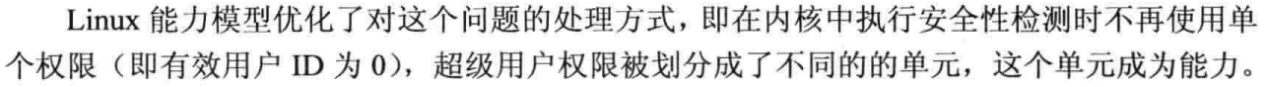
第三十九章

1. 能力
2. 能力的基本原理：

传统的UNIX能力模型是all-or-nothing，这样很不方便，这个模型的粗粒度划分是一个问题，会导致安全隐患。



1. 进程和文件的能力：

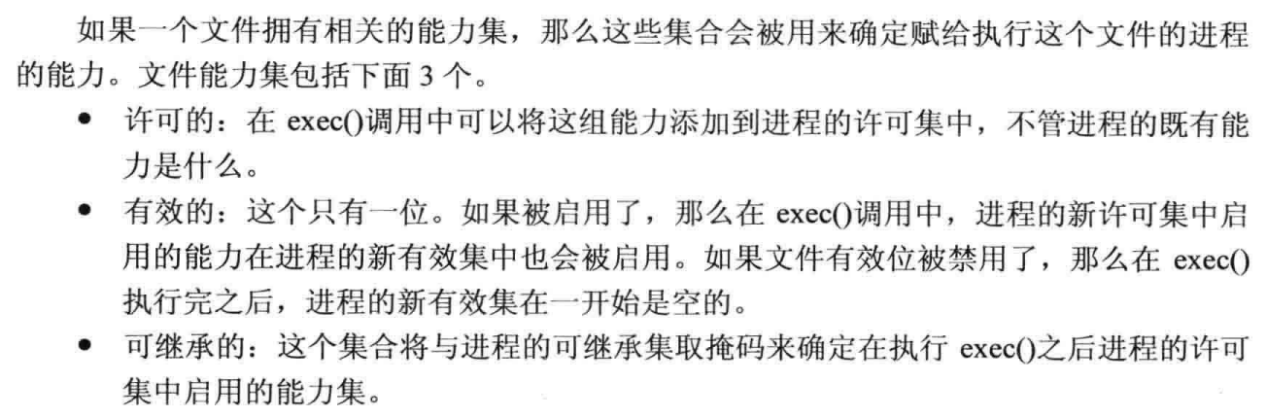
进程能力：



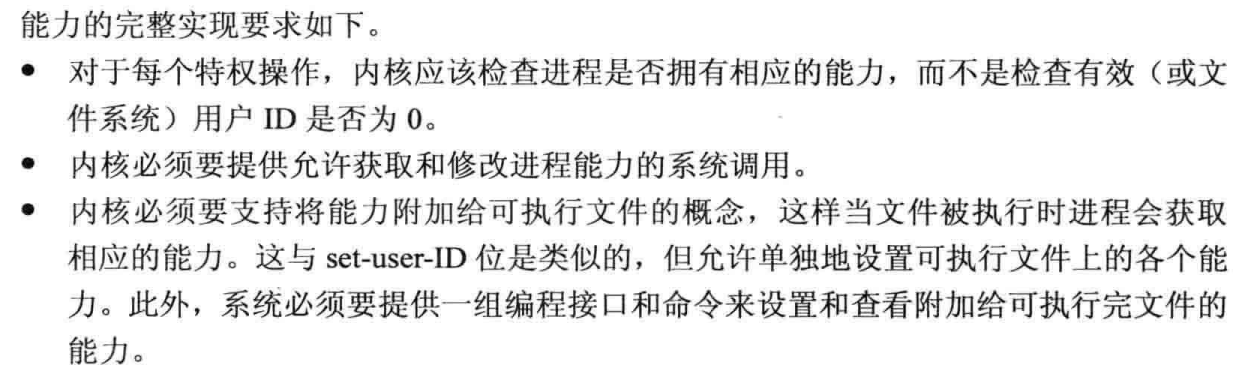
通过fork创建的子进程会继承其父进程的副本。

实际上，能力是线程级特性。

文件能力：

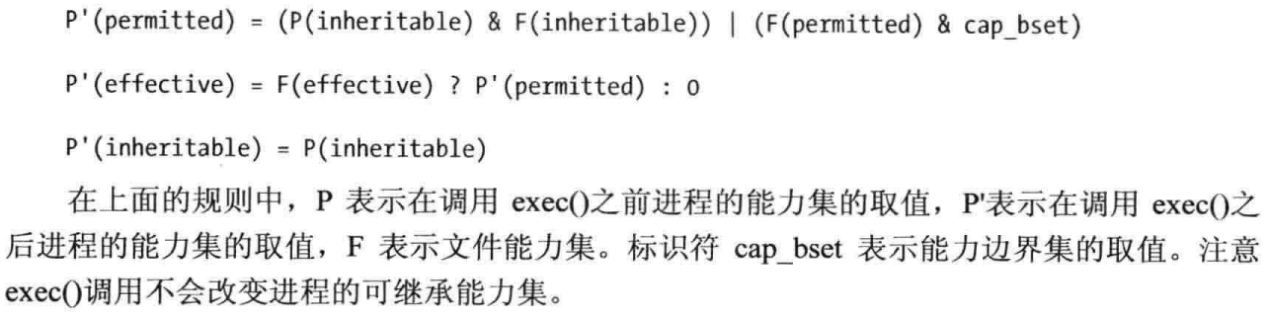


1. 现代能力的实现：



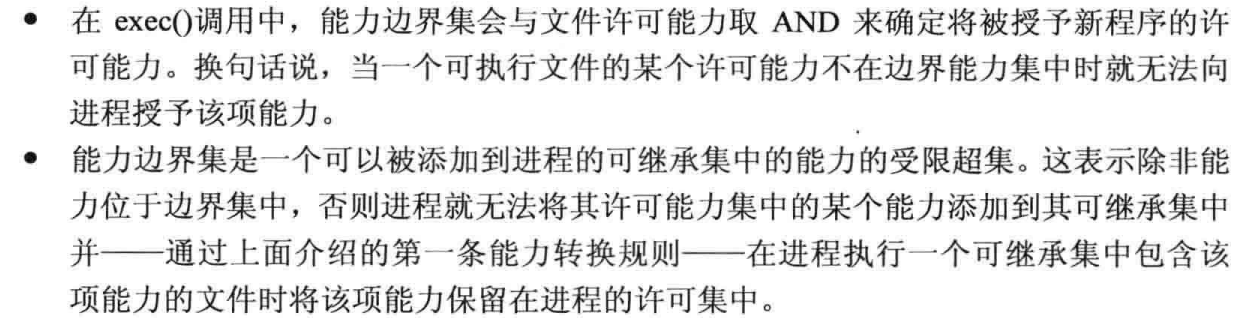
1. 在exec中转变进程能力：

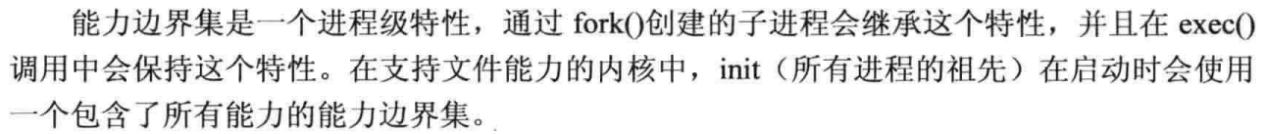
在exec执行期间，内核为进程设置新能力的规则如下：



1. 能力边界集：

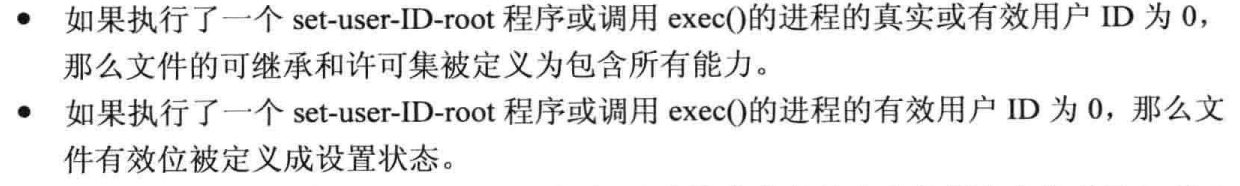
能力边界集作用：



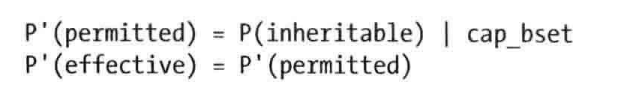


是一个线程级特性。在/proc/PID/status文件中用CapBnd表示。

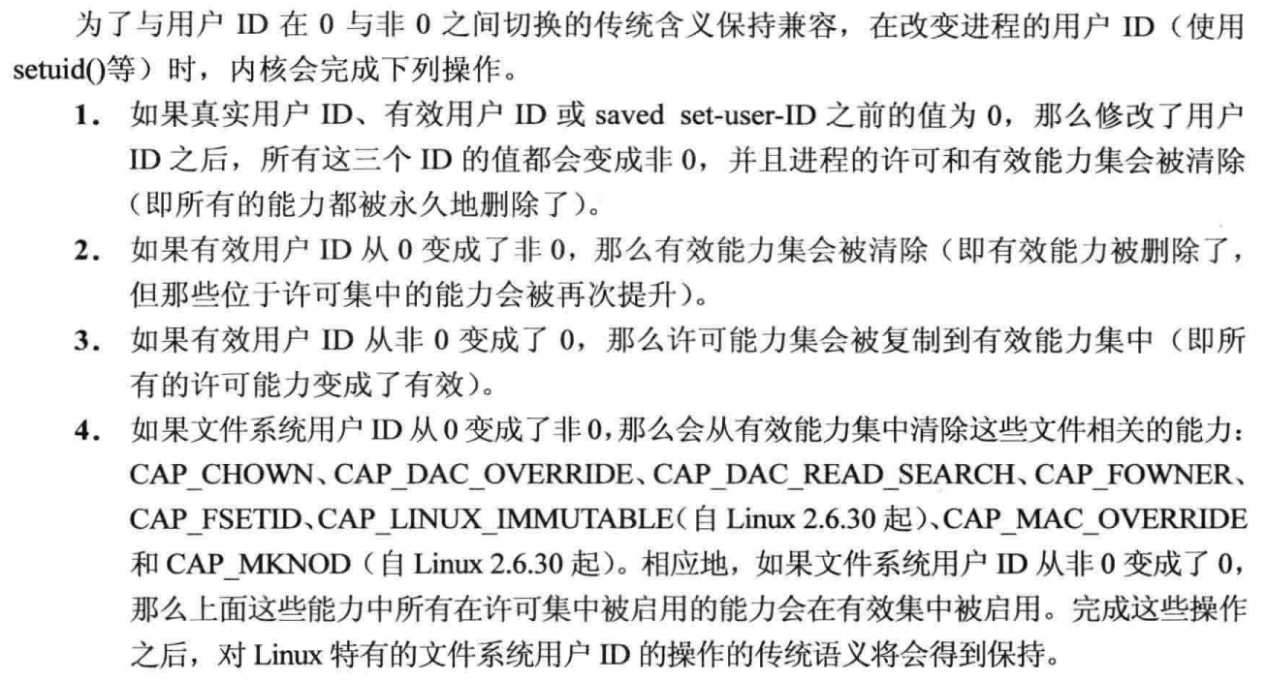
1. 保持root语义：



假设正在执行set-user-id-root程序，转换规律如下：



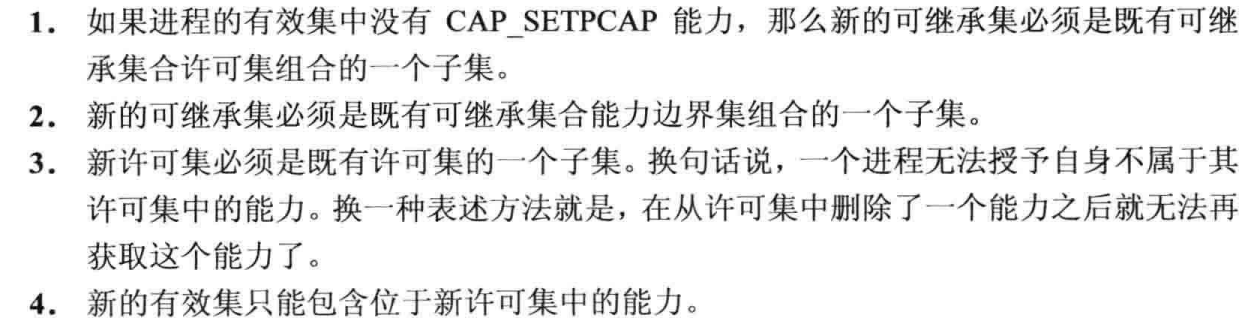
1. 改变用户ID对进程能力的影响：



1. 用编程的方式改变进程的能力：

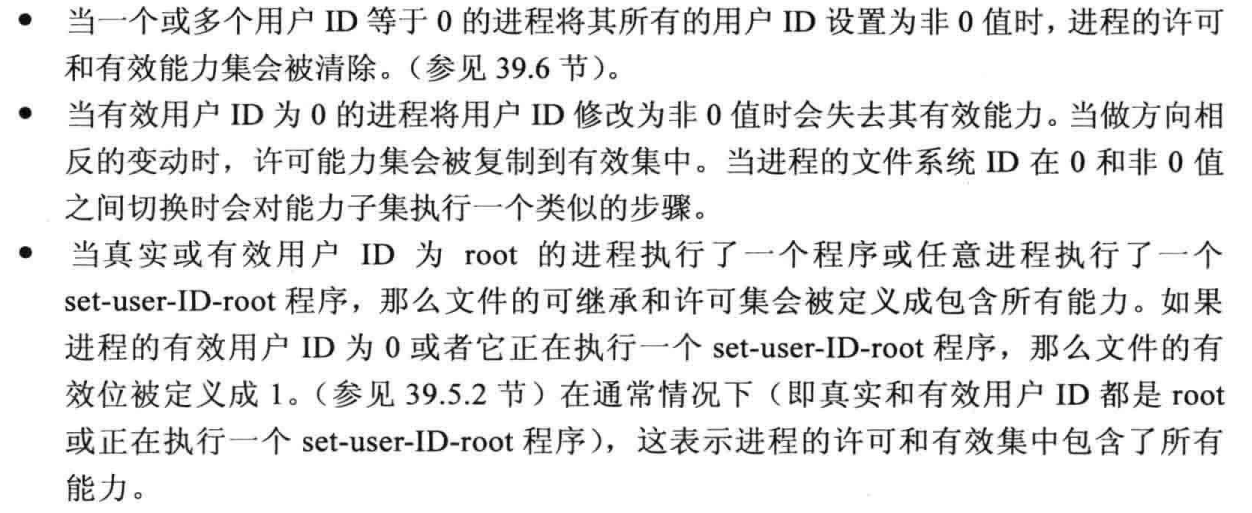
由于我的环境没有这个库，所以尝试不了。

修改进程能力需要遵循的规则：

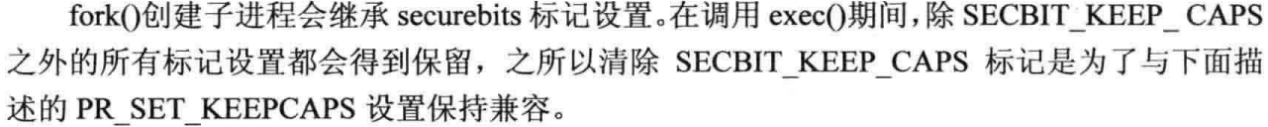


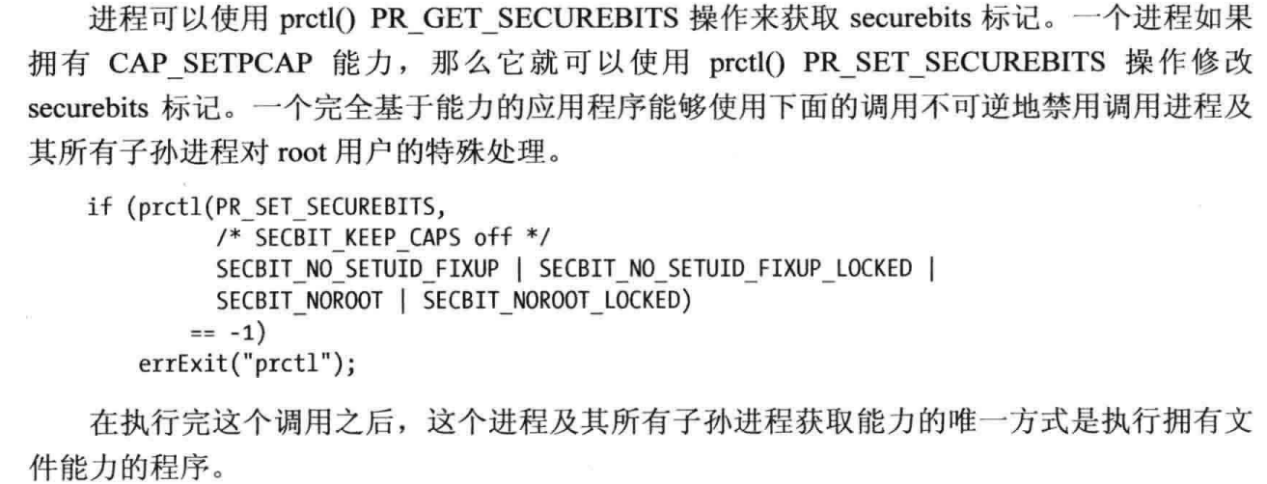
1. 创建仅包含能力的环境：

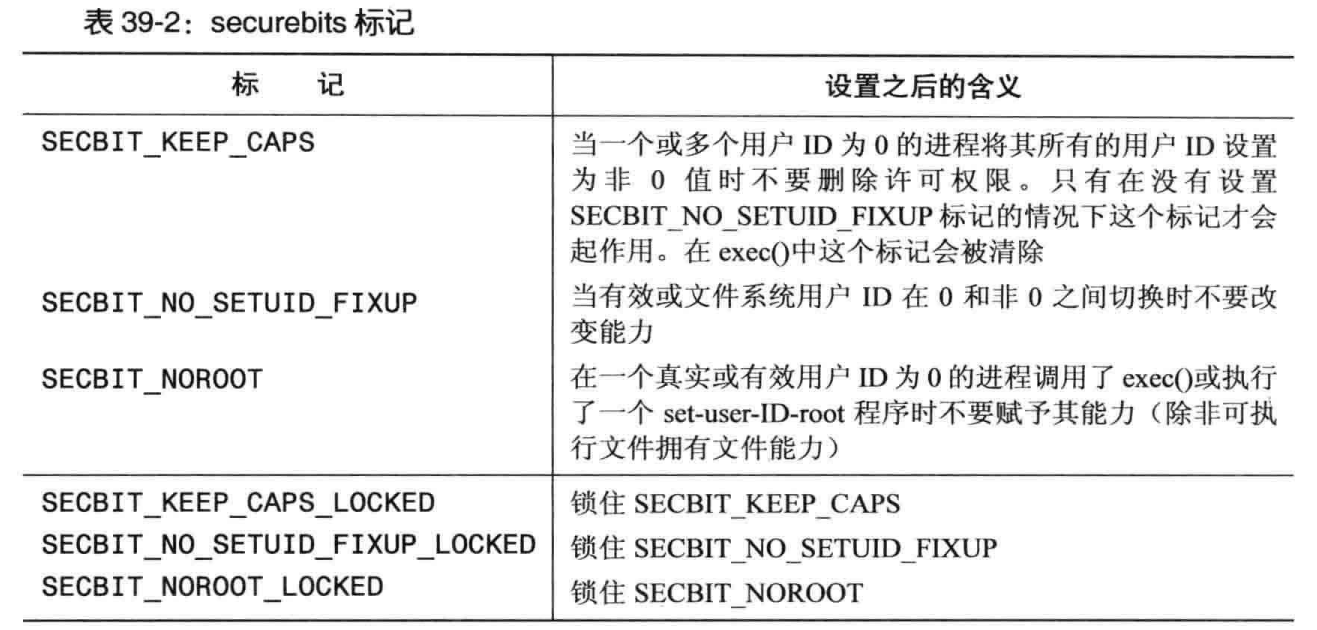
内核在能力方面对root进程进行了特殊的处理：



由于既有的程序不会使用文件能力基础架构，因此内核必须维持对root进程的特殊处理。但是，我们可以让程序在一个完全基于能力的环境中运行。

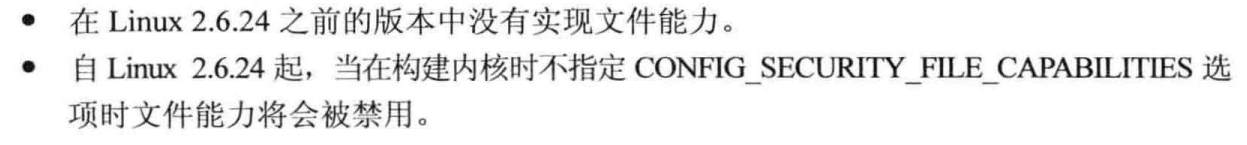






1. 发现程序所需的能力：
2. 使用strace。
3. 使用内核指针？？？
4. 不具备文件能力的老式内核和系统：

Linux在下面两个场景是不支持文件能力的：

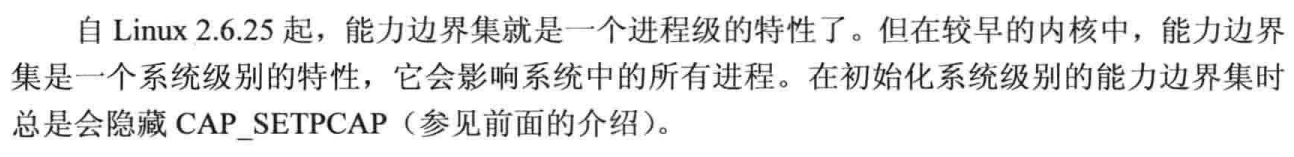


文件能力是使用第十六章的扩展特性实现的。

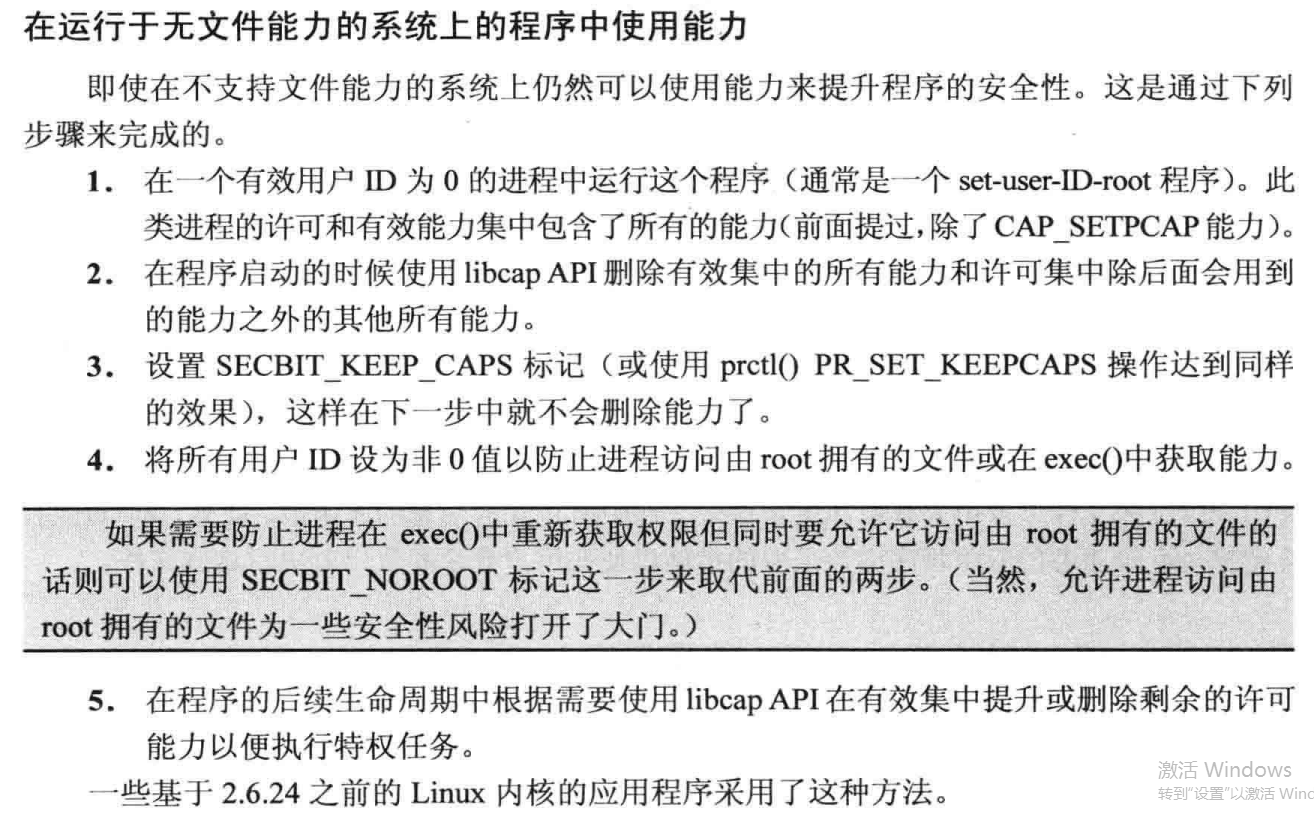
CAP\_SETPCAP能力：

在不支持文件能力的系统里，拥有这个能力的进程可以修改任何除了本身和init进程外所有进程的能力集。但只是理论上可行，因为这种情况下，能力边界集总是隐藏这个能力值。

能力边界值：



作为能够被添加进程的可继承集的受限集，也具有不同语义。



由上述推测：没有文件能力集的进程里，ID为非0的进程是没有任何能力的，而set-user-id-root或ID为0的进程里拥有除CAP\_SETPCAP能力之外的所有能力，要使用上面的方法限制。