第 1 章 初识HarmonyOS

1.1 Android 简介

1.1.1 Android 概述

Android 是一种基于 Linux 内核（不包含 GNU 组件）的自由及开放源代码的移动操作系统，由 Google 成立的 OHA（Open Handset Alliance，开放手机联盟）领导并开发，为各类智能手机及便携式设备提供可运行的操作系统。

Android 操作系统最早由 Andy Rubin、Rich Miner 和 Nick Sears 等创建并开发，后被 Google 于 2005 年收购。2007 年 11 月，Google 联合硬件制造商、软件开发商及电信营运商组建了开放手机联盟，随后以 Apache 开源许可证的授权方式，发布了 Android 的源代码，称为 AOSP（Android Open Source Project，Android 开放源代码项目）。

2008 年，Google 提出了 Android HAL 架构，并联合 HTC 发布了第一部 Android 智能手机。此后，Android 不断发展更新，并逐渐扩展到平板电脑及其他领域，如电视、数码相机、游戏机、智能手表等，并在 2011 年首次超过塞班系统，跃居全球移动操作系统市场份额第一，彼时的 iOS、塞班等移动操作系统完全没有与之抗衡的能力。Android 能在短时间内称雄移动操作系统市场，归功于 Android 系统的 AOSP。

随着 Android 系统的不断发展壮大，Android 在 2017 年 3 月成功超越 Microsoft Windows，成为全球装机量最多的操作系统。截至 2023 年 8 月，根据 StatCounter 统计，除了美国、英国、加拿大、巴哈马、冰岛等少数国家和地区外，Android 都是占比最高的智能手机操作系统。

1.1.2 Android 系统架构

Android 系统构架又被称为体系结构。和其他操作系统一样，Android 系统也采用了分层的架构，共分为四层五部分，从下到上分别是 Linux 内核层、硬件抽象层（HAL）、Android Runtime 系统库、Java API 框架层和应用层，如图 1-1 所示。

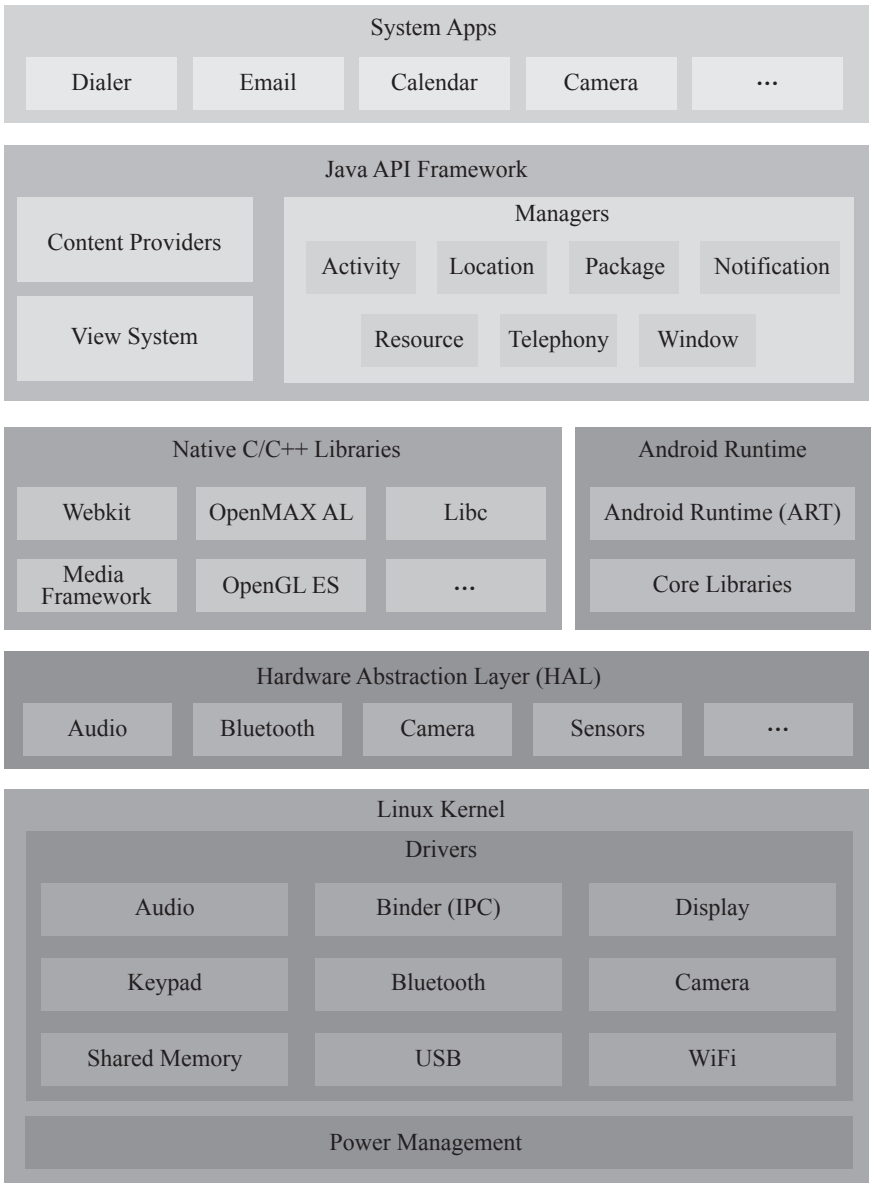

图 1-1　Android 系统架构图

Linux 内核层

Android 的核心系统服务依赖于 Linux 内核，如安全性、内存管理、进程管理、网络协议栈和驱动模型。同时，Linux 内核也是硬件和软件栈之间的抽象层，允许设备制造商为内核开发硬件驱动程序。

硬件抽象层

硬件抽象层提供标准接口，向更高级别的 Java API 框架提供接入能力。HAL 包含多个

库模块，其中每个模块都为特定类型的硬件组件实现一个界面，例如相机或蓝牙模块。当框架 API 要求访问硬件设备时，Android 系统将为该硬件加载相应的库模块。

Android Runtime 系统库

对于运行 Android 5.0 或更高版本的设备，每个应用都在自己的进程中运行，并且拥有自己的 Android Runtime（ART）实例。而在 Android 5.0 版本之前，Dalvik 就是 Android Runtime。ART 通过执行 DEX 文件，可以在设备上运行多个虚拟机，DEX 文件是一种专为 Android 设计的字节码格式文件，作为一种经过优化的字节码文件，DEX 文件占用的内存很少。

编译工具链（如 Jack）能够将 Java 源代码编译为 DEX 字节码，使其运行在 Android 设备上。ART 主要功能包括：预先（AOT）编译和即时（JIT）编译，优化垃圾回收（GC），以及调试方面的支持，如专用采样分析器、详细的诊断异常和崩溃报告等。

同时，Android 还包含一套核心运行时库，可提供 Java API 框架所使用的 Java 编程语言中的大部分功能，还包括一些 Java 8 语言功能。

事实上，Android 系统的许多核心系统组件和服务（如 ART 和 HAL）都构建自原生代码，所以需要使用 C 和 C++ 来编写对应的原生库，然后再提供 Java API 供应用层调用。例如，可以通过 Android 提供的 Java OpenGL API 来访问 OpenGL ES 的相关服务，实现在应用中绘制和操作 2D 和 3D 图形。

Java API 框架层

Java API 框架层主要提供构建应用程序可能用到的各种 API，Android 自带的一些核心应用就使用到了这些 API，开发者也可以使用这些 API 来构建属于自己的应用程序。

事实上，这些 API 是创建 Android 应用所必需的构建模块，它们是一些简化后的系统组件和服务，是可以重复使用的，具体包括如下：

- 丰富、可扩展的视图系统，可用于构建应用的 UI，如列表、网格、文本框、按钮以及浏览器等；
- 资源管理器，用于访问非代码资源，如本地化的字符串、图形和布局文件等；
- 通知管理器，用于在应用的状态栏中显示提醒消息；
- Activity 管理器，用于管理应用的生命周期，以及提供导航栈管理；
- 内容提供程序，用于访问其他应用的数据或者共享自己的数据。

应用层

Android 系统在发布时会默认提供一些核心应用程序包，如电子邮件、短信、日历和联系人等，它们一般使用 Java 语言进行编写。事实上，所有安装在手机上的应用程序都属于这一层。

1.2 HarmonyOS 简介

2020 年 9 月 10 日，华为公司在 2020 年华为开发者大会上发布了 HarmonyOS（鸿蒙

操作系统）2.0 版本。HarmonyOS 是一款面向全场景的分布式操作系统。不同于既有的 Android、iOS、Windows、Linux 等操作系统，HarmonyOS 面向的是 1+8+*N* 的全场景设备，能够根据不同内存级别的设备进行弹性组装和适配，并且支持跨设备交互信息。

HarmonyOS 提供了支持多种开发语言的 API，供开发者进行应用开发。支持的开发语言包括 Java、XML、C/C++、JavaScript（简称 JS）、CSS 和 HTML。

1.2.1 HarmonyOS 概述

HarmonyOS 是华为公司发布的一款面向万物互联的全新分布式操作系统。不同于既有的 Android、iOS、Windows、Linux 等操作系统，HarmonyOS 提出了基于同一套系统能力、适配多种终端形态的分布式理念，能够同时支持手机、平板、智能穿戴、智慧屏、车机等多种终端设备，提供全场景业务能力，实现极速发现、极速连接、硬件互助、资源共享的场景体验。

对消费者而言，HarmonyOS 能够将生活场景中的各类终端进行能力整合，形成一个超级虚拟终端，并且能够实现不同终端设备之间的快速连接、能力互助、资源共享，匹配合适的设备，提供流畅的全场景体验。

对应用开发者而言，HarmonyOS 采用了多种分布式技术，使得应用程序的开发实现与不同终端设备的形态差异无关，降低了开发难度和成本。让开发者聚焦上层业务逻辑，更加便捷、高效地开发应用。

对设备开发者而言，HarmonyOS 采用了组件化的设计方案，可以根据设备的资源能力和业务特征进行灵活裁剪，满足不同形态的终端设备对于操作系统的要求。

自 2019 年 8 月正式发布以来，HarmonyOS 一共经过了四次大的版本迭代。目前发布的最新版本 HarmonyOS Next 是一款同时支持手机、平板、智能穿戴、智慧屏、车机等多种终端设备的操作系统。

1.2.2 HarmonyOS 技术特性

HarmonyOS 具备分布式软总线、分布式数据管理和分布式安全三大核心能力，体现如下。

- 分布式软总线：作为多种终端设备的统一基座，为设备的互联互通提供统一的分布式通信能力，让多设备融为一体，带来设备内和设备间高吞吐、低时延、高可靠的流畅连接体验。
- 分布式数据管理：基于分布式总线提供的能力，实现应用程序数据和用户数据的分布式管理。用户数据不再与单一物理设备绑定，跨设备运行时数据无缝衔接，为打造一致、流畅的用户体验创造了基础条件。
- 分布式安全：HarmonyOS 能够把手机的内核级安全能力扩展到其他终端，进而提升全场景设备的安全性，通过设备互助、共同抵御攻击，保障智能家居网络安全。HarmonyOS 自定义数据和设备的安全级别，对数据和设备进行分类分级保护，确保

数据流通安全可信。

HarmonyOS 提供的用户程序框架、Ability 框架以及 UI 框架，支持在应用开发过程中多终端的业务逻辑和界面逻辑的复用，实现应用的一次开发、多端部署，提升跨平台应用开发的能力。

除此之外，HarmonyOS 通过组件化和小型化的设计方案，支持多终端设备的按需弹性部署，进而适配不同类别的硬件资源和功能需求；支持通过编译链自动生成组件化的依赖关系，形成组件树依赖图，支撑产品系统的便捷开发，降低硬件设备的开发门槛。

1.2.3　HarmonyOS 系统安全

搭载 HarmonyOS 的分布式终端设备，主要从“正确的人，通过正确的设备，正确地使用数据”三方面来保证用户数据的安全，体现如下。

- 通过分布式多端协同身份认证来保证正确的人。
- 通过在分布式终端上构筑可信运行环境来保证正确的设备。
- 通过分布式数据在跨终端流动的过程中，对数据进行分类分级管理来保证正确地使用数据。

正确的人

在分布式终端场景下，正确的人指通过身份认证的数据访问者和业务操作者。正确的人是确保用户数据不被非法访问、用户隐私不被泄露的前提条件。目前，HarmonyOS 通过三方面来实现协同身份认证。

- 零信任模型：HarmonyOS 基于零信任模型，实现对用户的认证和对数据的访问控制。当用户需要跨设备访问数据资源或者发起高安全等级的业务操作时，HarmonyOS 会对用户进行身份认证，确保其身份的可靠性和合法性。
- 多因素融合认证：HarmonyOS 通过用户身份管理，将不同设备上标识同一用户的认证凭据关联起来，提高了认证的准确度。
- 协同互助认证：HarmonyOS 通过将硬件和认证能力解耦，实现不同设备的资源池化以及能力的互助与共享，让高安全等级的设备协助低安全等级的设备完成用户身份认证。

正确的设备

在分布式终端场景中，只有保证用户使用的设备是安全可靠的，才能保证用户数据在终端上得到有效保护，避免用户隐私泄露的问题。

- 安全启动：确保源头每个虚拟设备运行的系统固件和应用程序是完整的、未经篡改的。通过安全启动程序，保证设备厂商的镜像包没有被非法替换为恶意程序，从而保护用户的数据和隐私安全。
- 可信执行环境：提供了基于硬件的可信执行环境来保护用户的个人敏感数据的存储和处理，确保数据不泄露。HarmonyOS 使用基于数学可证明的形式化开发和验证的 TEE 微内核，获得了商用 OS 内核 CC EAL5+ 的认证评级。

- 设备证书认证：支持为具备可信执行环境的设备预置设备证书，用于向其他虚拟终端证明自己的安全能力。设备证书在产线进行预置，设备证书的私钥写入并安全保存在设备的 TEE 环境中，且只在 TEE 内进行使用。在必须传输用户的敏感数据时，会在使用设备证书进行安全环境验证后，建立从一个设备的 TEE 到另一设备的 TEE 之间的安全通道，进而实现安全传输。

正确地使用数据

在分布式终端场景下，需要确保用户能够正确地使用数据。HarmonyOS 围绕数据的生成、存储、使用、传输以及销毁过程进行全生命周期的保护，从而保证用户数据与隐私以及系统的机密数据（如密钥）不被泄露。

- 数据生成：根据数据所在的国家或组织的法律法规与标准规范，对数据进行分类分级，并且根据分类设置相应的保护等级。每个保护等级的数据从生成开始，在其存储、使用、传输的整个生命周期都需要根据对应的安全策略提供不同强度的安全防护。虚拟超级终端的访问控制系统支持依据标签的访问控制策略，保证数据只能在可以提供足够安全防护的虚拟终端之间存储、使用和传输。
- 数据存储：HarmonyOS 通过区分数据的安全等级，存储到不同安全防护能力的分区，对数据进行安全保护，并提供密钥全生命周期的跨设备无缝流动和跨设备密钥访问控制能力，支撑分布式身份认证协同、分布式数据共享等业务。
- 数据使用：HarmonyOS 通过硬件为设备提供可信执行环境。用户的个人敏感数据仅在分布式虚拟终端的可信执行环境中进行使用，确保用户数据的安全和隐私不泄露。
- 数据传输：为了保证数据在虚拟超级终端之间的安全流转，需要各设备是正确可信的，建立信任关系并在验证信任关系之后，再建立安全的连接通道，然后按照数据流动的规则安全地传输数据。当设备之间进行通信时，需要基于设备的身份凭据对设备进行身份认证，并在此基础上建立安全的加密传输通道。
- 数据销毁：销毁密钥即销毁数据，数据在虚拟终端的存储都建立在密钥的基础上。当销毁数据时，只需要销毁对应的密钥即可。

1.2.4 HarmonyOS 系统架构

HarmonyOS 系统架构遵从分层设计的理念，从下向上依次是内核层、系统服务层、应用框架层和应用层。系统功能按照【系统】>【子系统】>【功能 / 模块】逐级展开，在多设备部署场景下，支持根据实际需求裁剪某些非必要的子系统或功能 / 模块。HarmonyOS 系统架构如图 1-2 所示。

内核层

HarmonyOS 系统的内核层分为内核子系统和驱动子系统，说明如下。

- 内核子系统：HarmonyOS 采用多内核设计，支持针对不同资源受限设备选用适合的 OS 内核。内核抽象层（Kernel Abstract Layer，KAL）通过屏蔽多内核差异，对上层

提供基础的内核能力，包括进程 / 线程管理、内存管理、文件系统、网络管理和外设管理等。

- 驱动子系统：HarmonyOS 驱动框架（HarmonyOS Driver Foundation，HDF）作为 HarmonyOS 硬件生态开放的基础，提供统一外设访问能力和驱动开发、管理框架。

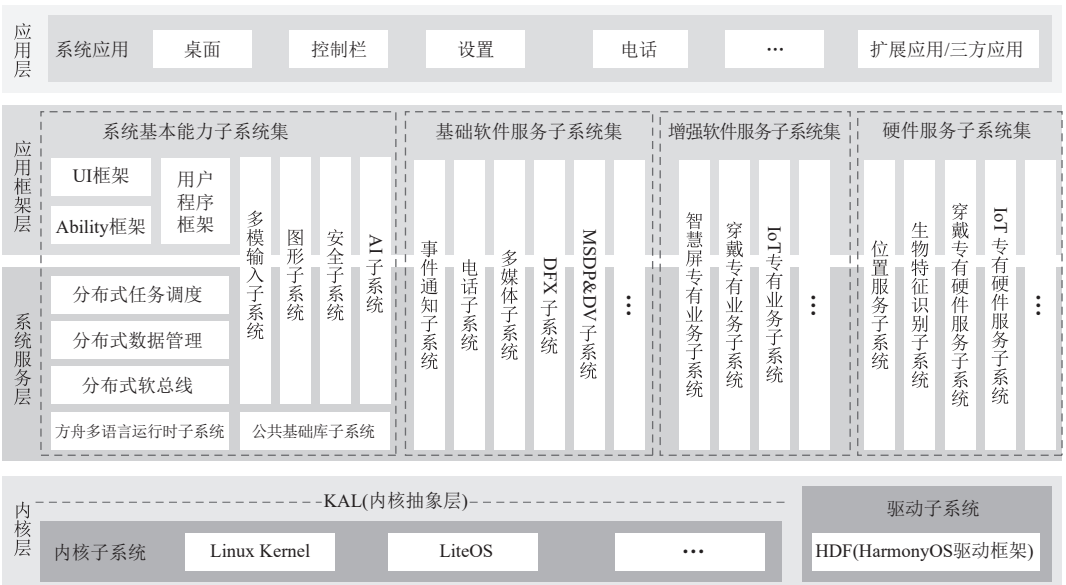

图 1-2　HarmonyOS 系统架构

系统服务层

系统服务层是 HarmonyOS 核心能力的集合，通过框架层对上层的应用程序提供服务。该层包含以下几部分。

- 系统基本能力子系统集：为分布式应用在 HarmonyOS 多设备上运行、调度、迁移等操作提供基础能力，由分布式软总线、分布式数据管理、分布式任务调度、方舟多语言运行时、公共基础库、多模输入、图形、安全、AI 等子系统组成。
- 基础软件服务子系统集：为 HarmonyOS 提供公共的、通用的软件服务，由事件通知、电话、多媒体、DFX、MSDP&DV 等子系统组成。
- 增强软件服务子系统集：为 HarmonyOS 提供针对不同设备的、差异化的能力增强型软件服务，由智慧屏专有业务、穿戴专有业务、IoT 专有业务等子系统组成。
- 硬件服务子系统集：为 HarmonyOS 提供硬件服务，由位置服务、生物特征识别、穿戴专有硬件服务、IoT 专有硬件服务等子系统组成。

根据不同设备形态的部署环境，基础软件服务子系统集、增强软件服务子系统集、硬件服务子系统集内部可以按子系统粒度进行裁剪，并且每个子系统内部又可以按功能粒度进行裁剪。

应用框架层

应用框架层为 HarmonyOS 应用程序提供 Java/C/C++/JavaScript 等多语言的用户程序框架和 Ability 框架，以及各种软硬件服务对外开放的多语言框架 API，同时为采用 HarmonyOS 的设备提供 C/C++/JavaScript 等多语言框架 API。需要注意的是，不同设备支持的 API 与系统的组件化裁剪程度相关。

应用层

应用层包括系统应用和第三方非系统应用。通常，HarmonyOS 应用由一个或多个 FA（Feature Ability）或 PA（Particle Ability）组成。其中，FA 有 UI，提供与用户交互的能力；而 PA 无 UI，提供后台运行任务的能力以及统一的数据访问抽象。基于 FA/PA 开发的应用，能够实现特定的业务功能，支持跨设备调度与分发，为用户提供一致、高效的应用体验。

1.3 HarmonyOS 程序包

1.3.1 HarmonyOS 程序包概述

在软件开发中，应用程序用来泛指运行在设备操作系统之上，为用户提供特定服务的程序，简称应用。一个应用所对应的软件包文件，称为应用程序包。

HarmonyOS 提供了应用程序包开发、安装、查询、更新、卸载管理机制，这些管理机制可以方便开发者开发和管理 HarmonyOS 应用，说明如下。

- 应用软件所涉及的文件多种多样，开发者可通过 HarmonyOS 提供的集成开发工具将其开发的可执行代码、资源、三方库等文件整合到一起制作成 HarmonyOS 应用程序包，便于开发者对应用程序的部署。
- 应用软件所涉及的设备类型多种多样，开发者可通过 HarmonyOS 提供的应用程序包配置文件指定其应用程序包的分发设备类型，便于应用市场对应用程序包的分发管理。
- 应用软件所包含的功能多种多样，将不同的功能特性按模块来划分和管理是一种良好的设计方式。HarmonyOS 提供了同一应用程序的多包管理的机制，开发者可以将不同的功能特性聚合到不同的包中，方便后续的维护与扩展。
- 应用软件涉及的芯片平台多种多样，有 x86、ARM 等，还有 32 位、64 位之分，HarmonyOS 为应用程序包屏蔽了芯片平台的差异，使应用程序包在不同的芯片平台都能够安装运行。
- 应用软件涉及的软件信息多种多样，有应用版本、应用名称、组件、申请权限等的信息，HarmonyOS 包管理为开发者提供了这些信息的查询接口，方便开发者在程序中查询所需要的包信息。

- 应用软件涉及的资源多种多样，有媒体资源、原生资源、字符资源以及国际化的资源等，HarmonyOS 包管理将不同的资源归档到不同的目录中，并集成资源索引文件，方便应用对资源的查找和使用。

1.3.2　HarmonyOS 包结构

在正式进行 HarmonyOS 应用 / 服务开发前，开发者需要掌握应用 / 服务的一些包结构和基本概念。通常，应用 / 服务的发布形态为 App Pack（Application Package），它是由一个或多个 HAP（Harmony Ability Package）包以及描述 App 属性的 pack.info 文件组成。

HAP 是 HarmonyOS 应用安装的基本单位，由编译后的代码、资源、三方库及配置文件构成，HAP 可以分为 Entry 和 Feature 两种类型，说明如下。

- Entry：应用 / 服务的主模块，可独立安装运行。对于同一类型的设备，App 可以包含一个或多个 Entry 类型的 HAP，如果同一类型的设备包含多个 Entry 模块，需要配置 distroFilter 分发规则，这样就可以在做应用的云端分发时，对不同规格的设备进行分发。
- Feature：应用 / 服务的动态特性模块。通常，一个 App 可以包含零到多个 Feature 类型的 HAP，并且只有包含 Ability 的 HAP 才能够独立运行。

HarmonyOS 应用开发分成了两种开发模型，分别是 Stage 模型和 FA 模型用。其中，FA 模型是鸿蒙早期版本就支持的模型，适合工程结构不是很复杂的应用；Stage 模型则是 API 9 新增的模型，是为了解决 FA 模型无法解决的开发场景而开发的新模型。

FA 模型与 Stage 模型最大的区别在于，FA 模型的每个应用组件独享一个虚拟机，多个应用组件共享同一个虚拟机。所以，相比 Stage 模型，FA 模型的应用程序包结构要简单许多，如图 1-3 所示。

可以看到，FA 模型将所有的资源文件、库文件和代码文件都放在 assets 文件夹中，然后在文件夹内部进一步进行区分，说明如下。

- config.json：应用配置文件，IDE 会自动生成一部分模块代码，开发者按需修改其中的配置。
- assets：HAP 所有的资源文件、库文件和代码文件集合，内部由 entry 和 js 文件夹构成。entry 文件夹存放的是资源文件和 resources.index 文件。
- resources：用于存放应用的资源文件，如字符串、图片等。
- resources.index：资源索引表，由 IDE 调用 SDK 工具自动生成。
- js：文件夹中存放的编译后的代码文件。
- pack.info：Bundle 中用于描述每个 HAP 属性的文件，由 IDE 工具生成 Bundle 包时自动生成。

Stage 模型基于 FA 模型的基础概念演化而来，开发者能够使用它开发出分布式场景下的复杂应用，Stage 模型应用包结构如图 1-4 所示。

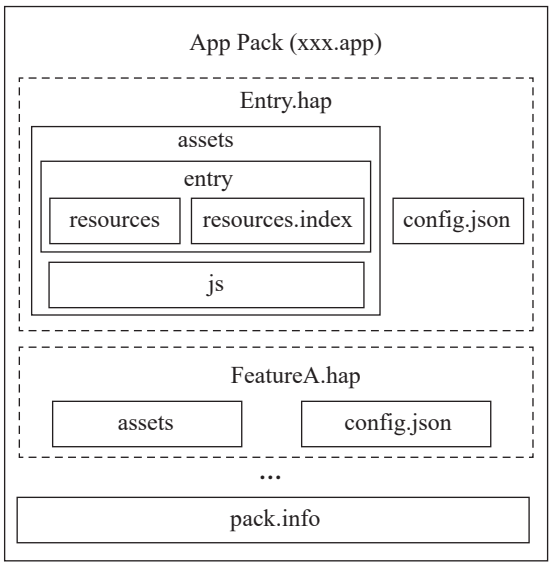

图 1-3　FA 模型应用包结构

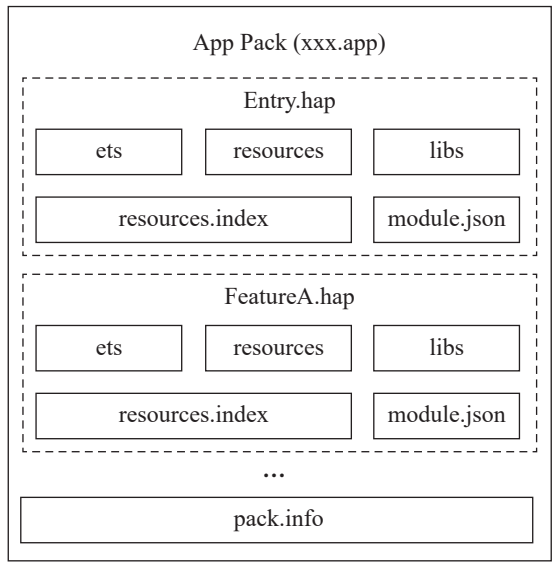

图 1-4　Stage 模型应用包结构

可以看到，每个 HarmonyOS 应用程序都可以包含多个 HAP 包文件，而 HAP 则由 ets、libs、resources 等文件夹和 resources.index、module.json、pack.info 等配置文件构成，说明如下。

- ets：用于存放应用代码编译后的字节码文件。
- libs：用于存放库文件，库文件是 HarmonyOS 应用依赖的第三方代码。
- resources：用于存放应用的资源文件，如图片、字符串和颜色等。
- resources.index：资源索引表，由 IDE 编译工程时生成。
- module.json：HAP 配置文件，内容由工程配置中的 module.json5 和 app.json5 组成，IDE 会自动生成一部分默认配置，开发者按需修改其中的默认配置。
- pack.info：用于描述每个 HAP 属性的文件，如应用的 bundleName 和 versionCode 信息、模块的 name、type 和 abilities 等信息。

需要特别说明的是，从 API 10 开始，HarmonyOS 将逐步弃用 FA 模型，并主推 Stage 模型，因为相较于 FA 模型，Stage 模型目录管理更加简单方便。

1.3.3　共享包

OpenHarmony 提供了两种共享包，分别是 HAR（Harmony Archive）静态共享包和 HSP（Harmony Shared Package）动态共享包。

HAR 与 HSP 都是为了实现代码和资源的共享，都包含有代码、C++ 库、资源和配置文件。二者最大的不同之处在于，HAR 中的代码和资源跟随使用方编译，如果有多个使用方，它们的编译产物中会存在多份相同“拷贝”。而 HSP 中的代码和资源可以独立编译，

运行时在一个单独的进程中，代码也只会存在一份，HAR 和 HSP 在 App 包中的形态如图 1-5 所示。

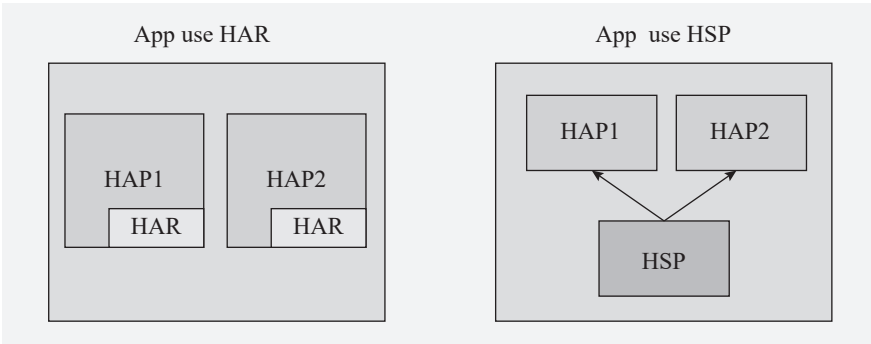

图 1-5　HAR 和 HSP 在 App 包中的形态

HAR 是 HarmonyOS 开发中的静态共享包，作用类似于 Android 开发中的 aar 包。HAR 包含代码、C++ 库、资源和配置文件，HAR 可以实现多个模块或多个工程之间 ArkUI 组件、资源等代码的共享。不同于 HAP，HAR 不能独立安装运行在设备上，只能作为应用模块的依赖项被引用。

HSP 是 HarmonyOS 开发中的动态共享包，目的是解决大型项目中多个 HAP 无法共享同一公共资源代码的问题，作用类似于 Android 开发中的本地 library 模块。HSP 只能在应用内部其他的 HAP 或 HSP 使用，实现应用内部代码和资源的共享。同时，应用内 HSP 会跟随其宿主应用的 App 包一起发布，与该宿主应用具有相同的包名和生命周期。

需要说明的是，在应用内使用 HSP 时，有以下几点约束。

- HSP 及其使用方都必须是 Stage 模型。
- HSP 及其使用方都必须使用 esmodule 编译模式。
- HSP 不支持在配置文件中声明 abilities、extensionAbilities 标签。