系统调用是可以接受的，那么狱卒进程会通知内核，由内核来执行该系统调用。通过使用这种方法，不正确的行为会在它引起麻烦之前被捕捉到。

囚禁有很多的实现方法。有一种方法可以在不需要修改内核的情况下，在几乎任何一个UNIX系统上实现，这种方法是Van't Noordende等人在2007年提出的。在nutshell中，这个方法使用普通的UNIX调试功能，让狱卒进程作为调试者而囚犯进程作为被调试者。这种情况下，调试者可以指示内核把被调试者封装起来，然后把被调试者的所有系统调用都传递给自己来监视。

9.8.5 基于模型的入侵检测

还有一种方法可以保护我们的机器，那就是安装一个IDS（Intrusion Detection System）。IDS有两种基本的类型，一种关注于监测进入电脑的网络包，另一种关注寻找CPU上的异常情况。之前在防火墙的部分我们简要地提到了网络IDS；现在我们对于基于主机的IDS进行一些讲解。出于篇幅限制，我们不能够审视全部的种类繁多的基于主机的IDS。相反地，我们选择一种类型来简单地了解它们是如何工作的。这种类型是基于静态模型的入侵检测（wagner和Dean，2001）。它可以用上面提到的囚禁技术来实现，同时也有其他的实现方法。

在图9-36a中我们看到了这样一个小程序，它打开一个叫data的文件，然后每次一个字符地读入，直到遇到了一个0字节，这时打印出文件开始部分的非0字节的个数然后程序退出。在图9-36b中，我们看到了这个程序的系统调用图（这里打印被叫做write）。

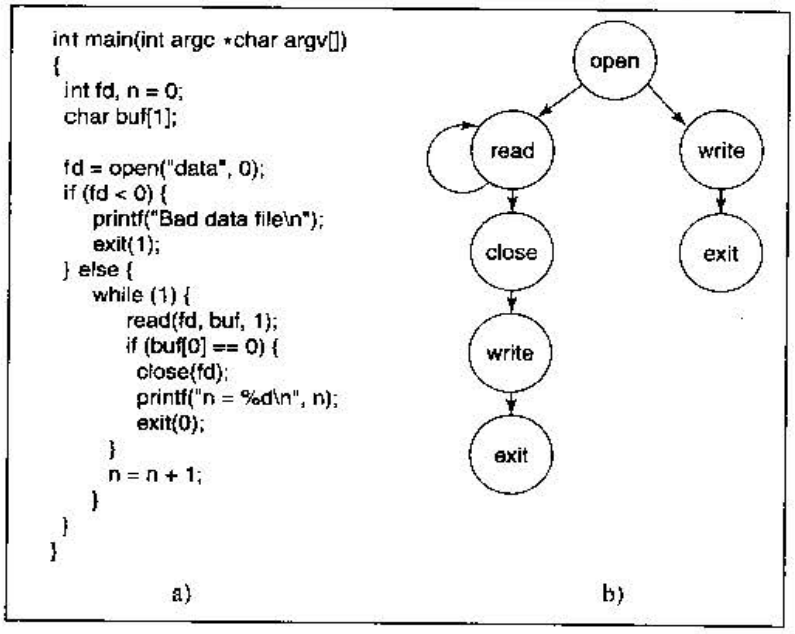

图9-36　a) 程序；b) 该程序的系统调用图

这个图告诉了我们什么呢？首先，在任何情况下，这个程序的第一个系统调用一定是open。第二个系统调用是read或者write，这要根据执行if语句的那个分支来决定。如果第二个系统调用是write，那么就意味着文件无法打开，然后下一个系统调用必须是exit。如果第二个系统调用是read，那么可能还有额外任意多次的read调用，并且最后调用close、write和exit。在没有入侵的情况下，其他序列是不可能的。如果这个程序被囚禁，那么狱卒程序可以看到所有的系统调用并很容易地验证某个序列是不是有效的。

现在假设某人发现了这个程序的一个bug，然后成功地引起了缓冲区溢出，插入并执行了恶意代码。当恶意代码运行的时候，极大的可能是会执行一个不同的系统调用序列。例如，恶意代码可能尝试打开某个它想要复制的文件或者可能和家里的电话建立网络连接。当第一次出现系统调用不符合原来的模式时，狱卒十分肯定地认定出现了攻击并会采取行动，比如结束这个进程并向系统管理员报警。这样，入侵检测系统就能够在攻击发生的时候检查到它们。静态系统调用分析只是很多IDS工作方法中的一种。

当使用这种基于静态模型的入侵检测的时候，狱卒必须知道这个模型（比如系统调用图）。最直接的方式就是让编译器产生它并让程序的作者签名同时附上它的证书。这样的话，任何预先修改可执行程