操作系统通常将驱动程序归类于少数的类别之一。最为通用的类别是块设备（block device）和字符设备（character device）。块设备（例如磁盘）包含多个可以独立寻址的数据块，字符设备（例如键盘和打印机）则生成或接收字符流。

大多数操作系统都定义了一个所有块设备都必须支持的标准接口，并且还定义了另一个所有字符设备都必须支持的标准接口。这些接口由许多过程组成，操作系统的其余部分可以调用它们让驱动程序工作。典型的过程是那些读一个数据块（对块设备而言）或者写一个字符串（对字符设备而言）的过程。

在某些系统中，操作系统是一个二进制程序，包含需要编译到其内部的所有驱动程序。这一方案多年以来对UNIX系统而言是标准规范，因为UNIX系统主要由计算中心运行，I/O设备几乎不发生变化。如果添加了一个新设备，系统管理员只需重新编译内核，将新的驱动程序增加到新的二进制程序中。

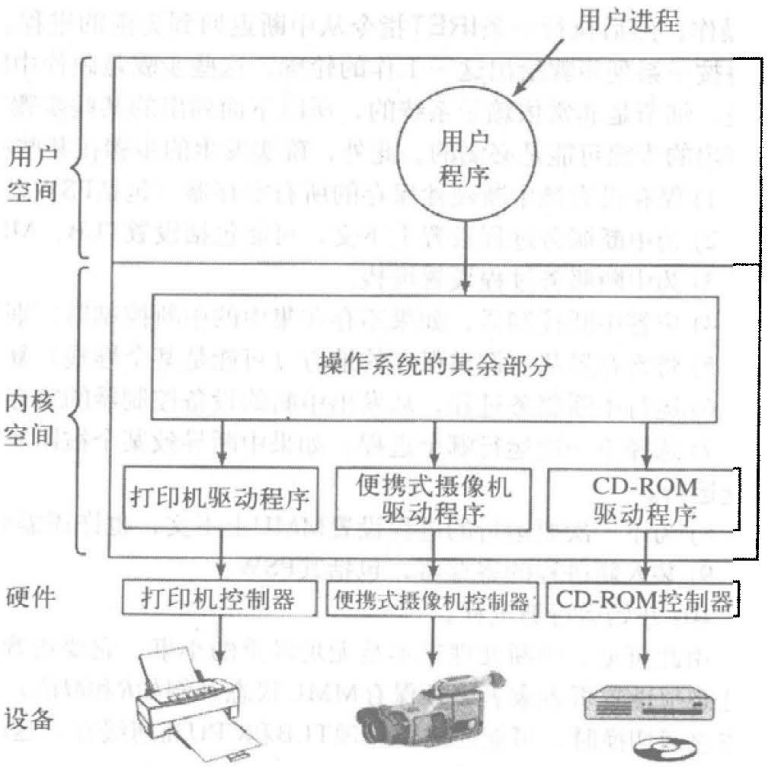

图5-12 设备驱动程序的逻辑定位。实际上，驱动程序和设备控制器之间的所有通信都通过总线

随着个人计算机的出现，这一模型不再起作用，因为个人计算机有太多种类的I/O设备。即便拥有源代码或目标模块，也只有很少的用户有能力重新编译和重新连接内核，何况他们并不总是拥有源代码或目标模块。为此，从MS-DOS开始，操作系统转向驱动程序在执行期间动态地装载到系统中的另一个模型。不同的操作系统以不同的方式处理驱动程序的装载工作。

设备驱动程序具有若干功能。最明显的功能是接收来自其上方与设备无关的软件所发出的抽象的读写请求，并且目睹这些请求被执行。除此之外，还有一些其他的功能必须执行。例如，如果需要的话，驱动程序必须对设备进行初始化。它可能还需要对电源需求和日志事件进行管理。

许多设备驱动程序具有相似的一般结构。典型的驱动程序在启动时要检查输入参数，检查输入参数的目的是搞清它们是否是有效的，如果不是，则返回一个错误。如果输入参数是有效的，则可能需要进行从抽象事项到具体事项的转换。对磁盘驱动程序来说，这可能意味着将一个线性的磁盘块号转换成磁盘几何布局的磁头、磁道、扇区和柱面号。

接着，驱动程序可能要检查设备当前是否在使用。如果在使用，请求将被排入队列以备稍后处理。如果设备是空闲的，驱动程序将检查硬件状态以了解请求现在是否能够得到处理。在传输能够开始之前，可能需要接通设备或者启动马达。一旦设备接通并就绪，实际的控制就可以开始了。

控制设备意味着向设备发出一系列命令。依据控制设备必须要做的工作，驱动程序处在确定命令序列的地方。驱动程序在获知哪些命令将要发出之后，它就开始将它们写入控制器的设备寄存器。驱动程序在把每个命令写到控制器之后，它可能必须进行检测以了解控制器是否已经接收命令并且准备好接收下一个命令。这一序列继续进行，直到所有命令被发出。对于某些控制器，可以为其提供一个在内存中的命令链表，并且告诉它自己去读取并处理所有命令而不需要操作系统提供进一步帮助。

命令发出之后，会牵涉两种情形之一。在多数情况下，设备驱动程序必须等待，直到控制器为其做某些事情，所以驱动程序将阻塞其自身直到中断到来解除阻塞。然而，在另外一些情况下，操作可以无延迟地完成，所以驱动程序不需要阻塞。在字符模式下滚动屏幕只需要写少许字节到控制器的寄存器中，由于不需要机械运动，所以整个操作可以在几纳秒内完成，这便是后一种情形的例子。