的数据写入文件 | 4 |
| | pos=lseek(fd,offset,whence) | 把文件指针移向文件中的某处 | 19 |
| | s=stat(name,&buf) | 从 i 节点返回文件状态 | 18 |
| | s=fstat(fd,buf) | 从 i 节点返回文件状态，输入参数与 STAT 不同 | 28 |
| | fd=dup(fd1) | 为一个打开的文件分配另一个文件描述字 | 41 |
| | s=pipe(&fd[0]) | 建立管道 | 42 |
| | s=ioctl(fd,request,argp) | 对字符特别文件(主要是终端)作特别操作 | 54 |
| 目录管理 | s=link(name1,name2) | 创建一个新目录项 name2，并把它连接到 name1 | 9 |
| | s=unlink(name) | 解除目录项连接 | 10 |
| | s=mount(special,name,rwflag) | 安装文件系统 | 21 |
| | s=umount(special) | 拆卸文件系统 | 22 |
| | s=sync( ) | 将所有缓冲池块回写磁盘 | 36 |
| | s=chdir(dirname) | 改变工作目录 | 12 |
| | s=chroot(dirname) | 改变根目录 | 61 |

续表

| 分属 | 系统调用名称 | 功　　能 | 编号 |
|---|---|---|---|
| 文件保护 | s=chmod(name,mode) | 修改文件保护位 | 15 |
| | uid=getuid( ) | 取调用者的 uid | 24 |
| | gid=getgid( ) | 取调用者的 gid | 47 |
| | s=setuid(uid) | 置调用者的 uid | 23 |
| | s=setgid(gid) | 置调用者的 gid | 46 |
| | s=chown(name,owner,group) | 改变文件主和用户组 | 16 |
| | oldmask=umask(complmode) | 置系统内部位屏蔽 | 60 |
| | s=access(name,mode) | 检查一个文件的保护位 | 33 |
| 时间管理 | seconds=time(&seconds) | 取系统当前时间 | 13 |
| | s=stime(tp) | 置系统当前时间 | 25 |
| | s=utime(file,timep) | 为文件置最近一次访问的时间 | 30 |
| | s=time(buffer) | 取进程记帐时间 | 43 |

# 第二章　MINIX 的进程管理

## §2.1　进程模型

进程是一个执行中的程序，还包含程序计数器、寄存器和变量的现行值。在 MINIX 进程模型中，所有在计算机上可运行的软件，包括操作系统在内，都被组织成为不同的进程。

1. 进程层次

MINIX 系统初启时，计算机把引导程序读进内存并转移到它那里，由引导程序把整个操作系统和文件系统校验程序传送到内存并启动它们运行。在内核、存贮管理和文件系统业已运行并且把它们自身初始化以后，控制权即传递给 init。init 从读文件/etc/ttys 开始，检查现时安装的终端有多少台(标准配置只有控制台)，然后为每台终端分岔出一个 shell 子进程。这些子进程中的每一个都执行/bin/login，一直等到有用户注册为止。用户注册后，由 shell 进程为用户键入的命令创建(fork)若干子进程，每个子进程执行一条命令。执行命令的子进程也可再创建子进程。这样一来，shell 是 init 的子进程，用户进程则是 shell 进程的子进程，系统中的所有进程都是以 init 为根的一棵树的子树，如图 1－2－1 所示。

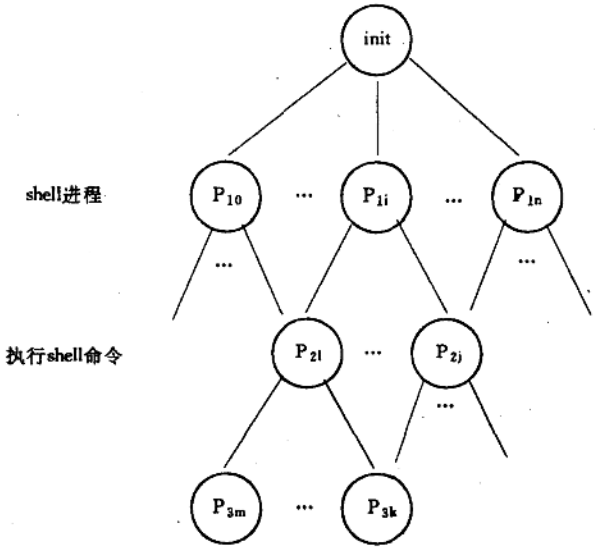


图 1－2－1　进程创建的层次关系

2. 进程表

MINIX 中配置了一张进程表(结构型数组)来描述进程模型。每个进程分配一个槽，槽里包含有关进程的全部信息，如表 1－2－1 所示。此表分为三个域：内核、存贮管理和文件系统，分别由文件 kernel/proc.h、mm/mproc.h 和 fs/fproc.h 说明。

MINIX 进程表的一个槽　　表 1－2－1

| 内　　核 | 存贮管理 | 文件系统 |
|---|---|---|
| 寄存器 | 正文段指针 | UMASK 屏蔽 |
| 程序计数器 | 数据段指针 | 根目录指针 |
| 程序状态字 | bss 段指针 | 工作目录指针 |
| 栈指针 | 出口状态 | 文件描述符 |
| 进程状态 | 信号状态 | 有效的 uid |
| 系统所用 CPU 时间 | 进程 id | 有效的 gid |
| 用户所用 CPU 时间 | 父进程标识号 | 系统调用参数 |
| 子进程的系统和用户时间 | 进程组(供信号用) | 各种标志位 |
| 下一次报警时间 | 真正的 uid | |
| 消息队列指针 | 有效的 uid | |
| 进程 id | 真正的 gid | |
| 各种标志位 | 有效的 gid | |
| | 供信号用的位图 | |
| | 各种标志位 | |

## §2.2　中断处理

中断处理是系统内核最基本的功能，也是整个操作系统赖以活动的基础，即操作系统的重要活动最终都将依赖于中断。如各种类型的系统调用、键盘命令的转入、进程调度、设备驱动及文件操作等，无不依赖于中断。这里以磁盘中断为例，考察 MINIX 的中断处理过程。

假设用户进程 2 正在运行时，磁盘中断发生。这时，中断硬件把程序计数器、程序状态字和工作寄存器压入堆栈，再转移到磁盘中断向量所指定的地址。随后，由软件(中断处理程序)进行相应处理。中断处理过程大致分为四个阶段：由汇编语言过程保存寄存器内容并设置新的栈；构造一条消息送往磁盘进程，并把它的状态由阻塞改为就绪；调度程序决定哪个进程接着运行，或许是磁盘进程，或许是被中断的用户进程，条件是优先级最高；最后由汇编语言过程为当前选中的进程装填寄存器与存贮映象，并启动它运行。

## §2.3　进程通信

MINIX 为进程之间的通信提供了三种工具：① 由软中断信号机制实现同一用户的诸进程之间少量信息的传输和同步；② 由消息会合通信机制实现进程间大量的消息传递；③ 基于文件系统的管道机制，可在进程间提供传输大量信息的通道。软中断通信将在 §4.5 有关信号问题中讨论，管道通信放在文件系统 §5.9 再述，这里着重介绍会合原理通信。

MINIX 的会合原理通信负责进程间的消息传递，它提供了 SEND 、RECEIVE 和 SENDREC 三个原语来发送和接收消息。

进程调用 SEND 发送消息时，首先查看目标进程是否在等待来自该发送者( 或任一发送者)的消息。若是，这条消息则从发送者的缓冲区复制到接收者的缓冲区里，两个进程都标记为可运行的；否则，该发送者被标记为阻塞，并排入等待向该接收者发送消息的进程队列。

进程调用 RECEIVE 接收消息时，先查看是否有进程在排队等待向它发送。若有，则把这

条消息从阻塞的发送者那里复制给接收者，两个进程都标记成可运行的；若无，则接收者阻塞，直至消息到来。

SENDREC 向一个进程发送一条消息，并等待该进程的回答。

参照图 1－1－2，每一进程或任务均可与其同层中的进程或任务发收消息，也能与直接在它下面的进程或任务发收消息，但用户进程不得直接跟 I/O 任务通信。

## §2.4 进程调度

MINIX 调度程序采用多级队列方法，如图 1－2－2 所示。系统中设置三个就绪队列，对应于图 1－1－2 中的 2、3、4 层，并赋予各队列不同的优先权。任务(TASK)享受最高优先权，存贮管理与文件服务(SERVER)其次，用户进程(USER)最低。在每个队列内部采用不同的调度算法：任务和服务员队列采用先来先服务算法，用户队列采用轮转法。

调度程序在挑选进程时，总是先调度任务执行；仅当任务队列空时，才调度服务员队列上的进程运行；在选不到服务员进程的情况下，再运行用户进程。倘若任何进程均未就绪，则系统空循环等待。每个用户进程在运行超过某一时间片（如 100ms)时，将被移到队尾。

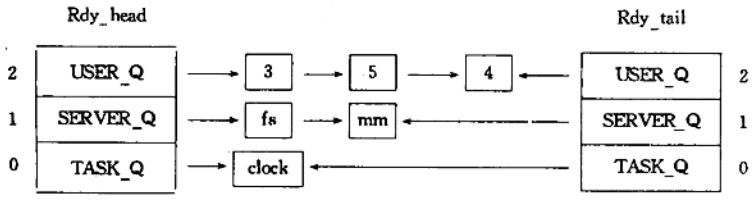


图 1－2－2 调度程序维护的三个队列

# 第三章　MINIX 的 I/O 管理

## §3.1　MINIX 中的 I/O

从 I/O 角度观察图 1－1－2，得到如图 1－3－1 所示的 MINIX I/O 系统的层次结构。每层意义如下：

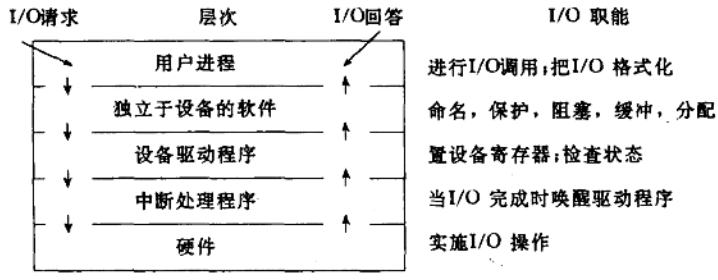


图 1－3－1　I/O 系统的层次和各层的主要职能

1. 中断处理程序

在 MINIX 中，当中断发生时，由中断处理程序将它们转变成消息，发送给被阻塞的进程（通常是设备驱动程序），使之变得能够运行。而许多设备驱动程序一般情况下都是阻塞起来等候消息到来。中断处理部分第二章已经讨论，在此不再详述。

2. 设备驱动程序

在 MINIX 中，每一类 I/O 设备都有一个独立的设备驱动程序（也称 I/O 任务），由它们驱动设备的实际操作。每一驱动程序的主体部分结构相同：初始化后阻塞，当消息来到时，调用一个过程执行这项工作，完成后发送回答。下面几节，将重点讨论 RAM 磁盘、软盘、时钟和终端四种驱动程序。

设备驱动程序借助于标准的消息机制实现相互间以及与文件系统的通信。对于块设备（字符设备与之类似），发送和接收的消息 m 的字段如表 1－3－1 所示。

块设备的请求与应答消息 m 的字段　　表 1－3－1

| 请求 | | | 应答 | | |
|---|---|---|---|---|---|
| 字　段 | 类型 | 含　义 | 字　段 | 类型 | 含　义 |
| m. m _ type | 整型 | 所申请的操作 | m. m _ TYPE | 整型 | 总是 TASK _ REPLY |
| m. DEVICE | 整型 | 要用的副设备 | m. REP _ PRCHR | 整型 | 同于请求中的 PROC _ NR |
| m. POSITION | 长型 | 在副设备上的位置 | m. REP _ STATUS | 整型 | 被传送的字节数或出错号 |
| m. PROC _ NR | 整型 | 申请该 I/O 的用户进程 | | | |
| m. ADDRESS | 字符 | 在 PROC _ NR 内的地址 | | | |
| m. COUNT | 整型 | 待传输的字节数 | | | |