简化I/O结构，因为当一个中断发生时，它就可以转化成在一个互斥量上的unlock，并且调用调度器以（潜在地）调度重新就绪的线程，而该线程曾阻塞在该互斥量之上。MINIX使用了这一方案，但是在UNIX、Linux和Windows Vista中，中断处理程序运行在一类“无主地带”中，而不是作为适当的线程可以被调度、挂起等。由于任何一个操作系统的大多数复杂性在于I/O之中，使其更加易于处理和封装的任何技术都是值得考虑的。

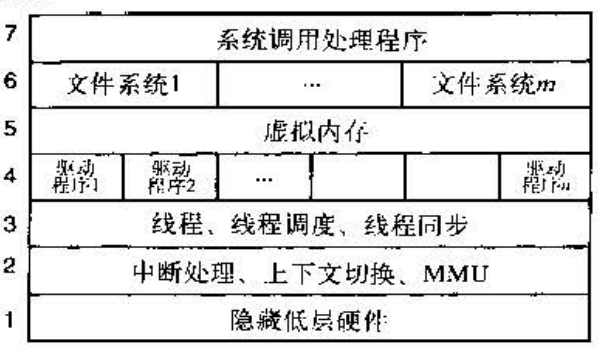

图13-2 现代分层操作系统的一种可能的设计

在第4层之上，我们预计会找到虚拟内存、一个或多个文件系统以及系统调用接口。如果虚拟内存处于比文件系统更低的层次，那么数据块高速缓存就可以分页出去，使虚拟内存管理器能够动态地决定在用户页面和内核页面（包括高速缓存）之间应该怎样划分实际内存。Windows Vista就是这样工作的。

2. 外内核

虽然分层在系统设计人员中间具有支持者，但是还有另一个阵营恰恰持有相反的观点（Engler等人，1995）。他们的观点基于端到端的论据（end-to-end argument）（Saltzer等人，1984）。这一概念说的是，如果某件事情必须由用户程序本身去完成，在一个较低的层次做同样的事情就是浪费。

考虑该原理对于远程文件访问的一个应用。如果一个系统担心数据在传送中被破坏，它应该安排每个文件在写的时候计算校验和，并且校验和与文件一同存放。当一个文件通过网络从源盘传送到目标进程时，校验和也被传送，并且在接收端重新计算。如果两者不一致，文件将被丢弃并且重新传送。

校验比使用可靠的网络协议更加精确，因为除了位传送错误以外，它还可以捕获磁盘错误、内存错误、路由器中的软件错误以及其他错误。端到端的论据宣称使用一个可靠的网络协议是不必要的，因为端点（接收进程）拥有足够的信息以验证文件本身的正确性。在这一观点中，使用可靠的网络协议的惟一原因是为了效率，也就是说，更早地捕获与修复传输错误。

端到端的论据可以扩展到几乎所有操作系统。它主张不要让操作系统做用户程序本身可以做的任何事情。例如，为什么要有一个文件系统？只要让用户以一种受保护的方式读和写原始磁盘的一个部分就可以了。当然，大多数用户喜欢使用文件，但是端到端的论据宣称，文件系统应该是与需要使用文件的任何程序相链接的库过程。这一方案使不同的程序可以拥有不同的文件系统。这一论证线索表明操作系统应该做的全部事情是在竞争的用户之间安全地分配资源（例如CPU和磁盘）。Exokernel是一个根据端到端的论据建立的操作系统（Engler等人，1995）。

3. 基于微内核的客户-服务器系统

在让操作系统做每件事情和让操作系统什么也不做之间的折衷是让操作系统做一点事情。这一设计导致微内核的出现，它让操作系统的大部分作为用户级的服务器进程而运行，如图13-3所示。在所有设计中这是最模块化和最灵活的。在灵活性上的极限是让每个设备驱动程序也作为一个用户进程而运行，从而完全保护内核和其他驱动程序，但是让设备驱动程序运行在内核会增加模块化程度。

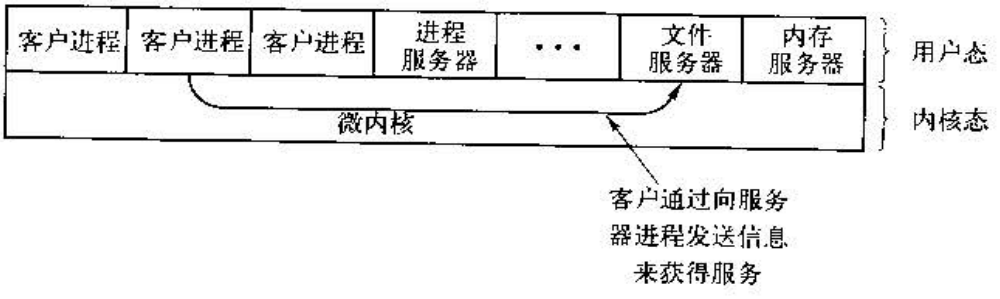

图13-3 基于微内核的客户-服务器计算

当设备驱动程序运行在内核态时，可以直接访问硬件设备寄存器，否则需要某种机制以提供这样的