是采用层次化构造，内层环比外层环有更高的级别（它们实际上是一样的）。当外环的过程欲调用内环的过程时，它必须执行一条等价于系统调用的TRAP指令。在执行该TRAP指令前，要进行严格的参数合法性检查。在MULTICS中，尽管整个操作系统是各个用户进程的地址空间的一部分，但是硬件仍能对单个过程（实际是内存中的一个段）的读、写和执行进行保护。

实际上，THE分层方案只是为设计提供了一些方便，因为该系统的各个部分最终仍然被链接成了完整的单个目标程序。而在MULTICS里，环形机制在运行中是实际存在的，而且是由硬件实现的。环形机制的一个优点是很容易扩展，可用以构造用户子系统。例如，在一个MULTICS系统中，教授可以写一个程序检查学生们编写的程序并给他们打分，在第n个环中运行教授的程序，而在第$n+1$个环中运行学生的程序，这样学生们就无法篡改教授所给出的成绩。

1.7.3 微内核

在分层方式中，设计者要确定在哪里划分内核-用户的边界。在传统上，所有的层都在内核中，但是这样做没有必要。事实上，尽可能减少内核态中功能的做法更好，因为内核中的错误会快速拖累系统。相反，可以把用户进程设置为具有较小的权限，这样，某一个错误的后果就不会是致命的。

有不少研究人员对每千行代码中错误的数量进行了分析（例如，Basilli 和Perricone，1984；Ostrand和Weyuker，2002）。代码错误的密度取决于模块大小、模块寿命等，不过对一个实际工业系统而言，每千行代码中会有10个错误。这意味着在有5百万行代码的单体操作系统中，大约有50 000个内核错误。当然，并不是所有的错误都是致命的，诸如给出了不正确的故障信息之类的某些错误，实际是很少发生的。无论怎样看，操作系统中充满了错误，所以计算机制造商设置了复位按钮（通常在前面板上），而电视机、立体音响以及汽车的制造商们则不这样做，尽管在这些装置中也有大量的软件。

在微内核设计背后的思想是，为了实现高可靠性，将操作系统划分成小的、良好定义的模块，只有其中一个模块——微内核——运行在内核态上，其余的模块，由于功能相对弱些，则作为普通用户进程运行。特别地，由于把每个设备驱动和文件系统分别作为普通用户进程，这些模块中的错误虽然会使这些模块崩溃，但是不会使得整个系统死机。所以，在音频驱动中的错误会使声音断续或停止，但是不会使整个计算机垮掉。相反，在单体系统中，由于所有的设备驱动都在内核中，一个有故障的音频驱动会很容易引起对无效地址的引用，从而造成恼人的系统立即停机。

有许多微内核已经实现并投入应用（Accetta等人，1986；Kirsch等人，2005；Heiser等人，2006；Herder等人，2006；Hildebrand，1992；Haertig等人，1997；Liedtke，1993，1995，1996；Pike等人，1992；Zuberi等人，1999）。微内核在实时、工业、航空以及军事应用中特别流行，这些领域都是关键任务，需要有高度的可靠性。知名的微内核有Integrity、K42、L4、PikeOS、QNX、Symbian，以及MINIX 3等。这里对MINIX 3做一简单的介绍，该操作系统把模块化的思想推到了极致，它将大部分操作系统分解成许多独立的用户态进程。MINIX 3遵守POSIX，可在www.minix3.org（Herder等人，2006a；Herder等人，2006b）站点获得免费的开放源代码。

MINIX 3微内核只有3200行C语言代码和800行用于非常低层次功能的汇编语言代码，诸如捕捉中断、进程切换等。C代码管理和调度进程、处理进程间通信（在进程之间传送信息）、提供大约35个内核调用，它们使得操作系统的其余部分可以完成其工作。这些调用完成诸如连接中断句柄、在地址空间中移动数据以及为新创建的进程安装新的内存映像等。MINIX 3的进程结构如图1-26所示，其中内核调用的句柄用Sys标记。时钟设备驱动也在内核中，因为这个驱动与调度器交互密切。所有的其他设备驱动都作为单独的用户进程运行。

在内核的外部，系统的构造有三层进程，它们都在用户态中运行。最底层中包含设备驱动器。由于它们在用户态中运行，所以不能物理地访问I/O端口空间，也不能直接发出I/O命令。相反，为了能够对I/O设备编程，驱动器构建了一个结构，指明哪个参数值写到哪个I/O端口，并生成一个内核调用，通知内核完成写操作。这个处理意味着内核可以检查驱动正在对I/O的读（或写）是否是得到授权使用的。这样，（与单体设计不同），一个有错误的音频驱动器就不能够偶发性地在硬盘上进行写操作。

在驱动器上面是另一用户态层，包含有服务器，它们完成操作系统多数的工作。有一个或多个文件服务器管理着文件系统，进程管理器创建、破坏和管理进程等。通过给服务器发送短消息请求POSIX系