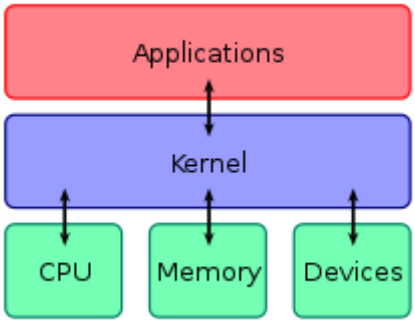
内核

维基百科，自由的百科全书

内核（英語：**Kernel**，又稱**核心**）在計算機科學中是一個用來管理軟體發出的資料I/O（輸入與輸出）要求的電腦程式，將這些要求轉譯為資料處理的指令并交由中央處理器（CPU）及電腦中其他電子元件進行處理，是現代操作系统中最基本的部分。它是为众多应用程序提供对计算机硬件的安全访问的一部分软件，这种访问是有限的，并由内核决定一个程序在什么时候对某部分硬件操作多长时间。直接对硬件操作是非常复杂的。所以内核通常提供一种硬件抽象的方法，来完成这些操作。有了這個，通过进程间通信机制及系统调用，应用进程可间接控制所需的硬件资源（特别是处理器及IO设备）。



電腦軟硬之間的架構與關係圖，可以看到内核进行的是应用软件和计算机硬件的交互工作

严格地说，内核并不是计算机系统中必要的组成部分。有些程序可以直接地被调入计算机中执行；这样的设计，说明了设计者不希望提供任何硬件抽象和操作系统的支持；它常见于早期计算机系统的设计中。但隨著電腦技術的發展，最终，一些辅助性程序，例如程序加载器和调试器，被设计到机器核心当中，或者写入在只读记忆体里。这些变化发生时，操作系统内核的概念就渐渐明晰起来了!

目录

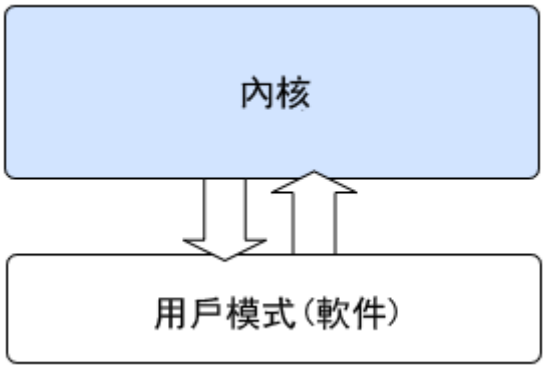
分類

内核在設計上可以概分為宏內核與微內核兩大架構。在宏內核與微內核之間，進行妥協的設計，這稱為混合內核，但是混合內核能否被列為第三大架構，目前仍然有爭議。在微內核之下，有一種極端的設計方式，稱為外內核，仍还在研究阶段，没有任何一个流行的操作系统采用了这种设计。

宏内核

宏内核结构在硬件之上，定义了一个高阶的抽象接口，应用一组原语（或者叫系统调用（System call））来实现操作系统的功能，例如进程管理，文件系统，和存储管理等等，这些功能由多个运行在核心态的模块来完成。

尽管每一个模块都是单独地服务这些操作，内核代码是高度集成的，而且难以编写正确。因为所有的模块都在同一个内核空间上运行，一个很小的bug都会使整个系统崩溃。然而，如果开发顺利，宏内核结构就可以从运行效率上得到好处。



宏内核的示意图

很多现代的宏内核结构内核，如Linux和FreeBSD内核，能够在运行时将模块调入执行，这就可以使扩充内核的功能变得更简单，也可以使内核的核心部分变得更简洁。

宏内核结构的例子：

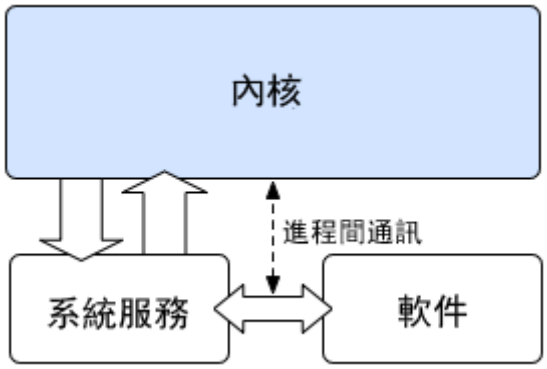
- 傳統Unix核心
  - BSD
  - Solaris
- 类Unix系统（Unix-like）的核心
  - FreeBSD
  - OpenBSD
  - NetBSD
  - Linux
  - LynxOS
  - Syllable Desktop
- DOS
  - DR-DOS
  - MS-DOS
    - Microsoft Windows 9x系列（95、98、98SE、Me）
- Mac OS（從最初版到Mac OS 8.6版）
- OpenVMS
- XTS-400

微内核

微内核结构由一个非常简单的硬件抽象层和一组比较关键的原语或系统调用组成；这些原语，仅仅包括了建立一个系统必需的几个部分；如线程管理，地址空间和行程間通訊等。

微核的目标是将系统服务的实现和系统的基本操作规则分离开来。例如，进程的输入/输出锁定服务可以由运行在微核之外的一个服务组件来提供。这些非常模块化的用户态服务器用于完成操作系统中比较高级的操作，这样的设计使内核中最核心的部分的设计更简单。一个服务组件的失效并不会导致整个系统的崩溃，内核需要做的，仅仅是重新启动这个组件，而不必影响其它的部分。

微内核将许多OS服务放入分离的进程，如文件系统，设备驱动程序，而进程通过消息传递调用OS服务。微内核结构必然是多线程的，第一代微内核，在核心提供了较多的服务，因此被称为'胖微内核'，它的典型代表是Mach，它是Mac OS X的核心，可以说，蒸蒸日上。第二代微内核只提供最基本的OS服务，典型的OS是QNX，QNX在黑莓手机BlackBerry 10系统中被采用。



微内核的示意图

微内核结构的例子：

- AIX
- BeOS
- L4微内核系列
- Mach（用于XNU、GNU Hurd）
- Minix
- MorphOS
- QNX
- RadiOS
- VSTa

宏内核与微内核的比较

宏内核结构是非常有吸引力的一种设计，由于在同一个地址空间上实现所有复杂的低阶操作系统控制代码的效率会比在不同地址空间上实现更高些。

20世纪90年代初，宏内核结构被认为是过时的。把Linux设计成为宏内核结构而不是微内核，引起了无数的争议（参见塔能鲍姆-林纳斯辩论）。

现在，单核结构正倾向于设计不容易出错，所以它的发展会比微内核结构更迅速些。两个阵营中都有成功的案例。微核经常被用于机器人和医疗器械的嵌入式设计中，因为它的系统的关键部分都处在相互分开的，被保护的存储空间中。这对于单核设计来说是不可能的，就算它采用了运行时加载模块的方式。

尽管Mach是众所周知的多用途的微内核，人们还是开发了除此之外的几个微内核。L3是一个演示性的内核，只是为了证明微内核设计并不总是低运行速度。它的后续版本L4，甚至可以将Linux内核作为它的一个进程，运行在单独的地址空间。

QNX是一个从20世纪80年代，就开始设计的微内核系统。它比Mach更接近微内核的理念。它可以被用于一些特殊的领域；在这些情况下，由于软件错误，导致系统失效是不允许的。例如航天飞机上的机械手，还有研磨望远镜镜片的机器，一点点失误就会导致上千美元的损失。

很多人相信，由于Mach不能够解决一些提出微内核理论时针对的问题，所以微内核技术毫无用处。Mach的爱好者表明这是非常狭隘的观点，但遗憾的是似乎所有人都开始接受这种观点。

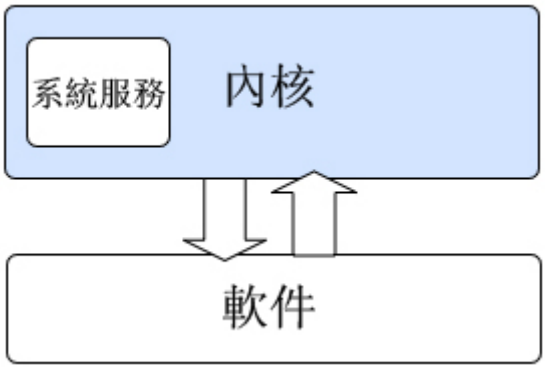
混合内核

混合内核的設計理念來自微内核，只不过它让一些微核结构运行在用户空间的代码运行在内核空间，这样让内核的运行效率更高些。这是一种妥协做法，微软视窗就是一个典型的例子。另外还有XNU，运行在苹果Mac OS X上的内核，也是一个混合内核。林纳斯·托瓦兹認為混合核心這種分類

只是一種市場行銷手法，因為它的架構實作與運作方式接近於宏內核。

混合內核的例子：

- Windows NT、Windows 2000、Windows XP、Windows Server 2003以及Windows Vista、Windows 7、Windows 8、Windows 8.1、Windows 10等基于NT技术的微软视窗操作系统
- Mac OS X（使用Mach內核來實作）
- BeOS内核
- DragonFly BSD
- ReactOS内核
- XNU（使用Mach內核）



混合内核的示意图

外内核

外内核系统，也被称为纵向结构操作系统，是一种比较极端的设计方法。

它的设计理念是让用户程序的设计者来决定硬件接口的设计。外内核本身非常的小，它通常只负责系统保护和系统资源复用相关的服务。

传统的内核设计（包括单核和微核）都对硬件作了抽象，把硬件资源或设备驱动程序都隐藏在硬件抽象层下。比方说，在这些系统中，如果分配一段物理存储，应用程序并不知道它的实际位置。

而外核的目标就是让应用程序直接请求一块特定的物理空间，一块特定的磁盘块等等。系统本身只保证被请求的资源当前是空闲的，应用程序就允许直接存取它。既然外核系统只提供了比较低级的硬件操作，而没有像其他系统一样提供高级的硬件抽象，那么就需要增加额外的运行库支持。这些运行库运行在外核之上，给用户程序提供了完整的功能。

理论上，这种设计可以让各种操作系统运行在一个外核之上，如Windows和Unix。并且设计人员可以根据运行效率调整系统的各部分功能。

现在，外核设计还停留在研究阶段，没有任何一个商业系统采用了这种设计。几种概念上的操作系统正在被开发，如剑桥大学的Nemesis，格拉斯哥大学的Citrix系统和瑞士计算机科学院的一套系统。麻省理工学院也在进行着这类研究。

参考文献

参见

- 操作系统
