计算机操作系统的设计与实现

——UNIX操作系统结构设计

仲萃豪　杨芙清　刘日昇　孙玉方

中国船舶工业总公司 第七研究院第七〇九研究所

一九八二年九月　武汉

重 印 说 明

本书以当前国际上著名的 UNIX 操作系统为背景，系统地介绍了操作系统的功能设计与结构设计方法。今年六月在成都由中国科学院计算技术服务社与中国计算技术服务公司联合举办的“UNIX 操作系统学习班”上刘日升和孙玉方两同志讲授了本书（初稿）。为了满足举办讲习班之急需和更广泛地听取各方面的意见，重印了本书。在重印前，主要由刘日升和孙玉方两同志对初稿作了重点修改。

在本书的编写及讲授过程中得到国内许多同志的热情支持，他们提出了许多保贵意见。这次重印过程中得到了七〇九所领导以及九室和印刷厂的同志们的大力支持。在此，我们表示衷心的感谢。

由于水平所限，时间仓促，书中肯定会有不少错误或不当之处。恳请读者给予指正。

作 者

一九八二年八月

前　言

UNIX操作系统是当前世界上最为著名的一个操作系统。国外的许多名牌计算机都配有这个系统，在小型机和高档微型机上它占据了统治地位。它的出现标志着操作系统技术已发展到一个新的里程碑。这个系统的主要特点是简洁，然而它却具备很强的功能，用户使用方便。为此我们编写了这本讲义，依此向国内读者介绍UNIX。

以某个操作系统作为实例来介绍操作系统的书已有不少，如 MVT、MULTICS 和 SOS 等，但成功的不多。有些系统本身过于庞大，结构复杂，难以交待清楚；有些实用性系统虽然可以使学生把理论和应用结合起来，但却忽略了实际系统中的复杂性。UNIX 系统的出现给操作系统的教学提供了一个合适的实例。一方面它本身短小精悍，另一方面在功能上又比较完善。因而，国外已有不少大学，把 UNIX 系统作为操作系统教材，收到了很好的效果。我们这本讲义以教材的方式编写，试图在一学期的课程内，使学生掌握较为先进的操作系统技术，并为自己动手设计一个操作系统打下良好的基础。这就是我们写这本书的动机和愿望。

作为教材，在内容上必须保持其系统性和完整性，为此我们把 UNIX 系统重新加以整理，有重点地、系统地讲述整个系统；另外还补充了某些必需的概念和原理及其它系统采用的一些技术。

另一方面，为了弥补当前一些操作系统讲义的不足，本书除了介绍操作系统功能外，重点介绍了操作系统的结构设计。按模块化的结构设计，整个系统可以分成若干层和若干个模块，每个模块集中管理某一种系统资源，每种资源的管理也就是系统中的某一个功能。根据这一设计原则，我们把 UNIX 系统的结构重新加以组织。这样，既便于剖析整个系统，又能介绍一些较为先进的结构设计技术和概念。

为了便于理解和修改，我们在保留 UNIX 原有功能和算法的基础上，用 XCY 语言重写了这个系统。改造后的 UNIX 系统结构是一个从底向上逐步扩充的抽象机，本书的叙述次序基本上也是按照这种方式进行的。但为了讲述方便，有时把实现相近功能的不同层次上的模块，或不同模块内的程序合在一起讲述，与此同时介绍与之有关的一些基本概念、原理及其它实现技术。

本书第一章介绍了 UNIX 系统的特点。第二章介绍了操作系统结构设计的几种常用方法。第三至第十章逐层介绍抽象机的各个组成模块。在我们读完全书以后，再回顾前两章，可以更进一步领会结构设计中的问题。本书的附录中介绍了 UNIX 的命令语言 shell，还收入了 XCY 语言的语法公式和使用的汇编子程序说明。

目　　录

第一章　UNIX操作系统及其主要特点

1．1 操作系统的形成过程

自从50年代末期开始，电子计算机进入了第二代，系统软件得到了发展，先后出现了各种语言及其编译程序。到60年代初期，计算机硬件提供了通道、中断和存储管理机制，为多道程序设计技术提供了物质基础。所谓多道程序就是在内存中同时存放几道用户程序，在单机情况下，任何时刻总有一道程序占用处理机执行计算。当该程序因等待外部设备传输而工作不下去时，系统马上令处理机执行另一道程序，从而使主机得到充分的利用。在这种思想下，负责管理多道程序工作的系统就是操作系统，它具有分配和管理系统的各种资源，以便充分发挥系统功效的能力的作用；操作系统另一作用是为了方便用户使用计算机，在用户与硬件机器之间起着接口的作用。

自60年代中期开始，计算机又进入到以集成电路为主要特征的第三代。这时出现了更多的高级语言和其它应用程序。操作系统的功能更加完善，已成为计算机系统中不可缺少的一部份。首先出现的大型操作系统是多道成批系统，系统成批地处理一批作业，在内存中有多个作业同时存在，在辅助存储器上还有大量的后备作业，可以随时补充。因此，这种系统有相当灵活的调度原则，易于选择搭配合理的作业，从而使系统中各种资源能够充分发挥它的利用率。然而“成批”处理不允许用户和机器之间发生交互作用，这样，就迫使用户改变他们原来的手工操作方式，必须针对作业运行中可能出现的各种情况，事先在操作说明书中规定好相应的措施。这对调试和试算作业是很不方便的，由此出现了分时操作系统。

分时系统允许一台计算机上可以挂上几个控制台和上百台终端，系统把处理机时间划分成很短的时间片，轮流地分配给各个联机作业使用。如果某个作业在时间片结束之前计算还未完成，那末该作业就被暂时中断，等待下一轮再继续计算，此时处理机让给另一作业使用。这样，每个用户的各次要求都能得到快速响应，因而给每个用户的印象是，好象只有他自己独占计算机一样。这种工作方式可以及时地编写、调试、修改和运行自己的程序，因而加快了解题周期，也利于在一台终端设备上随时随地使用计算机。

在今天，计算机硬件成本不断下降，计算机中各资源充分发挥效用已下降到次要地位，方便用户使用上升为主要矛盾，因此，分时系统就比成批系统更受用户欢迎。

操作系统的另一发展是实时系统，特别是在微型机和小型机大大普及的情况下，计算机的应用范围迅速扩大，从传统的科技计算推广到各行各业。例如，军事上的导弹发射控制、工厂的生产控制、医疗诊断及飞机订票等。实时系统分为实时信息控制及实时信息处理。他们的共同特点是响应时间要求快、完全性和可靠性要求高。一旦用户向实时系统提出服务请求，就要求系统立即安全可靠地处理。

自70年代中期开始，操作系统的主要发展方向有以下几个方面。

首先出现了一批通用的操作系统；如同时能处理分时作业和批作业的分时系统、同时处

理实时要求和批作业的实时系统等等，操作系统功能更加完善，规模更加庞大。

第二，大型数据库管理系统的出现，要求操作系统提供更强的支援，要求操作系统提供更灵活、更完善的文件系统和较强的实时和分时处理能力。为了适应网络软件的发展，要求操作系统中实现网络中的各级协议，配置完善的通讯软件和网络控制软件，并且要有较强的分时处理能力。

第三，大量廉价的微型机投入市场提出了由多台微型机组成一个大系统的要求，从而出现了分布式操作系统。

第四，操作系统设计的理论和工程研究。研制操作系统时的主要困难是：花费大、周期长、可靠性差，而且还难读、难修改、难扩充和难维护。其根本原因在于系统规模过于庞大、结构过于复杂。由此促使人们进一步探讨操作系统应如何进行设计，提出了许多理论性和工程性的问题。这主要包括以下三个方面。(1) 从理论上着手开展了一系列的研究，如系统模型的描述，各种实现算法的分析，死锁问题，并发程序的通讯等。(2) 从功能上着手总结和提炼，简化系统，UNIX 系统就是在这种情况下产生的，因此它的诞生标志着操作系统发展到新的里程碑。(3) 从结构着手探索新的设计方法，使系统有一个合理的清晰的结构，以利于它的正确性验证，近年来程序设计方法学和软件工程的发展，提出了不少有价值的方法。本书从第二章开始，介绍结构设计的发展概况，以及具体的解决途径，从而引出全书的各个章节。

1.2 UNIX 操作系统的产生

随着计算机应用的普及，计算机价格的不断下降，操作系统由分时系统代替了成批系统。又由于操作系统技术的逐步成熟，其结果必然会出现象 UNIX 那样的操作系统。它的出现，象征着操作系统技术已进入新的阶段。

UNIX 是美国贝尔实验室的 D· M· Ritche 和 K· Thompson 于 1969—1970 年共同研制的。该系统吸收和总结了当时一些成功的操作系统的重要成果，大胆地删去了一些并非必要的功能，在功能设计上进行了成功的提炼，由此获得了一个短小精悍的系统。自1971年在PDP—11机器上运行以来，很快传播开来，受到广大计算机用户的欢迎。接着它被移植到现有的各种名牌计算机上，到1979年底为止，世界各地已有2000多个文本。预计不久将来，它将在中小型计算机上占据统治地位。由此 UNIX 已成为 Bell 实验室的商标，在国内许多进口计算机上都陆续引进了 UNIX 系统或其变种。

1.3 UNIX 操作系统的特征

UNIX 系统是一个通用的分时系统。尽管系统设计中没有提出什么新的概念和方法，但它成功地继承和吸收了当前国际上操作系统设计中的许多成功经验和新的概念。UNIX 的设计者曾参与研制了对操作系统的发展产生过重大影响的 MULTICS 系统。在 UNIX 中，许多技术都是从这个系统以及它的前身 CTSS（世界上最早的分时系统）中吸取来的。但它大胆地删除了这些系统中与基本功能关系不大的部份，去粗取精，大大压缩了系统的规模，使得它能在一台小型机上实现了在许多大型机上也不常见的功能。UNIX 的主要特点如下：

1．具有层次结构且带有可装卸存储卷的文件系统。整个文件系统构成从根目录表开始的树形分级结构。用户可以把自己的存储卷上的文件系统安装到已有的文件系统上，形

成一个更大的层次结构。以后，用户又可以把它整个拆卸下来。

2．在 UNIX 文件系统中，普通数据文件、目录表和外部设备都统一作为文件处理。文件是无结构（即无记录）、无类型的字符流序列。它既可顺序存取，又可随机存取。它提供给用户的是一个简单划一的接口。实现简单，使用方便。

3．系统为用户提供了功能完备，使用灵活的命令语言，即 shell 语言。它既作为系统和用户之间接口，又具有很强的程序设计能力。它既可执行用户打入的命令，又可执行文件（程序）中的命令。因此，用户可以根据自己的需要编写命令过程，存入文件中使用。

4．UNIX 提供了十几种常见的程序设计语言，如 C，FORTRAN77，BASIC，SNOBOL，APL，ALGOL 68，PASCAL 等。此外还提供了大量子系统，如汇编程序，编辑程序，连接装配程序，公式排版程序等。它们不是系统内核部份，而是作为文件放在外存中，当需要时就作为进程的一部份调入内存执行。UNIX 系统的内核程序（即操作系统）只有一万多条语句，而其外围子系统都是由用户或者程序员逐步堆加上去的，目前已达几十万条指令，构成一个完整的系统。

5．UNIX 的绝大多数程序都是用 C 语言写的。这大大方便了系统的理解、修改和移植。反过来，由于 UNIX 系统在世界上的普及，也使 C 语言成为国际上最受欢迎的语言之一。

UNIX系统自从1970年第一个版本使用以来，已经经过了许多次修改，出了若干版本。不久前，已出到第七版。但是这些版本的基本思想一直保持不变。目前，在国际上，以UNIX操作系统为基础已经发展成许多不同种类、不同用途的软件系统。例如，网络 UNIX，微处理机 UNIX，小型卫星机 UNIX，数据库系统 INGRES，程序员工作台 PWB／UNIX 等。

UNIX 的设计采用了面向过程的方法。操作系统的主要功能都集中在内核里，例如有中断捕俘管理，进程管理，外设管理，文件管理，程序对换等。它们常驻内存。其它应用程序作为文件存于外存，按需要调入内存执行。命令解释程序 shell，作为系统与用户的接口，根据用户的命令动态地建立子进程，执行命令文件。

UNIX 系统在功能设计方面取得的成就是无可非议的，但它在结构设计方面存在一些问题。这主要反映在两个方面。第一，程序结构采用的是传统的模块接口方法，程序结构不够清晰。第二，并发进程的同步工具功能太弱。造成系统并发活动的关系复杂，效率下降，甚至导致死锁。除此之外，用于书写 UNIX 的 C 语言比起汇编语言虽然是一个很大的进步，但仍然是比较低级的高级语言，因此写出的程序相对来说要难读得多。此外，虽然 C 语言的作者（也是 UNIX 操作系统的作者之一）告诫别人不要滥用 goto 语句，但在 UNIX 源程序的不少地方却不适当地使用 goto 语句和非正常返回语句，从而破坏了程序的静态结构。

针对上述情况，我们准备这样来介绍 UNIX 系统，即在功能和算法上仍旧保持原来的不变，在结构上，我们把核心部份用 XCY 语言重新改写。这样，一方面便于讲述功能设计，另一方面也介绍了结构程序设计方面的新技术。在结构设计方面作的那些改变将在下面进一步介绍。

第二章　操作系统的结构设计

2.1 引　言

结构程序设计就是要使系统有一个合理的、清晰的结构，以利于它的理解和正确性验证，达到系统可靠、可移植、可维护的目标。操作系统结构设计的主要目的就是为了使操作系统有一个清晰的结构。

在操作系统发展的早期阶段，系统比较简单、规模小，因而未对它的结构进行认真的研究。但随着系统规模的迅速增大，其复杂性达到人们无法控制的地步，成为对人的智力的严重挑战。如例，IBM 360的OS／360操作系统，它是1966年开始设计的，共花费五千人年的工作量，而最后获得的却是一个可靠性很差的系统。每个版本都包含有上千个错误，尽管通过不断地改正，也无法使这个数目在数量级上减少。因此人们开始对操作系统的结构进行了研究，提出了一系列的概念、思想和方法。1968年 E·W·Dijkstra 在他的 THE 操作系统中，首先提出了顺序进程和抽象机的分层结构等设计思想。1970年，P· Brinch Hansen 在他的 RC 4000操作系统中提出了核扩充等思想，完善了 E·W·Dijkstra 的设计思想。到1974年，C·A·R·Hoare 和 P·Brinch Hansen 分别提出了管程结构的设计思想。把上述各种设计方法归纳起来可分为二大类：面向过程和面向消息的二种方法。这些方法的根本区别之处在于各组成部分之间的接口关系不同。

早期出现的和目前一些计算机厂家所设计的系统，大都是按模块结构来组成系统。然而这些模块的结构很不统一，因而接口方法差别甚大，我们统一都叫它们是模块接口法。它们的优点是代码紧凑，效率高，响应时间快，然而，系统接口混乱，不易理解，不易修改和维护，错误甚多。它们之间的接口方法主要通过子程序调用来实现，故称为面向过程的系统结构。70年代初提出的核扩充分层结构需要设置大量的进程（见1.3节），各进程为一个独立的模块，实现某一系统功能，它们之间的接口是统一的进程通讯，即通过消息传递来实现，故称为面向消息的系统结构。这种结构设计使系统结构统一，各进程模块独立性强，能做到结构清晰，适用于分布式系统和多机系统。然而系统的开销大，效率差，响应时间慢。70年代中期提出的管程结构是从以数据为中心的模块设计方法发展起来的。它提出了一种合理的模块结构和统一的模块接口，并使各模块的独立性强，结构清晰。它是面向过程方法的否定之否定，是发展到更高一级的模块设计方法。对早期按模块接口法设计的系统，经过对数据的重新整理和组织，采用统一的接口，就可以得到管程结构的模块。由于他在确保效率的情况下，仍旧保持结构清晰，因此是当前较好的一种设计方法。本书将具体地介绍如何把 UNIX 系统按管程结构来设计的方法。但是，分布式系统需要采用面向消息的结构设计方法，为了满足不同需要，目前提倡的结构设计方法，仍旧是面向过程和面向消息两种。

下面各节将进一步叙述这几种结构的设计方法。

2.2 传统的模块接口方法

这是操作系统设计中采用最早的传统设计方法。其基本思想是把操作系统按它所实现的

功能来划分模块。各模块所共享的一些公用数据作为全局量存在，各模块可随意存取；各模块之间的进程可以随意调用。这样，就使模块之间形成了非常复杂的调用关系，随着模块数目的增长，系统的复杂性按模块数的平方指数迅速上升。

在这样的系统中，任何一个模块中的程序和数据的改变都会影响到其它模块，任何一个模块的设计都要考虑清楚与其它模块的关系，任一个模块中的错误又会遍及整个系统。触一发而动全身。其复杂性是难以控制的。特别是在操作系统中，一个错误的出现常常是不可重演的，这给系统的维护和修改带来极大的困难。因此，我们期望在设计过程中就能消除这类错误。

在六十年代初，系统的规模较小，而要求效率高，因此采用这种方法还能适应设计要求。但随着系统规模的扩大，这种方法就越来越不适应于计算机发展的需要了。当时人们曾经产生过怀疑，依靠人的智力能不能设计出一个可靠的系统。这种现象当时称为软件危机。为了迎接这一挑战，产生了结构程序设计，以探求新的概念和方法，研究如何把复杂分解，以便依靠人的智力把巨大的系统组织起来，设计出便于理解，正确可靠的系统。

通过二十多年的发展，操作系统结构设计所采用的基本设计方法是：进程概念的引进，自底向上的抽象机层次结构，以数据为中心的管程模块。下面分别叙述这三种设计思想。

2.3 协同的顺序进程

操作系统所采用的主要技术是多道程序。它允许系统中有多个程序同时工作，它们共享着系统资源，从而使系统的资源能充分发挥作用。

如果多个程序在执行时，是交叉进行或重迭进行的，那末我们称它们是并发执行的（Concurrent）。操作系统与其它软件系统的主要区别在于它是并发的，而且这些并发执行的程序是共享着系统资源（包括处理机，存储器，外部设备，文件等）的。因此操作系统的主要特征就是它的并发性和共享性。操作系统的一切复杂性也来自它的并发性和共享性。为此，首先要弄清这些特征的性质，引进一些必要的概念，其中最主要的就是进程的概念。

在操作系统中每个程序的执行，可粗略地定义为进程，有时也叫做任务。操作系统要协调和调度并发进程的工作，使它们共享资源。每个进程都是顺序执行的程序，几个并发执行的程序之间必须协同工作，因此操作系统所研究的是协同的顺序进程。

每个进程通常包括以下三个部份，第一部份是完成这个任务时所需要执行的程序。它描述进程所要完成的功能。要完成一个任务可能需要串行地执行几个程序，这些程序可以是被某进程所专用，也可以被几个进程所共同使用。例如，要将一批数据记入某个文件这一项任务，那末它需要执行文件系统中的程序和盘设备管理程序。若系统中同时有多个题目都正在把自己的数据记入文件，那末它们都要执行文件系统中的程序，但是不必在系统中同时置设多个完全相同的文件系统程序，而只要都调用同一个程序执行就可以了。这里一个程序对应多个进程。反之，一个进程也可以对应多个程序。因此进程的概念不同于程序。程序是一个静态的文本，而进程动态地执行程序，是一个动态的概念。进程的第二部分是程序工作时的数据区（包括调用子程序的下推栈），每个进程都有它自己专用的数据区。第三部份是进程控制块 PCB，每个进程都占有一块，它包含进程的描述信息和控制信息，例如，进程名，当前进程的状态（即它正占用处理机运行还是在等待），进程重新执行时的现场（包括累加器、程序状态字等），资源分配情况、通讯信息以及各种使用资源的统计数据等。系统就是

依据进程控制块中的信息来调度和控制进程的运行（详见4.3节）。由于进程控制块决定了进程的个性，所以它是系统中唯一的进程标识。在 UNIX 系统中把上述的三部份信息叫做进程映象。

当进程真正工作时，还需要有处理机供其使用。为了使处理机充份发挥效率，实际处理机的数目往往比进程数目少得多，最常见的是只有一台处理机。在这种情况下，任一时刻只有一个进程占用处理机。该进程处于运行态，其它进程或者是正等待着处理机，或者是正等待着某个事件发生（如等待输出结束），前者称为绪态，后者称为等待态。因此进程在执行过程中是“走走停停”地活动着，由进程调度程序对它进行调度（见4.3节）。

所有进程都是彼此无关的，那末系统可以把它们看作独立的进程，这类进程称为互不相交进程。而实际上，系统中的各并发进程之间，是共享系统资源的，于是发生竞争资源的现象。这时各进程之间就发生了同步或互斥的关系。有些表现为明显的直接关系，有些表现为不太明显的间接关系。系统要为它们提供各种为发生联系用的通讯手段。

总之，系统要为进程分配必要的资源（如 CPU，内存），设置进程调度模块，为进程提供通讯机制等，这些是操作系统最核心的部份。除此之外的程序，我们都可以把它们设置成进程。核心部份是为进程运行创造运行环境的，它本身不是进程。进程和核心之间的关系如图2—1所示。

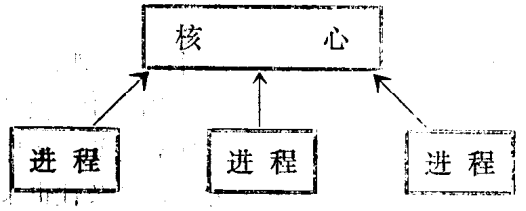

图2—1　核心与进程之间的关系

核心部份好比是一个司令部，进程运行中发生任何事情都要向司令部汇报，并由司令部指挥它们运行。

在裸机（即不配有任何软件的硬件计算机）上增加了这层核心程序之后，就可以把系统看作有许多台处理机。每个进程都占有一台处理机。核心程序给各进程提供为互相通讯所需要的“指令”。进程执行这种“指令”相当于向核心程序汇报，然后获得它的指挥。这种指令我们称为原语，原语可以看作是系统中的一些基本“指令”。在裸机提供的指令系统的基础上，构成了一个功能更强的系统，我们可以把它看作一台比裸机更强的虚拟机。有了这部份程序之后，其它部份就可以分别按顺序进程来设计，这样就大大简化了系统的复杂性，这种方法就是 P. Brinch Hansen 提出的核扩充方法。如果核心部份以外并不全部由进程组成，而大部份是由管程组成，那末就是管程结构的设计方法。

引入了进程概念以后，不仅使系统中的并发活动得以清晰的描述和有效的控制，而且使操作系统的结构清晰；同时增加了系统的安全性。核心部份为每个进程分配地址空间，并借助于硬件，保护它的程序和数据区不受侵犯；它为共享资源提供了各种通讯机制，也为进程的运行提供了相应的控制机制。由此使得每个进程都可以（至少绝大部份）无需考虑并发程

序设计中的问题，只要按照顺序设计的方式，通过执行某些原语传递消息来通讯，对这些结构我们称为面向消息的结构。

在这结构中，系统中的大部分程序都作为进程来处理，这样使系统的并发性得以充分地利用。但是由于设置了大量的多余进程，系统就要花费一定的开销。第一，为每个进程设置一个进程控制块，多占用了一部分存储空间。第二，对系统的请求必须借助于通讯机制，响应时间慢，效率降低。第三，进程引入过多，增加了进程调度的负担，影响效率，也容易发生死锁。

在多处理机或分布式系统中，各处理机之间只能通过传递消息的机制来实现，为了尽量利用各处理机的并发工作，这样设计方法是合适的。然而在单处理机情况下，引起系统的开销过大，例如新墨西哥大学曾将 UNIX 按这种结构改造，内存开销增加50%，时间的开销增加一倍以上。为此，本讲义侧重于介绍管程结构设计方法，它不但继承了核扩充中的各种概念和技术，而且也引进了管程的新概念。在讲述管程概念前，我们先介绍一下层次结构的设计方法。

2.4 层次结构设计方法

为了解决大型操作系统的复杂性，提出了结构程序设计这个课题。反过来，结构程序设计的发展又影响到操作系统的结构设计。结构程序设计中几乎所有的研究成果都可以应用在操作系统的设计中。其中最为重要的就是层次结构设计和模块结构设计。

结构程序设计方法中提出的自顶向下的求精和自底向上的抽象，反映到操作系统中就是抽象机的层次结构，即是按抽象机的观点来分层。层次结构方法是由 E·W·Dijkstra 在设计 THE 系统时提出的。即按照抽象机的观点，从裸机 A_0 出发，先增加一层软件，扩大了这台机器的功能，形成了一台能力更强的、更适合用户需要的新的抽象机 A_1，如此一层一层地扩充，最后形成一台用户所满意的抽象机 A_n。也就是我们设计的操作系统。

层次结构法通常主张自顶向下进行设计，然后再自底向上地反复修改。由于操作系统的层次结构是基本确定的，因此只要采用自底向上的设计就够了。当然在实际设计中，常常是需要经过多次反复的修改才能定型。层次结构可用于面向消息的系统结构中，也可用于面向过程的系统结构中，它是一种独立的设计方法。

根据虚拟机的分层观点，可把 UNIX 系统分为8层。各层含义如下：

裸机（PDP 11/40）
1层　地址再定位管理
2层　交通控制
3层　进程通讯
4层　外设与外设中断管理
5层　文件管理
6层　存储管理与程序对换
7层　进程控制
8层　中断总控与时钟管理
外层　进程模块

下面概述各层软件的主要功能。

第一层只有一个“再定位管理”模块。其主要任务是根据每个进程映象中各部份的大小在用户虚拟空间（逻辑地址空间）中分配地址。当某一进程由低级调度程序选出投入运行之前，按照该进程映象的实际内存地址，组装好页表，由此建立了虚地址向实地址的转换机构。所以这一层软件的主要功能是把整个内存空间转化成多个逻辑地址空间，每个进程一个空间。

第二层也仅有一个“交通控制调度”模块。它的功能是将一台处理机转化成多台虚拟处理机，调度各进程输流使用处理机，完成进程的状态管理。通过这二层的软件，使一台计算机，扩充成多台计算机，每个进程都有一台虚拟处理机和一个虚拟空间。在虚拟机上执行的程序便构成了一个进程。

第三层是进程通讯管理，它提供了各种通讯机制，包括实现同步互斥用的 P、V 操作原语，以及为传递消息用的消息缓冲通讯原语，以便协调各进程之间的协同工作。

第四层实现虚拟外部设备。控制外部设备操作，分配缓冲区。外设分为两类，面向块的设备（如磁盘、磁鼓、磁带等存贮设备，在本文中以磁盘为例）和面向字符的设备（如纸带机、行印机、终端等输入／输出设备），前者用于存放文件和被对换程序。后者用于感受或影响外部世界。

第五层是文件管理，为每个用户提供一个文件空间。

第六层为存储管理。其任务包括用户内存与磁盘对换区的分配，进程映象的建立，修改，更换和移动等。

第七层主要是进程控制及软中断处理。

第八层为中断总控和时钟中断管理。

以上八层组成了系统内核，外层都是进程模块。包括有二个系统进程，即 0 号和 1 号进程。后者为每个终端建立一个用户进程的命令解释程序，称为外壳（shell）。0 号进程负责各个进程的程序在内存和磁盘之间的对换，1 号进程起着调度作业进入系统的作用。

2.5 模块化概念的进一步发展

任何一个大型软件系统都必须要按照一定的准则把系统划分成若干个独立的部份，分别由不同的程序员编制。只要模块之间的接口关系不变，每个模块内部具体实现细节可以由各自的程序员随意修改，这种模块设计的技术自然容易被人们所接受。

模块化技术早在使用汇编时就出现了，后来 Fortran 语言提供了模块的概念。然而它们只是利用子程序，即按功能来划分。早期的模块接口法就是从子程序概念发展出来的。近年来，随着结构程序设计技术的发展，人民弄清了数据与程序之间的关系，提出了程序设计应该以数据为中心来设计的观点，这是七十年代软件发展中最重要的成果之一。

SIMULA 67 语言提出了协同程序（Coroutin）的概念，也就是说，各子程序之间有时存在着协同的关系。因此把一个子程序作为模块的独立性就不强了。C · A · R · Hoare 把 SIMULA 67 中的类（class）的概念加以发展，提出了‘数据表示抽象’，即把数据及其上的操作（由子程序来实现）集中起来作为一种抽象数据类型，这种概念进一步运用于并发程序，终于在1974年提出了管程（monitor）这个新的概念。

E · W · Dijkstra 于 1971 年发表的“协同顺序进程”一文中，首先提出应该把同一信号量上的 PV 操作集中在一起，以便使并发程序之间的相互作用更清晰、明确。他称这种

程序单位为“秘书”，并规定了“秘书”的职责。这就是管程概念的雏形。

与此同时 P · Brinch Hansen 在‘操作系统原理’一书中提出了一种保护机制，把数据和有关的子程序集中在一起，组成的新类型有助于编译程序的检查以确保系统的安全性。他于 1974 年发表了併发 PASCAL 语言，明确地提出了管程和类程（class）的新概念。

N · Wirth 和 D·L·Parnas 把管程不仅看为数据类型，而且作为模块来处理，并澄清了管程结构中的一些模糊概念，由此模块化技术开始运用于操作系统中。

在操作系统中，对于它的每一种资源，都可以用某一数据结构来表示。对这种资源的申请、释放和控制就是操作系统中的各种子程序，把它们组合在一起叫做管程或模块。C · A · R · Hoare 认为操作系统的功能就是对某种资源进行管理，因此整个操作系统就可以按资源的观点来划分模块。

管程概念及其相应语言表示提出以后，对于操作系统的结构设计产生了深远的影响，主要的优点可归纳成以下几个方面。

首先，模块提供给外部使用的只是这些数据结构本身和操作名称，即一种抽象的数据类型，至于具体数据结构的组成方式及其操作的实现细节在外部是看不见的。模块实现了抽象，隐藏了内部细节。例如，我们设计一个下推栈，栈的特性就可以用三种操作（即置初始栈、进栈和退栈）来表征。把它们组合在一起构成抽象栈的功能，至于栈是用数组来表示，还是用链结构来表示外部是无须了解的。外部只要抽象地了解这三种操作的功能，即懂得如何在调用这些子程序时给出相应的形式参数就行了。

这样设计的模块，可以做到真正的独立性。每个模块只要规定好其抽象说明，即其对外接口，就可以独立设计，独立地调试，独立地修改，不会影响其它模块的实现。

其次，这样的模块，由于每个模块的职责明确，与外部的接口关系清晰，使得阅读和理解整个程序很方便，既便于对模块进行正确性验证，也便于对每个模块给出形式说明（specification）。

最后，这种模块便于编译程序检查，易于保证安全性。同步通讯机制集中还有利于免除并发程序设计中出现的不可重演的错误。

在管程概念提出以后，曾发生过一场争论，按照 C · A · R · Hoare 和 P · Brinch Hansen 的定义，管程中的各子程序必须保证互斥执行，即管程内在任一时刻最多只允许有一个进程正在执行管程子程序。这给管理的嵌套调用带来了极大的困难，而在大型系统中要避免嵌套调用是不可能的。这就有必要对管理的属性重新加以认识。最近我国徐家福、仲萃豪和杨芙清提出的 XCY 语言重新定义了管程，即允许了管程内的子程序可以并发地执行，设计者可自行定义它的同步工具。确定了管程的基本属性是实现数据表示的抽象，保证数据的完整性，而不是保证各子程序的互斥执行。

按照重新定义的管程模块来设计操作系统，是更高一级的模块化结构设计，它属于面向过程的结构，访问模块都是通过调用模块子程序这种接口方法来实现的。按照这种设计方法，系统中大部分系统程序都由管程模块来组成，仅须设置少数几个需要併发执行的进程，从而大大减少了系统开销。

我们认为进程、层次及模块这三种设计思想之间本身是没有根本矛盾的，进程概念的引进，澄清了系统中的并发和共享所产生的各种关系，模块化设计提出了系统的基本组成单位