锁定或解锁一个预置的文件，且S不是在一个特别嘈杂的信道里，并不需要十分精确的时序，除非比特率很慢。使用一个双方确认的通信协议可以增强系统的可靠性和性能。这种协议使用了2个文件F1和F2。这两个文件分别被服务器和协作程序锁定以保持两个进程的同步。当服务器锁定或解锁S后，它将F1的状态反置表示送出了一个比特。一旦协作程序读取了该比特，它将F2的状态反置告知服务器可以送出下一个比特了，直到F1被再次反置表示在S中第二个比特已送达。由于这里没有使用时序技术，所以这种协议是完全可靠的，并且可以在繁忙的系统内使它们得以按计划快速地传递信息。也许有人会问：要得到更高的带宽，为什么不在每个比特的传输中都使用文件呢？或者建立一个字节宽的信道，使用从S0到S7共8个信号文件？

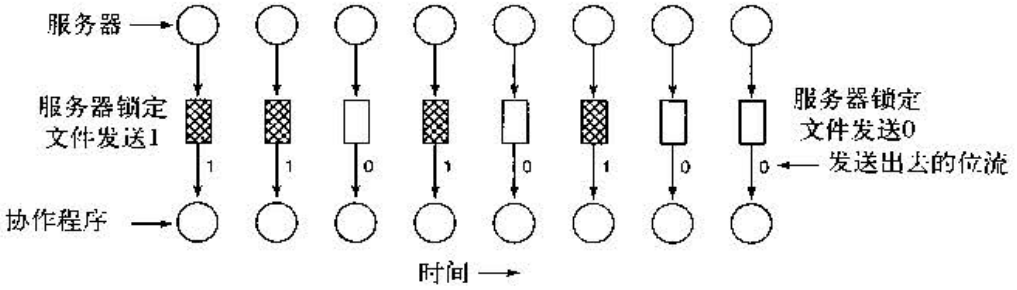

图9-15 使用文件加锁的隐蔽信道

获取和释放特定的资源（磁带机、绘图仪等）也可以用来作为信号方式。服务器进程获取资源时表示发送1信号，释放资源时表示发送0信号。在UNIX里，服务器进程创建文件表示1，删除文件表示0；协作程序可以通过系统访问请求来查看文件是否存在。即使协作程序没有使用文件的权限也可以通过系统访问请求来查看。然而很不幸，仍然还存在许多其他的隐蔽信道。

Lampson也提到了把信息泄露给服务器进程所有者（人）的方法。服务器进程可能有资格告诉其所有者，它已经替客户机完成了多少工作，这样可以要求客户机付账。如，假设真正的计算值为100美元，而客户收入是53 000美元，那么服务器就可以报告100.53美元来通知自己的主人。

仅仅找到所有的隐蔽信道已经是非常困难的了，更不用说阻止它们了。实际上，没有什么可行的方法。引入一个可随机产生页面调用错误的进程，或为了减少隐蔽信道的带宽而花费时间来降低系统性能等，都不是什么诱人的好主意。

隐写术

另一类稍微不同的隐蔽信道能够在进程间传递机密信息，即使人为或自动的审查监视着进程间的所有信息并禁止可疑的数据传递。例如，假设一家公司人为地检查所有发自公司职员的电子邮件来确保没有机密泄露给公司外的竞争对手或同谋。雇员是否有办法在审查者的鼻子下面偷带出机密的信息呢？结果是可能的。

让我们用例子来证明。请看图9-16a，这是一张在肯尼亚拍摄的照片，照片上有三只斑马在注视着金合欢树。图9-16b看上去和图9-16a差不多，但是却包含了附加的信息。这些信息是完整而未被删节的五部莎士比亚戏剧：《哈姆雷特》、《李尔王》、《麦克白》、《威尼斯商人》和《裘力斯恺撒》。这些戏剧总共加起来超过700KB的文本。

隐蔽信道是如何工作的呢？原来的彩色图片是1024×768像素的。每个像素包括三个8位数字，分别代表红、绿、蓝三原色的亮度。像素的颜色是通过三原色的线性重叠形成的。编码程序使用每个RGB色度的低位作为隐蔽信道。这样每个像素就有三位的秘密空间存放信息，一个在红色色值里，一个在绿色色值里，一个在蓝色色值里。这种情况下，图片大小将增加1024×768×3位或294 912个字节的空间来存放信息。

五部戏剧和一份简短说明加起来有734 891个字节。这些内容首先被标准的压缩算法压缩到274KB，压缩后的文件加密后被插入到每个色值的低位中。正如我们所看到的（实际上看不到），存放的信息完全是不可见的，在放大的、全彩的照片里也是不可见的。一旦图片文件通过了审查，接收者就剥离低位数据，利用解码和解压缩算法还原出743 891个字节。这种隐藏信息的方法叫做隐写术（steganography，来自于希腊语"隐蔽书写"）。隐写术在那些试图限制公民通信自由的独裁统治国家里不太流行，但在那些非常有言论自由的国家里却十分流行。