高等学校教材

操作系统高等教程

郑衍德 徐良贤 主编

6

上海交通大学出版社

操作系统高等教程

郑衍衡　徐良贤　主编

上海交通大学出版社

内 容 提 要

本书系按机电工业部的工科电子类专业教材1986～1990年编审出版规划，由计算机教材编审委员会征稿，推荐出版的。全书共分十三章，分别介绍了多机操作系统的原理；网络操作系统的功能及通信原语的设计；分布式系统的特点、命名系统、同步、容错技术及进程分配；保护机构；并发程序设计概念及其语言；操作系统的性能评价。本书内容新颖、选材适当、布局合理、叙述清晰，适于作研究生教材和本科生选修教材，亦可作为从事计算机工作的科技人员的参考书。

操 作 系 统 高 等 教 程

出　版：上海交通大学出版社
（淮海中路1984弄19号）
发　行：新华书店上海发行所
印　刷：崇明永南印刷厂
开　本：787×1092（毫米）1/16
印　张：15
字　数：365000
版　次：1991年5月 第1版
印　次：1991年6月 第1次
印　数：1—2650
科　目：246—318
ISBN7—313—00860—0/TP·39
定　价：3.90元

出 版 说 明

根据国务院关于高等学校教材工作分工的规定，我部承担了全国高等学校、中等专业学校工科电子类专业教材的编审、出版的组织工作。由于各有关院校及参与编审工作的广大教师共同努力，有关出版社的紧密配合，从1978年至1985年，已编审、出版了两轮教材，正在陆续供给高等学校和中等专业学校教学使用。

为了使工科电子类专业教材能更好地适应"三个面向"的需要，贯彻"努力提高教材质量，逐步实现教材多样化、增加不同品种、不同层次、不同学术观点、不同风格、不同改革试验的教材"的精神，我部所属的七个高等学校教材编审委员会和两个中等专业学校教材编审委员会，在总结前两轮教材工作的基础上，结合教育形势的发展和教学改革的需要，制订了1986～1990年的"七五"(第三轮)教材编审出版规划。列入规划的教材、实验教材、教学参考书等近400种选题。这批教材的评选推荐和编写工作由各编委会直接组织进行。

这批教材的书稿，是从通过教学实践、师生反映较好的讲义中经院校推荐，由编审委员会(小组)评选择优产生出来的。广大编审者、各编审委员会和有关出版社为保证教材的出版和提高教材的质量，作出了不懈的努力。

限于水平和经验，这批教材的编审、出版工作还会有缺点和不足之处，希望使用教材的单位，广大教师和同学积极提出批评建议，共同为不断提高工科电子类专业教材的质量而努力。

机械电子工业部教材办公室

前　言

本教材系按机电工业部的工科电子类专业教材1986～1990年编审出版规划，由计算机教材编审委员会征稿，推荐出版的，责任编委为东北工学院李华天教授。

本教材由上海工业大学郑衍衡．上海交通大学徐良贤担任主编，上海交通大学陈文颖、林冬梅、谈宏等参加了编写，复旦大学张然担任主审。

本课程的参考学时数为50学时。全书共分十三章，第一章、第二章介绍多机操作系统的原理；第三章、第四章介绍网络操作系统的功能及通信原语的设计；第五章至第九章分别介绍分布式系统的特点、命名系统、同步、容错技术及进程分配等内容；第十章介绍保护机构；第十一章，第十二章介绍并发程序设计概念及其语言；第十三章，讨论操作系统的性能评价。

本教材是在操作系统原理的基础上，介绍多机、网络和分布系统的构造和原理，以及随之而产生的通信、同步，一致性及保护等问题。要求使用本教材的读者具有操作系统、体系结构及其他相应课程的基本知识．本课程是一门实践性较强的课程，有条件的读者应注意结合实际学习。

本书第一章、第二章由谈宏等编写，第三章至第九章，第十三章由郑衍衡编写，第十章由陈文颖，林冬梅编写。第十一章、第十二章由徐良贤编写。

本书在编写过程中得到李华天教授的热忱关心和指导。张然详细审阅了全稿．并提出了许多改进意见。尤晋元，杭诚方对本书的编写给予了大力的支持和帮助，在此一并表示诚挚的感谢。由于编者水平有限，书中错误和不妥之处在所难免．恳请广大读者批评指正。

编者　　1990年8月

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第一章　多机操作系统

§1-1　多处理机系统结构

存储器用来存储像程序和数据这样的客体，处理机用来运行进程。计算机系统结构可按对进程和客体提供物理资源的不同来划分。操作系统要在用户之间分配资源，适于操作系统的资源分配算法要因不同的系统结构和使用目的而不同。

一个存储器单元用于存储所有活动客体，而一个处理器被所有进程分享，这种结构叫做多道程序单处理机(multiprogrammed uniprocessor)。这个处理器是这样被多个进程分享的：一个进程或已执行完，或不能继续进行(亦即被阻塞)，就把控制权交给另一进程。这被称为进程切换。处理机必须被合理调度，以便没有被阻塞的进程不会无限期地等待下去，因此，进程切换会频繁进行，而进程切换的速度对一个系统来说非常重要。

最早期的计算机就是单处理机，直到最近单处理机仍占主导地位。在改进其构件性能，提高其集成度方面已做了许多努力，如采用了更高速的电路，更快的内存和磁盘，以及高集成度、低能耗、散热性好的芯片等等。多道程序单处理机虽然是目前应用最为广泛的系统，但半导体技术的发展已使多处理机系统发展起来，下面介绍这类系统的构成。

一、多机系统

在一个多机系统中，有 n 个处理器，其优点是能进行真正的并行处理，亦即 n 条指令能被同时执行，其用途为：

(1) 如果一个程序能被分成几部分，并像进程一样分别独立运行，则会减少程序的运行时间。

(2) 并行执行用户的几个程序，提高了系统的吞吐量。

(3) 多处理机实行冗余操作，可实现高可靠性系统。

上述 (1) 用的很少，因把一个大型程序分解成 n 个较小的单元且能并行执行，需要用到相当高级的程序设计方法，这样的方法已经用来设计某些特定程序，如操作系统和实时系统，但在一般的程序设计中不常用。因此，一个操作系统的多个进程可分别在不同的处理机上独立运行，但用户程序在某特定时间内通常只在一个处理机上运行，像(2)中那样。注意，要允许一个用户程序能分成一个进程组在不同处理机上运行，也要求操作系统能识别并有支持用户程序多进程并行处理的机制。但目前多数商品化的操作系统还没有提供这种机制。

情况 (2) 是在多机系统应用中最流行的。多机系统的使用使进程切换时间减少，从而可以使众多用户分享价格昂贵但速度慢的外部设备(以下简称外设，)如打印机、磁盘等，使用这样的系统比较经济。长期以来，多机系统一直被认为是单处理机的逻辑扩充，至今为止所建立的多机系统还不很多，各处理机作用基本相同的并行处理机则更少。

情况 (3) 对实时系统应用较广，因要求高可靠性。通常是两台处理机一起使用，一台

正常运行；另一台是冗余的，做相同的工作。

二、处理器与存储器之间的连接

多处理机的一个明显特征就是多个处理器之间共享内存。处理器与存储器之间的不同连接方式则在很大程度上决定了多处理机的体系结构。

最简单的方式是通过多端口存储器来连接，如图 1-1，共享的存储器被分为几块，每一块有 n 个端口。每个处理器连到各块的其中一个端口上。这就使不同处理器能并行访问不同存储块，但对同一块的访问必须是顺序的。各块内的硬件控制器按固定算法控制其访问。这种连接方式最简单，通常用于处理器数目较少的情况。因这种方式的缺陷是，各存储块端口数是固定的，而每一个处理器必须与所有存储块相联，端口数目限制了处理器的数目。这个缺点可这样克服，把各存储块内的控制器换成一个集中控制开关，如图 1-2 所示。

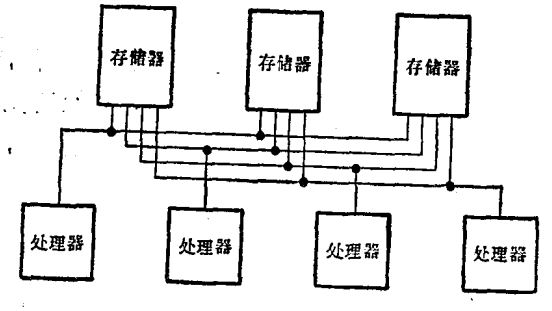


图 1-1　多端口存储器方式多处理器

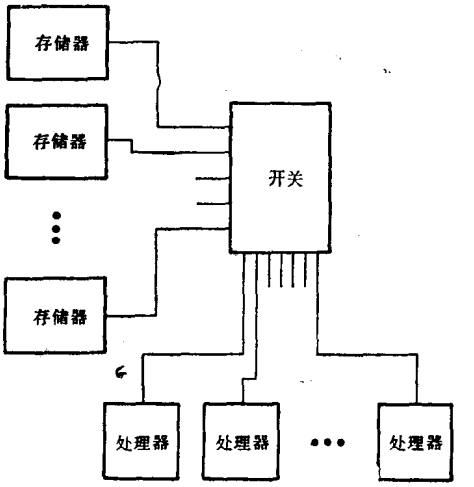


图 1-2　集中控制开关方式多处理器

集中控制开关称为纵横制门闩开关(crossbar switch)，它给出处理器到存储器的访问路径。其优点是比多端口存储器方式支持更多的处理器。缺点是控制开关单元比较复杂，降低了系统的可靠性。若有 m 个存储块，n 个处理器，则连接路径数目为 $n \times m$，且门闩开关接通这些路径时还不能有明显的延时。

通常，多机系统有一个严重的不足：当处理器竞争访问共享存储器时，就会相互间因冲突而延滞，从而使各处理器访问内存的平均时间增加，这就限制了连接到一个存储器上的处理器的数目。已经设计了几种方案来消除这种存储器竞争，如引入多端口存储块，使对每个存储块的竞争减少。还有一种方法是，在处理器和开关之间增加高速缓冲存储器(cache)，高速缓冲存储器中存有处理器刚访问过的共享存储单元中的内容，当处理器访问内存时，若内容在高速缓冲存储器中，则只要访问高速缓冲存储器即可；否则再访问存储器，并把一份拷贝存入高速缓冲存储器中，这样可使处理器访问内存次数减少。但高速缓冲存储器也带来了新问题：因有几个处理器，一个存储器单元中的内容可能同时在几个高速缓冲存储器中有副本，所以当一个处理器改变了这个单元内容时，就要通知其他处理器，以便更新其各自高速缓冲存储器中的内容。一般不会因此而设计特殊的硬件来实现这一过程，有时规定只有只读数据(比如指令)写入高速缓冲存储器。

还有一种方法是为每个处理器配备一个局部存储器，如图 1-3 所示。

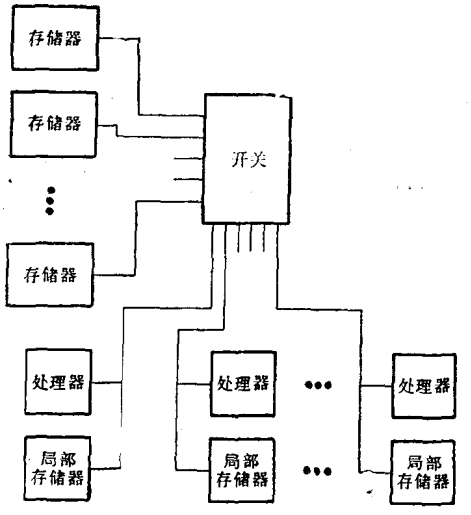


图 1-3 有局部存储器的处理器

局部存储器的引入导致了不对称性，任一处理器都不能访问所有存储器。而只能访问共享存储器和各自的局部存储器。局部存储器也由操作系统控制，把一个进程频繁使用的数据放到一个局部存储器中，可以使之起到一个高速缓冲存储器的作用。然而在运行一个进程时必须小心，当处理器能访问存有进程指令和数据的存储器时，把在一个处理器上执行的进程移到另外一个处理器上运行的代价是非常高的。

集中控制开关单元的另一个缺点是：连接线路数目是在开关设计时预先定好的，因此不能再增加处理器数目。要提高可靠性和可扩充性，分布式开关模式可解决这一问题，如图 1-4 所示。

这种模式中有几个开关，各连接到一个处理器及其局部存储器上，局部存储器可被各处理器访问。这样，所有处理器共享寻址空间，存储器在物理上是分布式的，每个存储器单元同一个处理器相连。一个处理器可直接访问其局部存储器，而访问其他处理器的局部存储器就要经过开关切换。切换策略称为包切换(packet switching)。

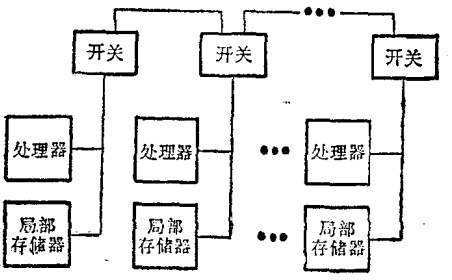


图 1-4 分布式多处理器

在包切换中，在一处理器及其局部存储器之间无直接电路相联，而是局部开关把请求信号作为一个消息(或信息包)发给目的开关，甚至开关之间也不必直接相联，只要通过中间开关联起来即可。当目的开关得到消息后，就把自己局部存储器中的数据作为一个信息包发回给源开关。当发出一非局部存储器请求信号后就开始等待，此期间它不占用局部总线，以便局部开关可接收来自其他处理目的的请求。这种结构称为分布式多处理器。它类似于通过电话线或电缆线相联的计算机网络，但有一个重要区别：分布式多处理器系统的所有处理器共享整个存储器。与此相反，网络中的计算机都有其局部存储器而没有共享存储器。

分布式多处理器的主要优点是没有瓶颈。但其在访问非局部存储器时有一定延时，这取决于访问的存储器与处理器的距离。在这样的系统中，操作系统的设计一定要考虑尽量减少这种延时等待。

三、输入/输出结构

上一节讨论了处理器与存储器连接的各种方式，这一节描述处理器和输入/输出设备间的连接模式。我们讨论下面几种方式：

(1) 多端口设备。

(2) 由门闩开关连接的处理器和设备。

(3) 只同一个处理器相连而不能被其他处理器访问的设备。

(4) 只同一个处理器相连但可被其他处理器访问的设备。

模式 (1) 中，设备的每个端口同一个处理器的总线相连接，设备可用此端口进行数据传输和中断处理。设备可以用不同端口进行不同处理。然而这与多端口存储器不同。很少有人设计多端口存储器的控制器，故这种模式应用不多。

模式 (2) 与处理器存储器的门闩开关相似，但设备开关要更复杂一些，因一个设备同一个处理器保持联系的时间在一个事务处理过程中(比如磁盘读、写)可能很长，因而用软件来裁决比用硬件实现会更方便。

模式 (3) 是很普遍的。一个处理器只能访问与之相联的外设，访问非局部设备只能用软件方式，向其他处理器发出一个请求，使能在其连接的外设上操作。

当使用分布式开关，且用存储器地址而不用I/O指令来访问I/O设备时，就用模式(4)。在这种方式下，因I/O操作是由往一个地址中写数据来初始化的，所以不用分布式开关做额外工作，处理器就可方便地访问远程设备。但从设备到处理器的路径却是不对称的，尽管处理器可访问任一设备，但设备发出的数据和中断信号却只能由与之连接的处理器处理。这种不对称性的原因如下：

(1) 设备发出的中断信号必须得到快速反应，这对远程处理器来说是做不到的。

(2) 到非局部存储器的数据传输有不同的时间延迟，此延迟有可能比设备读或设备写的速率大。

上述几种模式对操作系统的外设控制进程有不同的要求。模式 (1) 和模式 (2) 的设备进程要在每个处理器上运行。模式 (3) 的设备进程只要在可访问该设备的处理器上运行。模式 (4) 的设备进程操作要分成能在所有处理器上运行和在同设备相连的处理器上运行的两种。

四、处理器间通讯

处理器间共享的内存可用来进行进程间通讯。比如，两个进程可以通过驻留在共享存储器中的监控器进行通讯。但也有这种情况，通过监控器进行通讯有时不太方便，比如，若I/O结构是不对称的，一般进程初始化一个非局部设备，只能通过与能控制该设备的设备进程进行通讯。若通讯是通过监控器进行的，则只有当设备进程被调度执行时才能进行I/O设备初始化。这样，对进程请求的反应就太慢了。

为提高进程通讯的反应速度，需用另一种机制。处理机间中断(IPI)机制可以用中断信号中断其他处理机，当一处理机收到一IPI信号就把它交给相应的进程，这样可提高响应速度。不同的系统有不同的实现机制。有的处理机可接收两次中断。例：处理机A中断处理机B，处理机C也可中断处理机B，而在另一些系统中，处理机B只有当没有其他中断请求时才能接受A的中断。

§1-2 多机处理的基本问题

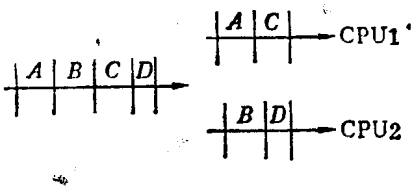
多机处理的中心设计问题是选择一种方法，把计算任务分配到各个处理机上，从而在每

此为试读，需要完整PDF请访问：www.ertongbook.com

个工作步执行期间，每个处理机上运行逻辑连续的指令及数据。考虑这样一个例子：在单处理机上有一串指令流，假设按图 1-5(a) 描述的那样，把这串指令流分成 A、B、C、D 四个计算任务。一个计算任务可能是一条简单的机器指令或是几千条机器指令。在多处理机系统中，设计的中心问题是如何划分这些计算任务，即给出一个把单处理机指令流分为几个计算任务的划分，这种划分可使这些计算任务同时在两台或更多台的计算机上重叠运行。图 1-5(b) 描述了一个重叠执行的任务划分。

在单处理机上与此类似的问题是：给出两个顺序的计算任务 A 和 B，如图 1-5 (a) 所示，在什么条件下可以重叠这些任务？例如，任务 B 处理 I/O 中断，任务 A 运行时要禁止 I/O 中断，那么要使任务 B 在任务 A 执行前或执行期间能被执行，即在任务 A 执行时允许 I/O 中断会产生什么结果呢？

若任务 B 逻辑依赖或数据依赖于任务 A，则单处理机上的这两个任务不能被重叠，即若 B 逻辑依赖于 A，那么 B 能否被执行则依赖于 A 的执行。



(a) 单处理机　　(b) 多处理机

图 1-5　计算任务的关系

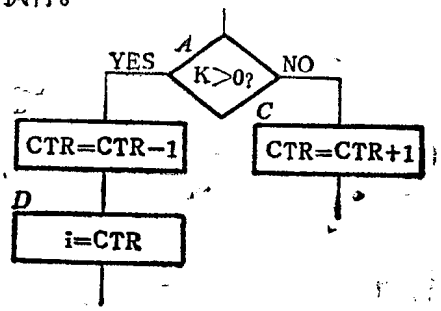


图 1-6　逻辑与数据依赖

在图 1-6 中，任务 B 和 C 逻辑依赖于任务 A，所以 B 或 C 能否被运行均依赖于 A 的执行结果。图 1-6 中的任务 B 与任务 D 之间存在数据依赖关系，任务 B 的输出结果是任务 D 的输入。如果把"数据"这个概念扩充到存储地址、寄存器或者其他任务的输入/输出，那么数据依赖会有其他的产生来源。

如果两个顺序的单处理机任务逻辑不依赖，并且数据也不依赖，它们就可以被重叠。在多处理机系统中，当计算任务被分配到各个处理机上时，必须注意任务间的逻辑与数据依赖关系。

如图 1-7(a) 所示，一个任务 T_1 可被看作是输入数据 I_1 到输出数据 O_1 之间的映射。图 1-7 的 (b)，(c)，(d)和(e)介绍了两个任务的输入/输出之间的各种关系。

在图 1-7(b) 中，两个任务的数据不交叠。在图 1-7(c) 中，输入数据重叠。图 1-7(b) 和1-7(c) 对于多机处理不会产生问题，但图 1-7(d)和1-7(e) 则会有不一致的结果发生，因此对这样的情况必须加以特别预防。

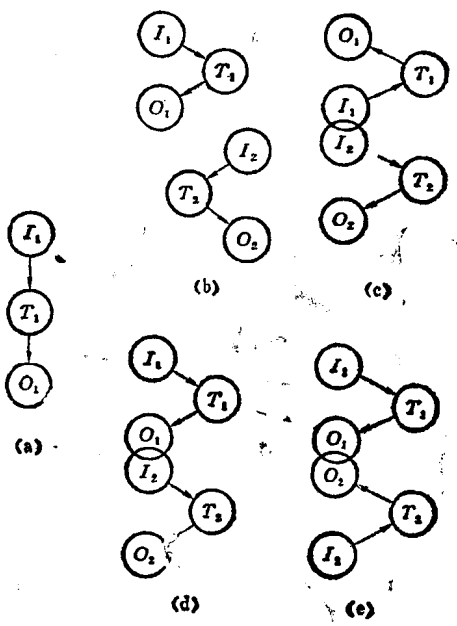


图 1-7　多机处理的数据依赖

处理数据依赖一般采用下面四种策略：

(1) 利用加锁机制对共享数据的使用顺序化。

(2) 把共享数据段复制给每个处理机。

(3) 强制所有使用共享数据段的任务在某一特定的处理机上运行。

(4) 重新设计任务，使数据冲突达到最小。

许多利用多处理机进行研究的理论问题往往在单一的程序中集中了许多并行计算。在单一程序的上下文中，逻辑依赖性的问题是非常突出的。这就要求用一些特殊的语言细节来区分程序中的哪些部分可以并行执行，哪些程序点又需要同步。虽然这种细颗粒的多机处理方法对许多运行环境来说是很重要的，但也有一些大型系统不采用这种办法。这些系统允许许多基本上没有逻辑依赖的任务竞争处理机资源，例如两个批处理作业或两个分时用户，它们除了进行系统调用以外基本不存在逻辑依赖关系。

§1-3 多处理机操作系统分类

概念上多处理机操作系统的要求与支持多道程序的大型计算机的操作系统没什么不同，然而，当多个处理器同时工作时，操作系统的复杂性就增大了。复杂性还表现在多处理机操作系统还要支持异步任务并发执行。

支持多道程序设计的操作系统功能包括：资源分配和管理、内存和数据集保护、死锁防止、流产进程中止处理及意外处理等。除此之外，多处理机操作系统还要充分利用系统资源、处理系统输入/输出，解决处理机负载均衡问题等。使用多处理机系统的主要原因是提高可靠性和在发生故障时能降级使用。因此，操作系统必须提供系统结构重组的能力，以支持系统的降级使用。如自动支持硬件和运行程序的并行性开发，从而使系统的负担加重。一个性能不好的操作系统会抵消多处理器的优点。所以为一个多处理器计算机设计一个高效的操作系统是最为重要的。

在一个系统中有多个处理器单元，这与传统的操作系统设计不同，众多数目的处理器对操作系统设计的影响仍值得研究。处理器的模块化，及其之间的相互连接也影响操作系统的设计。更进一步，通讯机制、同步机制、分配策略都决定着操作系统的效率。下面仅就现有多机系统中的一些基本结构予以介绍。

在多机操作系统设计中目前基本上有三种组织结构：主从结构(master-slave configuration)，独立监控结构(separate supervisor)和移动式监控控制(floating supevisror control)。大多数多处理机用的操作系统是第一种主从结构，这种结构的监控程序在一台处理器上最易于实现，且它最容易从单机多进程操作系统扩展实现。虽然这种结构比较简单，但对整个系统资源的控制、利用效率方面均不如另两种结构的操作系统。

主从结构的操作系统是由一台主处理器记录，控制其他从处理器的状态，并分配任务给从处理器。比如，Cyber-170 就是这种结构，操作系统在一个外围处理器 P_0 上运行，其余所有处理器，包括中心处理器及外围处理器都从属于 P_0。另一个例子是 DEC System10，有两台处理机，一台为主，另一台为从，操作系统在主处理机上运行，另一台作为子调度的资源。

因监控程序一直在同一台处理器上运行，从处理器的请求就要通过陷入或主控机呼叫传送给主控制处理器，然后主控机回答并执行相应的服务操作。主从结构操作系统还有其他一

些特点。监控程序及其相应过程不必重装入，因为只有一个处理器用到它们。然而，当不可恢复错误发生时，系统很容易导致崩溃，需要重新启动主处理器。此外，当主处理器不能迅速处理进程时，从处理器的使用率会比较低。主从结构当各处理器工作已定好或由于不对称性，从处理能力较弱时，效率比较高；当处理器数目不多时也可使用这种结构。

独立监控结构与前面介绍的主从结构差别很大。有点类似计算机网络，监控程序在每个处理器上运行，资源共享层次比较高，如共享一文件结构。每个处理器为自己的需要服务。然而因处理器间要发生相互作用，监控程序能重装入或在各处理器上复制独立的副本是很有必要的。虽然每个监控程序有自己的一组私有表，但也有部分表是公用的，供整个系统共享，这就产生了表访问的困难。访问共享资源的方法取决于处理器间关系紧密程度。这种结构不至于象主从结构那样易于崩溃。每个处理器有自己的I/O设备和文件系统，但I/O的重安装仍是手工操作的。

监控程序在各处理机中的复制会占用大量内存。有一种解决办法是，把经常使用的操作系统代码放到各自的高速缓冲存储器中，而把使用频度低的代码放在共享存储器中。但遗憾的是，判定哪一部分操作系统代码使用频度高是非常困难的，并且需依赖于工作负荷。

移动式监控控制模式把所有处理器与其他资源一视同仁，这是一种最难于实现、效率也最高的模式。虽然几个处理器可以同时执行监控服务程序，但监控程序仍然要从一个处理器移动到另一个处理器上。这种系统可以使所有资源负载比较均衡。服务请求冲突可以由优先级来解决，优先级是由统计决定或动态决定的。因共享程度高，所以监控程序代码都是可重入的。这种系统中表访问冲突和表锁定是不可避免的，因此应控制这些访问以保护系统。这种系统的一大优点是具备系统降级的能力，而且它提供了真正的冗余，实现了对资源的有效利用。这种操作系统的例子是MVS和VM，应用于IBM3081机上，及在C.mmp上的Hydra中。

很多操作系统不纯粹是上述三种类型中的一种。但有一点是可以肯定的，设计的第一个系统一般属主从结构的，而最终被研究探索的是移动式监控控制方式。下面简要总结前述三种类型的多机操作系统的优点，缺点及主要特点。

1．主从式操作系统

(1) 操作系统程序在一台处理机上运行。如果从处理机需要主处理机提供服务，则向主处理机发出请求，中断其进程，主处理机提供服务。因只有一个处理机上执行操作系统程序，所以该程序不必可重入。

(2) 只有一个处理器执行监控，所以不会发生表冲突及表锁定。整个系统的结构不易改变。系统只需相对比较简单的软件和硬件。

(3) 当主处理机失败或发生不可恢复错误时，整个系统会崩溃．这就要人工干预重新启动系统。

(4) 若主处理机的任务分配程序不能保证从处理机一直处于工作状态，则从处理机的空闲时间可能比较长。

(5) 这种类型的操作系统适于任务负荷已定好的系统或不对称系统，从处理机功能比主处理机少。

2．独立监控结构的操作系统

(1) 每台处理机为自己服务。事实上，每台处理机有自己的I/O设备、文件等。

(2) 监控程序必须可重入或复制，使每个处理机中有一个独立的副本。

(3) 每个处理机有自己的一组私有表，但有些表要全系统公用．这就产生了一些表访问冲突问题。

(4) 这种操作系统同主从结构系统一样不坚固，而且当系统失败后不容易重新启动。

(5) 由于（4）的原因，使 I/O 设备的重装通常需要人工干预。

3．移动式监控操作系统

(1) 监控程序会从一个处理机移动到另一处理机上，虽然有可能几个处理机同时运行监控服务程序。

(2) 这类操作系统可使所有资源负载均衡。

(3) 服务请求冲突由静态或动态设定优先级的方式来解决。

(4) 监控程序的大部分代码要求是可重入的，因几个处理机可能同时运行其同一服务程序。

(5) 会发生表访问冲突并会产生锁定，但没有方法避免；重要的是不能造成整个系统的崩溃。

第二章　多机操作系统原理

§2-1　多处理机调度策略

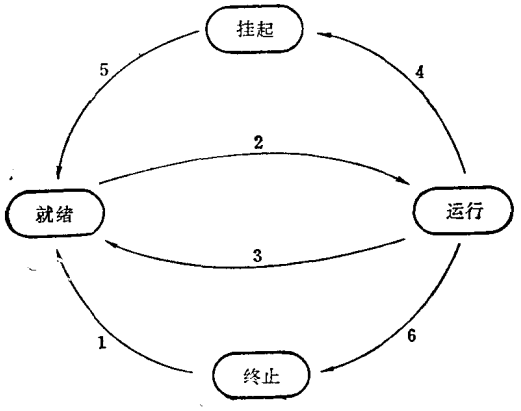
在本节中，我们讨论在多处理机系统中处理机的管理技术。多处理机的引入使调度问题复杂化。为了评价调度规则，必须确定模型和随机模型。总之，在多处理机系统中寻找处理机分配的最优算法的计算量是很大的，然而，有一些动态调度模型则很接近最优算法。

一、多处理机管理的维数

对于计算机工程领域的研究者来说，多处理机的管理和调度是非常引人入胜的。多重部件，特别是异体机和非对称机，增加了操作系统及其应用程序的管理工作量。从最一般的情况而言，为了达到最优化，必须在不同的处理机上调度进程。这包括从一组进程中选择一个供运行的进程。

在多处理机系统中，有两个最基本的资源分配问题：一个是在哪里存放代码和数据，即 placement decision；另一个是将进程分配在哪个处理机上运行，即 assignment decision。在单处理机系统中，这些问题显得无足轻重，因为分配是显而易见的。而且在单处理机系统中，处理机可以访问任何物理存储地址空间，所以不存在可访问性的问题，存储器竞争也可由交错访问解决。经常把分配问题称为处理机管理。它描述作为共享资源，在外部用户和内部进程之间如何管理处理机。所以，处理机管理包括两种基本调度：长期外部装载调度和短期内部进程调度。

一般而言，进程在它的生存期中要经历不同的状态转换。一个进程若正在使用处理机，它就处于运行状态。一个挂起的进程将进入挂起进程的队列。一个进程之所以被挂起是因为它在等待犹如 wake-up 之类的外部信号来唤醒。唤醒操作改变进程状态，使之成为占有处理机的就绪状态。图 2-1 示出进程状态的改变。



状态转复	事件
1	激活进程
2	运行进程
3	抢占进程
4	锁入进程
5	唤醒进程
6	终止进程

图 2-1　进程的状态及其改变

在这层的调度者实现短期进程调度：从就绪进程中选取一个执行进程。被选中的进程将在处理机上运行。中期和长期装载调度是用于选择和激活一个新的进程，使之进入运行环境，即使新进程进入就绪队列。长期装载调度也负责控制并行的程度(即活动进程的数目)，因为若活动进程数目太多，会引起系统颠簸。

每当运行进程被挂起或遇到更优先的进程时，进程调度者就工作。它要在就绪进程中选择一个运行进程。进程调度者存在于核中，可以被认为是就绪队列的管理者。因为它可能是系统中最经常被调用的程序，故应写得相当简洁，以减少操作系统的开销。

就绪队列既可以是局部的，也可以是全局的。一个局部就绪队列可以和一个具有局部内存的处理机联系起来。所以一个进程一旦被激活，最终会占有处理机。局部就绪队列减少了存取队列的时间，因而也减少了进程调度者的开销。然而，局部就绪队列的概念限制了进程迁移，而且在小型系统装载中，处理机的使用不能平均分布。为了允许进程迁移，使用了存储在共享内存中的全局就绪队列。这使得进程调度者要花费额外开销存取进程的状态，然而处理机使用的不均衡是很小的。

绝大多数理论上的调度算法的一般目标都是达到使用最少的处理机在最短时间内完成并行程序这样的处理机分配和进程调度技术，而且在处理机数目固定时，要改进处理机分配和进程调度的算法，尽量减少并行程序的运行时间。

有两种基本的进程调度模型：确定模型和随机模型。确定模型的定义是：在解决调度问题之前，有关问题的所有特征都已清楚。这些特征包括每个任务的执行时间和任务之间的相互关系。调度目标是要优化一个或更多的评价标准。例如，在确定模型中，每个进程的运行时间可以表示为最大运行时间或期望运行时间。对于前者，完成调度的时间被认为是系统完成的最大时间；对于后者，完成调度的时间表示了计算的大约估计时间。实现这个目标的动机是：在很多情况下，一个不好的调度程序会造成难以忍受的响应时间和极不合理的系统资源的利用。

确定模型并不十分现实，它没有考虑多机系统中不规则和非期待的要求。因而，随机模型就经常被用来研究动态调度技术。在随机模型中，进程运行时间是一个随机变量与一个给定的积累分布函数 $F(cdf)$。

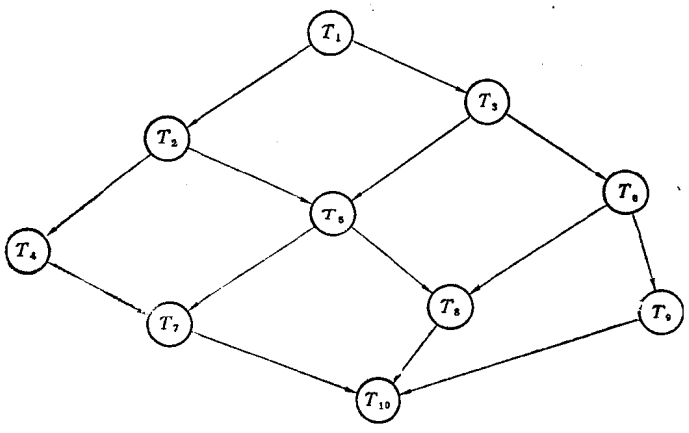


图 2-2　任务系统(工作)的表示，$T=\{T_1, \cdots, T_{10}\}$

动态调度意味着在一个特定的时刻，将进程或任务分配到一个特定的处理机上。因为同时有很多任务都可作为候选，所以需要有一个可以方便地体现任务之间关系的表示方法。一般地，我们把一个进程集合称为任务系统或工作（job)。一个包含进程集合的工作可以用一个优先图来表示，如图 2-2 所示。图中的结点表示具有独立操作且需要联系的任务。结点的集合代表了进程的集合 $T=\{T_1, \cdots, T_n\}$，结点之间的有向边表示进程之间存在着偏序或优先关系"$\prec$"。所以

此为试读，需要完整PDF请访问：www.ertongbook.com