UID为0的用户是一个特殊用户，称为超级用户（或者根用户）。超级用户能够读和写系统中的任何文件，不论这个文件为谁所有，也不论这个文件的保护模式如何。UID为0的进程拥有调用一小部分受保护的系统调用的权限，而普通用户是不能调用这些系统调用的。一般而言，只有系统管理员知道超级用户的密码，但是很多学生寻找系统安全漏洞想让自己能够不用密码就可以以超级用户的身份登录，并且认为这是一种了不起的行为。管理人员往往对这种行为很不满。

目录也是一种文件，并且具有普通文件一样的保护模式。不同的是，目录的x比特位表示查找权限而不是执行权限。因此，如果一个目录具有保护模式rwxr-xr-x，那么它允许所有者读、写和查找目录，但是其他人只可以读和查找，而不允许从中添加或者删除文件。

与I/O相关的特殊文件拥有与普通文件一样的保护位。这种机制可以用来限制对I/O设备的访问权限。例如，假设打印机特殊文件，/dev/lp，可以被根用户或者一个叫守护进程的特殊用户拥有，具有保护模式rw-------，从而阻止其他所有人对打印机的访问权限。毕竟，如果每个人都可以任意使用打印机，那么就会发生混乱。

当然，让/dev/lp被守护进程以保护模式rw-------拥有，意味着其他任何人都不可以使用打印机，但是这种做法限制了很多合法的打印要求。事实上，允许对I/O设备及其他系统资源进行受控访问的做法具有一个更普遍的问题。

这个问题通过增加一个保护位SETUID到之前的9个比特位来解决。当一个进程的SETUID位打开，它的有效UID将变成相应可执行文件的所有者的UID，而不是当前使用该进程的用户的UID。当一个进程试图打开一个文件时，系统检查的将是它的有效UID，而不是真正的UID。将访问打印机的程序设置为被守护进程所有，同时打开SETUID位，这样任何用户都可以执行该程序，并拥有守护进程的权限（例如访问/dep/lp），但是这仅限于运行该程序（例如给打印任务排序）。

许多敏感的Linux程序被根用户所有，但是打开它们的SETUID位。例如，允许用户改变密码的程序需要写password文件。允许password文件公开可写显然不是个好主意。解决的方法是，提供一个被根用户所有同时SETUID位打开的程序。虽然该程序拥有对password文件的全部权限，但是它仅仅改变调用该程序的用户的密码，而不允许其他任何的访问权限。

除了SETUID位，还有一个SETGID位，工作原理同SETUID类似。它暂时性地给用户该程序的有效GID。然而在实践中，这个位很少用到。

10.7.2 Linux中安全相关的系统调用

只有为数不多的几个安全性相关的系统调用。其中最重要的几个在图10-38中列出。最常用到的安全相关的系统调用是chmod。它用来改变保护模式。例如：

```
s=chmod("/usr/ast/newgame",0755);
```

它把newgame文件的保护模式修改为rwxr-xr-x，这样任何人都可以运行该程序（0755是一个八进制常数，这样表示很方便，因为保护位每三个分为一组）。只有该文件的所有者和超级用户才有权利改变保护模式。

系统调用	描　述
s = chmod(path, mode)	改变文件的保护模式
s = access(path, mode)	使用真实的UID和GID测试访问权限
uid = getuid( )	获取真实的UID
uid = geteuid( )	获取有效UID
gid = getgid( )	获取真实的GID
gid = getegid( )	获取有效GID
s = chown(path, owner, group)	改变所有者和组
s = setuid(uid)	设置UID
s = setgid(gid)	设置GID

图10-38 一些与安全相关的系统调用。当错误发生时，返回值s为-1；uid和gid分别是UID和GID。参数的意思不言自明