一种类型的I/O错误是编程错误，这些错误发生在一个进程请求某些不可能的事情时，例如写一个输入设备（键盘、扫描仪、鼠标等）或者读一个输出设备（打印机、绘图仪等）。其他的错误包括提供了一个无效的缓冲区地址或者其他参数，以及指定了一个无效的设备（例如，当系统只有两块磁盘时指定了磁盘3），如此等等。在这些错误上采取的行动是直截了当的：只是将一个错误代码报告返回给调用者。

另一种类型的错误是实际的I/O错误，例如，试图写一个已经被破坏的磁盘块，或者试图读一个已经关机的便携式摄像机。在这些情形中，应该由驱动程序决定做什么。如果驱动程序不知道做什么，它应该将问题向上传递，返回给与设备无关的软件。

软件要做的事情取决于环境和错误的本质。如果是一个简单的读错误并且存在一个交互式的用户可利用，那么它就可以显示一个对话框来询问用户做什么。选项可能包括重试一定的次数，忽略错误，或者杀死调用进程。如果没有用户可利用，惟一的实际选择或许就是以一个错误代码让系统调用失败。

然而，某些错误不能以这样的方式来处理。例如，关键的数据结构（如根目录或空闲块列表）可能已经被破坏，在这种情况下，系统也许只好显示一条错误消息并且终止。

4. 分配与释放专用设备

某些设备，例如CD-ROM刻录机，在任意给定的时刻只能由一个进程使用。这就要求操作系统对设备使用的请求进行检查，并且根据被请求的设备是否可用来接受或者拒绝这些请求。处理这些请求的一种简单方法是要求进程在代表设备的特殊文件上直接执行open操作。如果设备是不可用的，那么open就会失败。于是就关闭这样的一个专用设备，然后将其释放。

一种代替的方法是对于请求和释放专用设备要有特殊的机制。试图得到不可用的设备可以将调用者阻塞，而不是让其失败。阻塞的进程被放入一个队列。迟早被请求的设备会变得可用，这时就可以让队列中的第一个进程得到该设备并且继续执行。

5. 与设备无关的块大小

不同的磁盘可能具有不同的扇区大小。应该由与设备无关的软件来隐藏这一事实并且向高层提供一个统一的块大小，例如，将若干个扇区当作一个逻辑块。这样，高层软件就只需处理抽象的设备，这些抽象设备全都使用相同的逻辑块大小，与物理扇区的大小无关。类似地，某些字符设备（如调制解调器）一次一个字节地交付它们的数据，而其他的设备（如网络接口）则以较大的单位交付它们的数据。这些差异也可以被隐藏起来。

5.3.4 用户空间的I/O软件

尽管大部分I/O软件都在操作系统内部，但是仍然有一小部分在用户空间，包括与用户程序连接在一起的库，甚至完全运行于内核之外的程序。系统调用（包括I/O系统调用）通常由库过程实现。当一个C程序包含调用

```
count=write(fd, buffer, nbytes);
```

时，库过程write将与该程序连接在一起，并包含在运行时出现在内存中的二进制程序中。所有这些库过程的集合显然是I/O系统的组成部分。

虽然这些过程所做的工作不过是将这些参数放在合适的位置供系统调用使用，但是确有其他I/O过程实际实现真正的操作。输入和输出的格式化是由库过程完成的。一个例子是C语言中的printf，它以一个格式串和可能的一些变量作为输入，构造一个ASCII字符串，然后调用write以输出这个串。作为printf的一个例子，考虑语句

```
printf("The square of %3d is %6d\n", i, i*i);
```

该语句格式化一个字符串，该字符串是这样组成的：先是14个字符的串“The square of ”（注意of后有一个空格），随后是*i*值作为3个字符的串，然后是4个字符的串“is”（注意前后各有一个空格），然后是i^2值作为6个字符的串，最后是一个换行。

对输入而言，类似过程的一个例子是scanf，它读取输入并将其存放到一些变量中，采用与printf同样语法的格式串来描述这些变量。标准的I/O库包含许多涉及I/O的过程，它们都是作为用户程序的一部分运行的。