息来找出文件最后修改的时间。如果最后修改时间晚于本地缓存时间，那么旧的副本被丢弃，新副本从服务器取回。最后，每30秒缓存定时器到期一次，缓存中所有的“脏”块（即修改过的块）都发送到服务器。尽管并不完美，但这些修补使得系统在多数实际环境中高度可用。

4. NFS第4版

网络文件系统第4版是为了简化其以前版本的一些操作而设计的。相对于上面描述的第3版NFS，第4版NFS是有状态的文件系统。这样就允许对远程文件调用open操作，因为远程NFS服务器将维护包括文件指针在内的所有文件系统相关的结构。读操作不再需要包含绝对读取范围了，而可以从文件指针上次所在的位置开始增加。这就使消息变短，同时可以在一次网络传输中捆绑多个第3版NFS的操作。

第4版NFS的有状态性使得将第3版NFS中多个协议（在本节前面部分描述过）集成为一个一致的协议变得容易。这样就没有必要再为挂载、缓存、加锁或者安全操作支持单独的协议了。第4版NFS在Linux（和UNIX）和Windows文件系统语义下都工作得更好。

10.7 Linux的安全性

Linux作为MINIX和UNIX的复制品，几乎从一开始就是一个多用户系统。这段历史意味着Linux从早期开始就建立了安全和信息访问控制。在接下来的几节里，我们将关注Linux 安全性的一些方面。

10.7.1 基本概念

一个Linux系统的用户群体由一定数量的注册用户组成，其中每个用户拥有一个惟一的UID（用户ID）。UID是介于0到65 535之间的一个整数。文件（进程及其他资源）都标记了它的所有者的UID。尽管可以改变文件所有权，但是默认情况下，文件的所有者是创建该文件的用户。

用户可以被分组，其中每组同样由一个16位的整数标记，叫做GID（组ID）。给用户分组通过在系统数据库中添加一条记录指明哪个用户属于哪个组的方法手工（由系统管理员）完成。一个用户可以同时属于多个组。为简单起见，我们不再深入讨论这个问题。

Linux中的基本安全机制很简单。每个进程记录它的所有者的UID和GID。当一个文件被创建时，它的UID和GID被标记为创建它的进程的UID和GID。该文件同时获得由该进程决定的一些权限。这些权限指定所有者、所有者所在组的其他用户及其他用户对文件具有什么样的访问权限。对于这三类用户而言，潜在的访问权限为读、写和执行，分别由r、w和x标记。当然，执行文件的权限仅当文件是可执行二进制程序时才有意义。试图执行一个拥有执行权限的非可执行文件（即，并非由一个合法的文件头开始的文件）会导致错误。因为有三类用户，每类用户的权限由3个比特位标记，那么9个比特位就足够标记访问权限。图10-37给出了一些9位数字及其含义的例子：

二 进 制	标 记	允许的文件访问权限
111000000	rwx------	所有者可以读、写和执行
111111000	rwxrwx---	所有者和组可以读、写和执行
110100000	rw-r-----	所有者可以读和写；组可以读
110100100	rw-r--r--	所有者可以读和写；其他人可以读
111101101	rwxr-xr-x	所有者拥有所有权限，其他人可以读和执行
000000000	---------	所有人都不拥有任何权限
000000111	------rwx	只有组以外的其他用户拥有所有权限（奇怪但是合法）

图10-37 文件保护模式的例子

图10-37前两行的意思很清楚，允许所有者以及与所有者同组的人所有权限。接下来的一行允许所有者同组用户读权限但是不可以改变其内容，而其他用户没有任何权限。第四行通常用于所有者想要公开的数据文件。类似地，第五行通常用于所有者想要公开的程序。第六行剥夺了所有用户的任何权利。这种模式有时用于伪文件来实现相互排斥，因为想要创建一个同名的文件的任何行为都将失败。如果多个进程同时想要创建这样一个文件作为锁，那么只有一个能够创建成功。最后一个例子相当奇怪，因为它给组以外其他用户更多的权限。但是，它的存在是符合保护规则的。幸运的是，尽管没有任何文件访问权限，但是所有者可以随后改变保护模式。