第3章 移动生态

移动生态是移动终端操作系统及其周边服务、移动应用和硬件设备构成的综合体。安卓生态不仅是全球使用最广泛的移动终端操作系统和应用市场，还包含多样的第三方代码库和软件供应链。除此之外，随着小程序的普及和 AI 技术的快速发展，移动小程序生态和基于移动终端操作系统的新型智能系统也已经成为移动生态中重要的组成部分。

本章将从安卓系统生态、移动应用生态、移动小程序生态以及基于移动终端操作系统的新型智能系统四方面详细阐述移动生态的核心内容，带领读者从中体会移动生态的开放性、兼容性以及多样性。

3.1 安卓系统生态

在移动生态中，安卓系统是构建和运行移动终端和安卓应用的基础。安卓系统基于 Linux 内核进行扩展实现了稳定、安全、高性能的运行环境。在本节中将介绍安卓系统在 Linux 内核上的优化扩展以及定制化的安卓系统。

3.1.1 安卓系统

安卓系统建立在稳定可靠的 Linux 内核之上，通过优化和扩展 Linux 内核，实现了针对移动终端的高效性能和丰富功能。内核是 Linux 操作系统的核心，负责管理硬件资源、提供系统调用接口以及处理与硬件交互的底层任务，并通过驱动程序实现对处理器、内存、磁盘、网络接口等硬件设备的支持。安卓系统根据其需求，对 Linux 内核进程管理以及进程间通信机制进行优化，实现了高效的系统启动、应用运行以及远程过程调用。

在 Linux 操作系统中，进程是执行任务的基本单元，是程序的执行实例。进程管理涉及它们的调度、同步、通信等方面，对于系统的多任务处理和性能优化至关重要。在进程管理机制上，安卓系统引入了 Zygote 进程，作为应用程序的模板进程。Zygote 进程是安卓系统初始化创建的第一个 Java 进程，它是所有安卓应用的父进程，其主要任务是创建并预加载应用进程，这个过程称为“进程孵化”。

当启动新的应用时，系统会复制 Zygote 进程，以加快应用的启动速度。这种预加载机制可以显著减少应用的启动时间，提高用户体验。当用户启动一个安卓应用时，Zygote 会复制自身，创建一个新的 Java 虚拟机加载安卓应用的代码和资源。这样，每个安卓应

用都将在自己的虚拟机进程中运行而不会相互干扰。在进程创建时，Zygote 进程会预加载并共享系统的一些重要资源，包括系统库、字体、位图、系统服务等，因此，每个复制的应用进程都可以直接访问这些资源，而无须重复加载，这极大地降低了安卓应用运行时的内存占用，提高了系统的运行效率。

此外，安卓系统还引入了基于进程状态和用户行为的进程优先级调整机制，以确保系统能够优先响应用户操作和重要任务。在进程间通信的扩展中，安卓系统引入了第 2 章介绍的 Binder 进程间通信机制以实现高效、可靠的跨进程通信方式，并且支持跨进程调用和数据传输。在这一机制中安卓系统还引入了安全控制，通过对通信双方的身份验证和权限控制，实现进程通信中的精确访问控制。

同时，第 2 章介绍的安卓系统架构中，安卓系统通过其原生库、运行时环境以及应用框架层的扩展开发，实现了层次分明的封装和复杂的交互机制，塑造了一个既开放又兼容的操作系统。其中，应用框架层作为安卓应用开发和运行的核心基础，提供了统一的接口和功能，为移动应用生态的繁荣发展奠定了坚实的基础。

3.1.2　定制化安卓系统

定制化安卓系统是指根据用户或终端需求，对原生安卓系统进行修改和定制，以满足特定的功能、性能或用户界面需求的过程。这种定制化可以在不同层面进行，包括用户界面、系统配置以及底层框架的修改。通常开发者在完成定制化安卓系统的开发后，通常会将系统编译为只读存储器（read-only memory，ROM）镜像并发布给用户进行系统更新或变更。

1. 安卓系统定制化开发流程

定制化安卓系统的开发流程通常涉及多个关键步骤，本小节将依次介绍。首先，开发者需要进行需求分析和定制化范围的确定。这包括了解目标用户群体的需求、设备的硬件规格以及所需的功能和特性。其次，开发者会选择合适的定制化平台和工具，如基于安卓开放源代码项目（Android open source project，AOSP）源代码的定制化 ROM 构建工具，以及相关的定制化模块和框架。随后，开发者会根据需求进行具体的定制化开发。这可能包括修改系统用户界面、添加或移除预装应用、调整系统设置、优化性能、增强安全性等。在此过程中，开发者需要深入了解安卓系统的架构和相关代码，并确保所做的修改符合安卓系统的规范和要求。

在完成上述安卓源码修改的流程后，开发者会进行定制化系统的编译、调试和测试。这包括构建定制化 ROM、功能测试、性能测试、兼容性测试以及安全性评估等。在此阶段，开发者需要确保定制化系统的稳定性和安全性。最后，定制化安卓系统可以通过各种渠道进行发布和分发，包括开发者社区、设备制造商等。同时，开发者需要持续提供定制化系统的更新，以改进和优化用户体验。

2. ROM 生态

安卓系统中的 ROM 通常指的是安卓系统的镜像文件，可以用于安装或刷新设备的操作系统。安卓系统中的 ROM 可以分为两类，第一类是原生的官方发布系统固件，也称

底包、原生 ROM，可以从官网下载或用官方更新程序下载获取。原生 ROM 不包含制造商或第三方加入的个性化定制，其用户界面简洁，稳定性较好，比较安全。

第二类是被制造商或第三方修改后的 ROM，其中加入了个性化定制。这些个性化定制可能会优化性能、延长电池寿命或增强用户体验。一般来说，每个搭载安卓系统的设备厂商都会对其设备内部的 ROM 进行定制，如三星的 One UI、华为的 EMUI、小米的 MIUI 等基于安卓系统的第三方操作系统。除此之外，安卓社区的开发者也会基于个人需求开发第三方定制化 ROM。

用户也可以根据自己的需求和设备的兼容性选择一款合适的 ROM 给自己的移动终端刷机，更改使用的操作系统。值得注意的是，刷机是危险操作，在刷机前用户需要充分了解所选 ROM 的特性和风险，以保证刷机后设备的正常使用及其安全性。此外，并非所有的移动终端厂商均支持第三方 ROM，部分厂商限制用户使用第三方 ROM。

3.2 移动应用生态

由于安卓开发环境的开放性及安卓应用的便利性，移动应用生态成为移动生态中的重要组成部分，涵盖了应用的开发、分发和审查三个环节。这三个环节在移动应用生态中环环相扣，构成安卓应用从开发到使用的整个流程。应用开发是用户所使用产品的根本来源，在这一环节中开发者利用 Java、Kotlin 等编程语言以及丰富的开发框架构建功能丰富、用户友好的产品。第二个环节是应用分发，主要包括全球性和地区性两条分发渠道。在该环节中，各类分发渠道（如应用市场）充当开发者和用户之间的桥梁，需兼顾用户需求和开发者产品的推广，从而推动整体移动应用生态的发展。最后一个环节是移动应用审查，在这一环节中，执法机构和各类应用分发平台共同肩负起用户保护者的责任，对提交的应用进行审核，确保其符合特定的政策和相关准则，以保障用户能够安全地使用安卓应用。

在应用开发环节中，第三方代码库和软件供应链极大程度上增强了安卓应用的功能，其生态已经成为移动应用生态中的重要组成部分。应用分发和审查环节通常紧密结合，由监管机构和分发平台协作实现。本章将从第三方代码库、软件供应链和应用分发与审查三个角度介绍移动应用生态。

3.2.1 第三方代码库

安卓系统库指的是原生 C/C++ 库和 Java API，其为开发者提供了开发安卓应用所需的基本功能和服务。而安卓第三方代码库是与安卓系统库相对的概念，开发者在开发应用过程中使用到的除安卓系统库以外的其他代码库均为第三方代码库。

具体而言，安卓第三方代码库指的是由非官方开发者或组织创建并维护的可重用代码库。由于许多应用的设计中存在一些相同的功能，比如第三方登录、搜索过滤、新手引导等，实现这些相同功能的代码通常是相似的，因此一些独立的开发者或组织整合开发了第三方代码库以简化安卓应用的开发流程。第三方库能够为开发者提供实现某类特定功

能的接口，而无须重复实现。

第三方代码库的出现对安卓应用的开发产生了积极的影响，通过专业化的分工服务，能够帮助安卓应用快速实现业务功能、降低开发成本、缩短开发周期，具有广泛的应用场景和价值空间，与移动互联产业高度共生。具体而言，第三方代码库在移动应用中的重要作用主要体现在以下四方面。

(1) 丰富应用功能：第三方代码库对应用的功能进行扩展，提升应用功能的丰富性。第三方库通过提供现成的、经过验证的解决方案，如网络通信、图像处理、数据库管理等，使开发者能够轻松地添加复杂功能，大大丰富了应用的功能性。

(2) 加快开发流程：借助第三方代码库，开发者可以节省大量的时间和精力。这些库提供了已经被验证和测试过的代码，可以直接在项目中使用。开发者可以专注于应用的核心功能，不必为实现基本的功能花费大量时间。这种方式可以加快应用的开发速度，并缩短上线时间。

(3) 提高应用质量：许多第三方代码库是由经验丰富的开发者编写的，并经过了广泛的测试和优化。通过使用这些库，开发者可以确保他们的应用在性能、稳定性和安全性方面达到行业标准。例如，使用 Square 的 OkHttp 库可以帮助开发者实现快速、可靠的网络请求，提高应用的响应速度和稳定性。

(4) 促进社区合作：第三方代码库的开发通常是在开源社区中进行的，这意味着任何人都可以改进代码。这种开放的合作模式有助于推动技术的发展和进步，同时也促进了开发者之间的知识分享和交流。开发者可以通过提交漏洞报告、提出建议或者直接贡献代码来改进这些库，从而使其更加强大和稳定。

第三方代码库类型繁多，数量庞大，而且不同于应用代码，每个第三方代码库不仅依赖于多个其他第三方代码库，同时也被其他第三方代码库和应用代码所依赖。鉴于上述两个特点，第三方代码库已然成为与移动应用生态紧密相连的一种新型开发生态。下面将从第三方库的分类和链接调用流程两个角度介绍第三方代码库生态。

1. 第三方代码库分类

根据第三方代码库的功能特征，可以将常见第三方代码库分为五类：功能性代码库、第三方平台代码库、UI 组件库、安全功能库以及开发工具库，下面将逐一介绍。

(1) 功能性代码库。这类代码库提供了用于处理应用核心功能的工具和功能。它们大大简化了开发者处理常见任务的复杂性，使他们能够更专注于应用的业务逻辑。网络请求库(如 OkHttp、Retrofit)帮助开发者简化网络通信的处理，图像处理库(如 Glide、Picasso)强化了安卓应用中对图片的处理功能，数据库代码库(如 Room、Realm)使数据的持久化变得简单快捷。

(2) 第三方平台代码库。这类代码库基于第三方平台，提供数据分析、广告分发等特定任务。这些平台不仅提供了库本身的功能，还通过服务端提供了更多的增值服务，为应用开发者提供了更广泛的功能和支持。代表性的第三方平台代码库是 Firebase 代码库，提供了一个完整的生态系统，使得开发者可以轻松地构建高质量的应用，并通过分析、测试和改进来不断提升应用的性能和用户体验。

(3) UI 组件代码库。这类第三方代码库提供了各种 UI 组件和工具,帮助开发者创建精美的用户界面。UI 组件库(如 Material Components、FlexboxLayout)提供了丰富的可定制的 UI 组件,动画库(如 Lottie、AndroidViewAnimations)使得动画效果的实现变得容易,布局库(如 ConstraintLayout、FlowLayout)简化了复杂布局的设计和管理。

(4) 安全功能代码库。安全性库为应用提供了必要的安全保障,保护用户的个人信息和应用的敏感数据。加密库(如 Bouncy Castle、Android Keystore)提供了各种加密算法和工具,安全认证库(如 Firebase Authentication、OAuthAndroid)帮助开发者实现用户身份认证和授权。

(5) 开发工具代码库。开发工具类第三方代码库提供了各种辅助工具,帮助开发者更高效地进行开发、调试和构建。调试工具库(如 Stetho、LeakCanary)可以帮助开发者识别和解决应用的性能问题,构建工具库(如 Gradle 插件、Maven 依赖管理工具)简化了构建和打包过程。

这些第三方代码库从应用开发的各个环节为开发者提供便利,保障了安卓用户的使用体验和安全性。除以上第三方代码库外,还有许多库为开发者提供其他的功能和服务,建议感兴趣的读者自行了解,此处不再扩展。

2. 第三方代码库的链接调用流程

安卓生态中的第三方代码库大都被编译为.jar 文件或.aar 文件,以便引入到应用开发中。.jar 文件中只包含了类文件与清单文件。.aar 文件中除了包含 jar 包中的类文件外,还包含工程中使用的所有资源文件。第三方代码库如果是一个简单的类库,则编译成.jar 文件即可。如果是 UI 库,包含一些控件布局文件以及字体等资源文件,则需要编译成.aar 文件。接下来本节以使用 Java 进行安卓开发时使用第三方 jar 包为例介绍第三方代码库的链接调用过程。

1) 导入第三方代码库包

首先从第三方代码库的分发渠道(如公开代码平台或者官网)下载需要的 jar 包,然后按以下步骤将 jar 包导入到安卓应用项目中:

(1) 将 jar 包复制到 app 目录的 libs 目录下;

(2) 右击 jar 包,在弹出的快捷菜单中选择 add as library 选项;

(3) 单击 ok 按钮,即可以导入该 jar 包。

将 jar 包导入后,即可以引用 jar 包中的类。Java 语言中的类是放到包中的,包结构和文件夹的目录结构相似,以层层嵌套的形式存在于应用代码中。文件和文件夹拥有路径名,相应的类和包也有“路径名”,父子结构的包名之间以“.”作为分隔符。如 com 包下面有一个 example 包,example 包下面有一个 Test 类,那么该 Test 类的“路径名”为 com.example.Test。

在 Java 语言中可以使用 import 语句将需要的类导入,导入的方式有以下两种:

(1) 导入特定类。使用类的完全限定名,如在一个 Java 源文件中使用语句 import com.example.Test 即可在代码中引入 Test 类。

(2) 导入整个包或子包。使用通配符“*”导入一个包下所有的类以及子包,如在一

个 Java 源文件中使用语句 import com.example.* 即可导入 example 包下的所有类和子包。

2）函数调用

使用 import 关键字将第三方代码库中的类导入后，即可引用导入的类。

对于引入的第三方代码库中的类，可以使用 new 关键字创建一个对象，在创建完成后，可以使用对象的方法名来调用该对象的一个成员方法。例如，假定 Test 类中定义了一个名为 method 的方法，在创建完 testobject 对象后，可以使用代码语句 testobject.method()调用该方法。

3）第三方代码库类的继承

在类似 Java 语言的面向对象编程语言中，为了保证良好的代码结构和足够高的代码复用性，继承是不可或缺的特性。在使用第三方代码库的开发中，可以在定义一个类时使用 extends 关键字继承其中的类，从而可以增加新的数据或功能。例如，对于前文提及的 com.example.Test 类，可以采用代码语句 class Childtest extends Test 定义一个新的 Test 类的子类 Childtest，从而实现了对第三方代码库类的继承。

3.2.2　软件供应链

软件供应链是现代软件开发和交付过程中至关重要的概念之一。它涵盖了从软件开发的初始阶段到最终用户使用的整个过程，包括第三方组件的选择、集成、测试、交付以及产品维护等各个环节，是软件生产和交付过程中涉及的所有环节和资源的组合。

传统制造业中的供应链是将原始材料加工转换为最终产品。软件供应链类似于该过程。在软件供应链中，“原始材料”包括应用代码、第三方组件、开发工具等。软件开发者通过对这些“原始材料”组合、二次开发、集成、测试等过程快速构建软件。相比从零开始构建软件的开发方式，软件供应链大大提高了软件开发的效率。本节将进一步探讨软件供应链的特征以及安卓生态中的软件供应链。

1. 软件供应链的特征

软件供应链是现代软件开发和交付过程中涉及的所有环节和资源的组合，在这一过程中的各个环节均体现出不同于传统制造业供应链的独特特征。具体而言，软件供应链具有四个显著特征，这些特征共同构成了软件供应链的核心属性，对现代软件开发、交付流程的管理和审计有着不可忽视的价值。

（1）自动化和标准化：软件供应链的流程通常是自动化和标准化的，利用各种工具和技术来实现从开发到交付的全流程管理。这种自动化和标准化使得软件开发过程更加规范和可控，有助于提高产品质量和交付效率。

（2）模块化和可重用性：软件供应链中的组件通常是模块化和可重用的，开发人员可以通过选择并组合不同的组件来构建软件产品。这种模块化和可重用性提高了开发效率，并且有助于降低软件开发成本。

（3）持续交付和持续集成：软件供应链采用持续交付和持续集成的方式，将软件产品持续地交付给最终用户。持续集成确保开发人员频繁地将代码集成到共享代码库中，

并通过自动化测试来验证代码的质量，从而确保软件的稳定性和可靠性。

(4) 安全性和可信度：软件供应链的安全性和可信度至关重要。开发人员需要确保所使用的第三方组件和工具具有良好的安全性和可靠性，以避免引入潜在的安全漏洞和风险。

2. 安卓生态中的软件供应链

安卓生态中的软件供应链通常包括安卓应用开发、分发和维护的整个过程及涉及的参与者。安卓生态中的软件供应链需要从移动应用的开发、分发、维护的全生命周期角度出发进行理解。

首先，在安卓应用开发过程中，开发者通常会依赖于各种第三方组件，如开源库、SDK 等。这些第三方组件提供了丰富的功能和工具，帮助开发者快速构建应用。然而，开发者需要仔细评估和选择这些组件，确保它们的安全性和可靠性。此外，第三方组件的更新和维护也是一个重要的考虑因素，开发者需要及时更新组件以修复存在的漏洞和其他安全问题。

在安卓移动生态中，最具代表性的第三方组件供应商是 Maven 工具和 GitHub 平台。其中，Maven 工具在安卓应用开发中通常被用来管理项目的依赖关系和构建过程。通过 Maven 工具，开发者可以方便地引入第三方库、工具和插件，从而加速开发过程并提高代码质量。除此之外，Maven 工具在安卓软件供应链的生产环节也起到重要作用，其提供了一套强大的构建工具。开发人员可以使用 Maven 工具来完成构建和测试软件。而 GitHub 平台是一个基于 Git 版本控制系统的代码托管平台，全球超过 1 亿开发者将其开源项目托管在 GitHub 平台进行管理和分发。GitHub 平台同样实现了安卓软件供应链中生产环节的管理，其提供了一系列与持续集成和持续部署相关的工具和服务，如 GitHub Actions 服务等。开发人员可以利用这些工具来完成自动化构建、测试以及部署流程，从而提高软件交付的效率和可靠性。通过持续集成和部署，开发团队可以更快地发布新版本，及时修复已知的问题和漏洞。

此外，软件供应链还涉及应用商店等软件分发渠道，安卓生态中的应用通常通过应用商店进行分发和下载。Google Play 商店是最大的安卓应用商店之一，但安卓生态中也存在其他第三方应用商店。开发者需要选择合适的应用商店来发布其应用，确保这些商店的可信度和安全性。同时，用户在下载应用时也应谨慎选择来源，避免从不可信的第三方渠道下载，以防遭受安全风险。

在安卓生态中的软件供应链需要保证应用产品的长期维护，这涉及应用更新和漏洞修复的相关管理措施。应用的安全性不仅取决于发布时的审核，还包括后续的更新和漏洞修复。开发者需要及时发布应用的更新版本，修复已知的漏洞和问题。同时，应用商店也需要提供相关的更新机制和通知，帮助用户及时更新已安装的应用，以保障其安全性和稳定性。

在安卓应用生态中，上游第三方组件中的安全缺陷会沿着软件供应链传播到下游的组件和应用中。由于应用与第三方组件、第三方组件与其他第三方组件之间存在复杂的依赖关系，软件供应链中的任何环节出现安全缺陷或漏洞都将造成大范围的影响，严重危

害用户和移动生态。通过深入了解和分析安卓生态中的软件供应链，读者可以更好地认识其中存在的安全问题，并提出相应的解决方案和措施，以确保用户数据的安全和隐私。

3.2.3　移动应用分发与审查

1. 应用分发渠道

移动应用最主要的分发渠道是应用商店——通过网络向终端用户提供移动应用的信息和数据检索、移动应用发布、下载等服务的业务系统。应用商店的兴起促进了移动互联网的崛起，并深刻地影响了移动互联网的发展态势。

依赖于安卓系统的开发商——谷歌公司的推广与支持，基于谷歌公司发布的 Google Play 全球性应用商店迅速发展，其在提供移动应用分发功能的同时，还提供了相关开发者工具和接口，方便开发者进行移动应用开发、测试和发布，并在开发规范、商业模式、内容审核、隐私保护等层面提供参考。

同时，各个国家和地区会根据当地移动市场、应用开发生态等情况，形成具有地区性特征的应用商店。中国移动应用商店市场总规模高达上百亿元，每日活跃用户数超过 5000 万，且整体用户活跃度仍在不断提升。

应用商店的盈利主要来自用户付费抽成和开发商付费推广。若用户付费下载应用，或在应用中进行充值，则开发者需要按照一定比例向应用商店交纳抽成。此外，开发商如果希望获取更优良的推荐位置，也需要付费给应用商店。近年来，应用商店推出了应用订阅服务，即应用商店将在每个结算周期（如每周、每月）开始时，将代替应用开发商向用户收取费用。

应用中存在各种各样的广告形式，如启动页广告、广告横幅，以及应用的各个组件都可以用于广告宣传。移动应用市场拥有庞大的用户数量，可以对人群进行细分，根据广告购买者的意愿对象进行投放。应用的单击量、安装量越高，开发者获得的收益也越高。

最后，随着网络服务、云服务等技术的快速发展，应用开发者通常会精心设计以增值服务为核心的盈利模式。这一盈利模式通常是在初期将应用免费提供给用户使用，以培养用户的使用习惯。在用户养成了一定的使用习惯后，再针对部分功能进行付费增值服务，常见于各大音视频、游戏应用。

2. 移动应用审查

移动应用审查机制是指应用在上架前，应用分发商所需要对安卓应用进行的认证、审核、测试等一系列流程。《中华人民共和国网络安全法》要求应用软件下载服务提供者履行安全管理义务，对恶意程序、含有不良信息的程序采取禁止发布或者停止提供服务、消除等处置措施，这在法律层面规定了应用商店应承担应用的安全审查义务。

审查移动应用是应用商店和开发者保护用户安全所必须履行的义务，因此有关部门出台了多项法律法规以规范开发者的行为。国内相关法规对移动应用个人信息管理机制做出合法性规范。例如，《中华人民共和国网络安全法》规定，网络运营者收集、使用个人信息，应当遵循合法、正当、必要的原则，需经被收集者同意。而移动应用监管条例《常见类型移动互联网应用程序必要个人信息范围规定》中明确规范应用各项活动可收集个人

信息的具体范围。根据法律法规，应用商店的隐私合规标准通常包含个人信息管理、隐私政策声明、应用权限管理三个维度的监管，详细说明如下。

（1）个人信息管理：应用不得超范围收集个人信息，向第三方提供个人信息时需经用户同意，尽可能避免个人信息的跨境传输。

（2）隐私政策声明：首次运行应当有隐私政策弹窗，应用应当告知用户个人信息的用途并征得同意，应用也需要支持用户的授权同意可撤回等。

（3）应用权限管理：应用需要在首次启动时动态申请所需权限并同步告知用户申请该权限的目的，同时用户授权同意的权限必须支持撤回。

移动应用审查有提出审查、执行审查和审查监管三个环节。其中通常由开发者提出移动应用审查，应用商店执行审查流程，而监管机构将定期对应用审查环节做出监管。在这三个环节中，应用商店承担核心的监管能力。

应用商店在执行审查时主要关注以下审核重点。

（1）开发者资质：为保证应用来源可信，需要验证开发者是否实名认证的合法法人；同时为保证应用运营内容的合法性，需要对运营者资质进行验证，以确保应用运营服务符合其声明。

（2）内容规范：应用不得包含非法金钱交易、低俗内容、攻击性内容等违法内容；应用内不得是简单聚合无效内容，不得是不具备实用价值的内容等；应用需要对用户生成的内容进行有效管控，制定过滤机制，对内容中的违法有害信息进行防范处置并保存有关记录；应制定举报机制并及时做出响应；应实现服务关闭功能，对严重违规用户账号停止提供服务。

（3）应用安全：应用商店通常需要扫描应用的安全机制，防止恶意软件、手机病毒、软件漏洞、伪造应用等问题；同时应用不得含有试图滥用或不当使用任何网络、设备以及干扰其他应用的安全隐患。这些隐患包括但不限于通过可疑代码、文件及程序等形式，对系统造成负面影响或侵害用户权益，通过隐蔽执行、欺骗用户单击等手段订购各类收费业务，劫持系统操作、利用漏洞或采取欺骗手段监控用户、窃取数据。

随着有关部门的强力监管，相关工作举措、法律规定出台，各大应用商店的审查机制也趋于完善。这些措施保障了用户权益，改善了用户体验，并使得过去安卓应用经常滥用隐私的现象也得到了改善。此外，一些应用商店也为开发者提供了自动化合规工具，为应用上架提供了一定的便利。严格的审查机制也提高了开发者的开发成本，进一步促进了我国移动应用从一味地追求数量，转向质量与数量并重的变化趋势。

拓展知识

移动应用审查流程

在移动应用市场中，新应用上架前必须通过严格的审查流程。在安卓系统的各大应用商店中，审查机制和要求各有不同。

一般而言，开发者需要提交一系列审核申请，等待通过应用商店的初审。在提交应用时需要填写应用信息，并上传应用安装包、截图、备案信息等材料，然后提交应用审核申请。某些应用商店还要求开发者完成实名认证等认证流程。收到开发者提交的审核申请

后，应用商店会进行初步的审核准备工作，包括检查应用安装包的完整性、审核材料的准确性、备案信息的完整性等。

通过初审后，应用将进入详细审核阶段，通常包含多轮测试，涉及应用的功能、性能、安全性、用户体验等方面。审核人员会对应用进行测试，如果发现问题，将提出具体的反馈意见，开发者需对应用进行修改后重新提交审核。经过详细审核后，如果应用通过了审查，就可以在应用商店上架。上架后，用户便可通过搜索和下载来使用应用。审查流程确保了应用在上架前经过了严格的测试和审核，以保障用户体验和应用商店的安全性。如应用审核中发现问题，审核人员将给出具体的反馈意见，开发者需要对应用进行修改后重新提交审核。

3.3　移动小程序生态

随着移动互联网的不断发展，小程序作为一种轻量级的应用形式，正日益受到用户和开发者的关注。小程序生态系统的不断发展为用户提供了更丰富多样的服务、更低的开发成本和更广阔的市场，其平台的建设和完善为小程序的开发和管理提供了良好的保障。同时，小程序与传统移动应用之间的竞争与合作促进了整个移动应用生态系统的健康发展。本小节将介绍移动小程序生态，包括小程序的特征及其与传统移动应用之间的关系。

3.3.1　小程序特征

在移动互联网用户增速逐渐趋缓、市场规模逐渐饱和的背景下，新兴的移动应用面临着日益严峻的挑战。开发者面临着高成本、长周期、市场空间有限等问题，而用户则面临着流量消耗大、手机内存占用高、学习门槛高等难题。

为了提高流量资源的分发效率、满足用户多样化需求、进一步挖掘用户价值，移动互联网时代开发者开始探索新模式，即“超级应用＋小程序”的组合应用模式。这种模式下，超级应用如微信、支付宝等平台开始在其应用中搭载第三方小程序，为用户提供更丰富的服务形式和内容。小程序作为一种轻量级的应用形式，具有下列诸多独特的特点，使其逐渐在移动应用层生态中占据重要地位。

(1) 功能简洁明了：小程序平台严格限制了小程序代码包的大小，促使开发者将业务逻辑简化，更专注于核心功能的实现。这意味着小程序的功能简洁明了，用户可以迅速理解并上手使用，避免被繁杂的功能所困扰。

(2) 使用便捷：用户无须下载、安装、注册或卸载小程序，只需通过搜索、单击、授权等简单步骤即可直接进入小程序并享受服务。这种无须经历烦琐安装过程的使用方式大大降低了用户的使用门槛，提升了用户体验。

(3) 开发成本低：小程序开发相对于传统移动应用开发来说，具有更低的开发成本。开发者可以利用小程序平台提供的开发工具和框架，通过简单的组件化编程即可开发出具有原生应用体验的小程序应用。这种开发方式节约了开发时间和人力成本，使更多的开发者能够参与到小程序应用的开发中。

3.3.2 小程序与移动应用的关系

小程序与移动应用之间的关系主要体现在运行机制、用户信息管理和商业意义三个角度。本节将逐一介绍小程序与移动应用之间的关系。

1. 运行机制

小程序与传统移动应用之间的关系体现在二者之间的寄宿关系上，小程序通常运行在移动应用客户端内，基于客户端及其提供的基础库构成的宿主环境运行。其中，提供小程序运行平台的移动应用被称为超级应用。客户端所提供的运行环境为小程序提供功能性框架 API、用户信息收集和使用渠道等重要接口。具体而言，小程序可以使用平台提供的功能（如调用设备摄像头等），也可以通过平台获取相关用户信息（如用户的手机号码）。同时，超级应用需要负责小程序的监管。为保护用户个人信息，超级应用通常会对小程序提出相关要求。超级应用类似于小程序的操作系统平台。

下面以微信小程序为例介绍小程序的运行模式。如图 3-1 所示，应用为小程序提供逻辑层（logic layer）和视图层（view layer）的运行环境，分别由两个线程独立管理。逻辑层负责小程序的 JavaScript 代码执行，用于处理代码逻辑、数据请求等业务功能。视图层创建的 WebView 线程主要负责小程序 UI 渲染。而微信通过安卓系统的 WebView 组件的 addJavascriptInterface()方法实现了 JSBridge 层作为小程序开发与应用层之间的桥梁，用于执行诸如与第三方服务器通信的功能。在本书第 10 章将详细介绍微信应用的小程序架构。

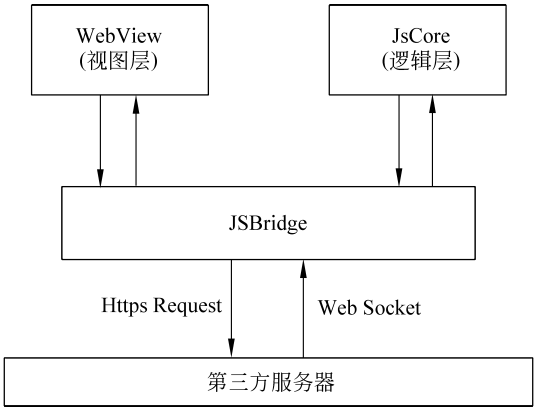

图 3-1　微信应用的小程序架构

2. 用户信息管理

超级应用通常存储大量用户信息，并且拥有收集和处理用户隐私的系统权限。小程序的业务功能同样依赖用户信息，但由于其运行在超级应用上，无法直接访问安卓系统中处理用户数据的 API。面对这一需求，超级应用会提供一系列 API 用于满足小程序的用户数据收集与使用需求，这些接口可以用于监控小程序对用户信息的处理是否符合其权限。因此，小程序与移动应用在用户信息管理这一维度上联系紧密。

一方面小程序可以通过小程序平台获取用户信息、设备信息和统计信息。其中，用户信息包括平台昵称、头像、所在地区、手机号等个人信息，以及设备信息如网络状态、系统版本等。这些信息可以帮助小程序开发者更好地了解用户需求和行为，从而优化小程序的功能和体验。

另一方面小程序平台为了保护用户个人信息，对小程序提出了一系列要求。首先，在数据收集、存储与授权方面，小程序必须经过用户同意才能收集用户数据，并且必须向用户如实披露数据内容、用途和使用范围；其次，在数据使用规范方面，小程序不得将用户个人信息用于未经授权的用途，也不得在未经用户同意的情况下使用用户数据；再次，在数据安全方面，平台要求小程序谨慎保管用户信息，确保用户授权的隐私信息和数据安全；此外，在地理位置方面，小程序只有在确有实际需要的情况下才能申请获取用户的实时位置信息，并且必须获得用户的同意。

3. 商业联系

从商业角度而言，小程序与移动应用之间是互惠互利的关系。小程序是一种不需要下载安装即可使用的应用，因此，带来了便捷的用户体验，也提供了丰富的功能和服务，使得超级应用的功能性得到了极大的扩展。通过小程序，单一的超级应用可以提供超出其本身原有业务功能的服务，例如，社交应用微信通过小程序可以提供包括购物、旅游、餐饮在内的多种服务，而无须安装和切换其他应用。而一个拥有高用户流量、优质开发团队的超级应用平台也将为小程序厂商提供更多的商业推广，带来更高的品牌曝光度和盈利。

综上所述，小程序与传统移动应用的关系是一种互相依存、合作与约束的关系。小程序通过平台获取用户信息，平台也对小程序进行严格的规范和监管，以保护用户的个人信息和数据安全。这种关系不仅促进了移动应用生态的发展，也保护了用户的权益和隐私。

3.4　基于移动终端操作系统的新型智能系统

随着新兴技术的发展，移动终端操作系统逐渐开始部署于新型智能系统中，进一步成为控制智能系统的核心。本节将基于移动终端操作系统的新型智能系统应用案例从特点与发展趋势两个角度探讨移动生态中的新兴领域。

3.4.1　新型智能系统案例

移动终端操作系统在各种智能系统中均有广泛应用，其中一个显著的应用案例是智能家居系统。谷歌公司意识到家用物联网设备的广大需求后，基于其发布的原生安卓系统针对物联网设备推出名为 Android Things 的物联网设备操作系统。Android Things 支持各种硬件平台，并提供了与搭载安卓系统移动终端进行交互的接口。

因此，通过将各种家庭设备连接到智能手机上，并利用移动终端操作系统提供的平台和接口，用户可以实现对家庭灯光、温度、安防等方面的远程控制和自动化管理。例如，用户可以使用智能手机上的应用远程监控家中的摄像头，并通过手机控制智能灯泡的亮度和颜色。这种智能家居系统不仅提高了生活的便利性，还能够节约能源和增强家庭安

全性。

近年来，各大移动厂商纷纷生产智能手表和其他可穿戴设备。面对这一市场需求，安卓系统开发团队基于安卓系统构建了安卓可穿戴系统，以支持信息查看、健康监测、语音助手和触屏交互等便捷功能。借助安卓可穿戴系统的传感器支持和数据处理能力，智能手机可以成为人们的个人健康监护仪。例如，用户可以使用智能手环或智能手表配合移动应用来实时监测自己的心率、睡眠质量、步数等健康指标，并将数据同步到云端进行分析和管理。

在车联网领域，安卓系统开发团队发现智能汽车中的操作系统需求与移动终端中的操作系统需求相似，在安卓系统的基础上设计了一套基于车载硬件运行的移动智能终端操作系统——Android Automotive。Android Automotive 在车载硬件的基础上提供了与安卓系统相同的应用框架和 API，这使得全球数十万安卓开发者的开发经验和成品软件得以重复使用。基于 Android Automotive 系统，开发者可以利用其导航系统、车载互联网功能开发丰富的智能子系统，例如，信息娱乐系统、辅助驾驶系统等，极大地提升了汽车的驾驶性能和用户体验。

3.4.2 新型智能系统特点与发展趋势

新型智能系统具有多重特点，这些特点不仅反映了移动终端操作系统在智能系统中的优势，还展示了其在未来发展中的潜力，其主要体现在以下四个维度。

（1）多样化的应用场景。移动终端操作系统的灵活性和开放性为智能系统多样化的应用场景提供了广阔的空间。从智能家居、智能健康监测到智能交通、智能娱乐，移动终端操作系统都在不同应用场景中发挥着重要作用。随着物联网技术的普及和智能设备的增多，基于移动终端操作系统的智能系统将涵盖更多的应用场景，为用户带来更多便利和更加智能化的体验。

（2）数据互联与智能分析。移动终端操作系统支持的传感器技术和云计算技术使得智能系统能够实时收集和分析大量的数据。通过对这些数据进行智能分析和挖掘，智能系统可以为用户提供个性化的服务和智能化的决策支持。例如，智能健康监测系统可以根据用户的健康数据推荐适合的运动和饮食方案，智能家居系统可以根据用户的生活习惯自动调节家庭设备的工作模式，从而提高用户的生活质量。

（3）AI 技术的集成。随着 AI 技术的不断发展和普及，基于移动操作系统的智能系统将会更加智能化。智能手机上的语音助手、人脸识别技术等已经成为智能系统中常见的功能，未来还将出现更多基于 AI 的应用场景。例如，智能语音助手可以通过自然语言理解和机器学习技术来更好地理解用户的需求，并提供更加智能化的交互体验。

（4）与物理设备的深度融合。随着物联网技术的成熟和普及，基于移动终端操作系统的智能系统将与各种智能设备和传感器实现更紧密的互联。通过与智能家居设备、智能可穿戴设备、智能车载设备等进行融合，智能系统可以提供更加智能化和个性化的服务。例如，智能家居系统可以通过与智能门锁、智能摄像头等设备的联动，实现更加智能化的家居安全管理和便捷的家庭生活体验。

3.5 本章小结

本章主要围绕安卓移动生态进行介绍，旨在通过详细阐述安卓系统生态、移动应用生态、移动小程序生态以及基于移动终端操作系统的新型智能系统，带领读者体会移动生态的开放性、兼容性以及多样性。安卓系统生态部分主要从安卓系统和定制化安卓系统两个角度进行阐述，对安卓系统基于 Linux 内核的优化和扩展技术进行分析，并且讨论了安卓系统中的 ROM 生态。移动应用生态主要介绍了第三方代码库、软件供应链以及移动应用分发与审查渠道。随后，结合目前的发展趋势，本章介绍了移动小程序生态和基于移动终端操作系统的新型智能系统，进一步展望了移动生态的未来发展趋势。

3.6 习题

1. 移动生态有什么特性，具体体现在哪些方面？
2. 安卓系统在 Linux 内核的基础上做了哪些优化和扩展？有何作用？
3. 请简述安卓系统启动至 UI 界面启动的流程。
4. 为什么需要第三方代码库？安卓第三方代码库如何分类？
5. 什么是开源软件，软件开源有何意义？
6. 软件供应链的生命周期是怎样的？有什么特点？
7. 小程序和超级应用平台（移动应用）存在哪些区别与联系？
8. 移动终端操作系统在新型智能系统中有什么作用？未来还会有哪些应用场景？