科研早培

Android系统与应用安全

——Android基础入门

黄建军

2023.01.13

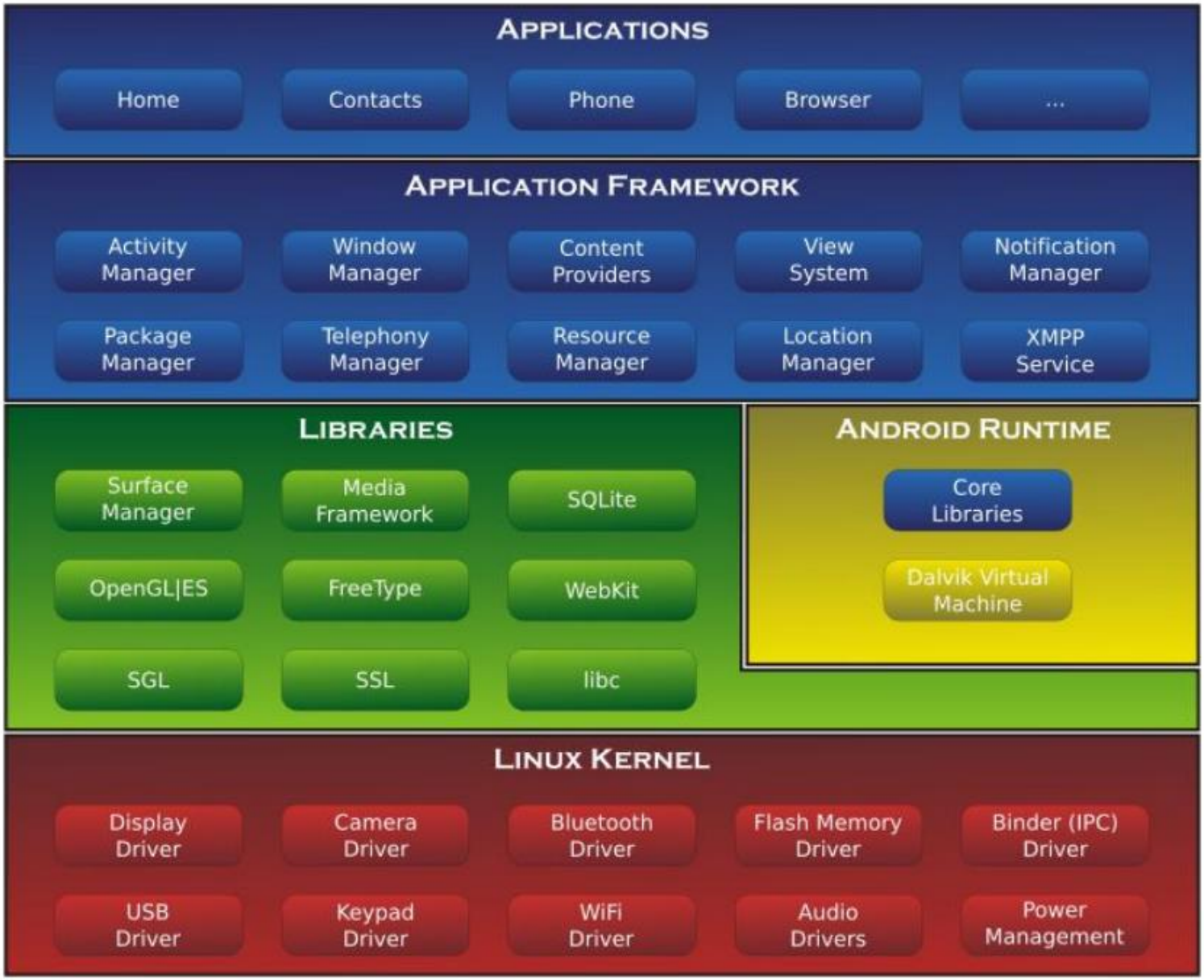
群聊：科研早培-Android 系统与应用安全

该二维码7天内(1月19日前)有效，重新进入将更新

内容

- Android系统架构
- Android应用的基本构造
- Android安全权限机制
- Android Studio用法简介
- Android应用开发示例
- adb使用演示
- ApkTool
- 练习

Android系统架构



系统架构

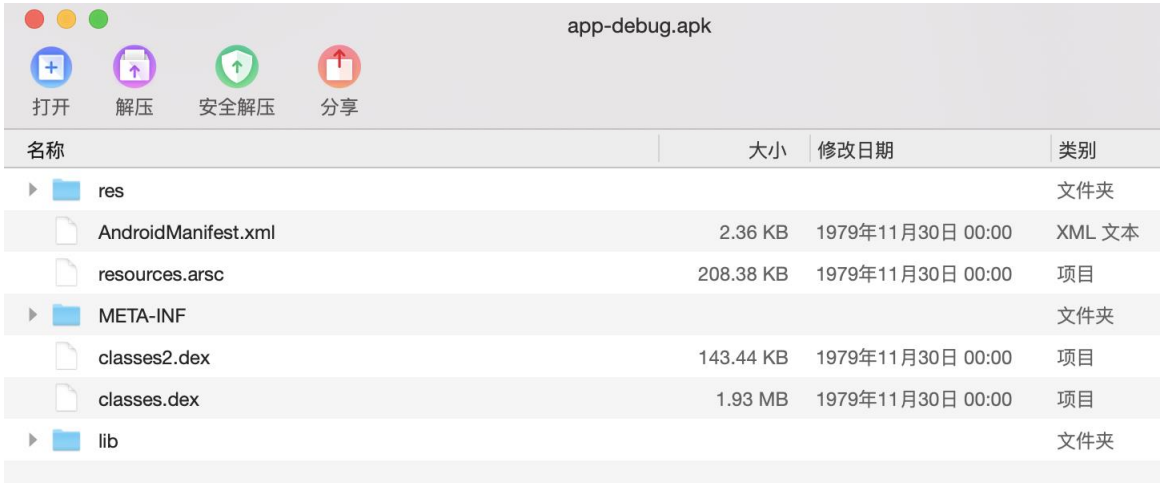
- 系统运行层
  - 通过一些C/C++库来为Android系统提供了主要的特性支持，如SQLite提供了数据库的支持、Webkit提供了浏览器内核的支持等
  - Android运行时库提供了使用Java编写的应用程序运行所需的核心库以及Dalvik虚拟机。该虚拟机使得每一个Android应用都运行在独立的进程中，相比于Java虚拟机，Dalvik对手机内存、CPU性能有限等情况做了优化处理
    - 从4.4开始，Android引入了ART模式（Android Runtime），在应用程序安装时，将DEX格式的Java字节码（由Dalvik虚拟机执行）编译为机器码，提升运行效率

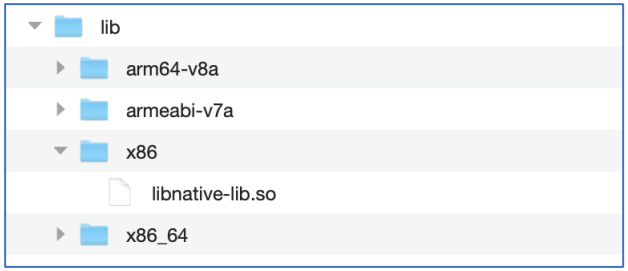
系统架构

- 应用框架层
  - 提供了构建应用时可能用到的API
  - 开发者可以使用这些API构建自己的应用程序
  - 提供了隐藏在每个应用程序后面的一系列的服务
    - 丰富可扩展的视图：列表、网格、文本框、按钮等
    - Content Provider：提供一层数据库访问操作的包装，使得应用程序可以访问另一个应用程序的数据或共享自己的数据
    - Resource Manager：提供非代码资源的访问，如本地字符串、图形、布局文件等
    - Activity Manager：管理应用程序生命周期并提供常用的导航回退功能
    - ……

Android应用的基本构造

- Android应用的安装包是APK格式（Android Application Package），是一种zip压缩格式，可通过常用的解压软件（如7-zip）解压
- 一个APK通常包括



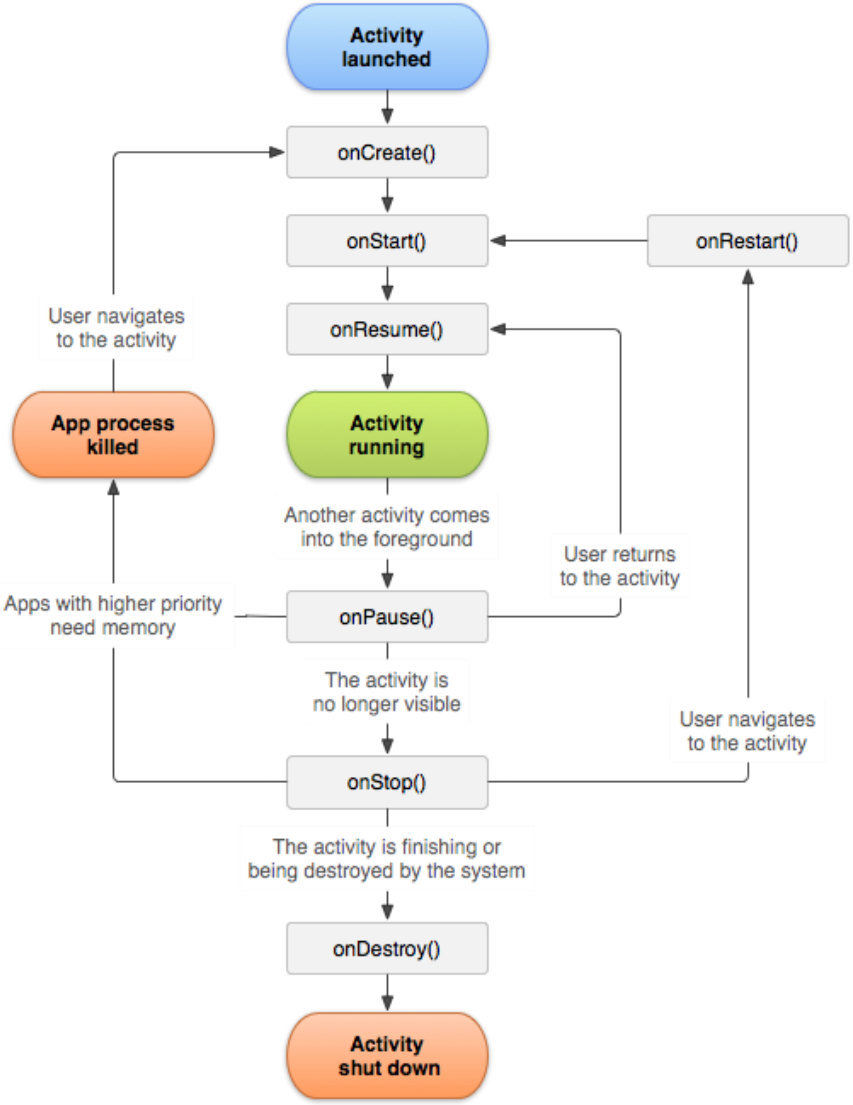


Android应用的组件

- Android应用通常包含四大组件
  - Activity：为用户提供了一个可以通过与之交互来完成某些操作的界面。通常来说，Activity是Android应用中最重要的组件。对绝大多数应用来说，与用户交互是其最基本的功能
  - Service：可以在后台长时间运行，且不提供用户界面，如E-Mail更新检查服务
  - Broadcast Receiver：能在设备系统中接收广播通知的组件，能接受诸如低电量、启动完成、耳机连接、网络更换等消息以及其他应用程序发出的消息
  - Content Provider：以表格的形式提供数据，可以用于与其他应用共享数据。所有的Content Provider使用content://开头的特殊格式的URI，如content://sms/inbox提供了内置短信应用收件箱的接口

Activity生命周期

- 用户使用、退出、返回应用程序，应用的一个Activity实例在生命周期（lifecycle）中不同状态间转换
- 系统通过调用回调函数使Activity知道状态转换
- 在不同的回调函数实现中，声明Activity应该执行什么操作



https://developer.android.com/guide/components/activities/activity-lifecycle

Application组件

- Android的Application：
  - Application是Android框架的一个系统组件，每个应用启动时，系统都会创建一个Application对象
  - Android系统为每个应用程序运行时创建的Application仅有一个，其生命周期即为整个应用程序的生命周期，诞生于其他任何组件对象之前，并一直存活直到应用进程结束
  - Application在应用程序运行过程中不会改变，而Activity、Service等都可能不停的在创建和销毁，但在每个Activity、Service等对象内部获取到地Application都是同一个
  - Application对象全程陪同应用进程，适合共享全局状态，初始化应用所需服务
  - Application的回调函数包括onCreate()，onConfigurationChanged()等
  - 开发者通常不需要指定一个Application，系统会在启动应用进程时自动创建
  - 自定义的Application需要在AndroidManifest.xml中注册

Android Intent

- Android组件或应用之间如何通信（如启动、发消息）？

- Android提供Intent机制来协助应用或组件间的通信
- Intent这个英语单词的本意是“目的、意向、意图”
- 通过Intent，应用程序可以向Android表达某种请求或者意愿，Android会根据意愿的内容选择适当的组件来响应
  - Intent负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述
  - Android则根据此Intent的描述，负责找到对应的组件，将 Intent传递给调用的组件，并完成组件的调用
- 一个Intent消息可以包含多个字段（属性）
  - 组件Component
  - 动作Action
  - 数据Data
  - ……

Intent解析

- Android系统收到一个Intent消息之后，解析其中包含的信息，寻找一个合适的目标应用程序或组件来响应该消息
- **显式Intent**
  - 通过setComponent、setClass等方法显式地指明消息接收方，Android系统直接启动对应的应用程序或组件，并将Intent消息传递给目标程序/组件
  - 目标应用程序/组件不需要在AndroidManifest.xml声明额外的信息，只需要能够被访问就行
- **隐式Intent**
  - 通过setAction、setData等方式为Intent添加信息，Android系统在所有已注册的相应组件中寻找声明了可以处理这些信息的组件，启动对应组件，将Intent消息传递给被启动的组件
  - 目标组件需要在AndroidManifest.xml声明能够接收响应的消息类型，如特定的action字段

```
<activity
    android:name=".SndActivity"
    android:permission="android.permission.SEND_SMS">
    <intent-filter>
        <action android:name="com.example.huang.ama2.snd" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

AndroidManifest.xml

- AndroidManifest文件声明了一个Android应用所需要的权限、组件（Activity、Service等）

SndActivity.java | native-lib.cpp | AndroidManifest.xml | MainActivity.java | acti

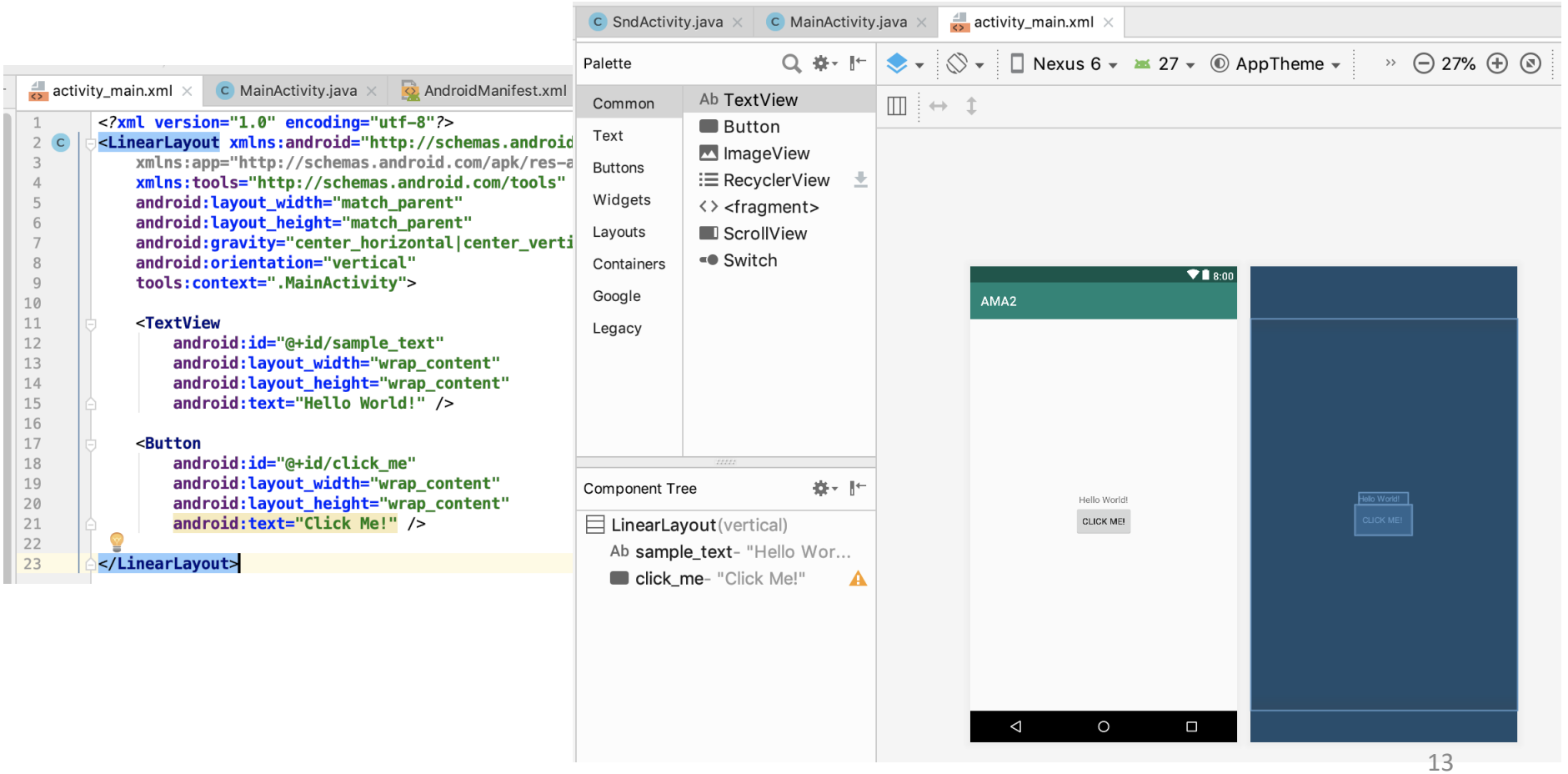
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.huang.ama2" >
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="@string/app_name"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@style/AppTheme" >
12         <activity android:name=".MainActivity" >
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15                 <category android:name="android.intent.category.LAUNCHER" />
16             </intent-filter>
17         </activity>
18         <activity android:name=".SndActivity" >
19             <intent-filter>
20                 <action android:name="com.example.huang.ama2.snd" />
21                 <category android:name="android.intent.category.DEFAULT" />
22             </intent-filter>
23         </activity>
24     </application>
25
26 </manifest>
```

自定义Application在<application>标签上注册

未直接指明接收消息Intent的类的时候，如果消息的action对应该字符串，系统会将消息发给SndActivity

布局文件（Layout XML）

- XML格式的布局文件（layout）用于静态定义用户界面。应用程序代码加载布局文件之后，系统渲染生成相应的界面
  - res/layout/activity_main.xml



Java代码

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView tv = (TextView) findViewById(R.id.sample_text);
        Button btn = (Button) findViewById(R.id.click_me);
        btn.setOnClickListener(this);
    }

    public void onClick(View view) {
        Intent intent = new Intent();
        intent.setAction("com.example.huang.ama2.snd");
        startActivity(intent);
    }
}
```

加载布局文件

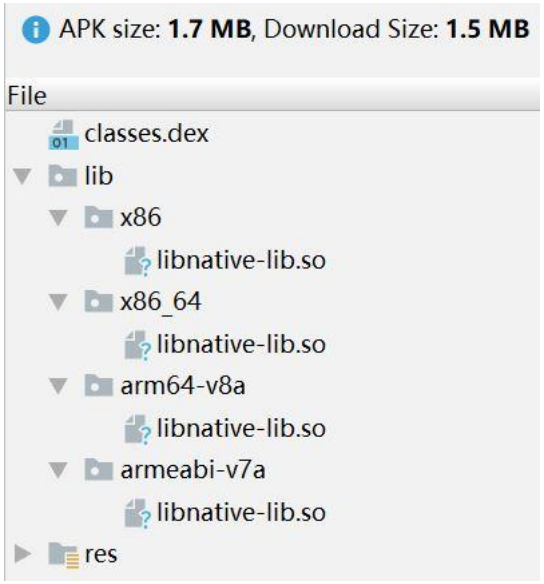
获取界面元素、设置事件监听器

发送Intent消息，由系统寻找并启动相应的Activity

一个Activity至少应该实现其生命周期中的onCreate方法，在其中加载并渲染对应的界面布局文件生成用户界面

本地（Native）代码

- 使用C/C++开发的本地代码能够为Android应用带来性能提升（相比于使用Java代码实现同样的功能）
  - 参考Java应用开发中的JNI部分
    - https://en.wikipedia.org/wiki/Java_Native_Interface
    - https://developer.android.com/ndk/samples/sample_hellojni

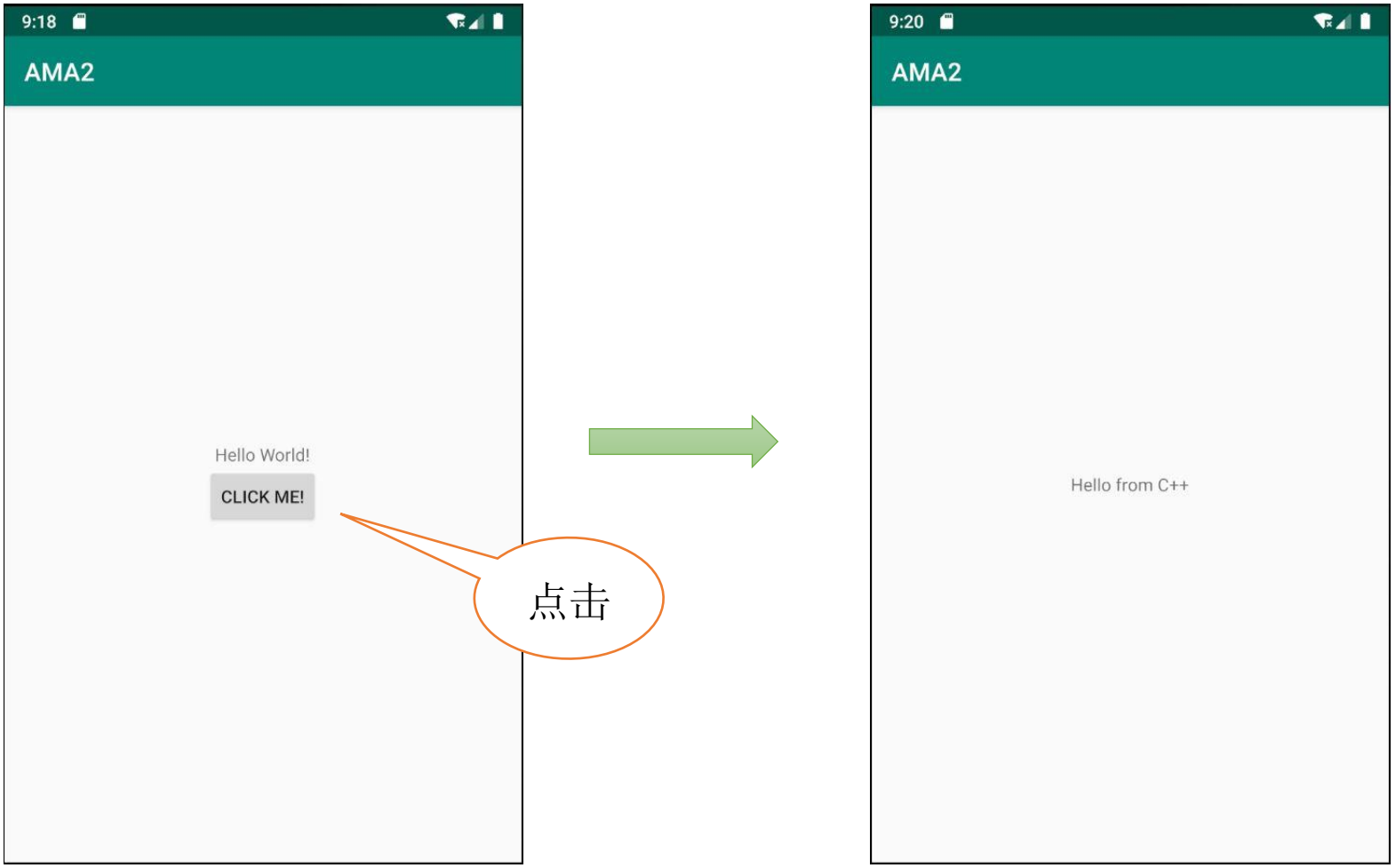



```
public class SndActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_snd);

        // Example of a call to a native method
        TextView tv = (TextView) findViewById(R.id.snd_sample_text);
        tv.setText(stringFromJNI());
    }

    /**
     * A native method that is implemented by the 'native-lib' native
     * which is packaged with this application.
     */
    public native String stringFromJNI();

    // Used to load the 'native-lib' library on application startup.
    static {
        System.loadLibrary( libname: "native-lib");
    }
```

```
#include <jni.h>
#include <string>

extern "C" JNIEXPORT jstring JNICALL
Java_com_example_huang_ama2_SndActivity_stringFromJNI(
        JNIEnv* env,
        jobject /* this */) {
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

运行结果

- 在模拟器（Emulator：Pixel 3 API 28/Android 9.0/x86）上运行结果



Android安全权限机制

- Android继承了Linux的安全机制，并发展出了**权限**（Permission）机制，系统更多的安全功能通过该机制提供
- 权限可以限制某个特定进程的特定操作，也可以限制每个URI权限对特定数据段的访问
- Android安全架构的核心思想：默认设置下，所有应用都没有权限对其他应用、系统或用户进行较大影响的操作，包括读写用户隐私数据（联系人或电子邮件）、读写其他应用文件、访问网络、发送短信等
- 常见的系统原生权限包括：发送短信/彩信、拨打电话、修改SD卡上的内容、读取联系人、获得精确的基于GPS的地理位置、振动控制、访问网络等

应用程序申请所需权限

- 开发者在AndroidManifest.xml中申请所需权限

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.permissiondemo">
   <!-- 请求网络权限 -->
   <uses-permission android:name=“android.permission.INTERNET” />
   <application android:allowBackup="true" android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name" android:supportsRtl="true“
        android:theme="@style/AppTheme">
            ……
    </application>
</manifest>
```

为组件设置权限

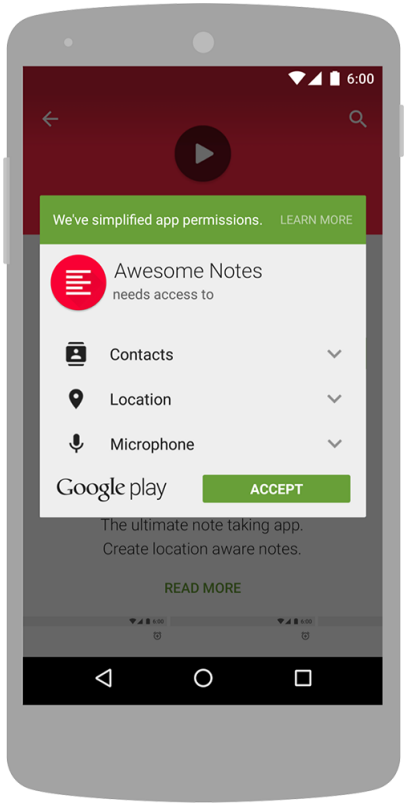
- 通过为组件设置权限，可以限制没有相应权限的其他应用程序访问、启动该组件（当前应用没有申请该权限，也可以访问其本身内部被保护的组件）

权限批准

- 如果应用程序包含Normal权限申请，系统自动地赋予应用程序相应的权限
- 如果应用包含Dangerous权限申请，需要用户显式地同意并批准相应权限
  - 安装时权限请求
  - 运行时权限请求

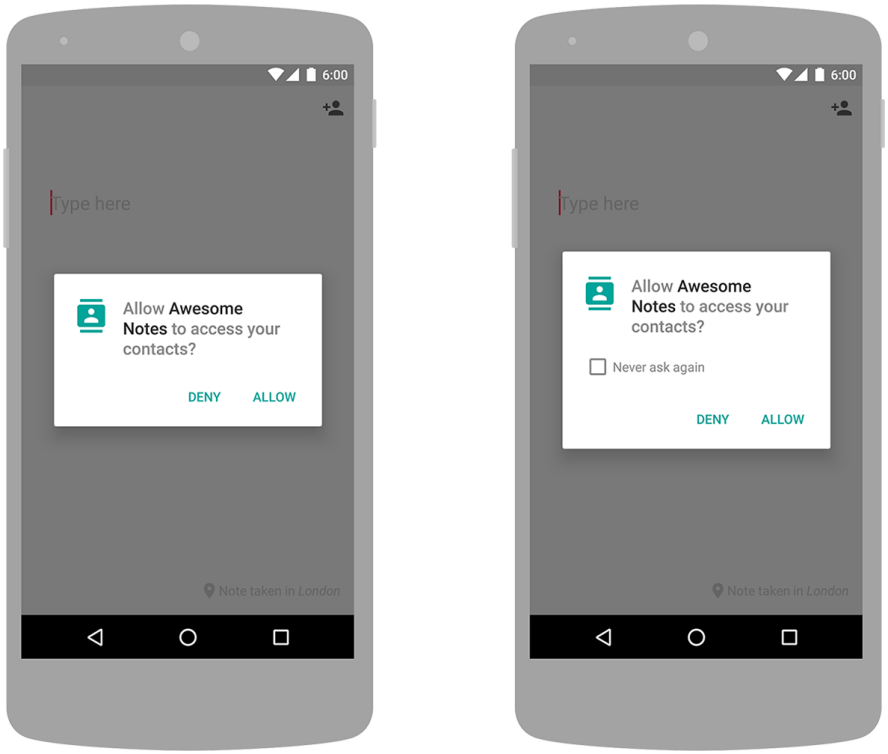
安装时权限请求

- Android 5.1及更低的版本中，应用程序申请的权限如果包含Dangerous权限，需要在安装时用户同意授权
  - 权限的层次划分请参阅相关资料
  - 常用的权限通常是Dangerous权限
- 用户不同意，则无法安装应用程序
- 应用程序的更新版本如果要求额外的权限，则更新时用户被询问是否同意新的权限请求



运行时权限请求

- 从Android 6.0开始，Android修改了危险权限的请求授权模型：应用必须在使用的时候进行申请，用户可以自主选择是否授权。
- 如果拒绝授权，应用也不会崩溃（视代码实现而定），只是用户无法使用相应功能



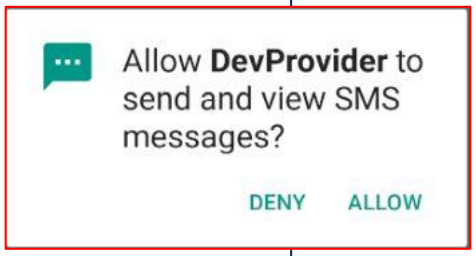
运行时权限请求-示例

- 在执行需要特定权限的操作之前，检测当前应用是否拥有相关权限；如果没有，则请求该权限（下方小图）

```
if (ContextCompat.checkSelfPermission( context: MainActivity.this,
        permission: "android.permission.READ_SMS") != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions( activity: MainActivity.this,
            new String[]{"android.permission.READ_SMS"}, requestCode: 1);
} else {
    // 实际需要该权限才能执行的操作
}
```

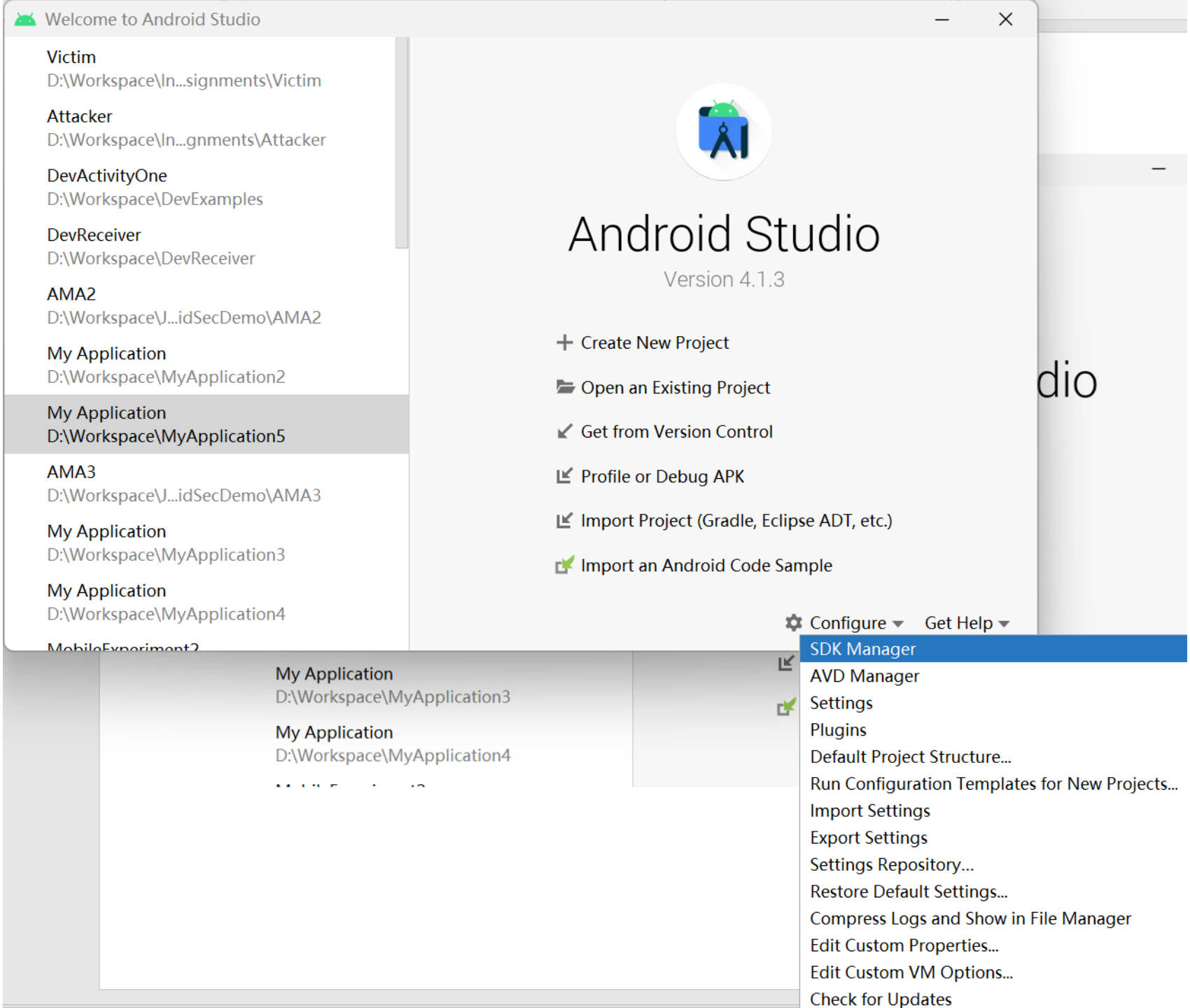
- 请求该权限后的回调函数如下，在其中判断，如果请求获批则可直接执行目标操作

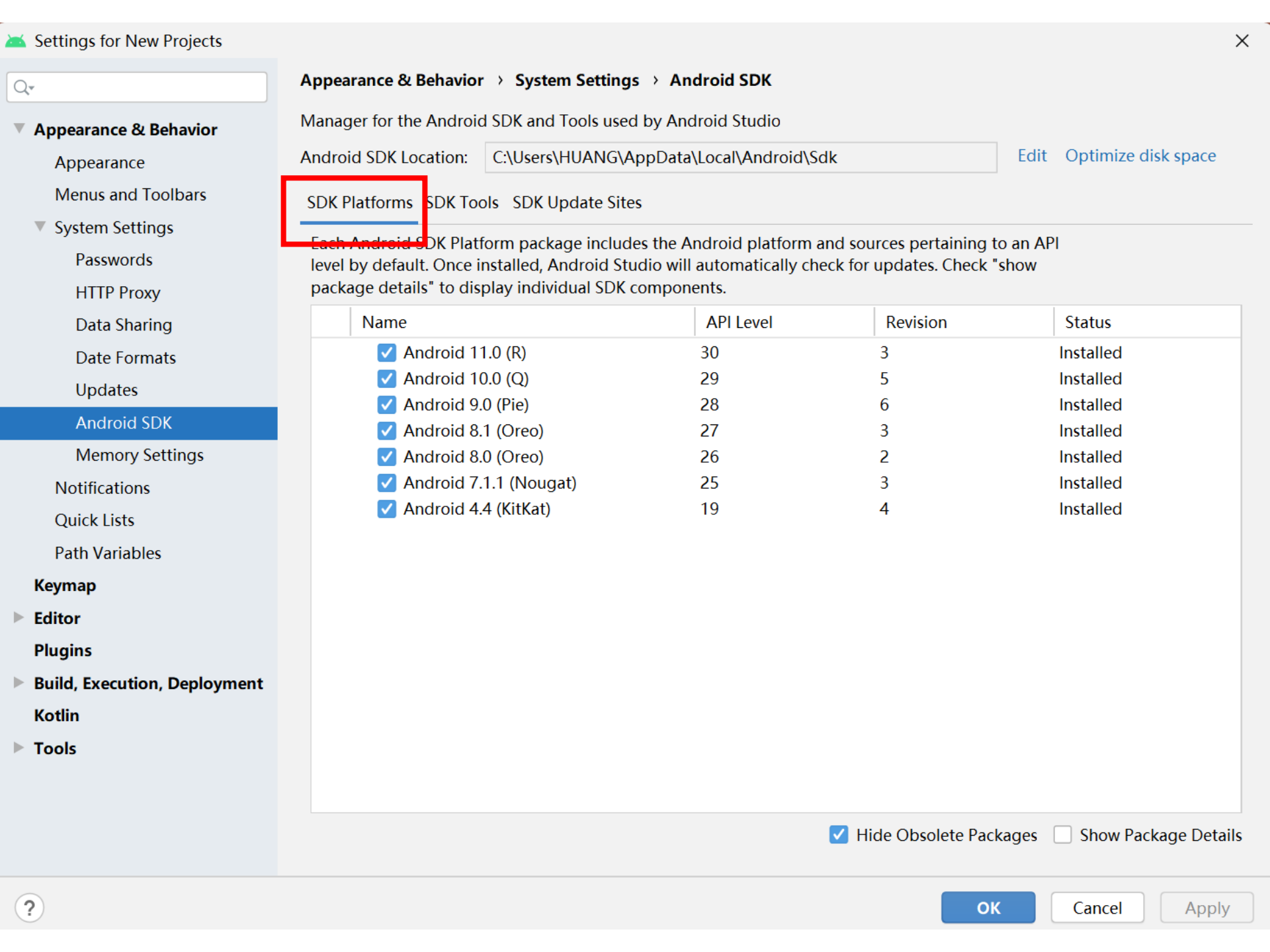
```
public void onRequestPermissionsResult(int requestCode,
                                       String permissions[], int[] grantResults) {
    switch (requestCode) {
        case 1:
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // 实际需要该权限才能执行的操作
            }
            break;
        default:
            break;
    }
}
```

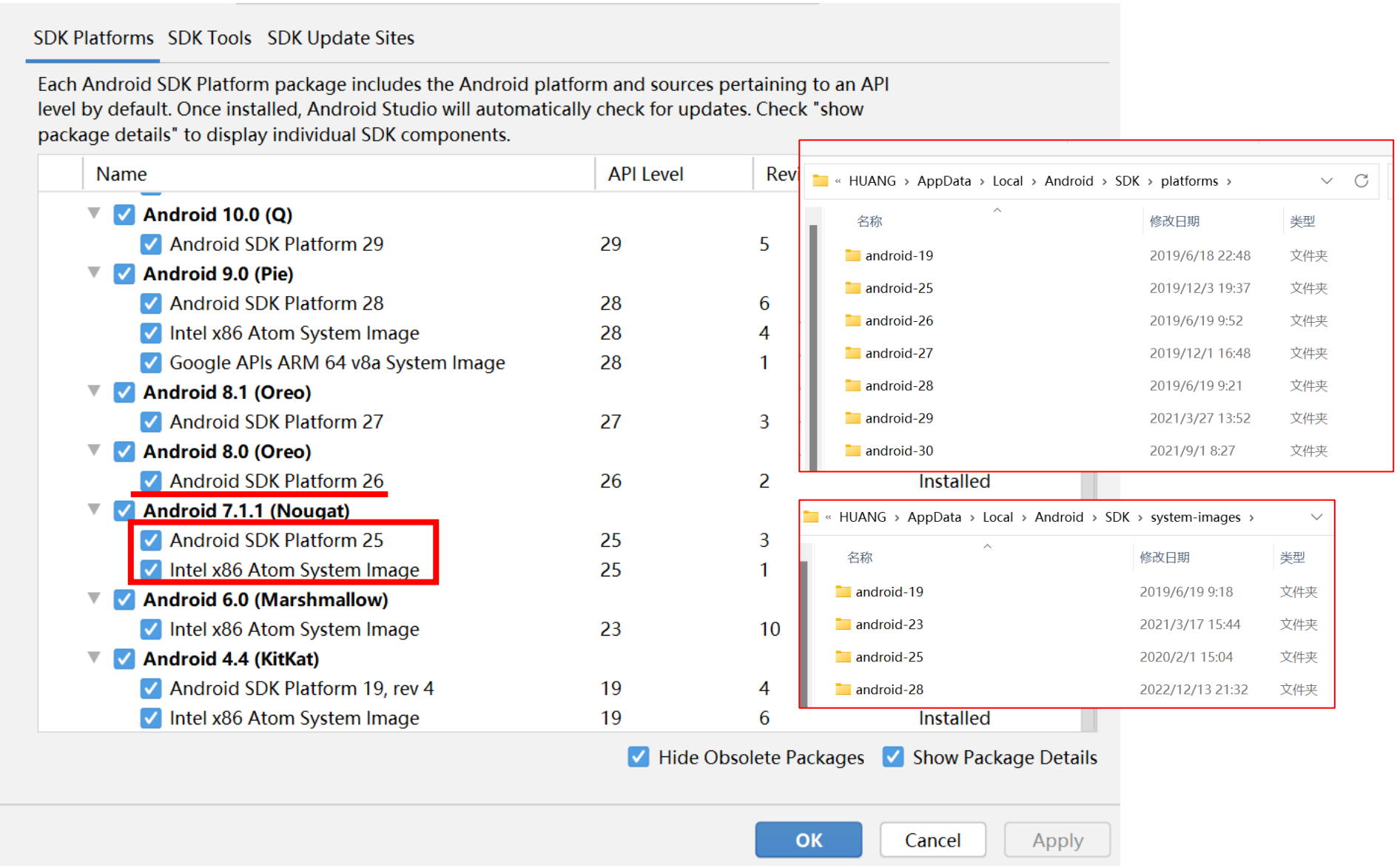


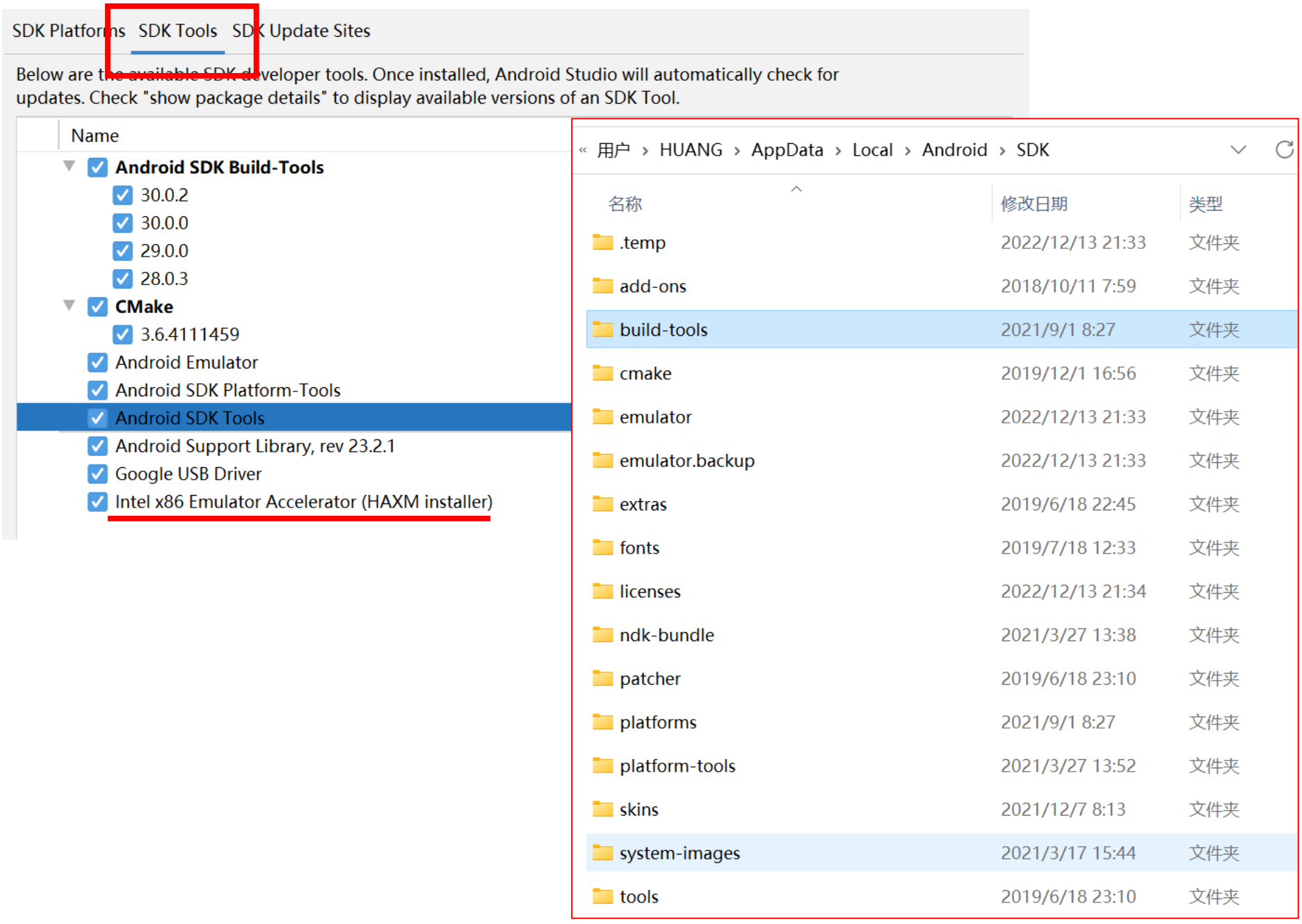
Android Studio用法简介

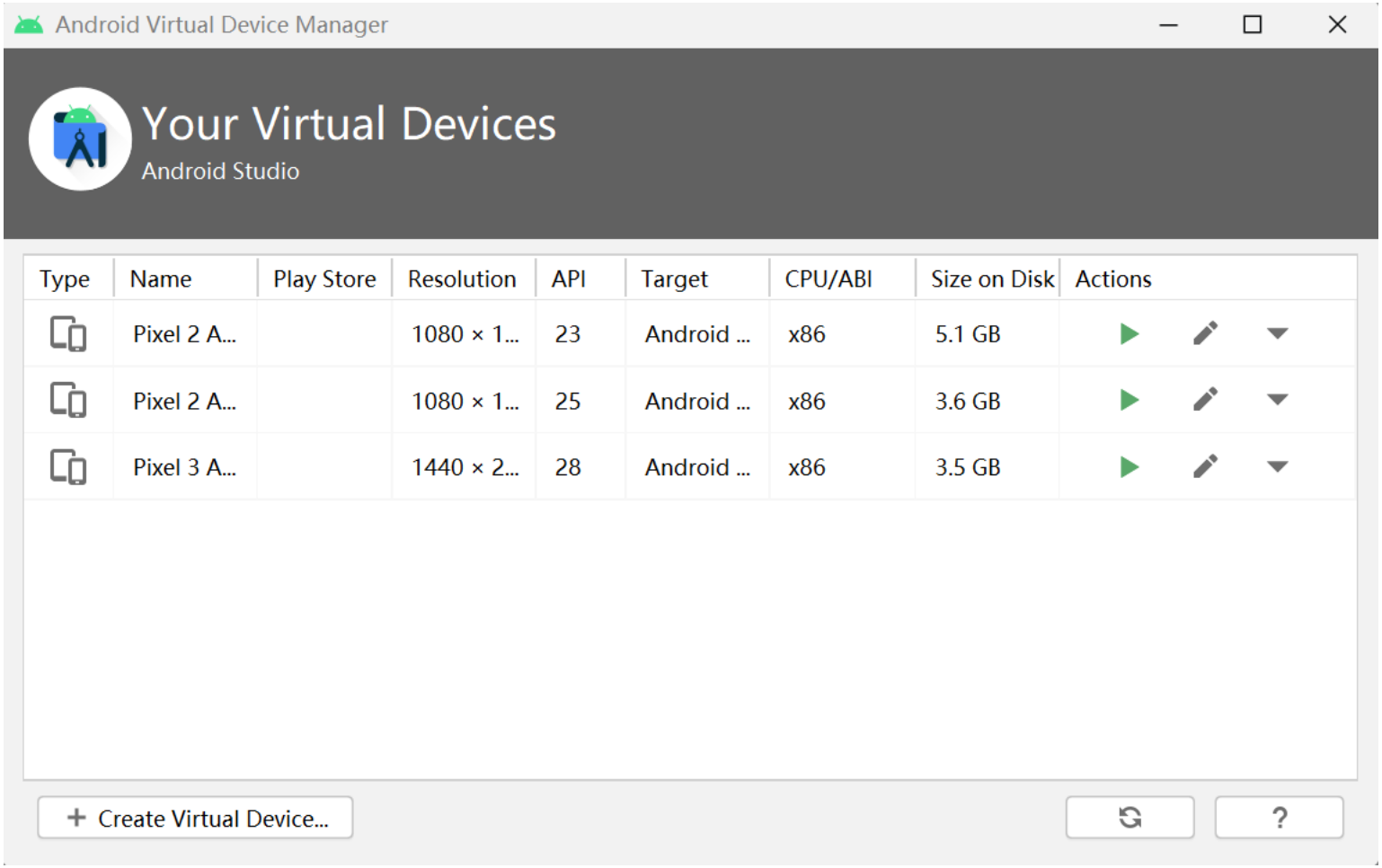
- Android Studio是Google官方定制的Android应用开发、调试IDE
- 可从官网直接下来（建议下载带SDK的版本）
- 以下是较旧的版本下载链接
- Windows 64-bit：https://dl.google.com/dl/android/studio/install/3.5.2.0/android-studio-ide-191.5977832-windows.exe
- Mac 64-bit ： https://dl.google.com/dl/android/studio/install/3.5.2.0/android-studio-ide-191.5977832-mac.dmg
- Linux 64-bit： https://dl.google.com/dl/android/studio/ide-zips/3.5.2.0/android-studio-ide-191.5977832-linux.tar.gz


Welcome to Android Studio
Victim
D:\Workspace\In...signments\Victim
Attacker
D:\Workspace\In...gnments\Attacker
DevActivityOne
D:\Workspace\DevExamples
DevReceiver
D:\Workspace\DevReceiver
AMA2
D:\Workspace\J...idSecDemo\AMA2
My Application
D:\Workspace\MyApplication2
My Application
D:\Workspace\MyApplication5
AMA3
D:\Workspace\J...idSecDemo\AMA3
My Application
D:\Workspace\MyApplication3
My Application
D:\Workspace\MyApplication4
Android Studio
Version 4.1.3
Create New Project
Open an Existing Project
Get from Version Control
Profile or Debug APK
Import Project (Gradle, Eclipse ADT, etc.)
Import an Android Code Sample
Configure
Get Help
SDK Manager
AVD Manager
Settings
Plugins
Default Project Structure...
Run Configuration Templates for New Projects...
Import Settings
Export Settings
Settings Repository...
Restore Default Settings...
Compress Logs and Show in File Manager
Edit Custom Properties...
Edit Custom VM Options...
Check for Updates
My Application
D:\Workspace\MyApplication3
My Application
D:\Workspace\MyApplication4


Settings for New Projects
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Data Sharing
Date Formats
Updates
Android SDK
Memory Settings
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Build, Execution, Deployment
Kotlin
Tools
Appearance & Behavior › System Settings › Android SDK
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location:
C:\Users\HUANG\AppData\Local\Android\Sdk
Edit
Optimize disk space
SDK Platforms
SDK Tools
SDK Update Sites
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.
Name
API Level
Revision
Status
Android 11.0 (R) 30 3 Installed
Android 10.0 (Q) 29 5 Installed
Android 9.0 (Pie) 28 6 Installed
Android 8.1 (Oreo) 27 3 Installed
Android 8.0 (Oreo) 26 2 Installed
Android 7.1.1 (Nougat) 25 3 Installed
Android 4.4 (KitKat) 19 4 Installed
Hide Obsolete Packages
Show Package Details
OK
Cancel
Apply


SDK Platforms
SDK Tools
SDK Update Sites
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.
Name
API Level
Rev
Android 10.0 (Q)
Android SDK Platform 29 29 5
Android 9.0 (Pie)
Android SDK Platform 28 28 6
Intel x86 Atom System Image 28 4
Google APIs ARM 64 v8a System Image 28 1
Android 8.1 (Oreo)
Android SDK Platform 27 27 3
Android 8.0 (Oreo)
Android SDK Platform 26 26 2 Installed
Android 7.1.1 (Nougat)
Android SDK Platform 25 25 3
Intel x86 Atom System Image 25 1
Android 6.0 (Marshmallow)
Intel x86 Atom System Image 23 10
Android 4.4 (KitKat)
Android SDK Platform 19, rev 4 19 4
Intel x86 Atom System Image 19 6 Installed
Hide Obsolete Packages
Show Package Details
OK
Cancel
Apply
« HUANG › AppData › Local › Android › SDK › platforms ›
名称 修改日期 类型
android-19 2019/6/18 22:48 文件夹
android-25 2019/12/3 19:37 文件夹
android-26 2019/6/19 9:52 文件夹
android-27 2019/12/1 16:48 文件夹
android-28 2019/6/19 9:21 文件夹
android-29 2021/3/27 13:52 文件夹
android-30 2021/9/1 8:27 文件夹
« HUANG › AppData › Local › Android › SDK › system-images ›
名称 修改日期 类型
android-19 2019/6/19 9:18 文件夹
android-23 2021/3/17 15:44 文件夹
android-25 2020/2/1 15:04 文件夹
android-28 2022/12/13 21:32 文件夹


SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.
Name
Android SDK Build-Tools
30.0.2
30.0.0
29.0.0
28.0.3
CMake
3.6.4111459
Android Emulator
Android SDK Platform-Tools
Android SDK Tools
Android Support Library, rev 23.2.1
Google USB Driver
Intel x86 Emulator Accelerator (HAXM installer)
« 用户 › HUANG › AppData › Local › Android › SDK
名称
修改日期
类型
.temp 2022/12/13 21:33 文件夹
add-ons 2018/10/11 7:59 文件夹
build-tools 2021/9/1 8:27 文件夹
cmake 2019/12/1 16:56 文件夹
emulator 2022/12/13 21:33 文件夹
emulator.backup 2022/12/13 21:33 文件夹
extras 2019/6/18 22:45 文件夹
fonts 2019/7/18 12:33 文件夹
licenses 2022/12/13 21:34 文件夹
ndk-bundle 2021/3/27 13:38 文件夹
patcher 2019/6/18 23:10 文件夹
platforms 2021/9/1 8:27 文件夹
platform-tools 2021/3/27 13:52 文件夹
skins 2021/12/7 8:13 文件夹
system-images 2021/3/17 15:44 文件夹
tools 2019/6/18 23:10 文件夹


Android Virtual Device Manager
Your Virtual Devices
Android Studio
Type
Name
Play Store
Resolution
API
Target
CPU/ABI
Size on Disk
Actions
Pixel 2 A...
1080 × 1...
23
Android ...
x86
5.1 GB
Pixel 2 A...
1080 × 1...
25
Android ...
x86
3.6 GB
Pixel 3 A...
1440 × 2...
28
Android ...
x86
3.5 GB
Create Virtual Device...

Virtual Device Configuration

System Image

Select a system image

Recommended | x86 Images | Other Images

Release Name	API Level	ABI	Target
R Download	30	x86	Android 11.0 (Google Play)
Q Download	29	x86	Android 10.0 (Google Play)
Pie Download	28	x86	Android 9.0 (Google Play)
Oreo Download	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)

R

API Level
30

Android
11.0

Google Inc.

System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

A system image must be selected to continue.

Previous | Next | Cancel | Finish

Android应用开发示例

- （现场演示通过Android Studio创建一个简单应用程序）
- 新建
- 添加GUI组件
- 编写代码
- 运行

adb使用演示

- （现场演示adb命令的部分功能）
- shell
- root
- pull
- uninstall
- install

ApkTool

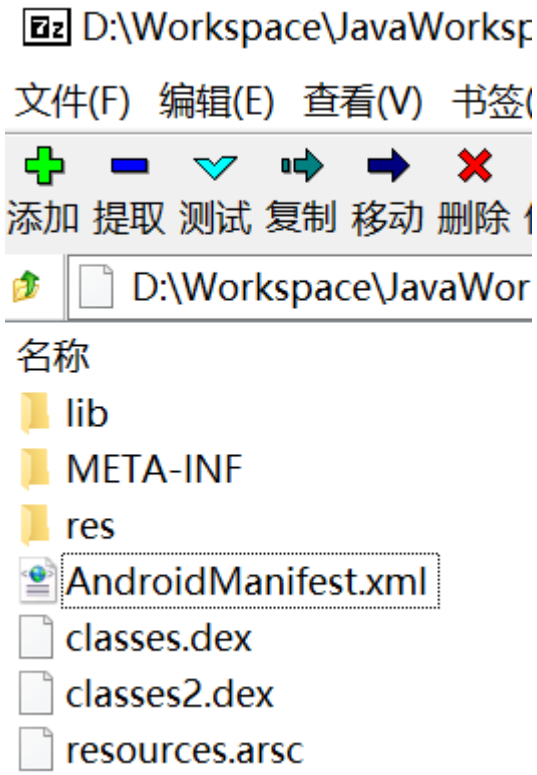
- 常用的APK反编译工具，可以将编译后的XML文件（二进制）还原为人可读的形式（几乎与开发时的状态一致），将编译后的DEX字节码反编译为一种smali格式的代码
- 对无法处理的.so文件等保持原样
- 攻击者修改XML或smali代码之后，ApkTool能够将所有文件重编译为APK包
- 官网：
  - https://ibotpeaches.github.io/Apktool
  - JAR下载地址：
    - https://bitbucket.org/iBotPeaches/apktool/downloads/
  - Wrapper Script下载地址：
    - https://raw.githubusercontent.com/iBotPeaches/Apktool/master/scripts/windows/apktool.bat
    - https://raw.githubusercontent.com/iBotPeaches/Apktool/master/scripts/linux/apktool
    - https://raw.githubusercontent.com/iBotPeaches/Apktool/master/scripts/osx/apktool
    - *Wrapper scripts are not needed, but helpful so you don't have to type* java -jar apktool.jar *over and over.*

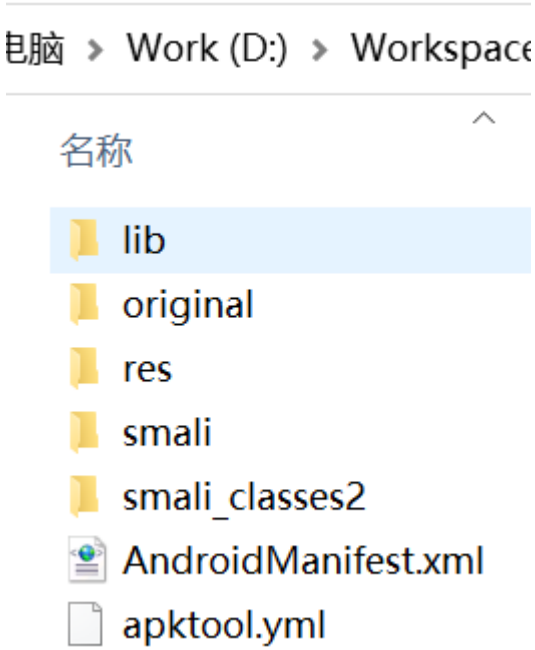
ApkTool反编译

- 执行命令：apktool d app-debug.apk
  - apktool的命令行选项请自行查阅相关资料

```
I: Using Apktool 2.3.4 on app-debug.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\HUANG\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Baksmaling classes2.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

- 当前目录下生成app-debug目录（与APK文件名一致），其中包含反编译后的smali代码、xml文件（AndroidManifest、layout等），以及来自于原APK包中未做修改的文件（原始的二进制格式的AndroidManifest以及META-INF目录）
- 下方左图是APK包结构；右图是反编译后的目录结构，其中smali与smali_classes2分别对应原APK中的classes.dex与classes2.dex





练习

- 学习Android应用开发，尝试开发一个简单的、有用的小应用，如闹钟、便签等
  - 学习多种Android组件的开发、理解各种组件的运行机制
  - 理解Android应用各组成部分的用途，尤其是AndroidManifest.xml的作用
  - 学习本次课程中未曾涉及的部分，如：通过WebView的方式，在用户界面中展示渲染网页内容并与主程序交互
  - 注意：对于Android安全分析而言，布局文件（layout XML）如何构建、用户界面是什么样，通常不重要，我们面对的应该是较为成熟的、具有良好用户界面的应用

- Android 开发者网站： https://developer.android.google.cn/reference/packages
- Java API 文档 IO 部分（标准 API 与 Android 所采用完全一样）：
  - https://docs.oracle.com/javase/8/docs/api/java/io/package-summary.html