一次完整的Android打包要进行以下的几步：编译、代码混淆、打包apk、签名apk、apk优化。

为了能包涵使用NDK的情况，在这里使用一个有native代码的工程TestJni。

在工程根目录下新建local.properties文件，输入sdk和ndk的路径

例如：

sdk.dir=D:\\Android\\android-sdk
ndk.dir=D:\\Android\\android-ndk

在工程根目录下新建build.xml，输入代码


```
<?xml version="1.0" encoding="UTF-8"?>
<project name="TestJni" default="my.package">
    <loadproperties srcFile="local.properties" />
    <loadproperties srcFile="project.properties" />

    <fail message="sdk.dir is missing." unless="sdk.dir" />
    <fail message="ndk.dir is missing." unless="sdk.dir" />

    <target name="native">
        <echo message="Building native libraries..." />
        <exec executable="${ndk.dir}/ndk-build.cmd" failonerror="true" />
        <echo message="DONE (Building native libraries)" />
    </target>

    <import file="${sdk.dir}/tools/ant/build.xml" />

    <target name="my.package" depends="native,release">
    </target>
</project>
```

my.package依赖于native和release，执行了一次ndk编译和release操作。

打开cmd，切换到工程根目录，输入ant my.package，等待build完成，在bin目录下生成了未签名的apk文件。

代码混淆

如果在**project.properties**中配置了**proguard.config=proguard.cfg**，release的时候会自动进行代码混淆。在${sdk.dir}/tools/ant/build.xml可以找到相应的target。

```
    <target name="-release-obfuscation-check">
        <echo level="info">proguard.config is ${proguard.config}</echo>
        <condition property="proguard.enabled" value="true" else="false">
            <and>
                <isset property="build.is.mode.release" />
                <isset property="proguard.config" />
            </and>
        </condition>
        <if condition="${proguard.enabled}">
```

```
            <then>
                <echo level="info">Proguard.config is enabled</echo>
                <!-- Secondary dx input (jar files) is empty since all the
                     jar files will be in the obfuscated jar -->
                <path id="out.dex.jar.input.ref" />
            </then>
        </if>
    </target>
```

proguard.enabled的依据就是在release模式下，并且设置proguard.config这个属性。

签名

在${sdk.dir}/tools/ant/build.xml查找release target

```
    <target name="release"
                depends="-set-release-mode, -release-obfuscation-check, -package, -post-package, -release-prompt-for-password, -release-nosign, -release-sign, -post-build"
                description="Builds the application in release mode.">
    </target>
```

我们看到有 -release-sign 这个 targer，继续查找这个target

查看第一行

```
<target name="-release-sign" if="has.keystore" >
```

如果执行签名的话 has.keystore 这个条件要成立，继续查找 has.keystore

```
    <!-- properties for signing in release mode -->
    <condition property="has.keystore">
        <and>
            <isset property="key.store" />
            <length string="${key.store}" when="greater" length="0" />
            <isset property="key.alias" />
        </and>
    </condition>
    <condition property="has.password">
        <and>
            <isset property="has.keystore" />
            <isset property="key.store.password" />
            <isset property="key.alias.password" />
        </and>
    </condition>
```

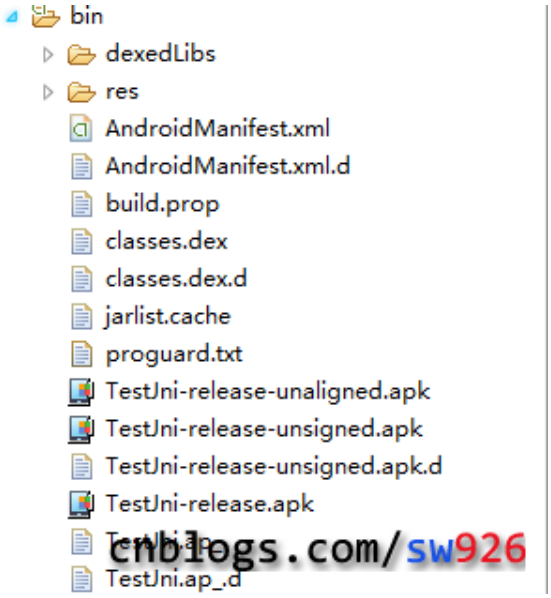
找到上面两个 condition，has.keystore成立的条件是设置了 key.store 和 key.alias 这两个property，并且 key.store 长度不能为0。

has.password 成立的条件是，has.keystore成立， 并且设置了 key.store.password 和 key.alias.password，如果没有这两个属性的话，在build过程中会要求输入密码。

在local.properties中添加这些属性

key.store=D:\\Android\\keystore\\test_key.keystore
key.alias=test_key
key.store.password=123456
key.alias.password=123456

打开cmd，切换到工程根目录，执行 ant my.package， 在bin目录中生成了 TestJni-release.apk 。



检查生成的apk是否进行了优化

```
D:\Android\workspace\TestJni\bin>D:\Android\android-sdk\tools\zipalign.exe -c -v
 4 TestJni-release.apk
Verifying alignment of TestJni-release.apk (4)...
      62 res/layout/activity_main.xml (OK - compressed)
     444 res/menu/main.xml (OK - compressed)
     755 AndroidManifest.xml (OK - compressed)
    1436 resources.arsc (OK)
    3704 res/drawable-hdpi/ic_launcher.png (OK)
    9732 res/drawable-mdpi/ic_launcher.png (OK)
   12908 res/drawable-xhdpi/ic_launcher.png (OK)
   22328 res/drawable-xxhdpi/ic_launcher.png (OK)
   40258 classes.dex (OK - compressed)
  230912 lib/x86/libTestJni.so (OK - compressed)
  232275 META-INF/MANIFEST.MF (OK - compressed)
  232779 META-INF/CERT.SF (OK - compressed)
  233312 META-INF/CERT.RSA (OK - compressed)
Verification succesful
```

至此，打包完成。