# 第一章 android 系统介绍

## 1.1 Android简介

**Android**（读音：英：['ændrɔɪd]，美：[ˈænˌdrɔɪd]），常见的非官方中文名称为**安卓**，是一个基于[Linux内核](https://zh.wikipedia.org/wiki/Linux%E6%A0%B8%E5%BF%83)的[开放源代码](https://zh.wikipedia.org/wiki/%E9%96%8B%E6%94%BE%E5%8E%9F%E5%A7%8B%E7%A2%BC)[移动操作系统](https://zh.wikipedia.org/wiki/%E8%A1%8C%E5%8B%95%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)，由[Google](https://zh.wikipedia.org/wiki/Google)成立的[Open Handset Alliance](https://zh.wikipedia.org/wiki/%E9%96%8B%E6%94%BE%E6%89%8B%E6%A9%9F%E8%81%AF%E7%9B%9F" \o "開放手機聯盟)（OHA，开放手持设备联盟）持续领导与开发，主要设计用于触屏移动设备如[智能手机](https://zh.wikipedia.org/wiki/%E6%99%BA%E6%85%A7%E5%9E%8B%E6%89%8B%E6%A9%9F" \o "智能手机)和[平板电脑](https://zh.wikipedia.org/wiki/%E5%B9%B3%E6%9D%BF%E9%9B%BB%E8%85%A6)与其他便携式设备。

Android最初由[安迪·鲁宾](https://zh.wikipedia.org/wiki/%E5%AE%89%E8%BF%AA%C2%B7%E9%B2%81%E5%AE%BE" \o "安迪·鲁宾)等人开发制作[[7]](https://zh.wikipedia.org/wiki/Android#cite_note-7)，最初开发这个系统的目的是创建一个数码相机的先进操作系统；但是后来发现市场需求不够大，加上智能手机市场快速成长，于是Android成为一款面向智能手机的操作系统。于2005年7月11日被[美国](https://zh.wikipedia.org/wiki/%E7%BE%8E%E5%9C%8B" \o "美国)科技企业Google收购[[8]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-gba-8) [[9]](https://zh.wikipedia.org/wiki/Android#cite_note-hh-9)。2007年11月，Google与84家硬件制造商、软件开发商及电信营运商成立[开放手持设备联盟](https://zh.wikipedia.org/wiki/%E9%96%8B%E6%94%BE%E6%89%8B%E6%A9%9F%E8%81%AF%E7%9B%9F" \o "开放手持设备联盟)来共同研发改良Android，随后，Google以[Apache免费开放源代码许可证](https://zh.wikipedia.org/wiki/Apache%E8%A8%B1%E5%8F%AF%E8%AD%89" \o "Apache许可证)的授权方式，发布了Android的源代码[[10]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-AndroidOverview-10)，开放源代码加速了Android普及，让生产商推出搭载Android的智能手机[[11]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-OHAhome-11)[[12]](https://zh.wikipedia.org/wiki/Android#cite_note-12)[[13]](https://zh.wikipedia.org/wiki/Android#cite_note-AndroidAnnouncement-13)[[14]](https://zh.wikipedia.org/wiki/Android#cite_note-14)，Android后来更逐渐拓展到平板电脑及其他领域上[[15]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-15)。

2010年末数据显示，仅正式推出两年的Android操作系统在市场占有率上已经超越称霸逾十年的[诺基亚](https://zh.wikipedia.org/wiki/%E8%AB%BE%E5%9F%BA%E4%BA%9E" \o "诺基亚)[Symbian](https://zh.wikipedia.org/wiki/Symbian)系统[[16]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-canalysQ42010-16)[[17]](https://zh.wikipedia.org/wiki/Android#cite_note-17)，成为全球第一大智能手机操作系统[[18]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-18)。

在2014年[Google I/O](https://zh.wikipedia.org/wiki/Google_I/O" \o "Google I/O)开发者大会上Google宣布过去30天里有10亿台活跃的安卓设备，相较于2013年6月则是5.38亿[[19]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-19)。

2017年3月，Android全球网络流量和设备超越[Microsoft Windows](https://zh.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows)，正式成为全球第一大操作系统[[20]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-20)。

2017年8月，Android O发布。

## 1.2 Android 系统设计

Linux内核

Android操作系统的核心属于[Linux内核](https://zh.wikipedia.org/wiki/Linux%E6%A0%B8%E5%BF%83" \o "Linux内核)的一个分支，具有典型的Linux调度和功能，[[44]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-44)除此之外，Google为了能让Linux在移动设备上良好的运行，对其进行了修改和扩充。Android去除了Linux中的本地[X Window System](https://zh.wikipedia.org/wiki/X_Window%E7%B3%BB%E7%B5%B1" \o "X窗口系统)，也不支持标准的[GNU库](https://zh.wikipedia.org/w/index.php?title=Gnulib&action=edit&redlink=1" \o "Gnulib（页面不存在）)，这使得Linux平台上的应用程序移植到Android平台上变得困难[[45]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-45)。2008年，Patrick Brady于[Google I/O](https://zh.wikipedia.org/wiki/Google_I/O" \o "Google I/O)演讲“Anatomy & Physiology of an Android”，并提出的Android HAL架构图。[HAL](https://zh.wikipedia.org/wiki/%E7%A1%AC%E9%AB%94%E6%8A%BD%E8%B1%A1%E5%B1%A4" \o "硬件抽象层)以\*.so档的形式存在，可以把Android framework与Linux kernel隔开，这种中介层的方式使得Android能在移动设备上获得更高的运行效率。这种独特的系统结构被Linux内核开发者[Greg Kroah-Hartman](https://zh.wikipedia.org/wiki/Greg_Kroah-Hartman" \o "Greg Kroah-Hartman)和其他核心维护者称赞。Google还在Android的核心中加入了自己开发制作的一个名为“wakelocks”的移动设备电源管理功能，该功能用于管理移动设备的电池性能，但是该功能并没有被加入到Linux内核的主线开放和维护中，因为Linux内核维护者认为Google没有向他们展示这个功能的意图和代码。

2010年2月3日，由于Google在Android核心开发方面和Linux社区方面开发的不同步，Linux内核开发者Greg Kroah-Hartman将Android的驱动程序从Linux内核“[状态树](https://zh.wikipedia.org/w/index.php?title=%E7%8A%B6%E6%80%81%E6%A0%91&action=edit&redlink=1)”（“staging tree”）上除去。[[46]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-46)2010年4月，Google宣布将派遣2名开发人员加入Linux内核社区，以便重返Linux内核。2010年9月，Linux内核开发者Rafael J. Wysocki添加了一个修复程序，使得Android的“wakelocks”可以轻松地与主线Linux内核合并。2011年，[Linus Torvalds](https://zh.wikipedia.org/wiki/Linus_Torvalds" \o "Linus Torvalds)说：“Android的核心和Linux的核心将最终回归到一起，但可能不会是4-5年。”在Linux 3.3中大部分代码的集成完成。[[47]](https://zh.wikipedia.org/wiki/Android" \l "cite_note-47)

## 1.3 Android特点

目前的Android操作系统具有以下几个功能：

**显示布局**

Android操作系统支持更大的分辨率，VGA，2D显示，3D显示都给予OpenGL ES 3.0标准规格（4.3版本开始支持OpenGL ES 3.0），并且支持传统的智能手机。

**数据存储**

Android操作系统内置[SQLite](https://zh.wikipedia.org/wiki/SQLite" \o "SQLite)小型关联式资料库管理系统来负责存储数据。

**网络**

Android操作系统支持所有的网络制式，包括[GSM](https://zh.wikipedia.org/wiki/GSM" \o "GSM)/[EDGE](https://zh.wikipedia.org/wiki/EDGE)、[IDEN](https://zh.wikipedia.org/wiki/IDEN" \o "IDEN)、[CDMA](https://zh.wikipedia.org/wiki/CDMA" \o "CDMA)、[TD-SCDMA](https://zh.wikipedia.org/wiki/TD-SCDMA" \o "TD-SCDMA) 、[EV-DO](https://zh.wikipedia.org/wiki/EV-DO" \o "EV-DO)、[UMTS](https://zh.wikipedia.org/wiki/UMTS" \o "UMTS)、[Bluetooth](https://zh.wikipedia.org/wiki/Bluetooth" \o "Bluetooth)、[Wi-Fi](https://zh.wikipedia.org/wiki/Wi-Fi" \o "Wi-Fi)、[LTE](https://zh.wikipedia.org/wiki/LTE" \o "LTE)、[NFC](https://zh.wikipedia.org/wiki/%E8%BF%91%E5%A0%B4%E9%80%9A%E8%A8%8A" \o "近场通信)和[WiMAX](https://zh.wikipedia.org/wiki/WiMAX" \o "WiMAX)。

**信息**

作为原设计给智能手机使用的操作系统，Android操作系统原生支持短信和邮件，并且支持所有的云信息和服务器信息。

**语言**

Android操作系统支持多语言。

**浏览器**

Android操作系统中内置的网页浏览器基于[WebKit](https://zh.wikipedia.org/wiki/WebKit" \o "WebKit)核心，并且采用了[Chrome](https://zh.wikipedia.org/wiki/Google_Chrome" \o "Google Chrome) [V8](https://zh.wikipedia.org/wiki/V8_(JavaScript%E5%BC%95%E6%93%8E))引擎。在[Android 4.0](https://zh.wikipedia.org/wiki/Android_4.0" \o "Android 4.0)内置的浏览器测试中，[HTML5](https://zh.wikipedia.org/wiki/HTML5" \o "HTML5)和[Acid3](https://zh.wikipedia.org/wiki/Acid3" \o "Acid3)故障处理中均获得了满分，并且于2.2版至4.0版之前能原生支持[Flash](https://zh.wikipedia.org/wiki/Adobe_Flash_Player" \o "Adobe Flash Player)，4.4版本后去除对Flash的支持。

**支持**[**Java**](https://zh.wikipedia.org/wiki/Java)

虽然Android操作系统中的应用程序大部分都是由[Java](https://zh.wikipedia.org/wiki/Java" \o "Java)编写的，但是Android却是以转换为Dalvik executables的文件在[Dalvik虚拟机](https://zh.wikipedia.org/wiki/Dalvik%E8%99%9A%E6%8B%9F%E6%9C%BA" \o "Dalvik虚拟机)上运行的。由于Android中并不自带[Java虚拟机](https://zh.wikipedia.org/wiki/Java%E8%99%9B%E6%93%AC%E6%A9%9F" \o "Java虚拟机)，因此无法直接运行Java程序。不过Android平台上提供了多个Java虚拟机供用户下载使用，安装了Java虚拟机的Android系统可以运行[Java\_ME](https://zh.wikipedia.org/wiki/J2ME" \o "J2ME)的程序。5.0版（Lolipop）开始以[Android Runtime](https://zh.wikipedia.org/wiki/Android_Runtime" \o "Android Runtime)（ART）取代Dalvik虚拟机。

**媒体支持**

Android操作系统本身支持以下格式的音频/视频/图片媒体：WebM、H.263, H.264（in 3GP or MP4 container）、MPEG-4 SP、AMR, AMR-WB（in 3GP container）、AAC, HE-AAC（in MP4 or 3GP container）、MP3、MIDI、Ogg Vorbis、FLAC、WAV、JPEG、PNG、GIF、BMP。如果用户需要播放更多格式的媒体，可以安装其他第三方应用程序。

**流媒体支持**

Android操作系统支持RTP/RTSP（3GPP PSS, ISMA）的流媒体以及（HTML5 <video>）的流媒体，同时还支持Adobe的Flash，在安装了[RealPlayer](https://zh.wikipedia.org/wiki/RealPlayer" \o "RealPlayer)之后，还支持苹果公司的流媒体。

**硬件支持**

Android操作系统支持识别并且使用视频/照片摄像镜头，多点电容/电阻触屏，GPS，加速计，陀螺仪，气压计，磁力仪(高斯计)，键盘，鼠标，USB Disk，专用的游戏控制器，体感控制器，游戏手把，蓝牙设备，无线设备，感应和压力感测器，温度计，加速2D位位块传输（硬件方向，缩放，像素格式转换）和3D图形加速。

**多点触控**

Android支持本地的多点触控，在最初的[HTC Hero](https://zh.wikipedia.org/wiki/HTC_Hero" \o "HTC Hero)智能手机上即有这个功能。该功能是内核级别（为了避免对苹果公司的触屏技术造成侵权）。

**蓝牙**

Android支持A2DP，AVRCP，发送文件（OPP），访问电话簿（PBAP），语音拨号和方送智能手机之间的联系。同时支持键盘，鼠标和摇杆（HID）。

**多任务处理**

Android操作系统支持本地的多任务处理。

**语音功能**

除了支持普通的电话通话之外，Android操作系统从最初的版本开始就支持使用语音操作来使用Google进行网页搜索等功能。而从Android 2.2开始，语音功能还可以用来输入文字、语音导航等功能。

**无线共享功能**

Android操作系统支持用户使用本机充当“无线路由器”，并且将本机的网络共享给其他智能手机，其他机器只需要通过[WiFi](https://zh.wikipedia.org/wiki/WiFi" \o "WiFi)查找到共享的无线热点，就可以上网。而在Android 2.2之前的操作系统则需要通过第三方应用或者其他定制版系统来实现这个功能。

**截图功能**

从Android 4.0开始，Android操作系统便支持截图功能，该功能允许用户直接抓取智能手机显示屏上的任何画面，用户还可以通过编辑功能对截图进行处理，还可以通过蓝牙/E-mail/共享等方式发送给其他用户或者上传到网络上，也可以拷贝到电脑中。

## 1.4 Android 系统架构

### 1.4.1 应用程序

Android系统是基于Linux内核开发，使用[Java](https://zh.wikipedia.org/wiki/Java" \o "Java)作编程语言，使[界面](https://zh.wikipedia.org/wiki/%E4%BB%8B%E9%9D%A2_(%E8%B3%87%E8%A8%8A%E7%A7%91%E6%8A%80)" \o "界面 (信息技术))到功能，都有层出不穷的变化，其中Activity等同于J2ME的MIDlet，一个Activity类别负责创建视窗，一个活动中的Activity就是在foreground（前景）模式，背景执行的程序叫做Service。两者之间透过由ServiceConnection和AIDL连结，达到复数程序同时执行的效果。如果执行中的Activity全部画面被其他Activity取代时，该Activity便被停止，甚至被系统清除。

View等同于J2ME的Displayable，程序人员可以透过View类别与“XML layout”档将UI放置在视窗上，并可以利用View打造出所谓的Widgets，其实Widget只是View的一种，所以可以使用xml来设计layout。至于ViewGroup是各种layout的基础抽象类别，ViewGroup之内还可以有ViewGroup。View的构造函数不需要在Activity中调用，但是Displayable的是必须的，在Activity中，要通过findViewById()来从XML中获取View，Android的View类的显示很大程度上是从XML中读取的。View与事件息息相关，两者之间透过Listener结合在一起，每一个View都可以注册event listener，例如：当View要处理用户触碰的事件时，就要向Android框架注册View.OnClickListener。另外还有Image等同于J2ME的BitMap。

### 1.4.2 中介软件

操作系统与应用程序的沟通桥梁，并用分为两层：[函数](https://zh.wikipedia.org/wiki/%E5%87%BD%E5%BC%8F)层和[虚拟机器](https://zh.wikipedia.org/wiki/%E8%99%9A%E6%8B%9F%E6%9C%BA%E5%99%A8)。 [Bionic](https://zh.wikipedia.org/wiki/Bionic_(%E8%BB%9F%E9%AB%94))是Android改良libc的版本。Android包含了Chrome浏览器引擎。Surface flinger是就2D或3D的内容显示到萤幕上。Android使用工具链为Google自制的Bionic Libc。

Android采用OpenCORE作为基础多媒体框架。OpenCORE可分7大块：PVPlayer、PVAuthor、Codec、PacketVideo Multimedia Framework（PVMF）、Operating System Compatibility Library（OSCL）、Common、OpenMAX。

Android使用[Skia](https://zh.wikipedia.org/wiki/Skia_Graphics_Library" \o "Skia Graphics Library)为核心图形引擎，搭配OpenGL/ES。Skia与Linux Cairo功能相当，但相较于Linux Cairo，Skia功能还只是阳春型的。2005年Skia公司被Google收购，2007年初，Skia GL源码被公开，目前Skia也是Google Chrome的图形引擎。

Android的多媒体资料库采用[SQLite](https://zh.wikipedia.org/wiki/SQLite" \o "SQLite)资料库系统。资料库又分为共用资料库及私用资料库。用户可透过ContentProvider类别获取共用资料库。

Android的中间层多以Java实现，4.4版之前使用特殊的[Dalvik虚拟机器](https://zh.wikipedia.org/wiki/Dalvik%E8%99%9A%E6%8B%9F%E6%9C%BA" \o "Dalvik虚拟机)。Dalvik虚拟机器是一种“暂存器型态”的Java虚拟机器，变数皆存放于[暂存器](https://zh.wikipedia.org/wiki/%E6%9A%AB%E5%AD%98%E5%99%A8" \o "寄存器)中，虚拟机器的指令相对减少。5.0版起改用[Android Runtime](https://zh.wikipedia.org/wiki/Android_Runtime" \o "Android Runtime)（ART）。

Dalvik虚拟机器可以有多个实例，每个Android应用程序都用一个自属的Dalvik虚拟机器来执行，让系统在执行程序时可达到最优化。Dalvik虚拟机器并非执行Java字节码，而是执行一种称为.dex格式的档案。

### 1.4.2 硬件抽像层（HAL）

Android的硬件抽像层是能以封闭源码形式提供硬件驱动模块。HAL的目的是为了把Android framework与Linux kernel隔开，让Android不至过度依赖Linux kernel，以达成“内核独立”（kernel independent）的概念，也让Android framework的开发能在不考量驱动程序实现的前提下进行发展，以达到垄断GPU市场的目的。

HAL stub是一种代理人的概念，stub是以\*.so档的形式存在。Stub向HAL“提供”操作函数，并由Android runtime向HAL获取stub的操作，再[回调](https://zh.wikipedia.org/wiki/%E5%9B%9E%E8%B0%83%E5%87%BD%E6%95%B0" \o "回调函数)这些操作函数。HAL里包含了许多的stub（代理人）。Runtime只要说明“类型”，即module ID，就可以获取操作函数。

## 1.5 Android硬件支持

由于Android操作系统的开放性和可移植性，它可以被用在大部分电子产品上。包括：智能手机、上网本、平板电脑、个人电脑、笔记本电脑、电视、机顶盒、电子书阅读器、MP3播放器、MP4播放器、掌上游戏机、家用游戏机、电子手表、电子收音机、耳机、汽车设备、导航仪、DVD机以及其他设备。

Android操作系统大多搭载在使用了[ARM](https://zh.wikipedia.org/wiki/ARM" \o "ARM)架构的硬件设备上。但是同样也有支持X86架构的Android操作系统，比如Google的Google TV就是使用一个特别定制的X86架构版本的Android操作系统。

华硕曾推出一系列使用x86 cpu的android手机，但兼容性较差、小问题较多；而华硕的android手机中，采用arm指令集者就比较稳定。[[来源请求]](https://zh.wikipedia.org/wiki/Wikipedia:%E5%88%97%E6%98%8E%E6%9D%A5%E6%BA%90)

同样，苹果公司的iOS设备，比如iPhone、iPod Touch以及iPad产品(iOS 4以前，需越狱) 都可以安装Android操作系统，并且可以通过双系统引导工具OpeniBoot或者iDroid来运行Android操作系统。微软的Windows Mobile、Windows Phone产品也一样可以。另外Android亦已成功移植到搭载[WebOS](https://zh.wikipedia.org/wiki/WebOS" \o "WebOS)系统HP [TouchPad](https://zh.wikipedia.org/wiki/TouchPad" \o "TouchPad)以及搭载[Meego](https://zh.wikipedia.org/wiki/Meego" \o "Meego)系统的[Nokia N9](https://zh.wikipedia.org/wiki/Nokia_N9" \o "Nokia N9)等设备。

世界上第一部真正意义上使用Android操作系统的设备是2008年10月22日发布的[HTC Dream](https://zh.wikipedia.org/wiki/HTC_Dream" \o "HTC Dream)。

Android可以像用电脑上网一样，使用任何服务[[49]](https://zh.wikipedia.org/wiki/Android#cite_note-49)。Android对手主要有[苹果公司](https://zh.wikipedia.org/wiki/%E8%8B%B9%E6%9E%9C%E5%85%AC%E5%8F%B8" \o "苹果公司)的[iOS操作系统](https://zh.wikipedia.org/wiki/IOS)，[Microsoft](https://zh.wikipedia.org/wiki/%E5%BE%AE%E8%BD%AF)的[Windows Phone](https://zh.wikipedia.org/wiki/Windows_Phone" \o "Windows Phone)操作系统及[RIM](https://zh.wikipedia.org/wiki/Research_In_Motion" \o "Research In Motion)使用的BlackBerry OS系统。

## 1.6 下载安装android studio

Android studio 下载网址： <https://developer.android.com/studio/index.html>

如需在 Windows 系统中安装 Android Studio，请执行以下操作：

1: 启动您下载的 .exe 文件。

2: 根据安装向导的指示安装 Android Studio 和所有所需的 SDK 工具。

在有些 Windows 系统中，启动器脚本无法找到 JDK 的安装位置。如果您遇到此问题，您需要设置指示正确位置的环境变量。

选择**“Start”菜单 > Computer > System Properties > Advanced System Properties**。然后打开**“Advanced”选项卡 > Environment Variables**，添加指向您的 JDK 文件夹位置（例如 C:\Program Files\Java\jdk1.8.0\_77）的新系统变量 JAVA\_HOME。

# 第二章 Android 基础知识

Android 应用采用 Java 编程语言编写。Android SDK 工具将您的代码 — 连同任何数据和资源文件 — 编译到一个 APK：*Android 软件包*，即带有 .apk 后缀的存档文件中。一个 APK 文件包含 Android 应用的所有内容，它是基于 Android 系统的设备用来安装应用的文件。

安装到设备后，每个 Android 应用都运行在自己的安全沙箱内：

Android 操作系统是一种多用户 Linux 系统，其中的每个应用都是一个不同的用户；

默认情况下，系统会为每个应用分配一个唯一的 Linux 用户 ID（该 ID 仅由系统使用，应用并不知晓）。系统为应用中的所有文件设置权限，使得只有分配给该应用的用户 ID 才能访问这些文件；

每个进程都具有自己的虚拟机 (VM)，因此应用代码是在与其他应用隔离的环境中运行；

默认情况下，每个应用都在其自己的 Linux 进程内运行。Android 会在需要执行任何应用组件时启动该进程，然后在不再需要该进程或系统必须为其他应用恢复内存时关闭该进程。

Android 系统可以通过这种方式实现最小权限原则。也就是说，默认情况下，每个应用都只能访问执行其工作所需的组件，而不能访问其他组件。 这样便营造出一个非常安全的环境，在这个环境中，应用无法访问系统中其未获得权限的部分。

不过，应用仍然可以通过一些途径与其他应用共享数据以及访问系统服务：

可以安排两个应用共享同一 Linux 用户 ID，在这种情况下，它们能够相互访问彼此的文件。 为了节省系统资源，可以安排具有相同用户 ID 的应用在同一 Linux 进程中运行，并共享同一 VM（应用还必须使用相同的证书签署）。

应用可以请求访问设备数据（如用户的联系人、短信、可装载存储装置 [SD 卡]、相机、蓝牙等）的权限。 用户必须明确授予这些权限。

## 2.1 Android 应用组件

应用组件是 Android 应用的基本构建基块。每个组件都是一个不同的点，系统可以通过它进入您的应用。 并非所有组件都是用户的实际入口点，有些组件相互依赖，但每个组件都以独立实体形式存在，并发挥特定作用 — 每个组件都是唯一的构建基块，有助于定义应用的总体行为。

共有四种不同的应用组件类型。每种类型都服务于不同的目的，并且具有定义组件的创建和销毁方式的不同生命周期。

**Activity**

*Activity* 表示具有用户界面的单一屏幕。例如，电子邮件应用可能具有一个显示新电子邮件列表的 Activity、一个用于撰写电子邮件的 Activity 以及一个用于阅读电子邮件的 Activity。 尽管这些 Activity 通过协作在电子邮件应用中形成了一种紧密结合的用户体验，但每一个 Activity 都独立于其他 Activity 而存在。 因此，其他应用可以启动其中任何一个 Activity（如果电子邮件应用允许）。 例如，相机应用可以启动电子邮件应用内用于撰写新电子邮件的 Activity，以便用户共享图片。

**服务**

*服务*是一种在后台运行的组件，用于执行长时间运行的操作或为远程进程执行作业。 服务不提供用户界面。 例如，当用户位于其他应用中时，服务可能在后台播放音乐或者通过网络获取数据，但不会阻断用户与 Activity 的交互。 诸如 Activity 等其他组件可以启动服务，让其运行或与其绑定以便与其进行交互。

**内容提供程序**

*内容提供程序*管理一组共享的应用数据。您可以将数据存储在文件系统、SQLite 数据库、网络上或您的应用可以访问的任何其他永久性存储位置。 其他应用可以通过内容提供程序查询数据，甚至修改数据（如果内容提供程序允许）。 例如，Android 系统可提供管理用户联系人信息的内容提供程序。 因此，任何具有适当权限的应用都可以查询内容提供程序的某一部分（如 [ContactsContract.Data](https://developer.android.com/reference/android/provider/ContactsContract.Data.html)），以读取和写入有关特定人员的信息。

内容提供程序也适用于读取和写入您的应用不共享的私有数据。 例如，[记事本](https://developer.android.com/resources/samples/NotePad/index.html)示例应用使用内容提供程序来保存笔记。

**广播接收器**

*广播接收器*是一种用于响应系统范围广播通知的组件。 许多广播都是由系统发起的 — 例如，通知屏幕已关闭、电池电量不足或已拍摄照片的广播。应用也可以发起广播 — 例如，通知其他应用某些数据已下载至设备，并且可供其使用。 尽管广播接收器不会显示用户界面，但它们可以[创建状态栏通知](https://developer.android.com/guide/topics/ui/notifiers/notifications.html)，在发生广播事件时提醒用户。 但广播接收器更常见的用途只是作为通向其他组件的“通道”，设计用于执行极少量的工作。 例如，它可能会基于事件发起一项服务来执行某项工作。

Android 系统设计的独特之处在于，任何应用都可以启动其他应用的组件。 例如，如果您想让用户使用设备的相机拍摄照片，很可能有另一个应用可以执行该操作，那么您的应用就可以利用该应用，而不是开发一个 Activity 来自行拍摄照片。 您不需要集成甚至链接到该相机应用的代码，而是只需启动拍摄照片的相机应用中的 Activity。 完成拍摄时，系统甚至会将照片返回您的应用，以便您使用。对用户而言，就好像相机真正是您应用的组成部分。

当系统启动某个组件时，会启动该应用的进程（如果尚未运行），并实例化该组件所需的类。 例如，如果您的应用启动相机应用中拍摄照片的 Activity，则该 Activity 会在属于相机应用的进程，而不是您的应用的进程中运行。因此，与大多数其他系统上的应用不同，Android 应用并没有单一入口点（例如，没有 main() 函数）。

由于系统在单独的进程中运行每个应用，且其文件权限会限制对其他应用的访问，因此您的应用无法直接启动其他应用中的组件， 但 Android 系统却可以。因此，要想启动其他应用中的组件，您必须向系统传递一则消息，说明您想启动特定组件的 Intent。 系统随后便会为您启动该组件。

## 2.2 Android 启动组件

四种组件类型中的三种 — Activity、服务和广播接收器 — 通过名为 Intent 的异步消息进行启动。Intent 会在运行时将各个组件相互绑定（您可以将 Intent 视为从其他组件请求操作的信使），无论组件属于您的应用还是其他应用。

Intent 使用 [Intent](https://developer.android.com/reference/android/content/Intent.html) 对象创建，它定义的消息用于启动特定组件或特定类型的组件 — Intent 可以是显式的，也可以是隐式的。

对于 Activity 和服务， Intent 定义要执行的操作（例如，“查看”或“发送”某个内容），并且可以指定要执行操作的数据的 URI（以及正在启动的组件可能需要了解的信息）。 例如， Intent 传达的请求可以是启动一个显示图像或打开网页的 Activity。 在某些情况下，您可以启动 Activity 来接收结果，在这种情况下，Activity 也会在 [Intent](https://developer.android.com/reference/android/content/Intent.html) 中返回结果（例如，您可以发出一个 Intent，让用户选取某位联系人并将其返回给您 — 返回 Intent 包括指向所选联系人的 URI）。

对于广播接收器， Intent 只会定义要广播的通知（例如，指示设备电池电量不足的广播只包括指示“电池电量不足”的已知操作字符串）。

Intent 不会启动另一个组件类型 - 内容提供程序，后者会在成为 [ContentResolver](https://developer.android.com/reference/android/content/ContentResolver.html) 的请求目标时启动。 内容解析程序通过内容提供程序处理所有直接事务，使得通过提供程序执行事务的组件可以无需执行事务，而是改为在 [ContentResolver](https://developer.android.com/reference/android/content/ContentResolver.html) 对象上调用方法。 这会在内容提供程序与请求信息的组件之间留出一个抽象层（以确保安全）。

每种类型的组件有不同的启动方法：

您可以通过将 [Intent](https://developer.android.com/reference/android/content/Intent.html) 传递到 [startActivity()](https://developer.android.com/reference/android/content/Context.html" \l "startActivity(android.content.Intent)) 或 [startActivityForResult()](https://developer.android.com/reference/android/app/Activity.html" \l "startActivityForResult(android.content.Intent,%20int))（当您想让 Activity 返回结果时）来启动 Activity（或为其安排新任务）。

您可以通过将　[Intent](https://developer.android.com/reference/android/content/Intent.html) 传递到 [startService()](https://developer.android.com/reference/android/content/Context.html" \l "startService(android.content.Intent)) 来启动服务（或对执行中的服务下达新指令）。 或者，您也可以通过将 [Intent](https://developer.android.com/reference/android/content/Intent.html) 传递到 [bindService()](https://developer.android.com/reference/android/content/Context.html" \l "bindService(android.content.Intent,%20android.content.ServiceConnection,%20int)) 来绑定到该服务。

您可以通过将 [Intent](https://developer.android.com/reference/android/content/Intent.html) 传递到 [sendBroadcast()](https://developer.android.com/reference/android/content/Context.html" \l "sendBroadcast(android.content.Intent))、[sendOrderedBroadcast()](https://developer.android.com/reference/android/content/Context.html" \l "sendOrderedBroadcast(android.content.Intent,%20java.lang.String)) 或 [sendStickyBroadcast()](https://developer.android.com/reference/android/content/Context.html" \l "sendStickyBroadcast(android.content.Intent)) 等方法来发起广播；

您可以通过在 [ContentResolver](https://developer.android.com/reference/android/content/ContentResolver.html) 上调用 [query()](https://developer.android.com/reference/android/content/ContentProvider.html#query(android.net.Uri,%20java.lang.String[],%20android.os.Bundle,%20android.os.CancellationSignal)) 来对内容提供程序执行查询。

## 2.3 Android 清单文件

在 Android 系统启动应用组件之前，系统必须通过读取应用的 AndroidManifest.xml 文件（“清单”文件）确认组件存在。 您的应用必须在此文件中声明其所有组件，该文件必须位于应用项目目录的根目录中。

除了声明应用的组件外，清单文件还有许多其他作用，如：

确定应用需要的任何用户权限，如互联网访问权限或对用户联系人的读取权限

根据应用使用的 API，声明应用所需的最低 [API 级别](https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

声明应用使用或需要的硬件和软件功能，如相机、蓝牙服务或多点触摸屏幕

应用需要链接的 API 库（Android 框架 API 除外），如 [Google 地图库](http://code.google.com/android/add-ons/google-apis/maps-overview.html)

## 2.4 Android 声明组件

清单文件的主要任务是告知系统有关应用组件的信息。例如，清单文件可以像下面这样声明 Activity：

<?xml version="1.0" encoding="utf-8"?>  
<manifest ... >  
    <application android:icon="@drawable/app\_icon.png" ... >  
        <activity android:name="com.example.project.ExampleActivity"  
                  android:label="@string/example\_label" ... >  
        </activity>  
        ...  
    </application>  
</manifest>

在 [<application>](https://developer.android.com/guide/topics/manifest/application-element.html) 元素中，android:icon 属性指向标识应用的图标所对应的资源。

在 [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html) 元素中，android:name 属性指定 [Activity](https://developer.android.com/reference/android/app/Activity.html) 子类的完全限定类名，android:label 属性指定用作 Activity 的用户可见标签的字符串。

您必须通过以下方式声明所有应用组件：

Activity 的 [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html) 元素

服务的 [<service>](https://developer.android.com/guide/topics/manifest/service-element.html) 元素

广播接收器的 [<receiver>](https://developer.android.com/guide/topics/manifest/receiver-element.html) 元素

内容提供程序的 [<provider>](https://developer.android.com/guide/topics/manifest/provider-element.html) 元素

您包括在源代码中，但未在清单文件中声明的 Activity、服务和内容提供程序对系统不可见，因此也永远不会运行。 不过，广播接收器可以在清单文件中声明或在代码中动态创建（如 [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) 对象）并通过调用 [registerReceiver()](https://developer.android.com/reference/android/content/Context.html" \l "registerReceiver(android.content.BroadcastReceiver,%20android.content.IntentFilter)) 在系统中注册。

**声明组件功能**

如上文[启动组件](https://developer.android.com/guide/components/fundamentals.html#ActivatingComponents)中所述，您可以使用 [Intent](https://developer.android.com/reference/android/content/Intent.html) 来启动 Activity、服务和广播接收器。 您可以通过在 Intent 中显式命名目标组件（使用组件类名）来执行此操作。 不过，Intent 的真正强大之处在于隐式 Intent 概念。 隐式 Intent 的作用无非是描述要执行的操作类型（还可选择描述您想执行的操作所针对的数据），让系统能够在设备上找到可执行该操作的组件，并启动该组件。 如果有多个组件可以执行 Intent 所描述的操作，则由用户选择使用哪一个组件。

系统通过将接收到的 Intent 与设备上的其他应用的清单文件中提供的 *Intent 过滤器*进行比较来确定可以响应 Intent 的组件。

当您在应用的清单文件中声明 Activity 时，可以选择性地加入声明 Activity 功能的 Intent 过滤器，以便响应来自其他应用的 Intent。 您可以通过将 [<intent-filter>](https://developer.android.com/guide/topics/manifest/intent-filter-element.html) 元素作为组件声明元素的子项进行添加来为您的组件声明 Intent 过滤器。

例如，如果您开发的电子邮件应用包含一个用于撰写新电子邮件的 Activity，则可以像下面这样声明一个 Intent 过滤器来响应“send” Intent（以发送新电子邮件）：

<manifest ... >  
    ...  
    <application ... >  
        <activity android:name="com.example.project.ComposeEmailActivity">  
            <intent-filter>  
                <action android:name="android.intent.action.SEND" />  
                <data android:type="\*/\*" />  
                <category android:name="android.intent.category.DEFAULT" />  
            </intent-filter>  
        </activity>  
    </application>  
</manifest>

然后，如果另一个应用创建了一个包含[ACTION\_SEND](https://developer.android.com/reference/android/content/Intent.html#ACTION_SEND) 操作的 Intent，并将其传递到 [startActivity()](https://developer.android.com/reference/android/app/Activity.html" \l "startActivity(android.content.Intent))，则系统可能会启动您的 Activity，以便用户能够草拟并发送电子邮件。

如需了解有关创建 Intent 过滤器的详细信息，请参阅 [Intent 和 Intent 过滤器](https://developer.android.com/guide/components/intents-filters.html)文档。

**声明应用要求**

基于 Android 系统的设备多种多样，并非所有设备都提供相同的特性和功能。 为防止将您的应用安装在缺少应用所需特性的设备上，您必须通过在清单文件中声明设备和软件要求，为您的应用支持的设备类型明确定义一个配置文件。 其中的大多数声明只是为了提供信息，系统不会读取它们，但 Google Play 等外部服务会读取它们，以便当用户在其设备中搜索应用时为用户提供过滤功能。

例如，如果您的应用需要相机，并使用 Android 2.1（[API 级别](https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels) 7）中引入的 API，您应该像下面这样在清单文件中以要求形式声明这些信息：

<manifest ... >  
    <uses-feature android:name="android.hardware.camera.any"  
                  android:required="true" />  
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />  
    ...  
</manifest>

现在，没有相机且 Android 版本低于 2.1 的设备将无法从 Google Play 安装您的应用。

不过，您也可以声明您的应用使用相机，但并不要求必须使用。 在这种情况下，您的应用必须将 [required](https://developer.android.com/guide/topics/manifest/uses-feature-element.html#required) 属性设置为 "false"，并在运行时检查设备是否具有相机，然后根据需要停用任何相机功能。

## 2.3 Android 应用资源

Android 应用并非只包含代码 — 它还需要与源代码分离的资源，如图像、音频文件以及任何与应用的视觉呈现有关的内容。 例如，您应该通过 XML 文件定义 Activity 用户界面的动画、菜单、样式、颜色和布局。 使用应用资源能够在不修改代码的情况下轻松地更新应用的各种特性，并可通过提供备用资源集让您能够针对各种设备配置（如不同的语言和屏幕尺寸）优化您的应用。

对于您的 Android 项目中包括的每一项资源，SDK 构建工具都会定义一个唯一的整型 ID，您可以利用它来引用应用代码或 XML 中定义的其他资源中的资源。 例如，如果您的应用包含一个名为 logo.png 的图像文件（保存在 res/drawable/ 目录中），则 SDK 工具会生成一个名为 R.drawable.logo 的资源 ID，您可以利用它来引用该图像并将其插入您的用户界面。

提供与源代码分离的资源的其中一个最重要优点在于，您可以提供针对不同设备配置的备用资源。 例如，通过在 XML 中定义 UI 字符串，您可以将字符串翻译为其他语言，并将这些字符串保存在单独的文件中。 然后，Android 系统会根据向资源目录名称追加的语言限定符（如为法语字符串值追加 res/values-fr/）和用户的语言设置，对您的 UI 应用相应的语言字符串。

Android 支持许多不同的备用资源限定符。限定符是一种加入到资源目录名称中，用来定义这些资源适用的设备配置的简短字符串。 再举一例，您应该经常会根据设备的屏幕方向和尺寸为 Activity 创建不同的布局。 例如，当设备屏幕为纵向（长型）时，您可能想要一种垂直排列按钮的布局；但当屏幕为横向（宽型）时，应按水平方向排列按钮。 要想根据方向更改布局，您可以定义两种不同的布局，然后对每个布局的目录名称应用相应的限定符。 然后，系统会根据当前设备方向自动应用相应的布局。

# 第三章 个税计算器的开发

## 3.1 activity 简介

[Activity](https://developer.android.com/reference/android/app/Activity.html) 是一个应用组件，用户可与其提供的屏幕进行交互，以执行拨打电话、拍摄照片、发送电子邮件或查看地图等操作。 每个 Activity 都会获得一个用于绘制其用户界面的窗口。窗口通常会充满屏幕，但也可小于屏幕并浮动在其他窗口之上。

一个应用通常由多个彼此松散联系的 Activity 组成。 一般会指定应用中的某个 Activity 为“主”Activity，即首次启动应用时呈现给用户的那个 Activity。 而且每个 Activity 均可启动另一个 Activity，以便执行不同的操作。 每次新 Activity 启动时，前一 Activity 便会停止，但系统会在堆栈（“返回栈”）中保留该 Activity。 当新 Activity 启动时，系统会将其推送到返回栈上，并取得用户焦点。 返回栈遵循基本的“后进先出”堆栈机制，因此，当用户完成当前 Activity 并按“返回”按钮时，系统会从堆栈中将其弹出（并销毁），然后恢复前一 Activity。

当一个 Activity 因某个新 Activity 启动而停止时，系统会通过该 Activity 的生命周期回调方法通知其这一状态变化。Activity 因状态变化—系统是创建 Activity、停止 Activity、恢复 Activity 还是销毁 Activity— 而收到的回调方法可能有若干种，每一种回调都会为您提供执行与该状态变化相应的特定操作的机会。 例如，停止时，您的 Activity 应释放任何大型对象，例如网络或数据库连接。 当 Activity 恢复时，您可以重新获取所需资源，并恢复执行中断的操作。 这些状态转变都是 Activity 生命周期的一部分。

### 3.1.1 创建Activity

要创建 Activity，您必须创建 [Activity](https://developer.android.com/reference/android/app/Activity.html) 的子类（或使用其现有子类）。您需要在子类中实现 Activity 在其生命周期的各种状态之间转变时（例如创建 Activity、停止 Activity、恢复 Activity 或销毁 Activity 时）系统调用的回调方法。 两个最重要的回调方法是：

[onCreate()](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle))

您必须实现此方法。系统会在创建您的 Activity 时调用此方法。您应该在实现内初始化 Activity 的必需组件。 最重要的是，您必须在此方法内调用 [setContentView()](https://developer.android.com/reference/android/app/Activity.html" \l "setContentView(android.view.View))，以定义 Activity 用户界面的布局。

[onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause())

系统将此方法作为用户离开 Activity 的第一个信号（但并不总是意味着 Activity 会被销毁）进行调用。 您通常应该在此方法内确认在当前用户会话结束后仍然有效的任何更改（因为用户可能不会返回）。

您还应使用几种其他生命周期回调方法，以便提供流畅的 Activity 间用户体验，以及处理导致您的 Activity 停止甚至被销毁的意外中断。 后文的[管理 Activity 生命周期](https://developer.android.com/guide/components/activities.html#Lifecycle)部分对所有生命周期回调方法进行了阐述。

3.2 fragment 简介

3.3 sqlite 简介

3.4 个税计算器的设计