access系统调用检验用实际的UID和GID对某文件是否拥有特定的权限。对于根用户所拥有的并设置了SETUID的程序，我们需要这个系统调用来避免安全违例。这样的程序可以做任何事情，有时需要这样的程序判断是否允许用户执行某种访问。让程序通过访问判断显然是不行的，因为这样的访问总能成功。使用access系统调用，程序就能知道用实际的UID和GID是否能够以一定的权限访问文件。

接下来的四个系统调用返回实际的和有效的UID和GID。最后的三个只能够被超级用户使用，它们改变文件的所有者以及进程的UID和GID。

10.7.3 Linux中的安全实现

当用户登录的时候，登录程序login（为根用户所有且SETUID打开）要求输入登录名和密码。它首先计算密码的散列值，然后在/etc/passwd文件中查找，看是否有相匹配的项（网络系统工作得稍有不同）。使用散列的原因是防止密码在系统中以非加密的方式存在。如果密码正确，登录程序在/etc/passwd中读取该用户选择的shell程序的名称，例如可能是bash，但是也有可能是其他的shell，例如csh或者ksh。然后登录程序使用setuid和setgid来使自己的UID和GID变成用户的UID和GID（注意，它一开始的时候是根用户所有且SETUID打开）。然后它打开键盘作为标准输入（文件描述符0），屏幕为标准输出（文件描述符1），屏幕为标准错误输出（文件描述符2）。最后，执行用户选择的shell程序，因此终止自己。

到这里，用户选择的shell已经在运行，并且被设置了正确的UID和GID，标准输入、标准输出和标准错误输出都被设置成了默认值。它创建任何子进程（也就是用户输入的命令）都将自动继承shell的UID和GID，所以它们将拥有正确的UID和GID，这些进程创建的任何文件也具有这些值。

当任何进程想要打开一个文件，系统首先将文件的i节点所记录的保护位与用户的有效UID和有效GID对比，来检查访问是否被允许。如果允许访问，就打开文件并且返回文件描述符；否则不打开文件，返回−1。在接下来的read和write中不再检查权限。因此，当一个文件的保护模式在它被打开后修改，新模式将无法影响已经打开该文件的进程。

Linux安全模型及其实现在本质上跟其他大多数传统的UNIX 系统相同。

10.8 小结

Linux一开始是一个开源的完全复制UNIX的系统，而今天它已经广泛应用于各种系统，从笔记本到超级计算机。它有三种主要接口：shell、C函数库和系统调用。此外，通常使用图形用户界面以简化用户与系统的交互。shell 允许用户输入命令来执行。这些命令可能是简单的命令、管线或者复杂的命令结构。输入和输出可以被重定向。C函数库包括了系统调用和许多增强的调用，例如用于格式化输出的printf。实际的系统调用接口是依赖于体系结构的，在x86平台上大约有250个系统调用，每个系统调用做需要做的事情，不会做多余的事情。

Linux 中的关键概念包括进程、内存模型、I/O和文件系统。进程可以创建子进程，形成一棵进程树。Linux 中的进程管理与其他的UNIX系统不太一样，Linux 系统把每一个执行体——单线程进程，或者多线程进程中的每一个线程或者内核——看做不同的任务。一个进程，或者统称为一个任务，通过两个关键的部分来表示，即任务结构和描述用户地址空间的附加信息。前者常驻内存，后者可能被换出内存。进程创建是通过复制父进程的任务结构，然后将内存映像信息设置为指向父进程的内存映像。内存映像页面的真正复制仅当在共享不允许和需要修改内存单元时发生。这种机制称为写时复制。进程调度采用基于优先级的算法，给予交互式进程更高的优先级。

每个进程的内存模型由三个部分组成：代码、数据和堆栈。内存管理采用分页式。一个常驻内存的表跟踪每一页的状态，页面守护进程采用一种修改过的双指针时钟算法保证系统有足够多的空闲页。

可以通过特殊文件访问I/O设备，每个设备都有一个主设备号和次设备号。块设备I/O使用内存缓存磁盘块，以减少访问磁盘的次数。字符I/O可以工作在原始模式，或者字符流可以通过行规则加以修改。网络设备稍有不同，它关联了整个网络协议模块来处理网络数据包流。

文件系统由文件和目录所组成的层次结构组成。所有磁盘都挂载到一个有惟一根的目录树中。文件可以从文件系统的其他地方连接到一个目录下。要使用文件，首先要打开文件，这会产生一个文件描述符用于接下来的读和写。文件系统内部主要使用三种表：文件描述符表、打开文件描述表和i节点表。