[Android中如何修改系统时间（应用程序获得系统权限）](http://blog.csdn.net/liujian885/article/details/5404834)

标签： [android](http://www.csdn.net/tag/android" \t "_blank)[user](http://www.csdn.net/tag/user)[eclipse](http://www.csdn.net/tag/eclipse)[虚拟机](http://www.csdn.net/tag/%e8%99%9a%e6%8b%9f%e6%9c%ba)[api](http://www.csdn.net/tag/api)[工具](http://www.csdn.net/tag/%e5%b7%a5%e5%85%b7)

2010-03-22 16:12 49230人阅读 [评论](http://blog.csdn.net/liujian885/article/details/5404834#comments)(31) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/liujian885/article/details/5404834#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

Android（11） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

在 [**Android**](http://lib.csdn.net/base/15) 的API中有提供 SystemClock.setCurrentTimeMillis()函数来修改系统时间，可惜无论你怎么调用这个函数都是没用的，无论模拟器还是真机，在logcat中总会得到"Unable to open alarm driver: Permission denied ".这个函数需要root权限或者运行与系统进程中才可以用。

        本来以为就没有办法在应用程序这一层改系统时间了，后来在网上搜了好久，知道这个目的还是可以达到的。

        第一个方法简单点，不过需要在Android系统源码的环境下用make来编译：

        1. 在应用程序的AndroidManifest.xml中的manifest节点中加入android:sharedUserId="android.uid.system"这个属性。

        2. 修改Android.mk文件，加入LOCAL\_CERTIFICATE := platform这一行

        3. 使用mm命令来编译，生成的apk就有修改系统时间的权限了。

        第二个方法麻烦点，不过不用开虚拟机跑到源码环境下用make来编译：

        1. 同上，加入android:sharedUserId="android.uid.system"这个属性。

        2. 使用eclipse编译出apk文件，但是这个apk文件是不能用的。

        3. 用压缩软件打开apk文件，删掉META-INF目录下的CERT.SF和CERT.RSA两个文件。

        4. 使用目标系统的platform密钥来重新给apk文件签名。这步比较麻烦，首先找到密钥文件，在我的Android源码目录中的位置是"build/target/product/security"，下面的platform.pk8和platform.x509.pem两个文件。然后用Android提供的Signapk工具来签名，signapk的源代码是在"build/tools/signapk"下，用法为"signapk platform.x509.pem platform.pk8 input.apk output.apk"，文件名最好使用绝对路径防止找不到，也可以修改源代码直接使用。

        这样最后得到的apk和第一个方法是一样的。

        最后解释一下原理，首先加入android:sharedUserId="android.uid.system"这个属性。通过Shared User id,拥有同一个User id的多个APK可以配置成运行在同一个进程中。那么把程序的UID配成android.uid.system，也就是要让程序运行在系统进程中，这样就有权限来修改系统时间了。

        只是加入UID还不够，如果这时候安装APK的话发现无法安装，提示签名不符，原因是程序想要运行在系统进程中还要有目标系统的platform key，就是上面第二个方法提到的platform.pk8和platform.x509.pem两个文件。用这两个key签名后apk才真正可以放入系统进程中。第一个方法中加入LOCAL\_CERTIFICATE := platform其实就是用这两个key来签名。

        这也有一个问题，就是这样生成的程序只有在原始的Android系统或者是自己编译的系统中才可以用，因为这样的系统才可以拿到platform.pk8和platform.x509.pem两个文件。要是别家公司做的Android上连安装都安装不了。试试原始的Android中的key来签名，程序在模拟器上运行OK，不过放到G3上安装直接提示"Package ... has no signatures that match those in shared user android.uid.system"，这样也是保护了系统的安全。

        最最后还说下，这个android:sharedUserId属性不只可以把apk放到系统进程中，也可以配置多个APK运行在一个进程中，这样可以共享数据，应该会很有用的。

文件位置：可以在platform/build/target/product/security/中找到platform.pk8 platform.x509.pem等签名文件，对应不同的权限。  
signapk.jar：由/platform/build/tools/signapk/编译产出,可以在/out/host/linux-x86/framework/中找到。  
  
签名：  
java -jar signapk.jar platform.x509.pem platform.pk8 MyDemo.apk MyDemo\_signed.apk 得到具有对应权限的APK  
  
优化APK：  
zipalign -v 4 MyDemo\_signed.apk MyDemo\_new.apk  
  
查循APK是否优化过：  
zipalign -c -v 4 MyDemo.apk