HCIA-HarmonyOS Device Developer V1.0 培训教材

HUAWEI

目录

HarmonyOS 介绍

HUAWEI

前言

- HarmonyOS是一款面向万物互联时代的、全新的分布式操作系统。
- 在传统的单设备系统能力的基础上，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，能够支持手机、平板、智能穿戴、智慧屏、车机等多种终端设备。
- 本章主要介绍HarmonyOS分布式操作系统的概念、关键技术与能力以及HarmonyOS典型的应用场景。

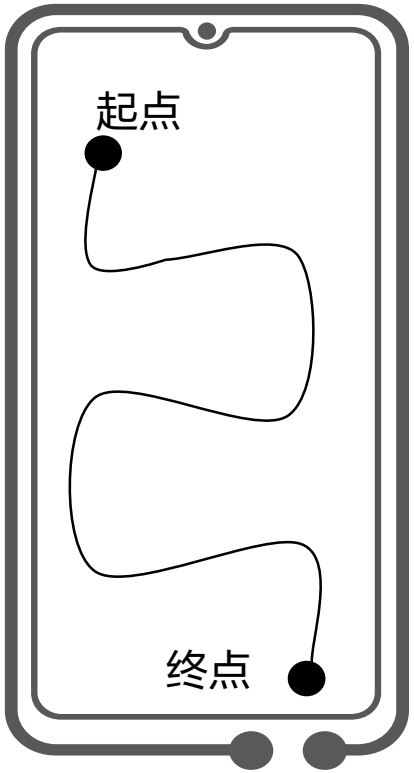
目标

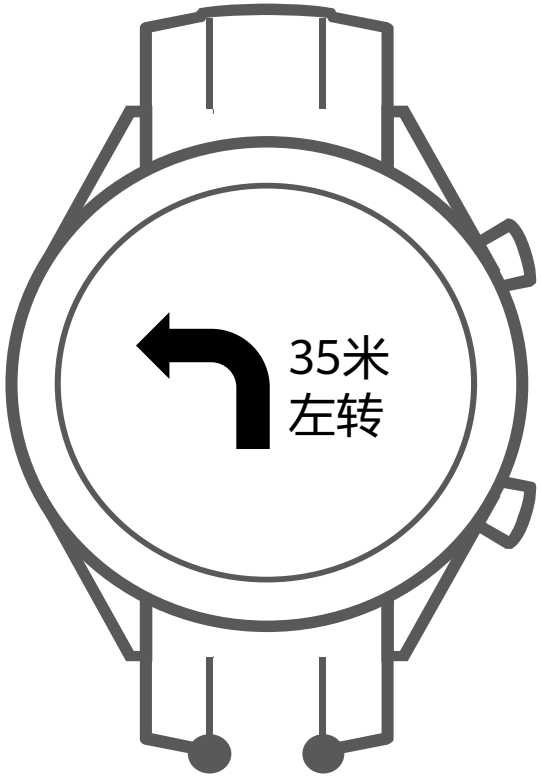
- 学习完本课程后，您将能够：
  - 了解HarmonyOS的相关概念和产品定位；
  - 了解HarmonyOS的技术架构；
  - 了解HarmonyOS的关键特性。

目录

HarmonyOS小场景 - 导航信息流转

- HarmonyOS与地图应用深度融合，实现手机+手表的无缝导航信息流转。





讨论：场景中需要实现哪些功能点?

- 请试着讨论并分析在HarmonyOS的多终端信息流转中，涉及到哪些关键点技术需要去实现?
- 信息的传递协议：
  - Wi-Fi、蓝牙、移动网络等；
- 信息的精准传递：
  - 设备绑定、信息认证等；
- 场景切换的判断：
  - 是否流转的判定条件；
- ……

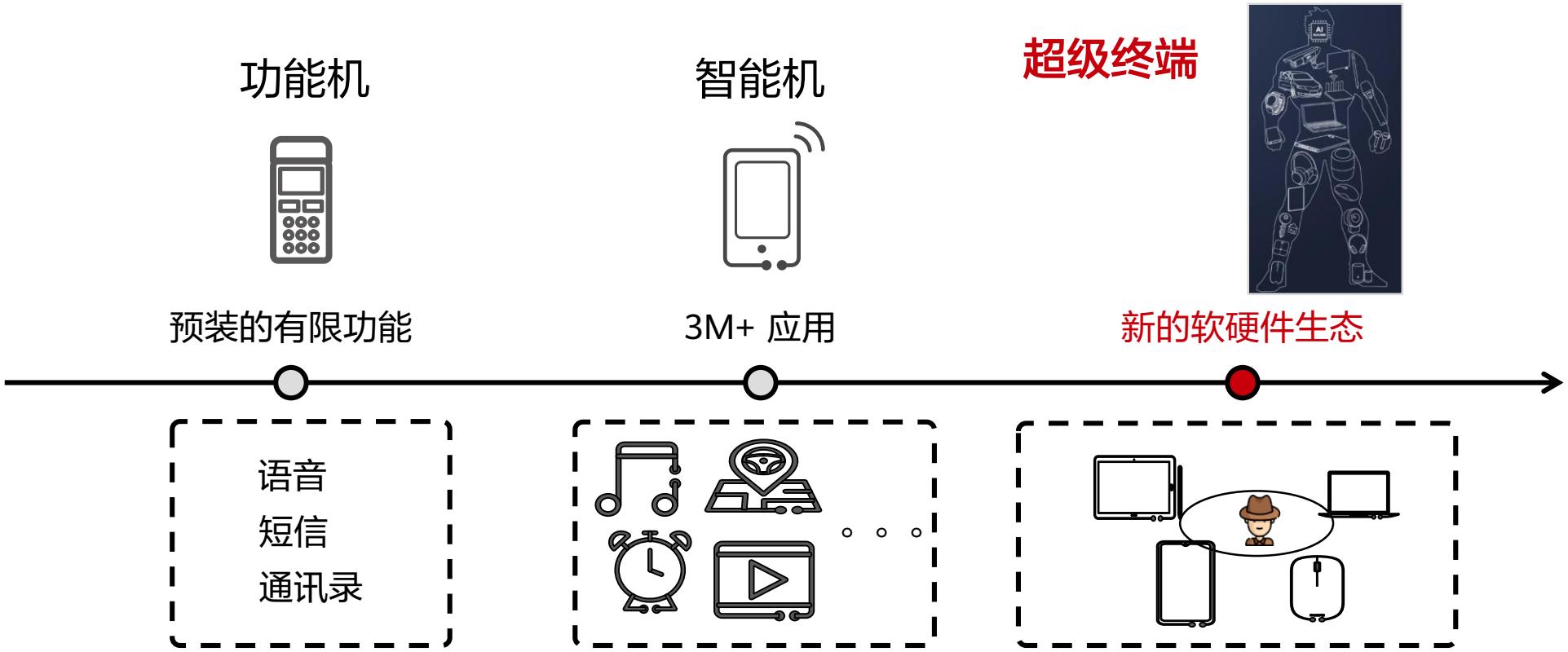
初识HarmonyOS

- HarmonyOS是全场景分布式智慧系统。

注：登录https://developer.harmonyos.com/cn/home/开发者网站，点击“HarmonyOS 概述”即可获取。

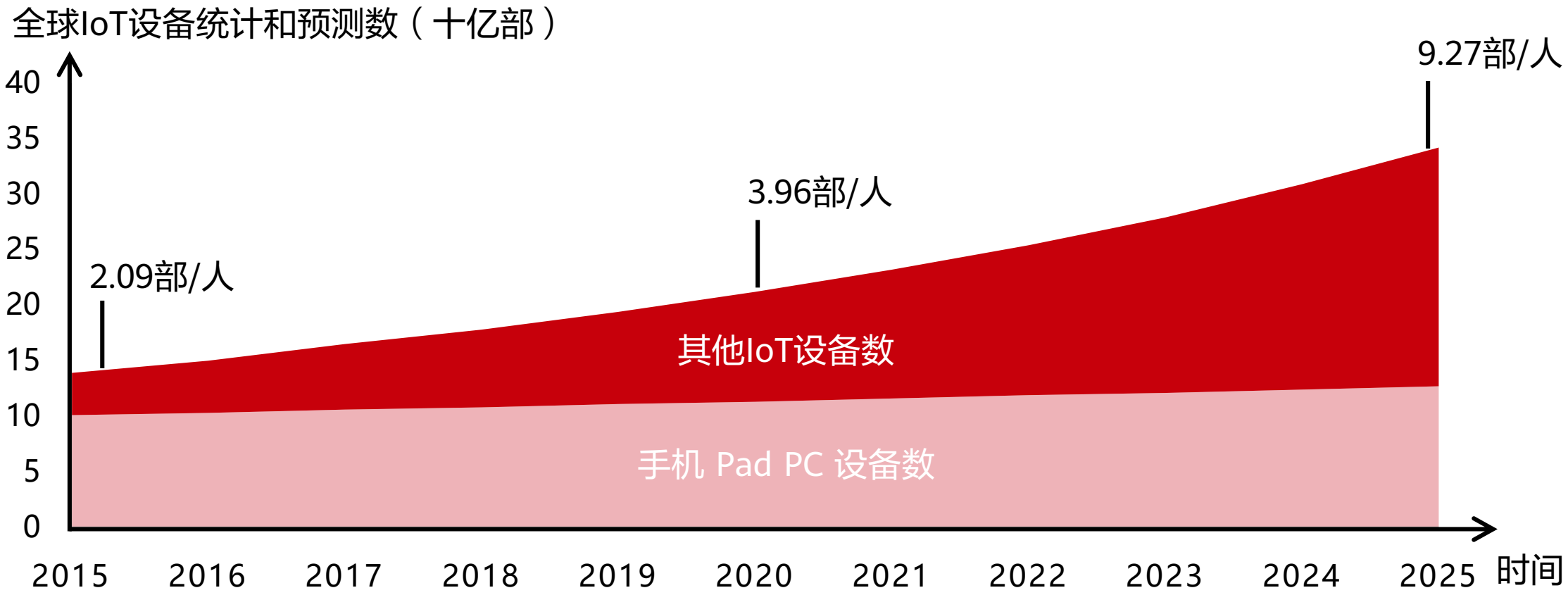
HarmonyOS简介

- HarmonyOS是一款面向万物互联时代的、全新的分布式操作系统。



HarmonyOS系统定位 - 万物互联的操作系统

- HarmonyOS是一款面向万物互联的操作系统。

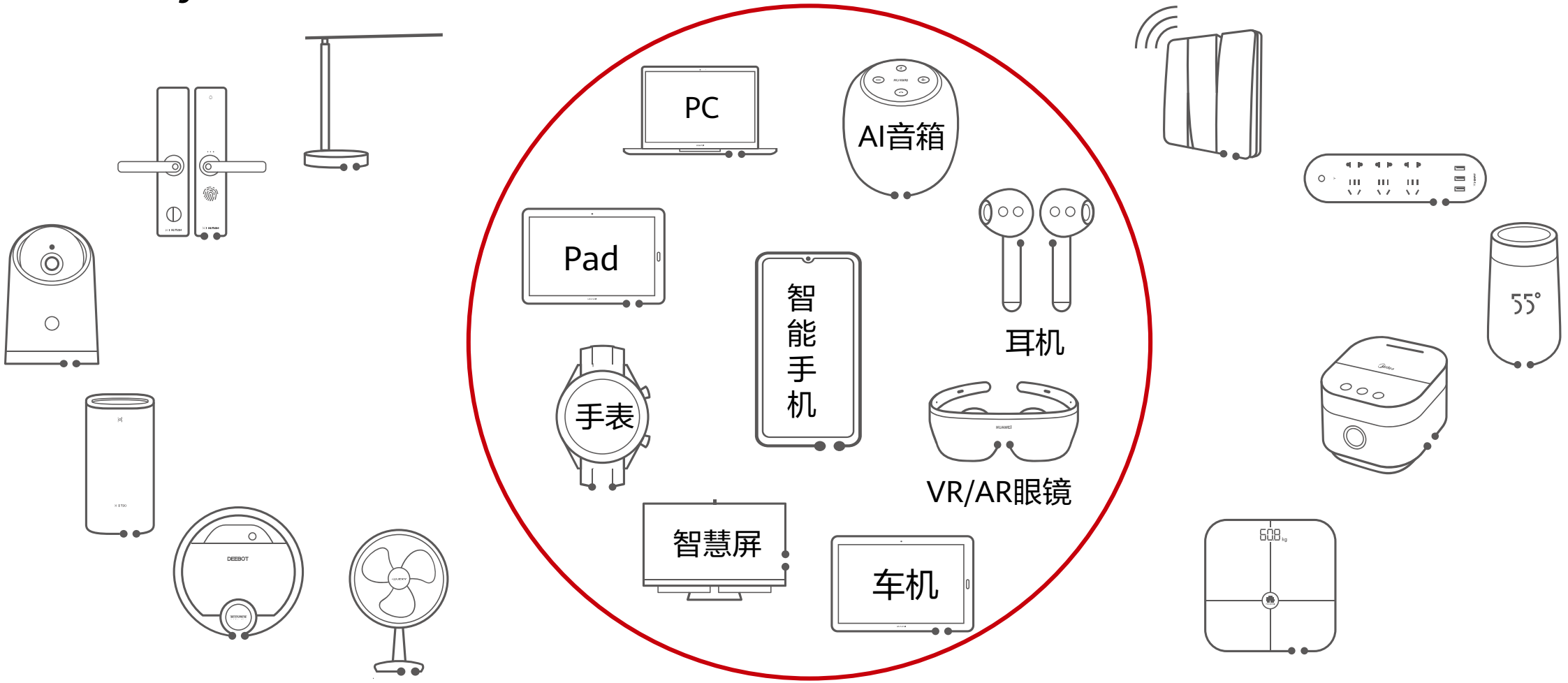


数据来源: iot-analytics.com,www.researchgate.net

目录

HarmonyOS应用场景总览

- HarmonyOS以手机为核心，构建1+8+N全场景应用。



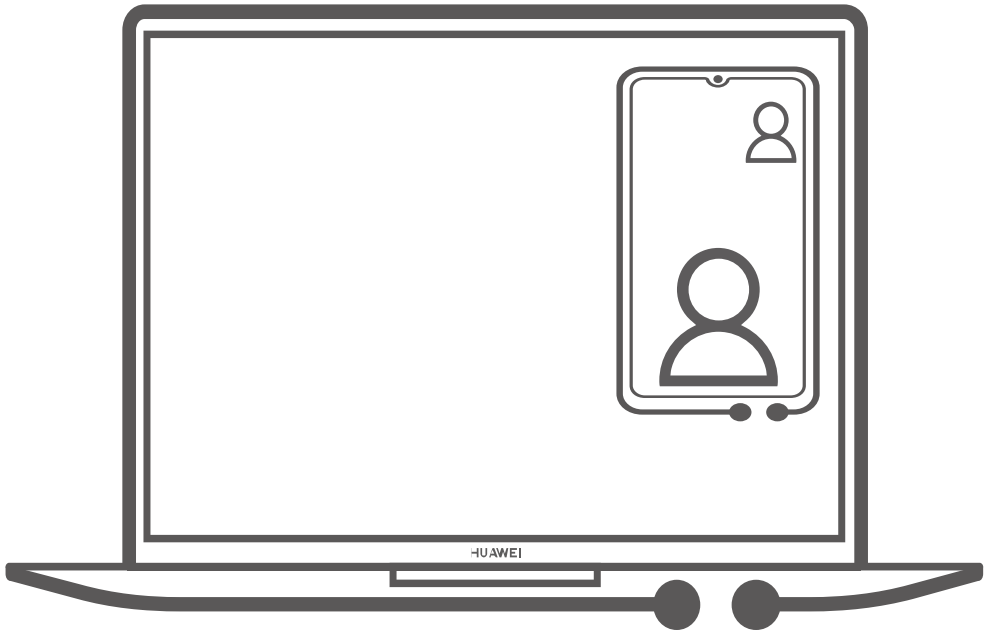
HarmonyOS典型应用场景 - 全新办公模式

- PC+手机构建具有强大通信能力、高效人机交互和丰富应用生态的新设备。

便携移动性强

丰富应用生态

强大通信能力



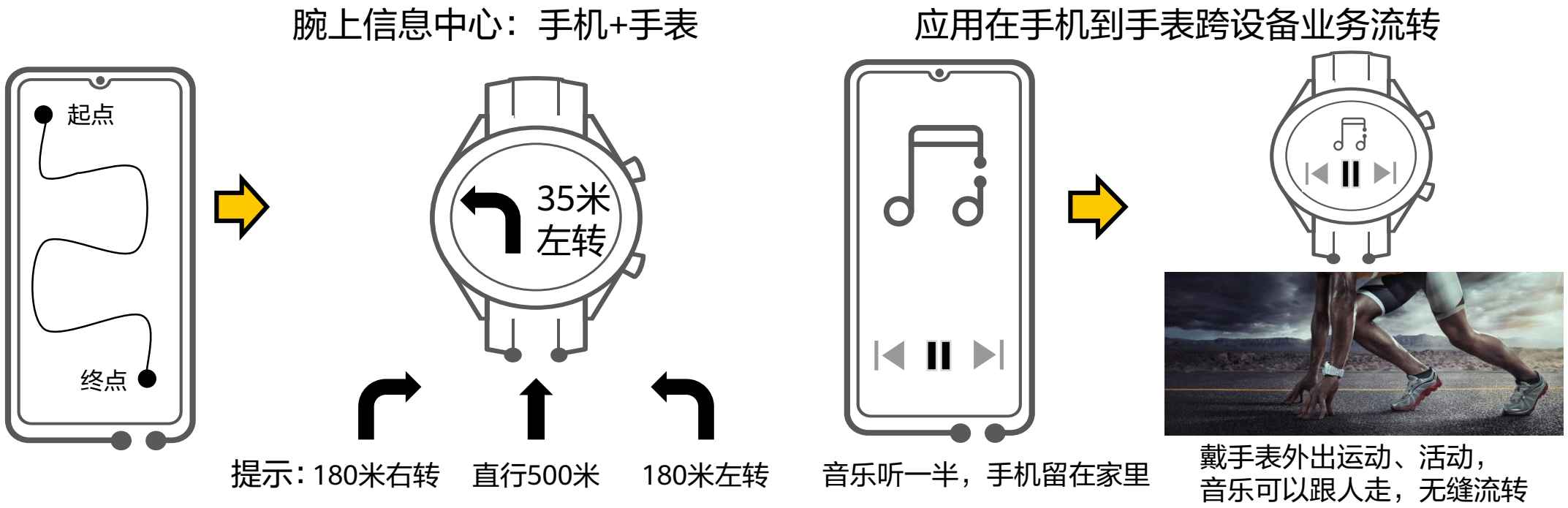
键鼠：高效输入

大屏：高效显示

应用：高效处理

HarmonyOS典型应用场景 - 跨设备信息流转

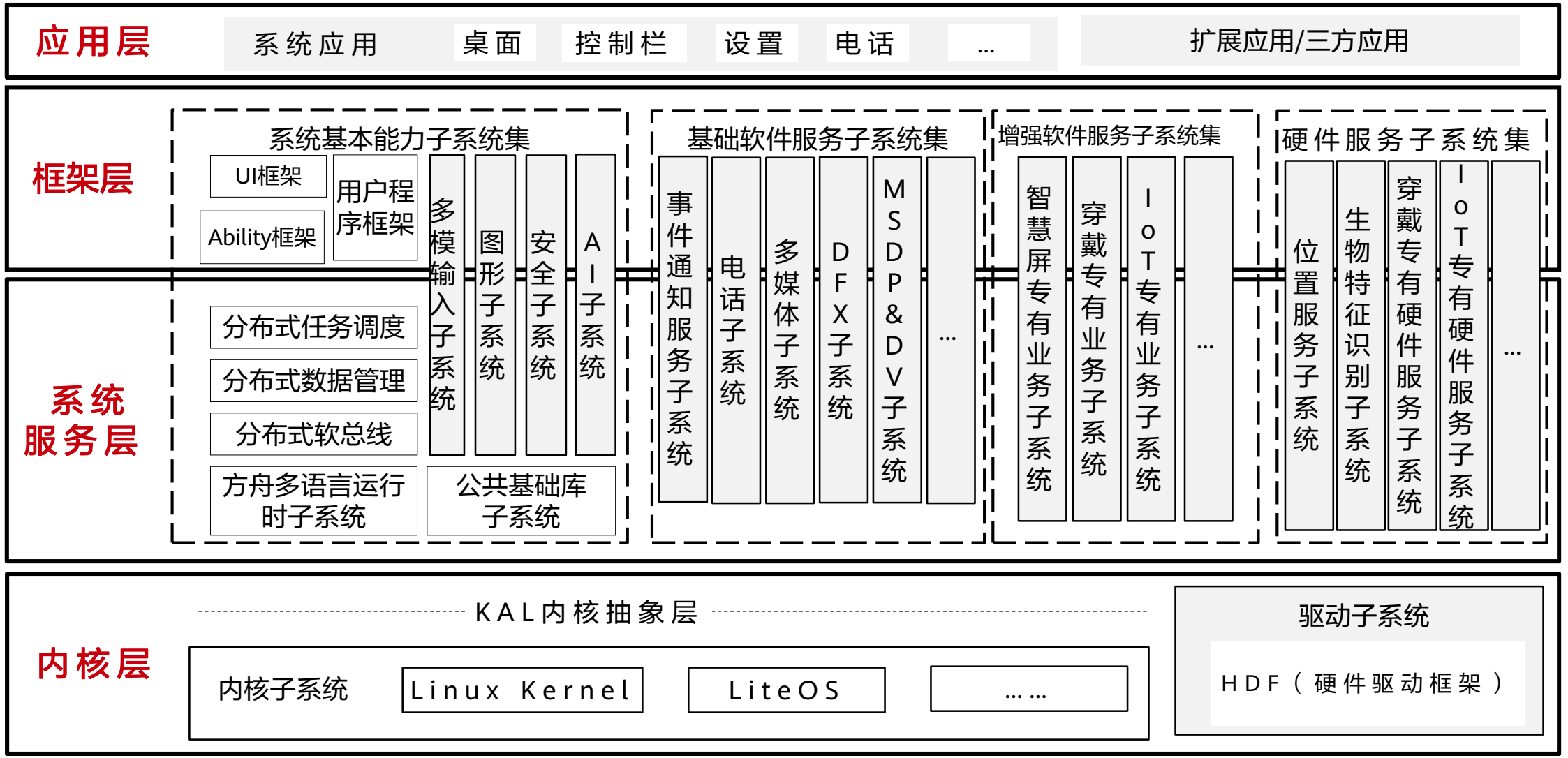
- 手机到手表跨设备业务流转，开创新的生活方式。



备注：展示图片非实际效果图，仅作示意

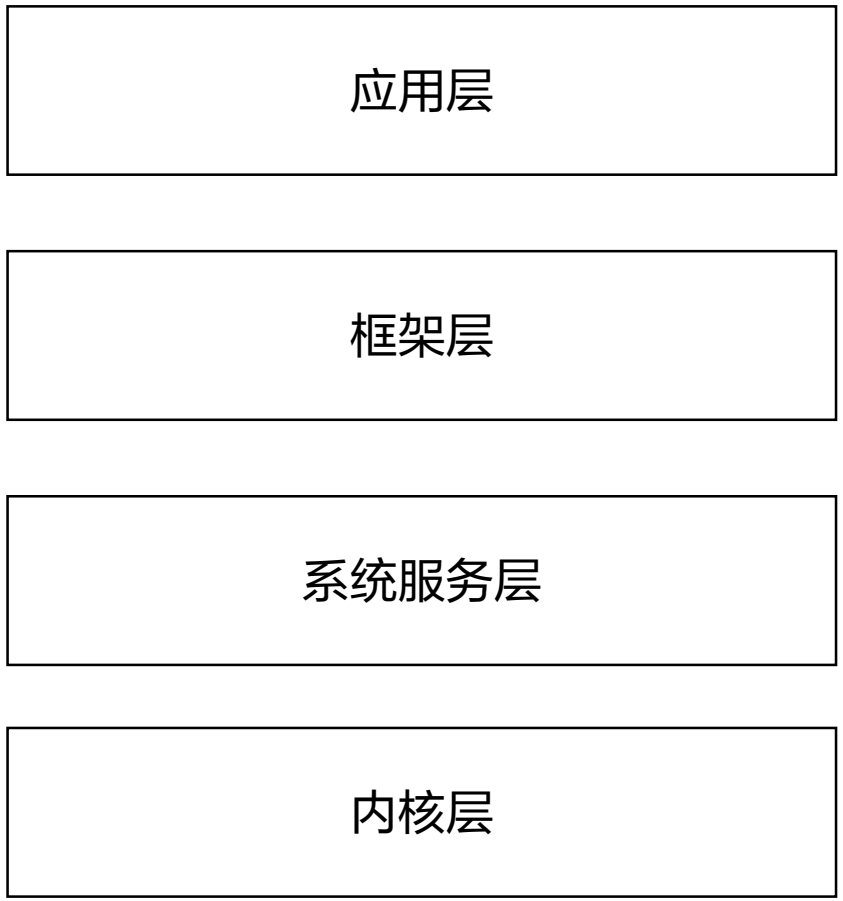
目录

HarmonyOS架构图



HarmonyOS架构解析

- HarmonyOS整体遵从分层设计，从下向上依次为：内核层、系统服务层、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。

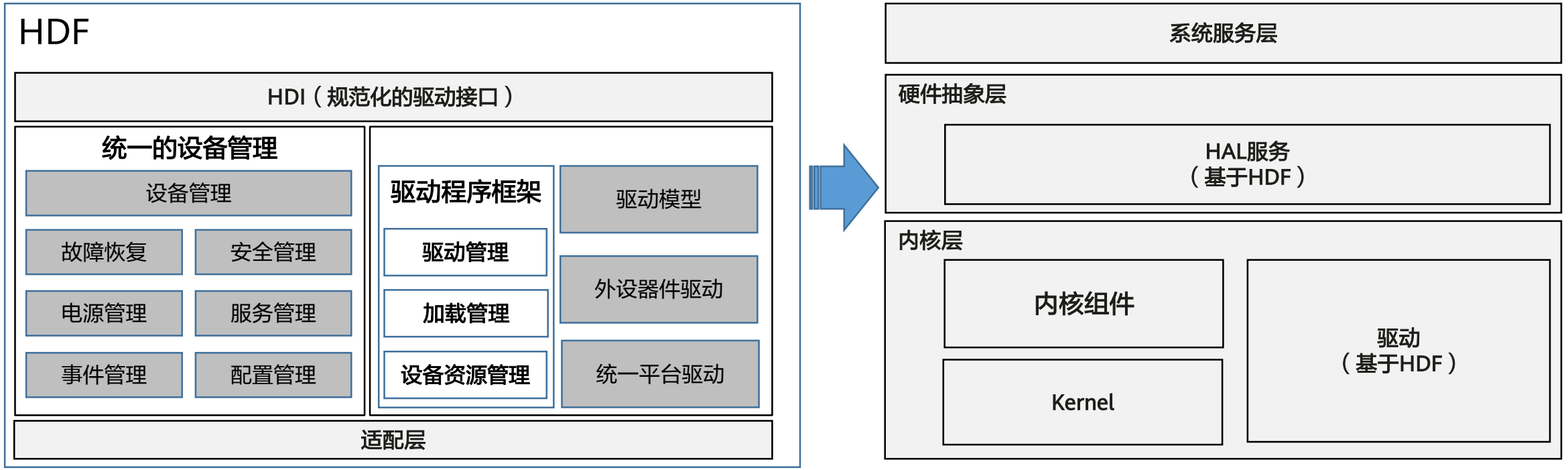


HarmonyOS架构解析 - 内核层

- 内核层主要包括内核子系统和驱动子系统两个部分。
  - 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(KAL，Kernel Abstract Layer)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：硬件驱动框架(HDF)是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。

统一驱动框架：驱动与内核解耦，支持运行动态加载，让更多IoT设备接入超级终端

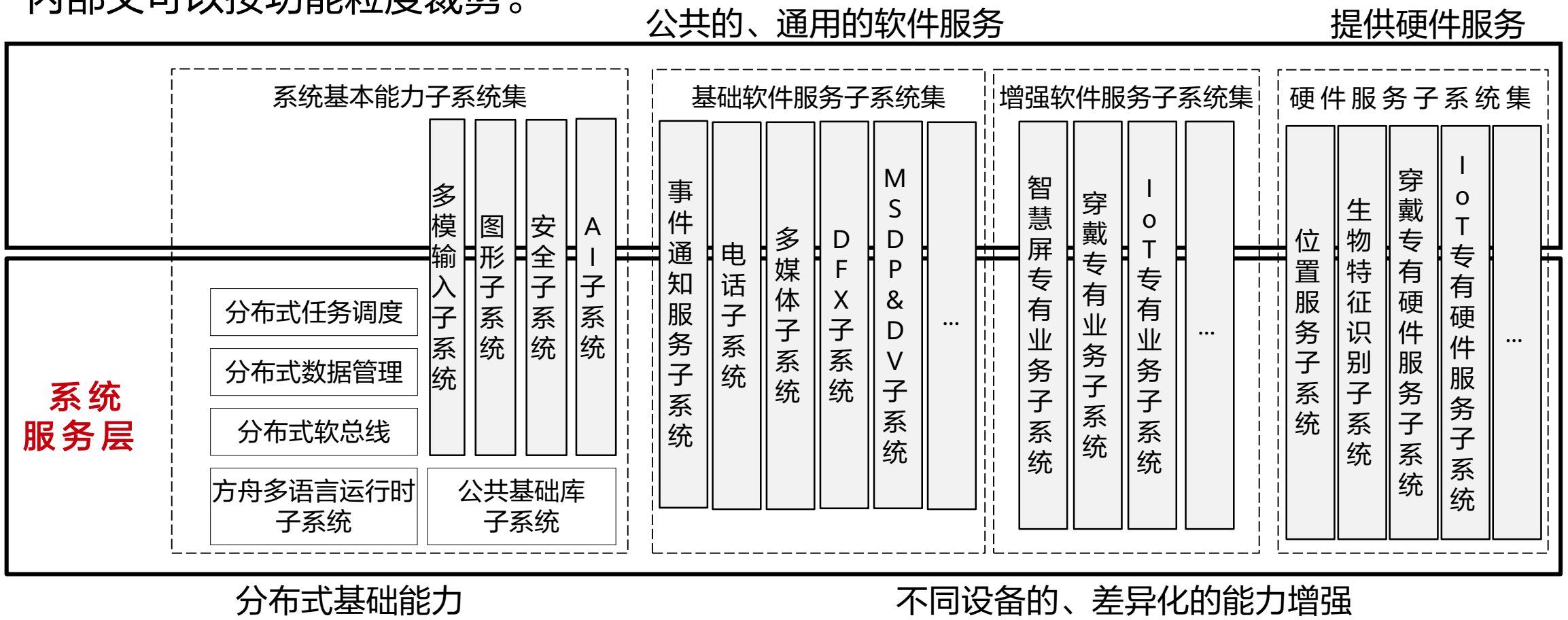
- 通过平台、系统接口解耦的构建统一的驱动平台底座兼容如Linux、LiteOS等不同内核；
- 支撑百K级~G级容量的1+8+N设备部署；
- 根据不同设备形态，支持用户态部署和内核态部署。



HUAWEI

HarmonyOS架构解析 - 系统服务层

- 根据不同设备形态的部署环境，各个子系统集内部可以按子系统粒度裁剪，每个子系统内部又可以按功能粒度裁剪。



HarmonyOS架构解析 - 框架层

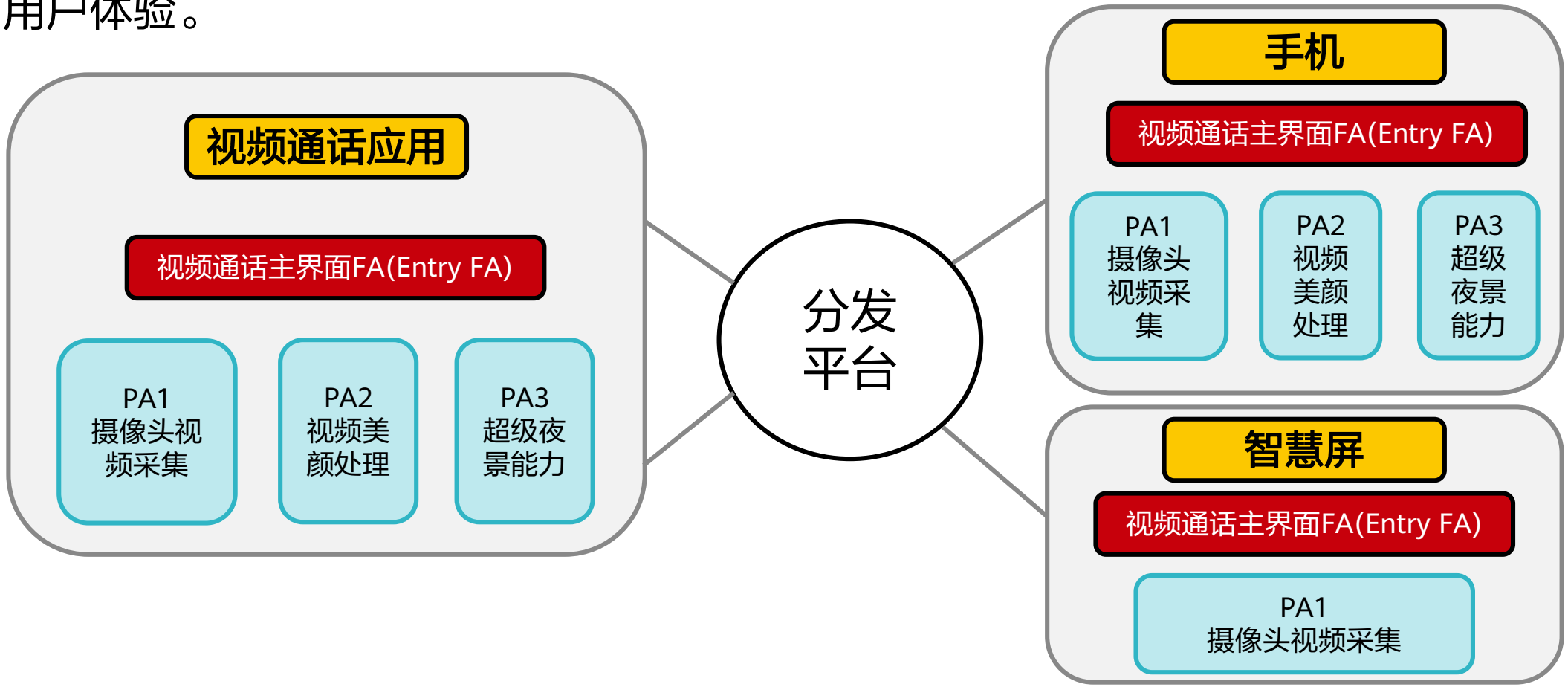
- 框架层为HarmonyOS应用开发提供：
  - 用户程序框架：支持Java/C/C++/JS等多种语言；
  - Ability框架：应用所具备能力的抽象；
  - 两种UI框架：适用于Java语言的Java UI框架和适用于JS语言的JS UI框架；
  - 多语言框架API：支持多种软硬件服务对外开放的语言框架。
- 根据系统的组件化裁剪程度，HarmonyOS设备支持的API也会有所不同。

HarmonyOS架构解析 - 应用层

- 应用层包括系统应用和扩展/第三方非系统应用。
- HarmonyOS的应用分为一个或多个FA(Feature Ability)或PA(Particle Ability)组成。
  - FA有UI界面，提供与用户交互的能力；而PA无UI界面，提供后台运行任务的能力以及统一的数据访问抽象。
  - FA在进行用户交互时所需的后台数据访问也需要由对应的PA提供支撑。
  - 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。

HarmonyOS应用服务智能分发

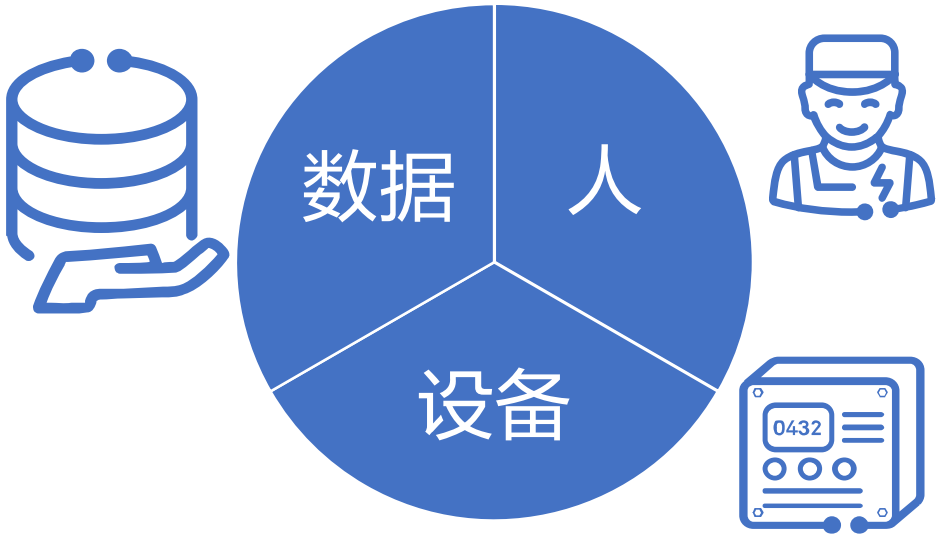
- 基于FA/PA构建的新型应用生态，能够实现三方服务跨设备智能分发，提供一致、高效的用户体验。



目录

HarmonyOS系统安全

- 在搭载HarmonyOS的分布式终端上，可以保证“正确的人，通过正确的设备，正确地使用数据”。
  - 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、用户隐私不泄露的前提条件。HarmonyOS通过以下三个方面来实现协同身份认证：

- 通过将硬件和认证能力解耦（即信息采集和认证可以在不同的设备上完成），来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。

- 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（例如，对安防设备的操作）时，HarmonyOS 会对用户进行身份认证，确保其身份的可靠性。

- 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，用于标识一个用户，来提高认证的准确度。

正确的设备

- 在分布式终端场景下，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虚拟终端上得到有效保护，避免用户隐私泄露。HarmonyOS通过以下三个方面来确保设备可靠：

- 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。

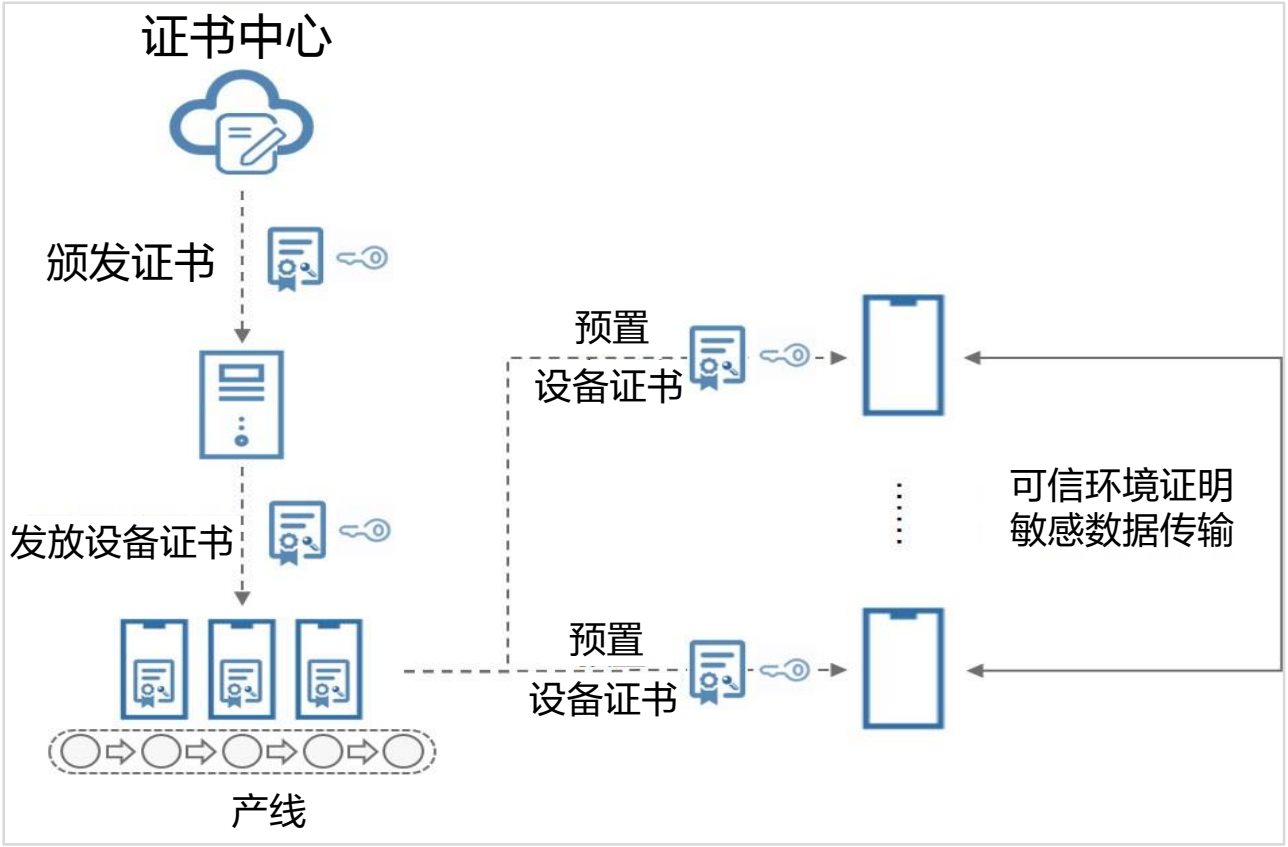
- 确保源头每个虚拟设备运行的系统固件和应用程序是完整的、未经篡改的。
- 通过安全启动，各个设备厂商的镜像包就不易被非法替换为恶意程序，从而保护用户的数据和隐私安全。

- 提供了基于硬件的可信执行环境(TEE)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。

设备证书使用示意图

- 在必须传输用户的敏感数据（如密钥、加密的生物特征等信息）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，实现安全传输。

  - 对于有TEE环境的设备，通过预置PKI设备证书给设备身份提供证明，确保设备是合法制造生产的；
  - 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。



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，从而保证个人数据与隐私、以及系统的机密数据（如密钥）不泄漏。

- 对数据分类分级，根据分类设置相应的保护等级。
- 通过硬件为设备提供可信执行环境。
- 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。

- 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。
- 建立安全的连接通道和安全的加密传输通道。

思考题

1. （多选题）HarmonyOS系统主要分为（　　）。

   A. 内核层

   B. 系统服务层

   C. 框架层

   D. 应用层

2. （多选题）HarmonyOS的UI框架支持以下哪些语言?

   A. JAVA

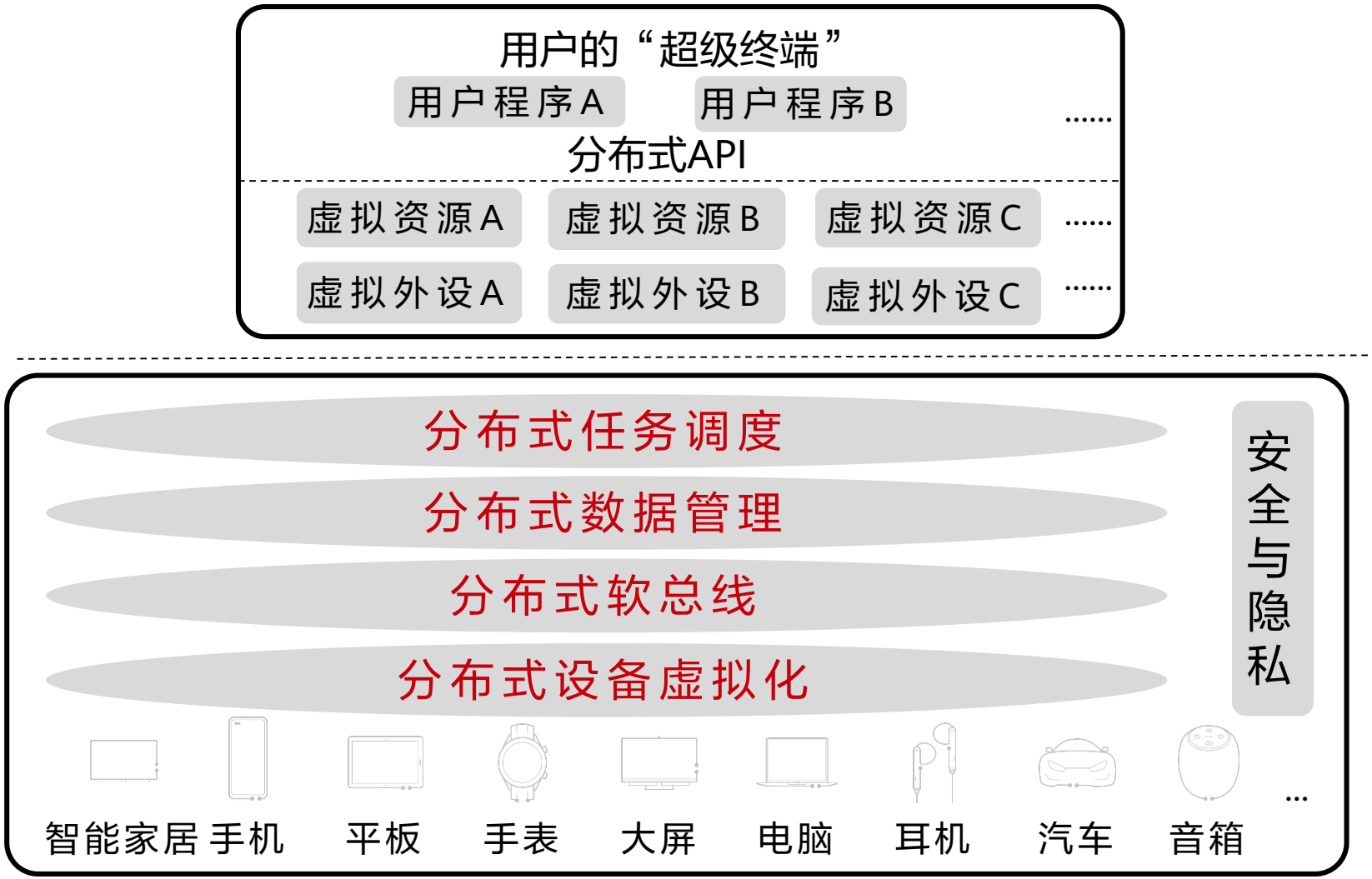
   B. JS

   C. PHP

   D. Python

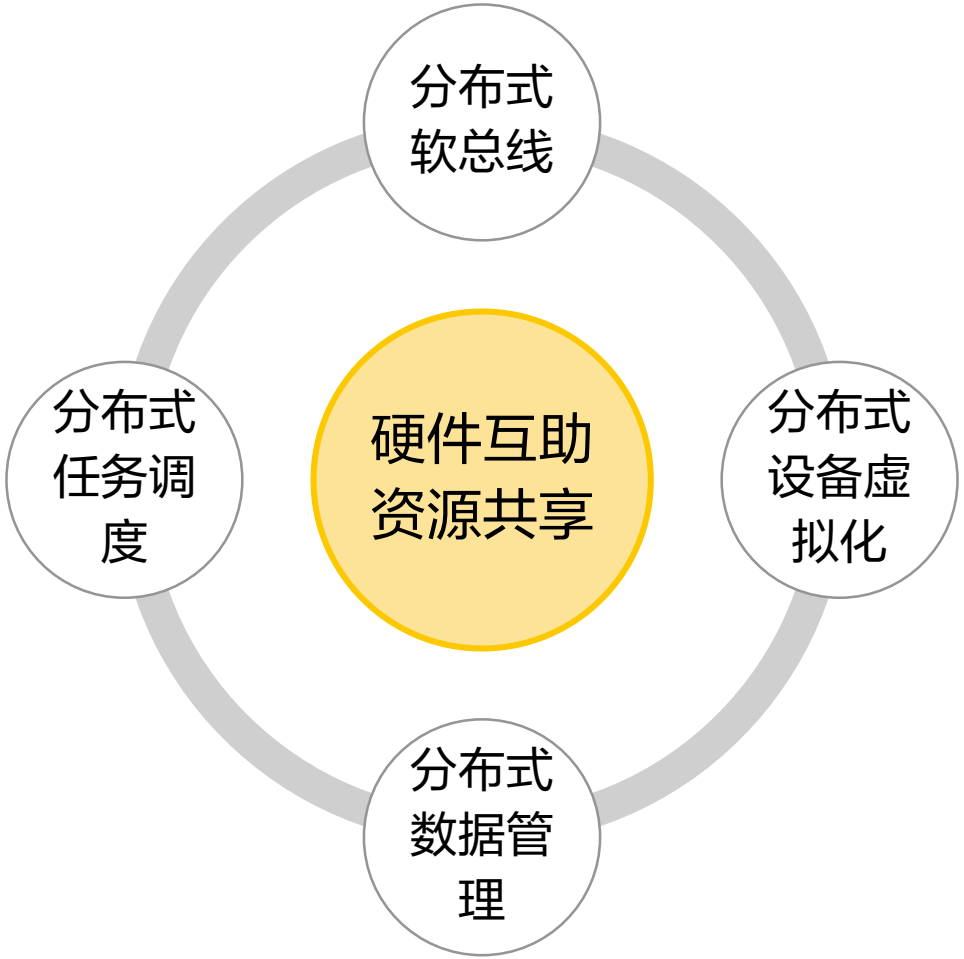
目录

“超级终端”概览图



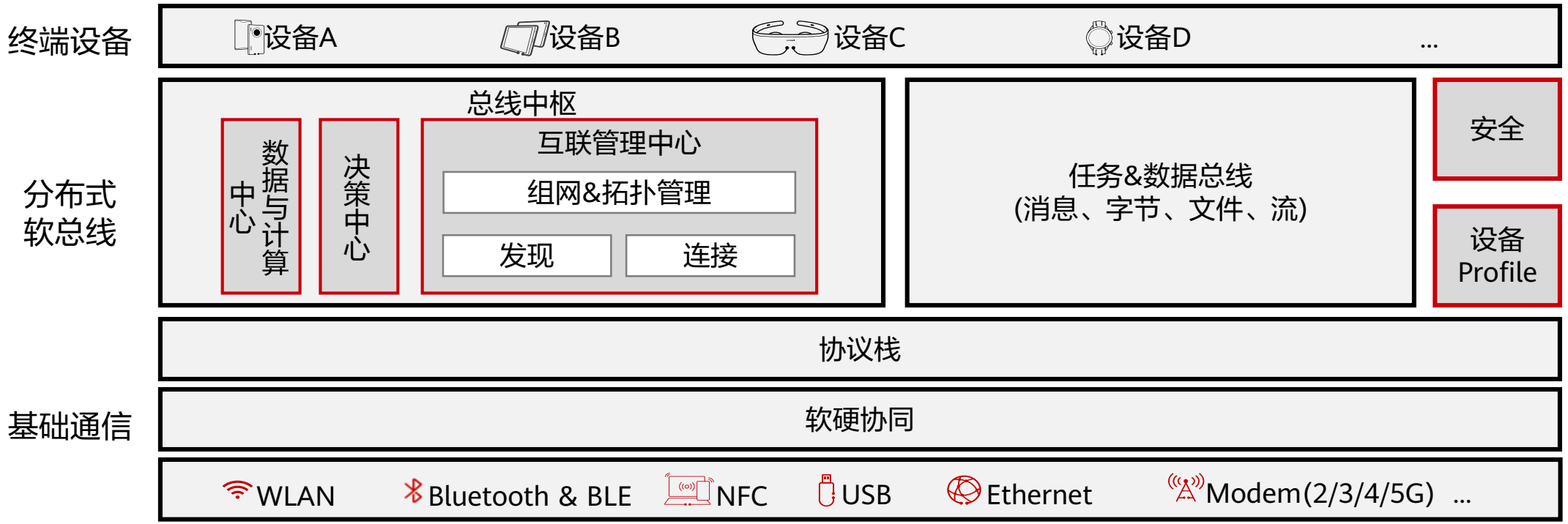
硬件互助，资源共享关键技术

- 多种设备之间能够实现硬件互助、资源共享，主要依赖以下四个关键分布式技术。



分布式软总线

- 分布式软总线是手机、平板、智能穿戴、智慧屏、车机等分布式设备的通信基座，为设备之间的互联互通提供了统一的分布式通信能力，为设备之间的无感发现和零等待传输创造了条件。开发者只需聚焦于业务逻辑的实现，无需关注组网方式与底层协议。

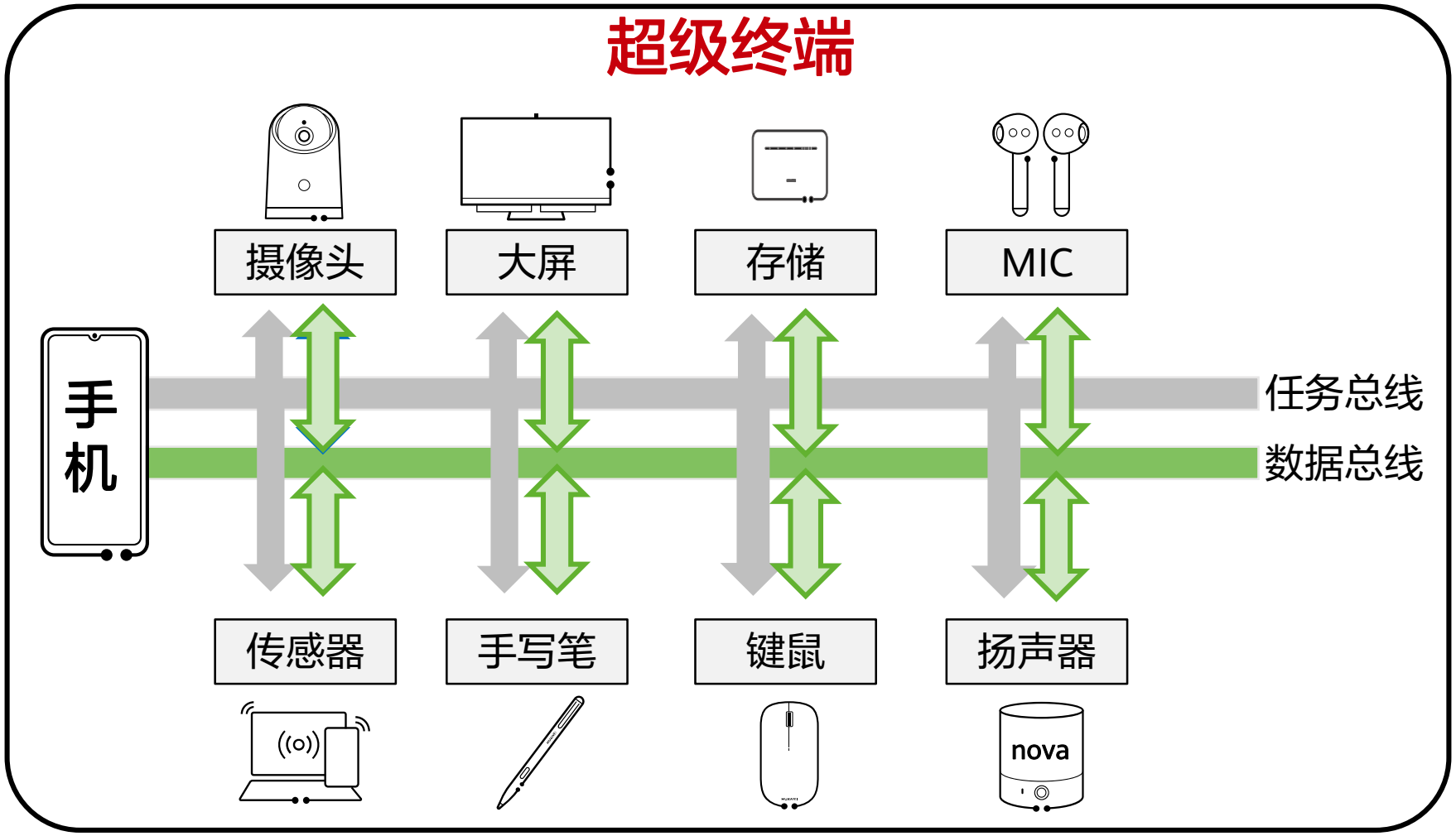


不同设备组成超级终端的核心基础能力，无感自组网

全连接组网

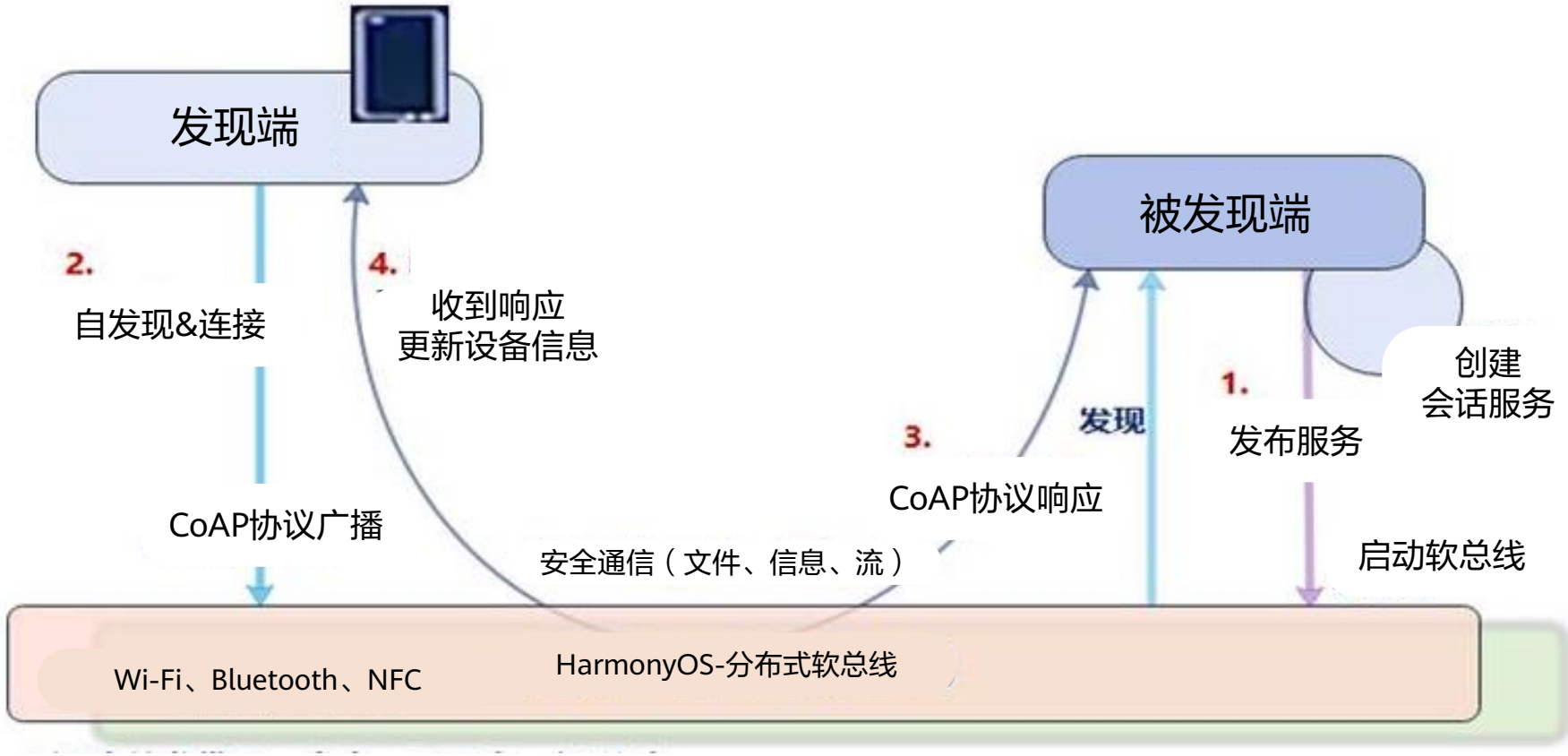
自发现

极速传输



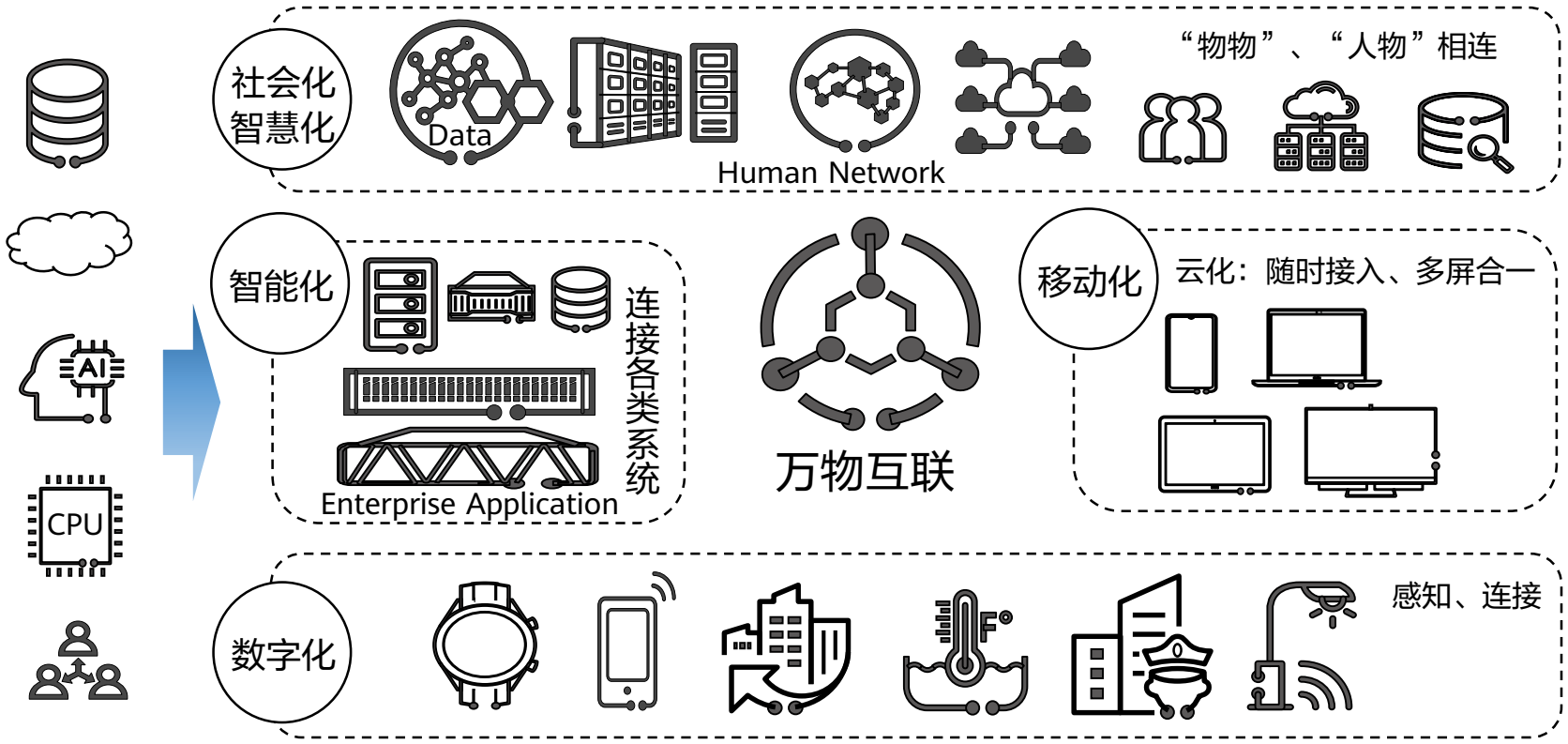
分布式软总线 - 自发现&连接

- 分布式软总线提出自动发现设备，实现用户零等待的自发现体验，附近同账号的设备自动发现无需等待，自动安全连接。



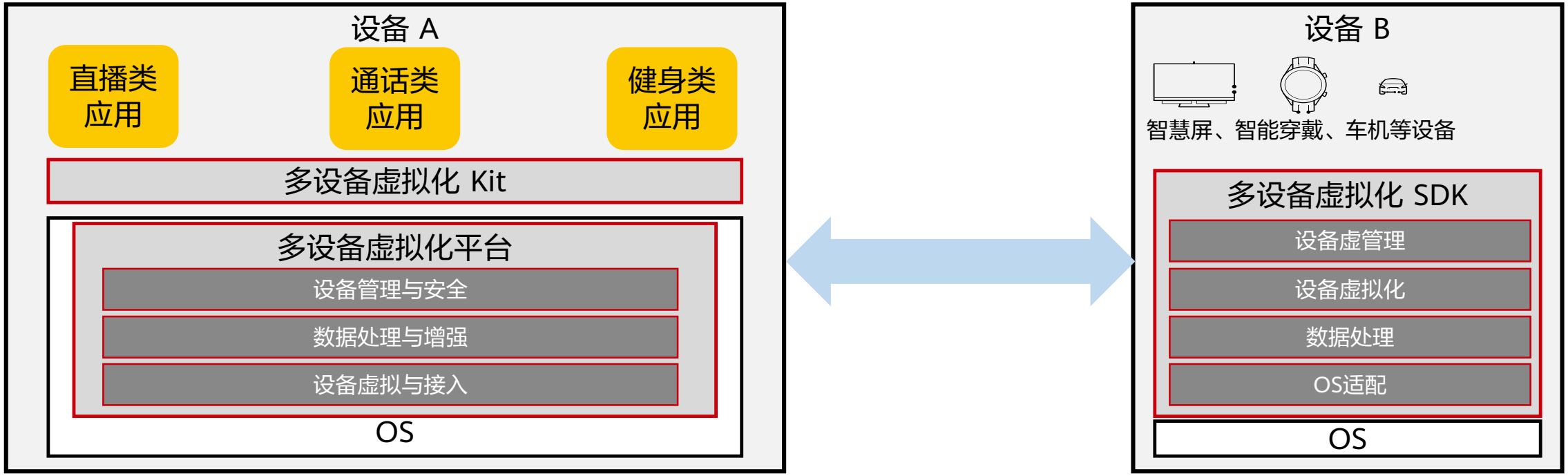
分布式软总线 - 多设备互联、组网

- 分布式软总线提出了异构网络组网，自动构建一个逻辑全连接网络，以解决设备间不同协议交互的问题。



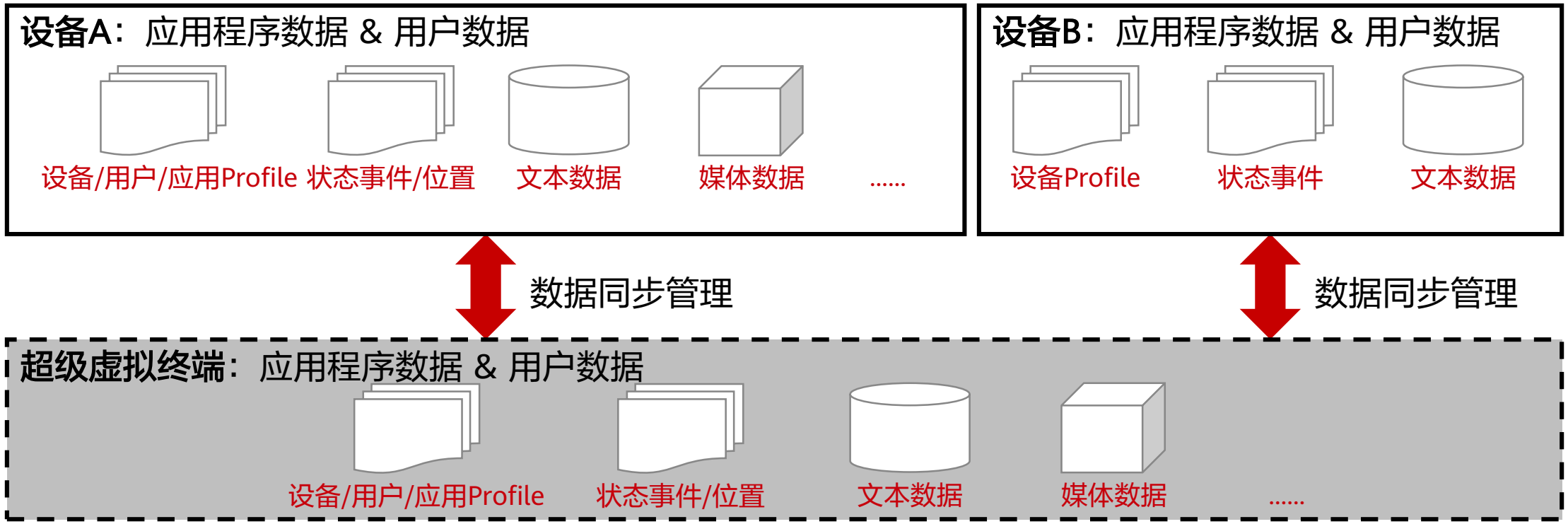
分布式设备虚拟化

- 分布式设备虚拟化平台可以实现不同设备的资源融合、设备管理、数据处理，多种设备共同形成一个超级虚拟终端。
- 针对不同类型的任务，为用户匹配并选择能力合适的执行硬件，让业务连续地在不同设备间流转，充分发挥不同设备的能力优势，如显示能力、摄像能力、音频能力、交互能力以及传感器能力等。

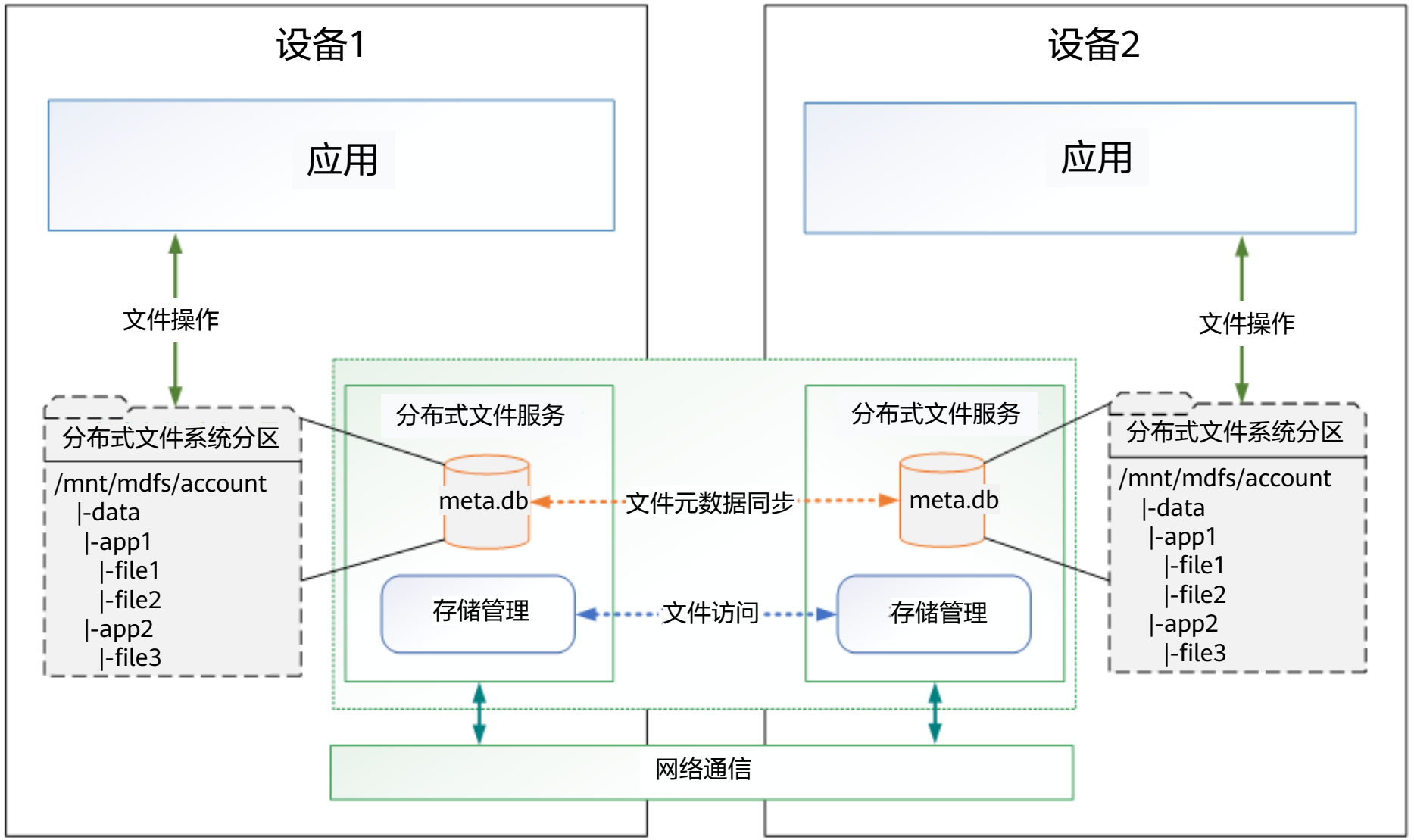


分布式数据管理

- 用户数据不再与单一物理设备绑定，业务逻辑与数据存储分离，跨设备的数据处理如同本地数据处理一样方便快捷，让开发者能够轻松实现全场景、多设备下的数据存储、共享和访问，为打造一致、流畅的用户体验创造了基础条件。

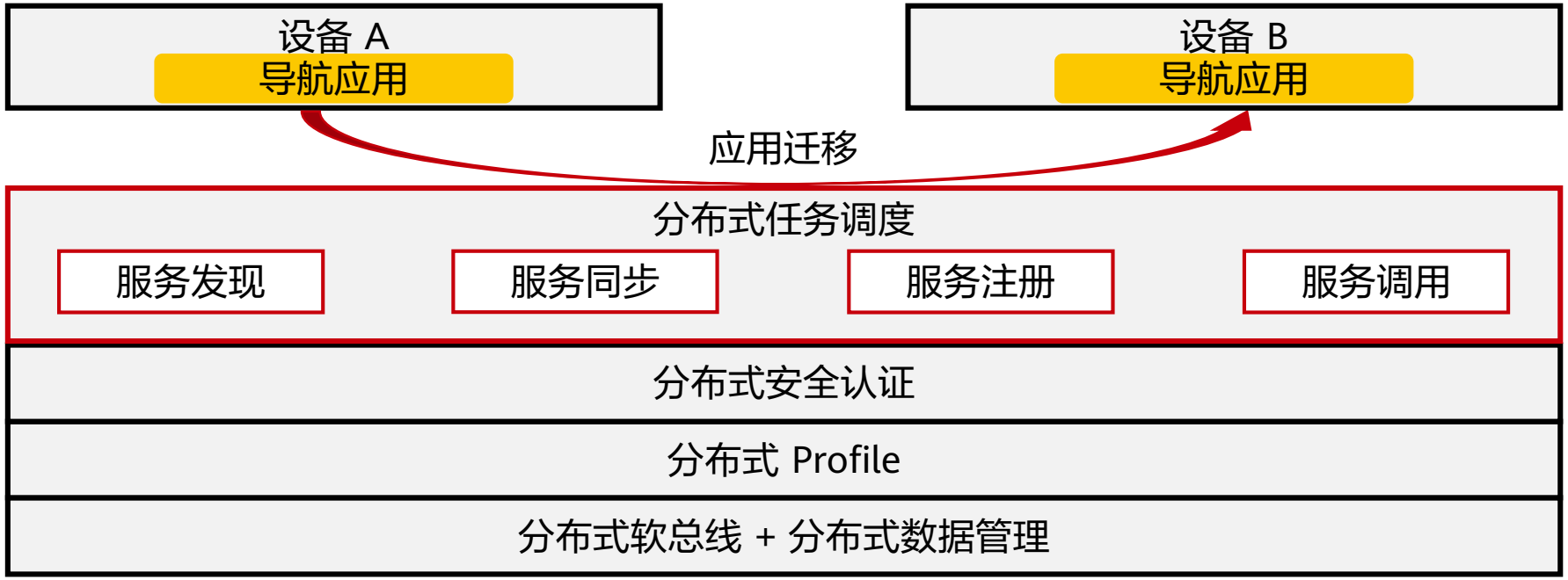


分布式文件管理



分布式任务调度

- 分布式任务调度基于分布式软总线、分布式数据管理、分布式Profile等技术特性，构建统一的分布式服务管理（发现、同步、注册、调用）机制，支持对跨设备的应用进行远程启动、远程调用、远程连接以及迁移等操作，能够根据不同设备的能力、位置、业务运行状态、资源使用情况，以及用户的习惯和意图，选择合适的设备运行分布式任务。



多设备信息融合，体验智能化健身模式

讨论：

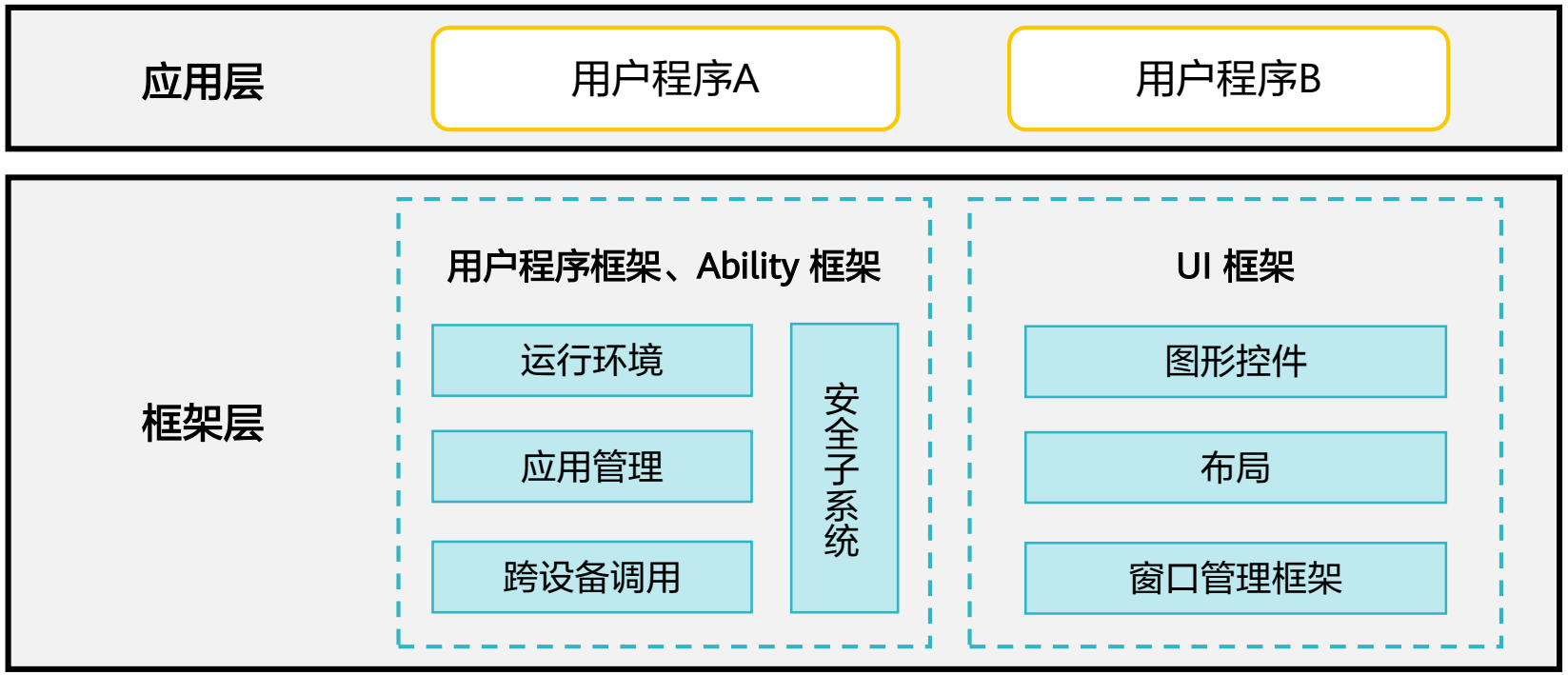
该场景中主要应用了哪些HarmonyOS的分布式技术?

注：登录https://developer.huawei.com/consumer/cn/training/，通过HarmonyOS->HarmonyOS先行者，查看视频。

目录

一次开发，多端部署

- HarmonyOS提供了用户程序框架、Ability框架以及UI框架，支持应用开发过程中多终端的业务逻辑和界面逻辑进行复用，能够实现应用的一次开发，多端部署，提升了跨设备应用的开发效率。
- 采用业界主流设计方式，提供多种响应式布局方案，支持栅格化布局，满足不同屏幕的界面适配能力。



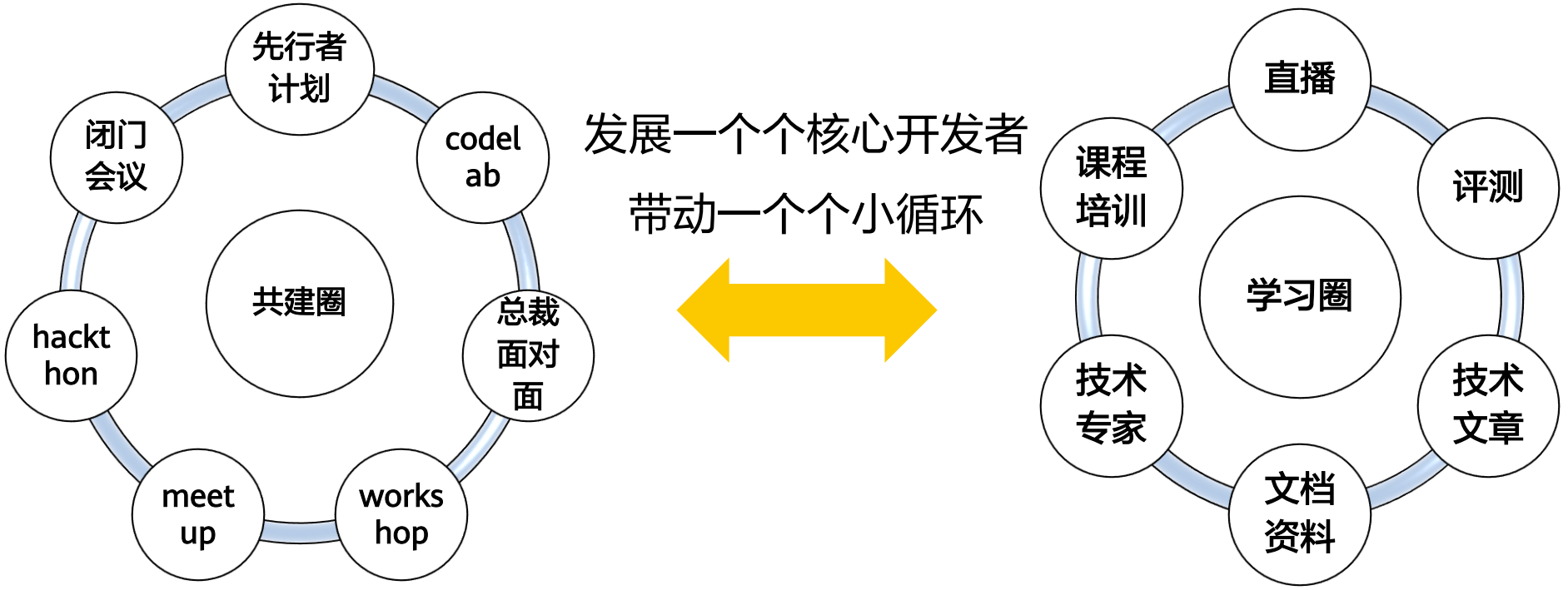
目录

统一OS，弹性部署

- HarmonyOS通过组件化和小型化等设计方法，支持多种终端设备按需弹性部署，能够适配不同类别的硬件资源和功能需求。支撑通过编译链关系去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。
  - 支持各组件的选择（组件可有可无）：
    - 根据硬件的**形态和需求**，可以选择所需的组件。
  - 支持组件内功能集的配置（组件可大可小）：
    - 根据硬件的**资源情况和功能需求**，可以选择配置组件中的功能集。例如，选择配置图形框架组件中的部分控件。
  - 支持组件间依赖的关联（平台可大可小）：
    - 根据**编译链关系**，可以自动生成组件化的依赖关系。例如，选择图形框架组件，将会自动选择依赖的图形引擎组件等。

目录

HarmonyOS社区运营规划



开发套件及社区礼品	社区专属定制证书及勋章	总裁面对面交流机会	免费体验在线培训课程	优先获得线下活动名额	技术沙龙等活动门票

思考题

1. (判断题)通过HarmonyOS的分布式数据管理技术，能够让开发者轻松实现全场景、多设备下的数据存储、共享和访问。（　　）

   A. 正确

   B. 错误

2. (多选题)HarmonyOS支持根据（　　）来实现组件弹性部署。

   A. 硬件价格

   B. 硬件形态和需求

   C. 硬件资源情况和功能需求

   D. 编译链关系

本章总结

- 本章讲述了HarmonyOS的相关概念，介绍了HarmonyOS的设计理念和技术架构，了解了HarmonyOS全场景系统的关键特性，帮助大家认识全新的“面向未来”的HarmonyOS操作系统。

学习推荐

- HarmonyOS官网社区：
  - https://www.harmonyos.com/cn/home/
- HarmonyOS应用开发文档：
  - https://developer.harmonyos.com/cn/home/
- HarmonyOS设备开发文档：
  - https://device.harmonyos.com/cn/home/
- OpenHarmony开源地址：
  - https://gitee.com/openharmony
- 华为人才在线：
  - https://e.huawei.com/cn/talent/#/

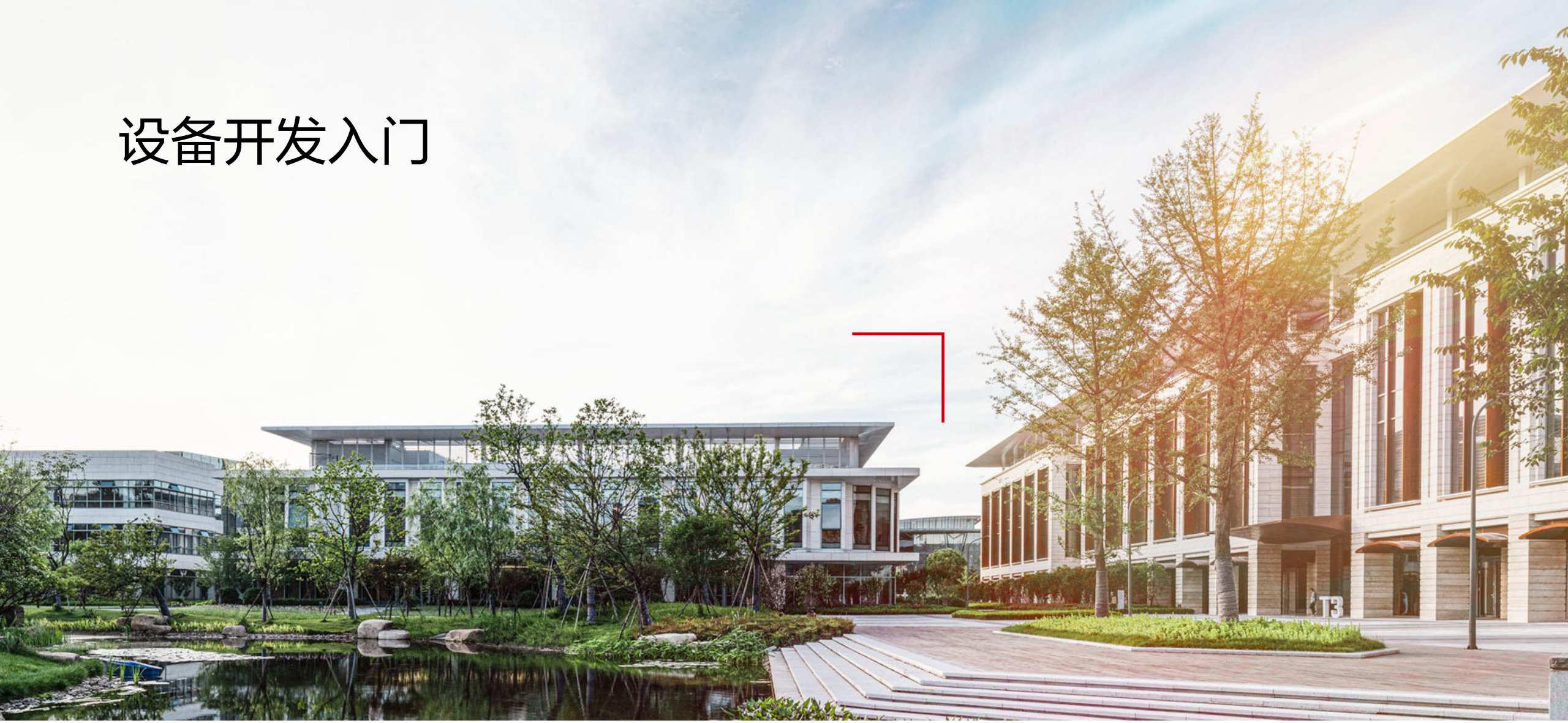
Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

设备开发入门



HUAWEI

前言

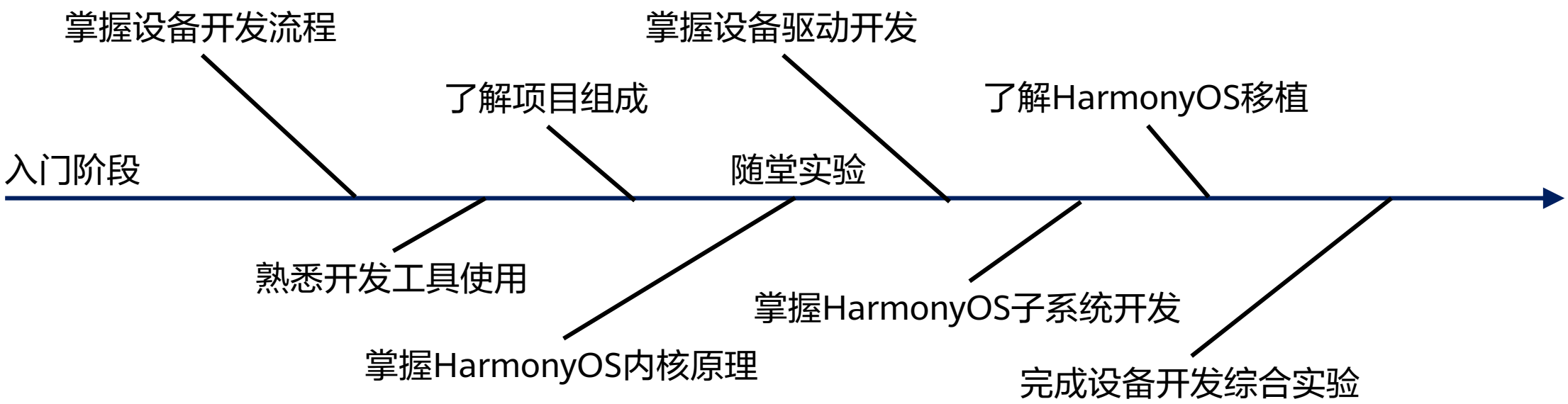
- 工欲善其事，必先利其器。HarmonyOS设备开发的首要环节是搭建开发环境，熟悉开发环境的使用，看懂项目的目录结构，了解组件开发概念并掌握常用的包管理工具HPM。

目标

- 学完本课程后，您将能够：
  - 掌握 HarmonyOS设备开发的环境搭建与新项目创建过程；
  - 了解OpenHarmony的目录结构，CMSIS与POSIX的概念；
  - 了解组件开发概念并熟悉包管理工具HPM的使用。

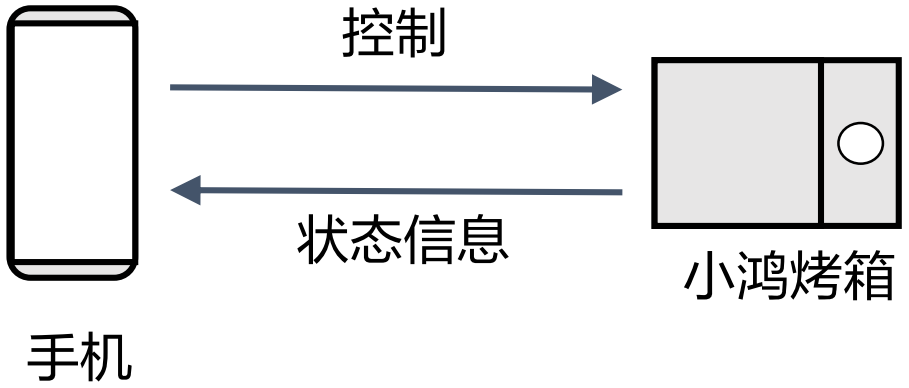
目录

HarmonyOS设备开发学习路径



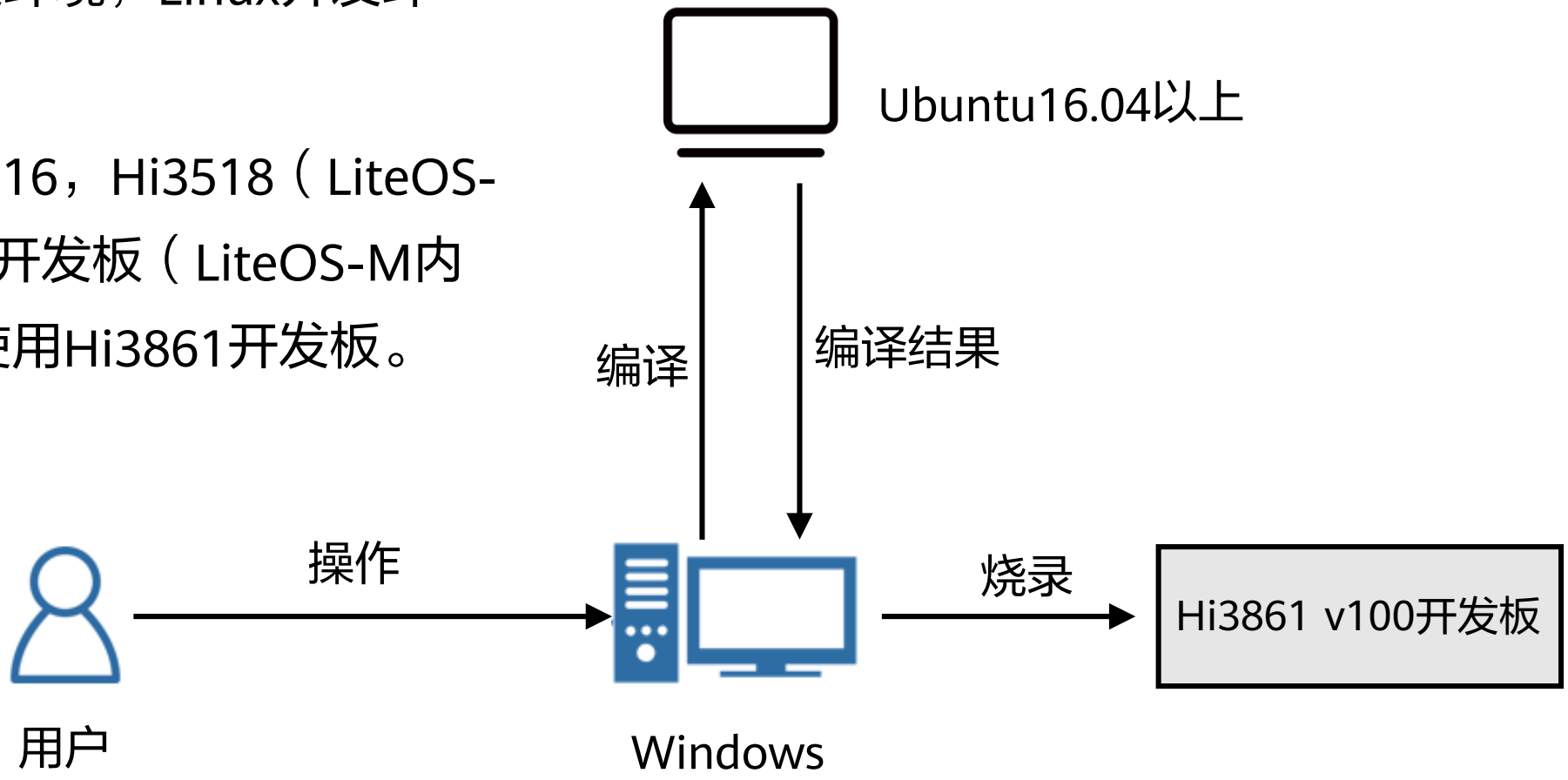
开发目的

- 最终实验将会完成一个“小鸿烤箱”，小鸿烤箱具有这样的功能特点：
  - 手机可以控制烤箱的开启和关闭；
  - 手机可以设置烤箱的蒸烤时间、火力强度；
  - 完成蒸烤之后，在手机上给出提醒。
- HarmonyOS设备开发需要的背景知识：
  - 掌握开发语言：C或C++；
  - 了解Ubuntu操作系统的使用。



设备开发环境准备

- 准备Windows开发环境，Linux开发环境。
- 硬件开发板有Hi3516，Hi3518（LiteOS-A内核）和Hi3861开发板（LiteOS-M内核），实验课程使用Hi3861开发板。

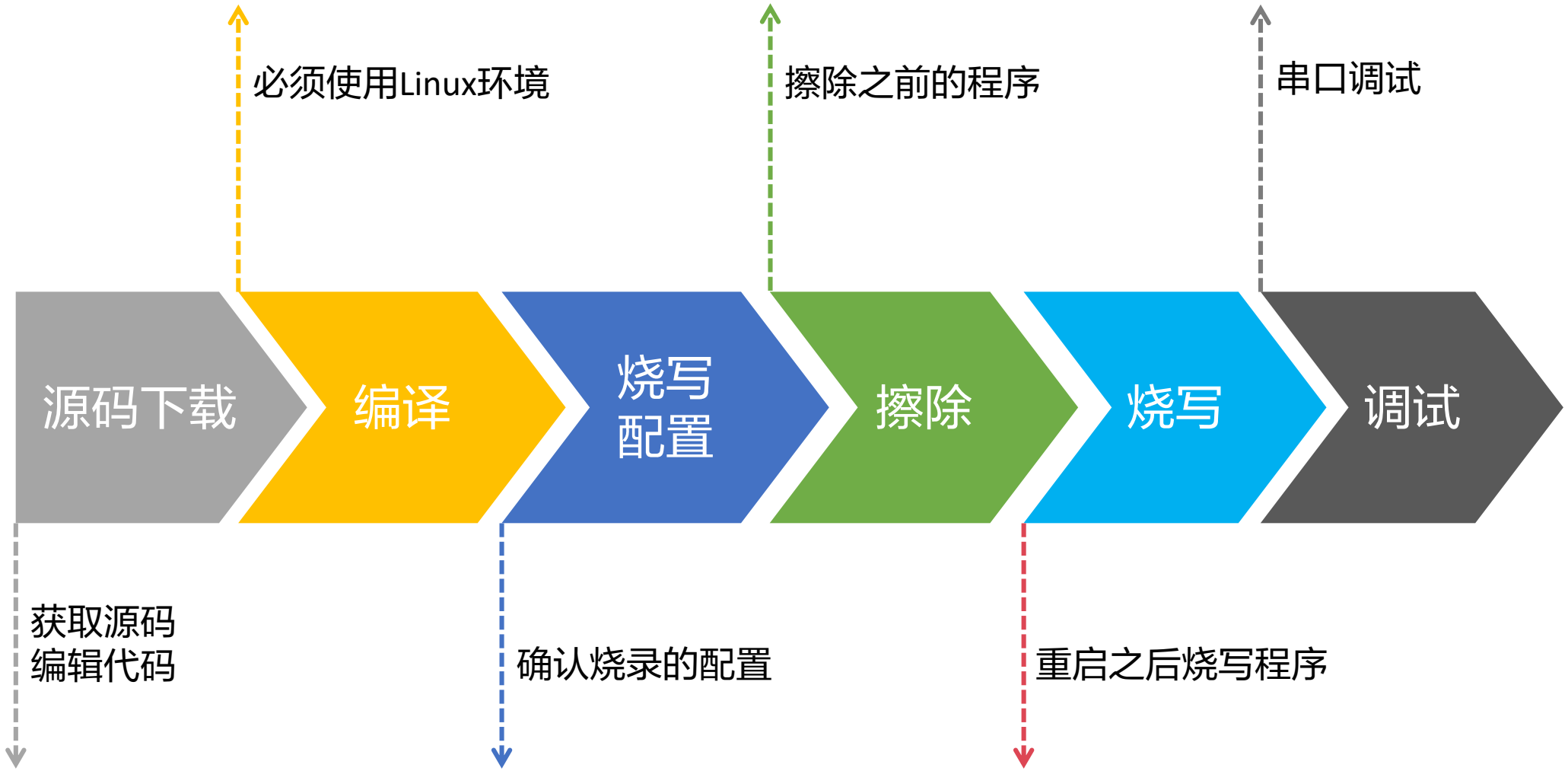


开发环境搭建软件安装列表

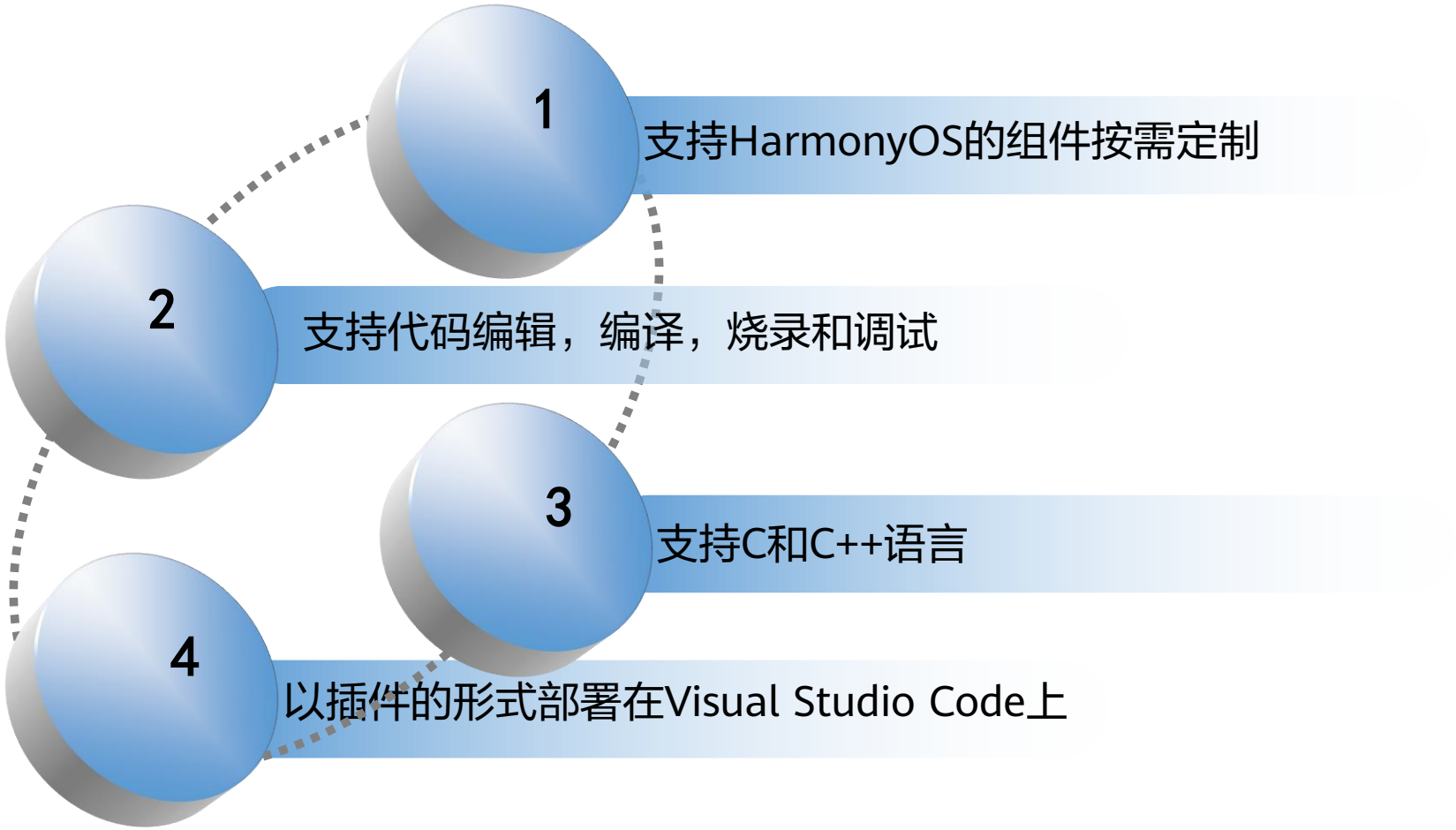
- Windows提供代码编辑和程序烧录的环境。
- Ubuntu提供编译环境。

安装环境	安装内容	作用	版本说明
Ubuntu20.04	Python	编译构建工具	3.8.5的64位版本
Ubuntu20.04	samba	构建samba服务器	最新版本
Ubuntu20.04	gn, ninja, gcc_riscv32	WLAN模组类编译工具链	指定版本
Ubuntu20.04	Scons	自动化构建工具	最新版本
Windows10	Visual Studio Code	代码编辑工具	64位最新版本
Windows10	Python	编译构建工具	≥3.7.3的64版本
Windows10	Node.js	提供npm环境	最新版本
Windows10	hpm	包管理工具	最新版本
Windows10	DevEco Device Tool	源码的编辑，烧录，调试	最新版本
Windows10	（可选安装）HiBurn	烧录工具	最新版本

设备开发流程



HUAWEI DevEco Device Tool



DevEco Device Tool 界面介绍

- 工具控制区提供工程导入、配置、烧录、调试功能。
- 代码编辑区域提供代码的查看、编写和调试功能。
- 输出控制台提供操作日志的打印、调试命令的输入。

思考题

1. (判断题)对于当前版本，在HarmonyOS设备开发中，代码编译需要使用的操作系统是Linux。(　　)

   A. 正确

   B. 错误

2. (判断题)HarmonyOS设备开发需要使用到DevEco Studio。(　　)

   A. 正确

   B. 错误

思考题

3. (单选题)HarmonyOS设备开发主要使用的开发语言是什么？（　　）

A. 汇编

B. C或C++

C. Java

D. Python

目录

OpenHarmony 整体目录结构

- 在OpenHarmony项目的目录结构中，共有16个1级目录文件夹。
  - applications目录中主要存放了用户的应用程序；
  - base目录是HarmonyOS Framework的基础能力集合，定位了大多数设备开发需要的能力模块；
  - untils目录作为公共基础库，存放通用的基础组件。

applications 目录详解

- applications目录主要存放用户的应用程序，或是HarmonyOS 预置的系统应用程序。
- applications展开：
  - applications.sample：提供Hi3516/Hi3518/Hi3861基础应用，这些应用预置在设备中；
  - applications.camera：主要用于Hi35xx AI Camera的基础应用；
  - applications.camera.app：此目录为用户自己开发的目录，可以通过该目录下的BUILD.gn文件适配是否要预置到系统中；
  - applications.camera.communication：通话模块；
  - applications.camera.example：示例模块；
  - applications.camera.hap：预置的app；
  - applications.camera.media：视频模块实例。

base 目录详解

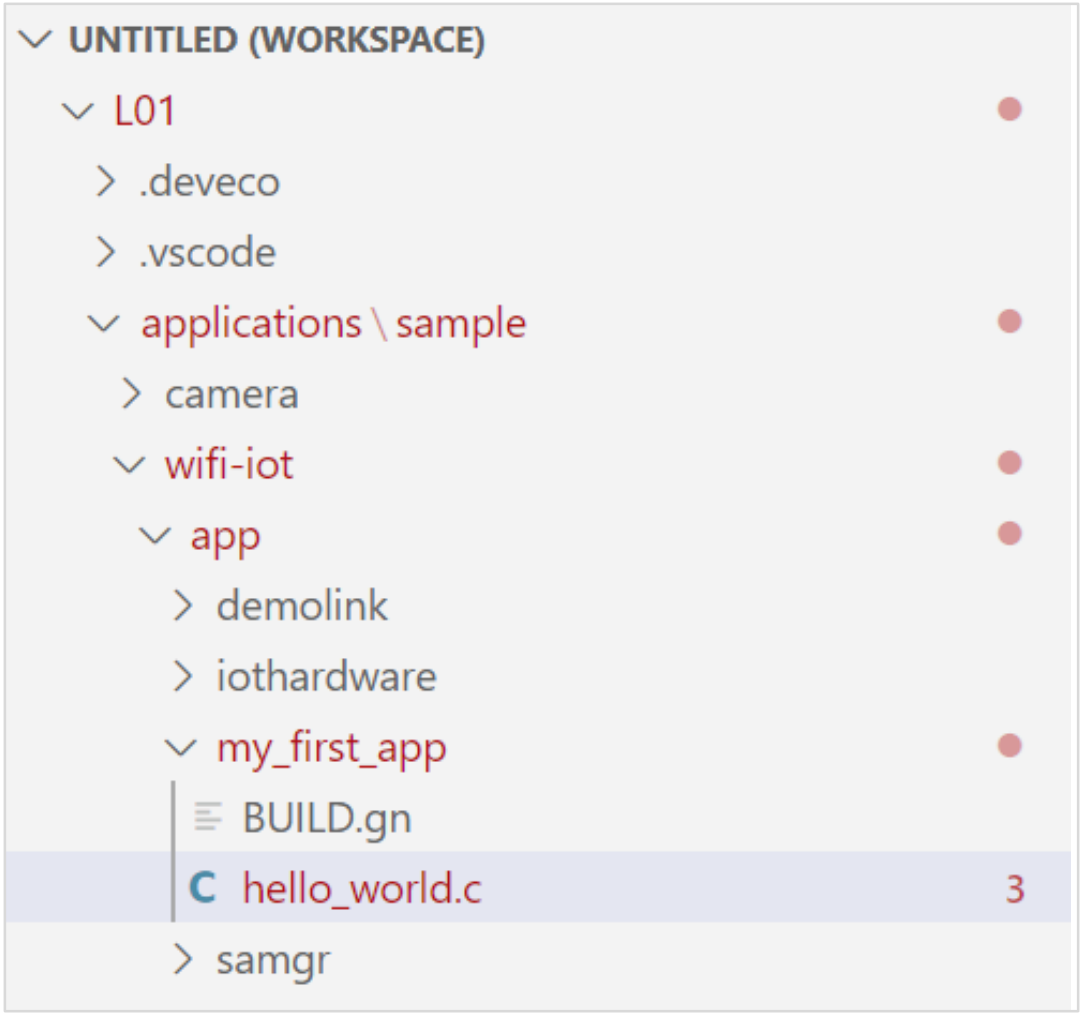
- HarmonyOS Framework基础能力集合，定位于大多数设备开发都需要能力模块。
- base目录展开：
  - base.global：全球化模块，是设备的基础能力模块，可被裁剪；
  - base.hiviewdfx：DFX模块；
  - base.iot_hardware：IoT外设能力模块（GPIO/I2C/SPI/AD/DA等）；
  - base.kits：IoT外设控制模块接口，与frameworks/wifiiot_lite配合；
  - base.security：安全模块；
  - base.syspara_lite：系统属性模块文件；
  - 系统启动相关模块：base.appspawn_lite应用孵化模块，base.bootstrap_lite启动服务模块，base.init_lite启动引导模块。

foundation目录详解

- foundation提供了更为高级的系统基础能力模块，包含分布式调度与分布式通信等。
- foundation目录展开：
  - aafwk：ability开发框架接口，ability的管理服务；
  - ace：JS应用开发框架，提供了一套跨平台的类web应用开发框架；
  - ai\engine：AI引擎框架；
  - appexecfwk：包管理服务；
  - communication：分布式通信；
  - distributedschedule：分布式任务调度；
  - graphic：图像模块；
  - multimedia：多媒体。

OpenHarmony 目录展开

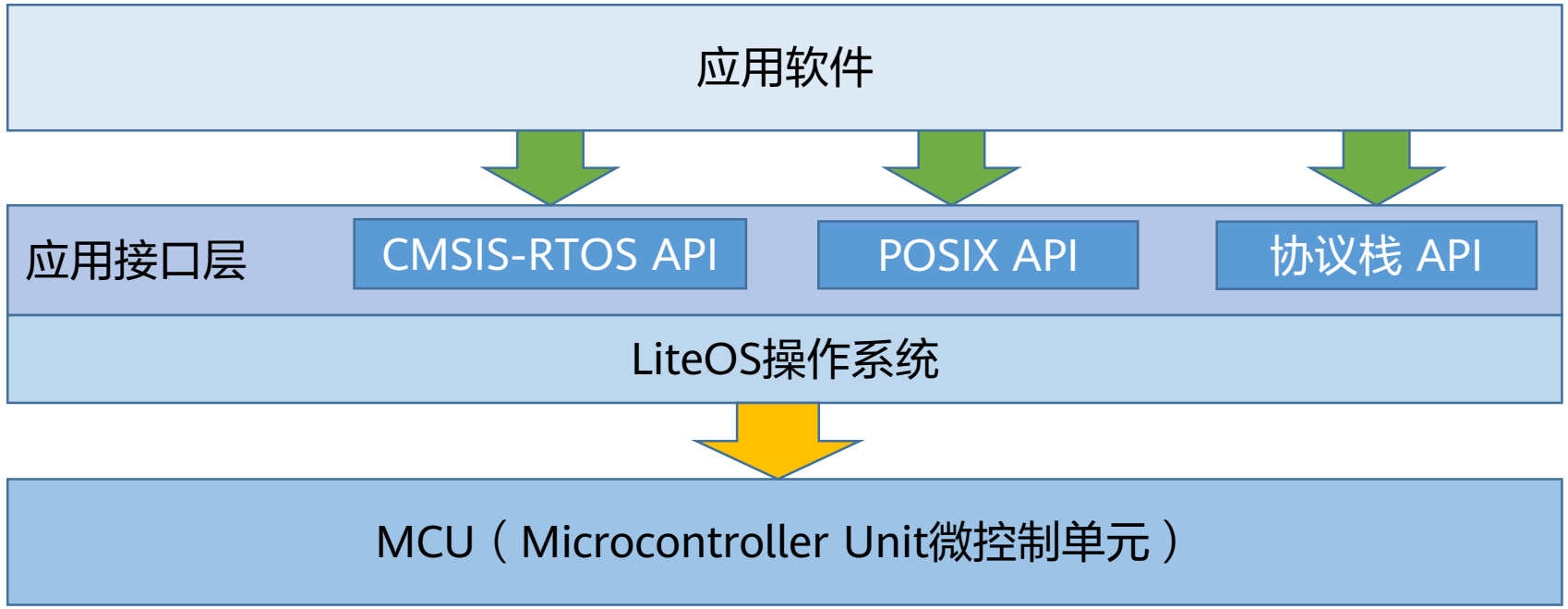
- 例子：wifi-iot的demolink：
  - application：应用；
  - wifi-iot：框架层级目录；
  - app：应用层级目录；
  - demolink：demo示例；
  - BUILD.gn：应用配置文件；
  - helloworld.c：demo源码。



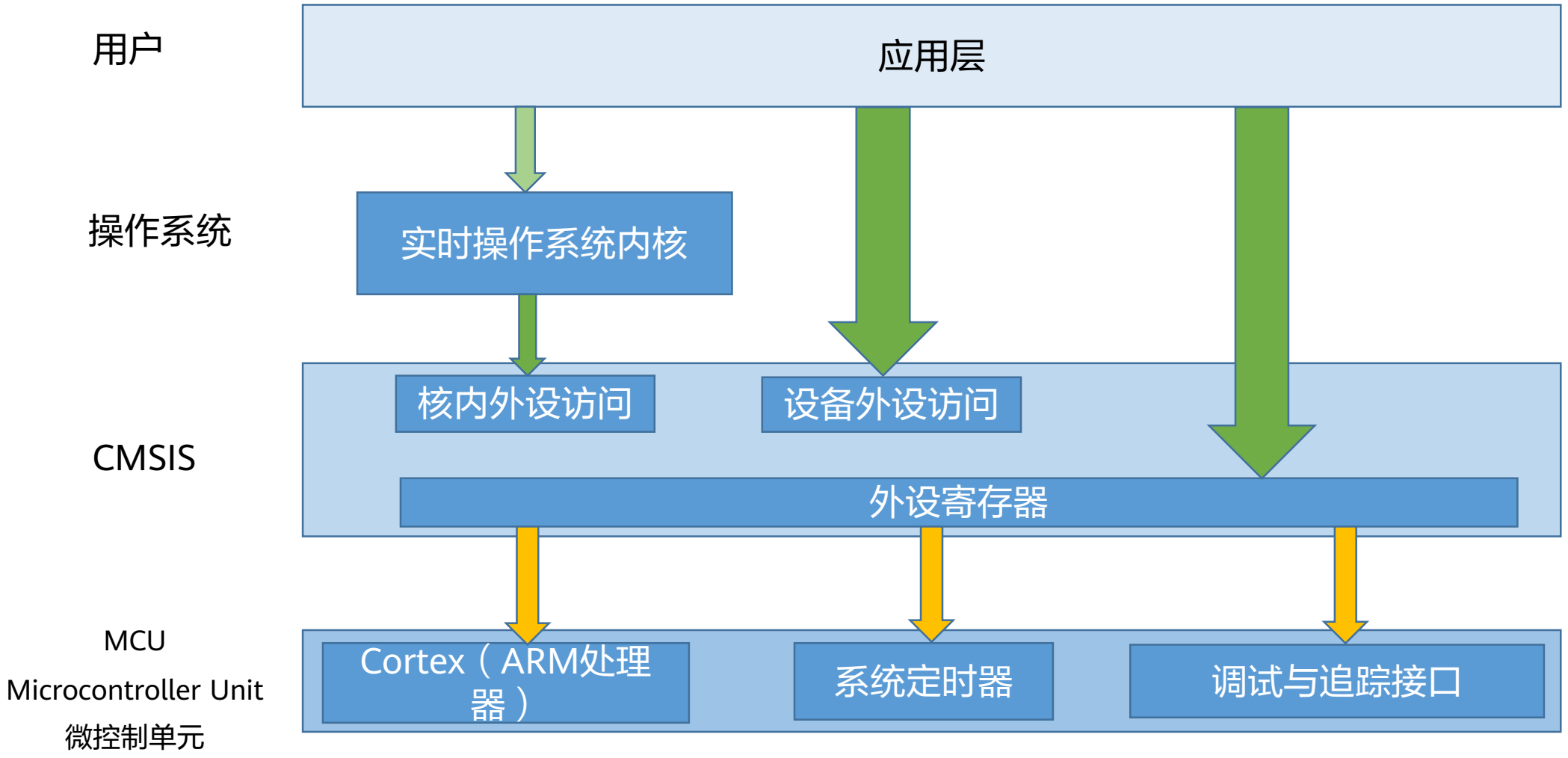
目录

CMSIS 和 POSIX 整体架构

- CMSIS（Cortex Microcontroller Software Interface Standard，微控制器软件接口标准）和POSIX（Portable Operating System Interface，可移植操作系统接口）都是可移植操作接口的标准，属于应用接口层，可增强应用软件的可移植性，降低开发难度。

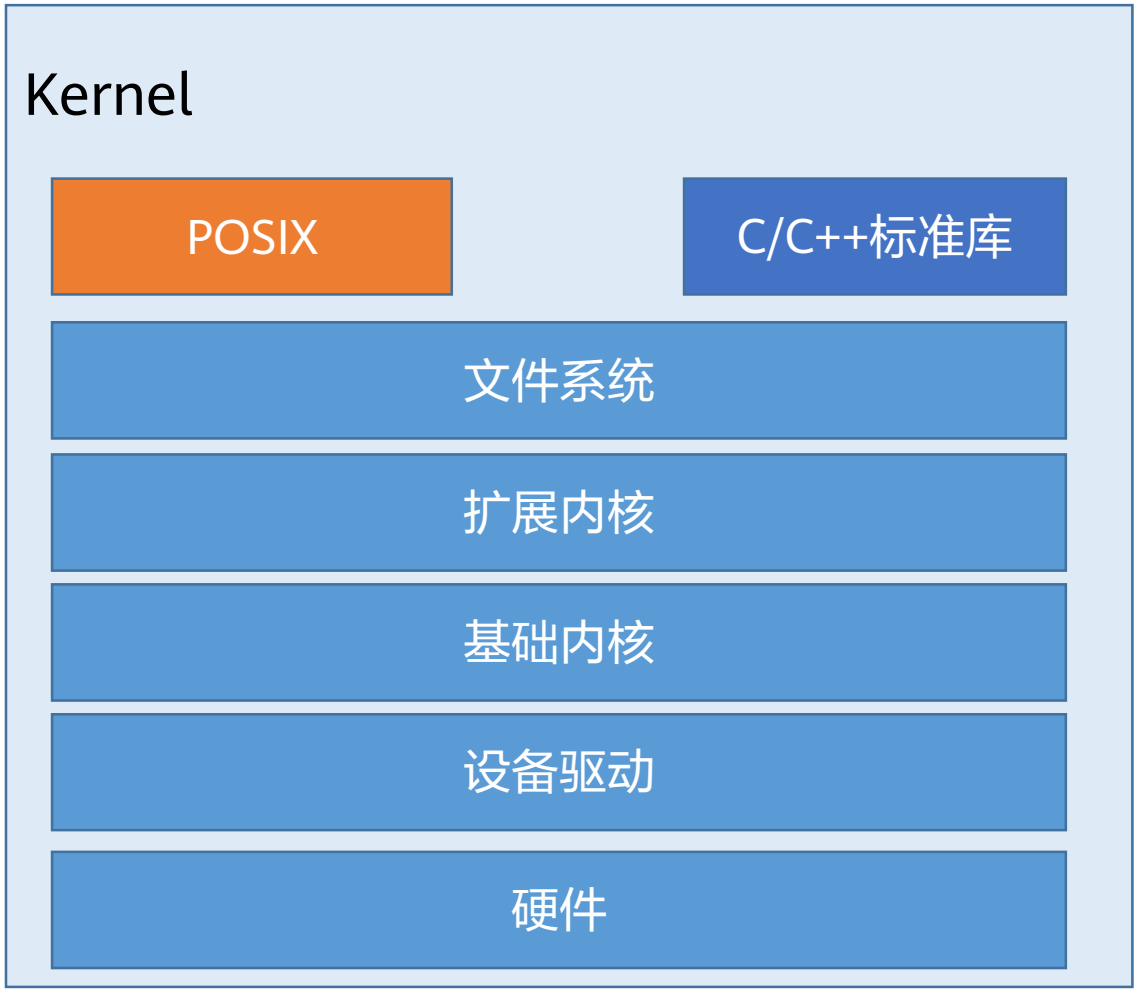


CMSIS 架构



POSIX 架构

- POSIX是API的一系列互相关联的标准的总称。
- 对于目前大多数的RTOS实时操作系统，不支持使用内存管理单元。
- 文件系统包括虚拟文件系统，网络文件系统，文件配置表等内容。
- 扩展内核：异常管理，动态加载等。
- 基础内核：时间管理，内存管理，IPC通信，任务同步，硬件管理，任务调度。



思考题

1. (判断题)CMSIS是ARM公司为Cortex芯片设计的一种标准。(　　)

   A. 正确

   B. 错误

2. (判断题)CMSIS与POSIX都可以增强软件的可移植性，降低开发难度。(　　)

   A. 正确

   B. 错误

思考题

3. (多选题)在CMSIS架构中，MCU层包含哪些组成部分？(　　)

A. Cortex（ARM处理器）

B. 系统定时器

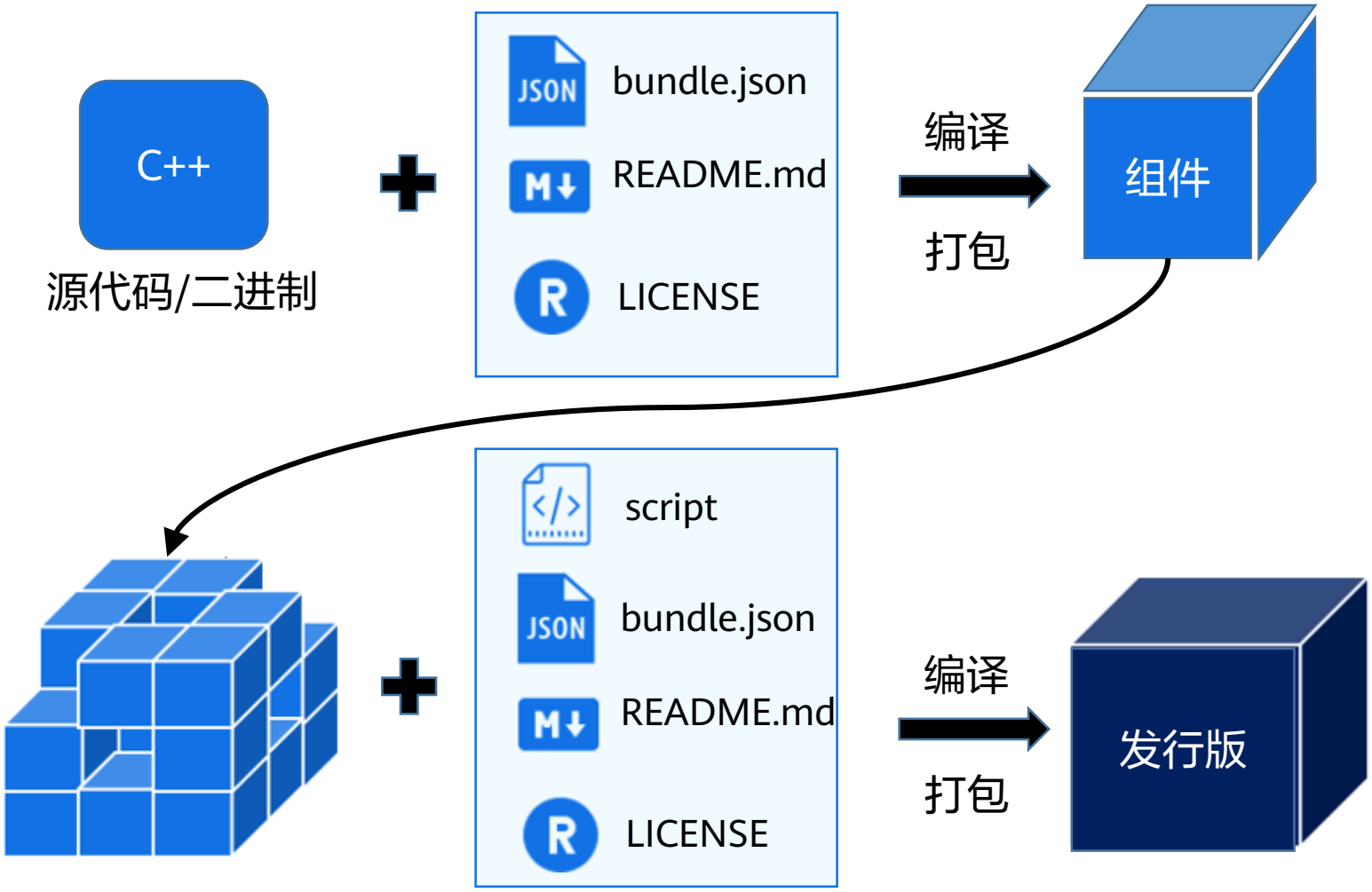
C. 外设寄存器

D. 调试与追踪接口

目录

组件开发介绍

- HarmonyOS组件开发遵从**模块化开发**的思想：独立的小组件拼接为一组组件，再次编译打包构成带有完整系统镜像的发行版。



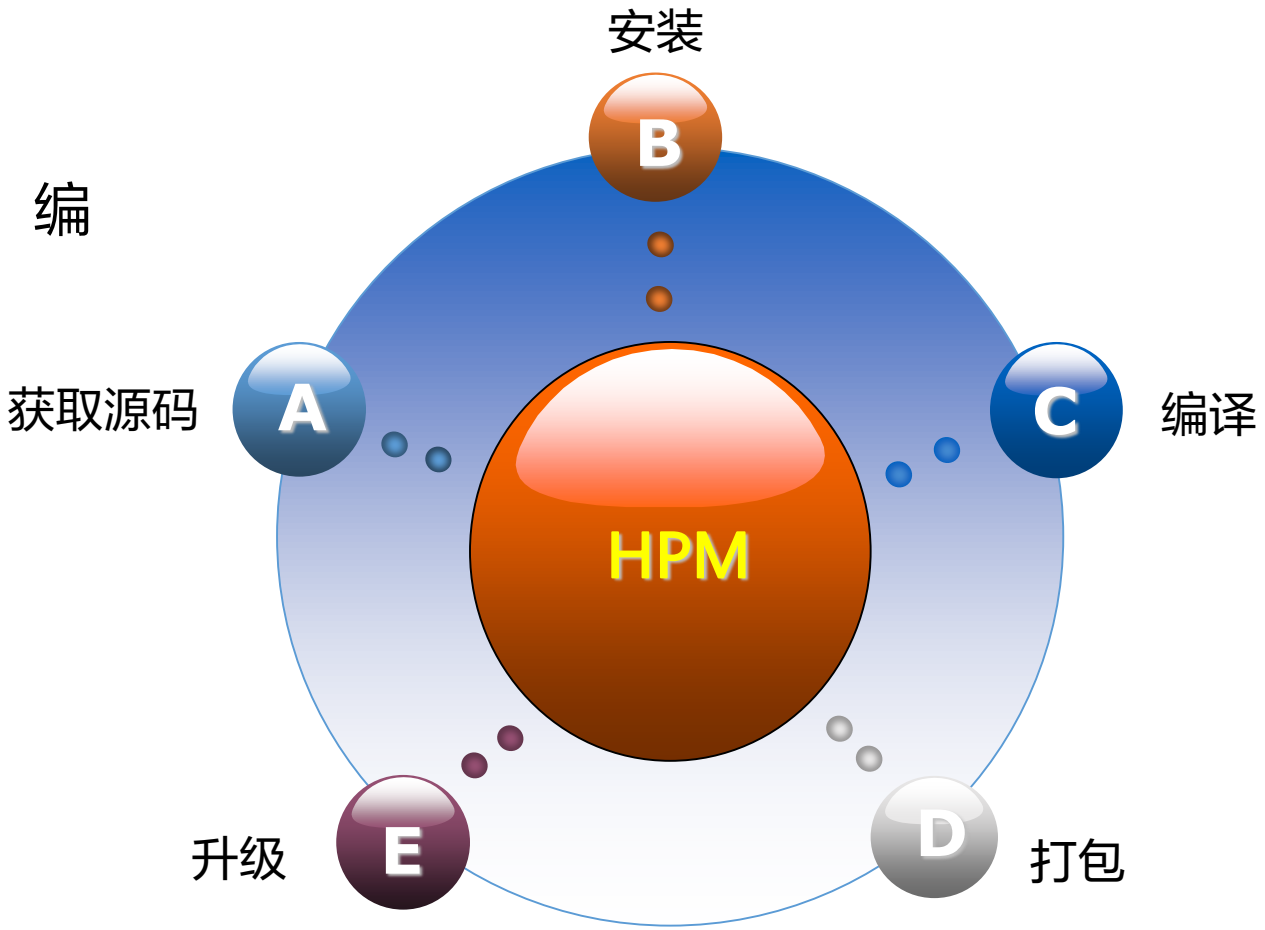
组件与发行版的异同点

- 一个组件(bundle)通常和一个代码仓库对应，在代码的基础上增加bundle.json、README文件、LICENSE描述文件。
- 一个发行版(distribution)是由多个组件构成的。发行版中集合了一个完整系统的各种组件（如驱动、内核、框架、应用），可以用于设备的烧录。

异同点	组件	发行版
应用场景	面向功能特性开发	面向系统开发
内容	功能或特性的实现代码或二进制库	依赖的组件清单及编译构建脚本
完整程度	操作系统的一部分	一个完整操作系统版本
编译后结果	组件包	系统镜像

HPM

- HarmonyOS Package Manager是HarmonyOS的包管理工具。
- HPM主要功能：获取源码，执行安装、编译、打包、升级操作。



HPM操作实例

- 以Hi3861平台为例
  - 第一步，初始化安装目录：

```
mkdir test3861
cd test 3861
hpm init –t dist
```

  - 第二步，安装wifi-iot发行版：

```
hpm install @ohos/wifi_iot
```

  - 第三步，执行发行命令，编译打包：

```
hpm dist
```

    - 编译打包成功之后，将在./out目录下生成编译结果，可用于烧录。

本章总结

- 本章主要讲述了以下几点内容：
  - 设备开发环境简介与开发环境的搭建过程，主要介绍了DevEco Device Tool工具；
  - 详细介绍了OpenHarmony的目录结构与CMSIS接口；
  - 介绍了组件的概念与包管理工具HPM的使用。

学习推荐

- HarmonyOS官网社区：
  - https://www.harmonyos.com/cn/home/
- HarmonyOS应用开发文档：
  - https://developer.harmonyos.com/cn/home/
- HarmonyOS设备开发文档：
  - https://device.harmonyos.com/cn/home/
- OpenHarmony开源地址：
  - https://gitee.com/openharmony
- 华为人才在线：
  - https://e.huawei.com/cn/talent/#/

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

内核基础

HUAWEI

前言

- 对于任何一个操作系统而言，内核的运行机制与原理是最为关键的部分。
- 本章内容从多角度了解HarmonyOS的内核运行机制，涵盖进程与线程的概念，内存管理机制，网络特性，文件系统，软件定时器，信号量，互斥锁，消息队列相关的内核知识。

目标

- 学完本章内容后，您将能够：
  - 掌握HarmonyOS的进程与线程的概念与调度机制；
  - 熟悉HarmonyOS的内存管理机制与网络特性；
  - 熟悉HarmonyOS的文件系统，VFS相关概念；
  - 了解HarmonyOS的软件定时器，信号量，事件管理，互斥锁，消息队列相关内容。

目录

线程的应用场景

- 考虑这样的业务场景：某开发人员需要让设备读取文件信息，每隔一定时间需要将读取出的文件内容通过网络发送到服务器。

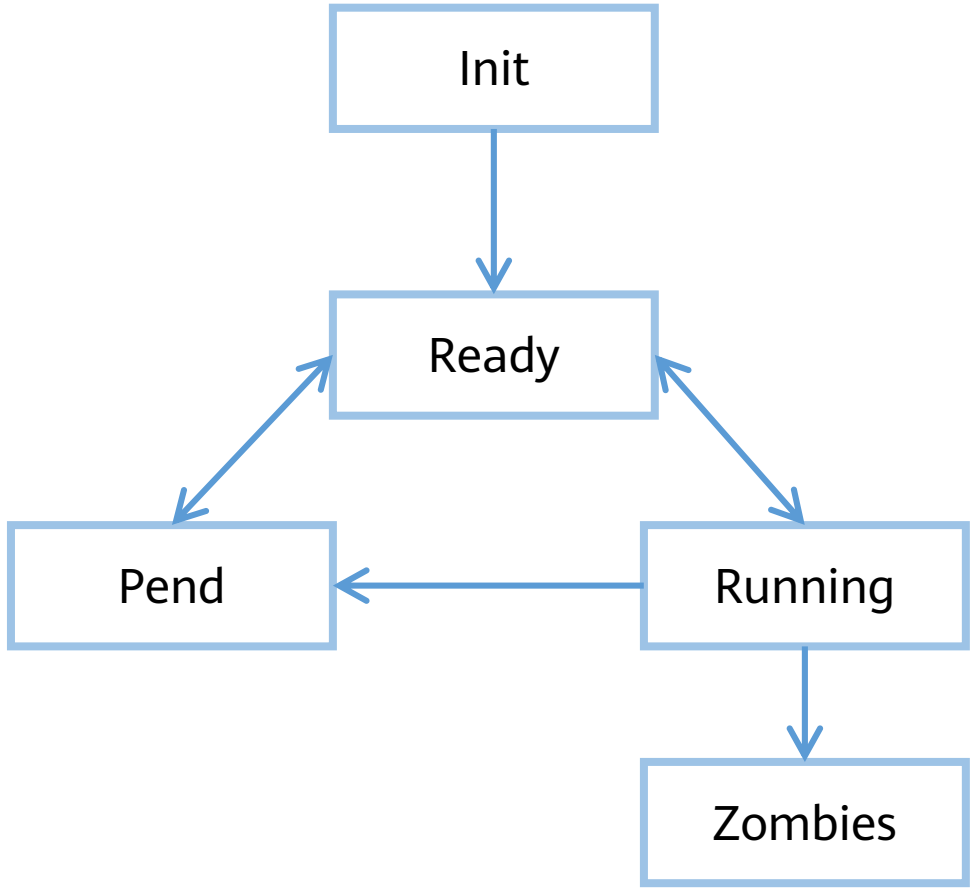
- 方案一：一段时间读取文件内容，停止文件读取，通过网络完成文件发送，之后继续读取文件，循环往复。
- 方案二：持续读取文件内容，不停止文件读取的操作，启动一个新的**线程**，将读取出的文件通过网络发送，完成文件发送之后关闭此线程，循环往复。

HarmonyOS 进程基本概念

- HarmonyOS内核中的进程采用抢占式调度机制，支持**时间片轮转调度方式**。
- HarmonyOS内核进程共有0~31的**进程优先级**，用户进程可配置的优先级有22个(10~31)。
- 高优先级进程**抢占**低优先级进程，低优先级进程必须等待高优先级进程释放CPU资源。
- 每一个用户态进程均拥有自己独立的进程空间，相互之间不可见，进程之间**相互隔离**。
- 用户态**根进程init**由内核态创建，其他用户进程均由init进程**fork**而来。

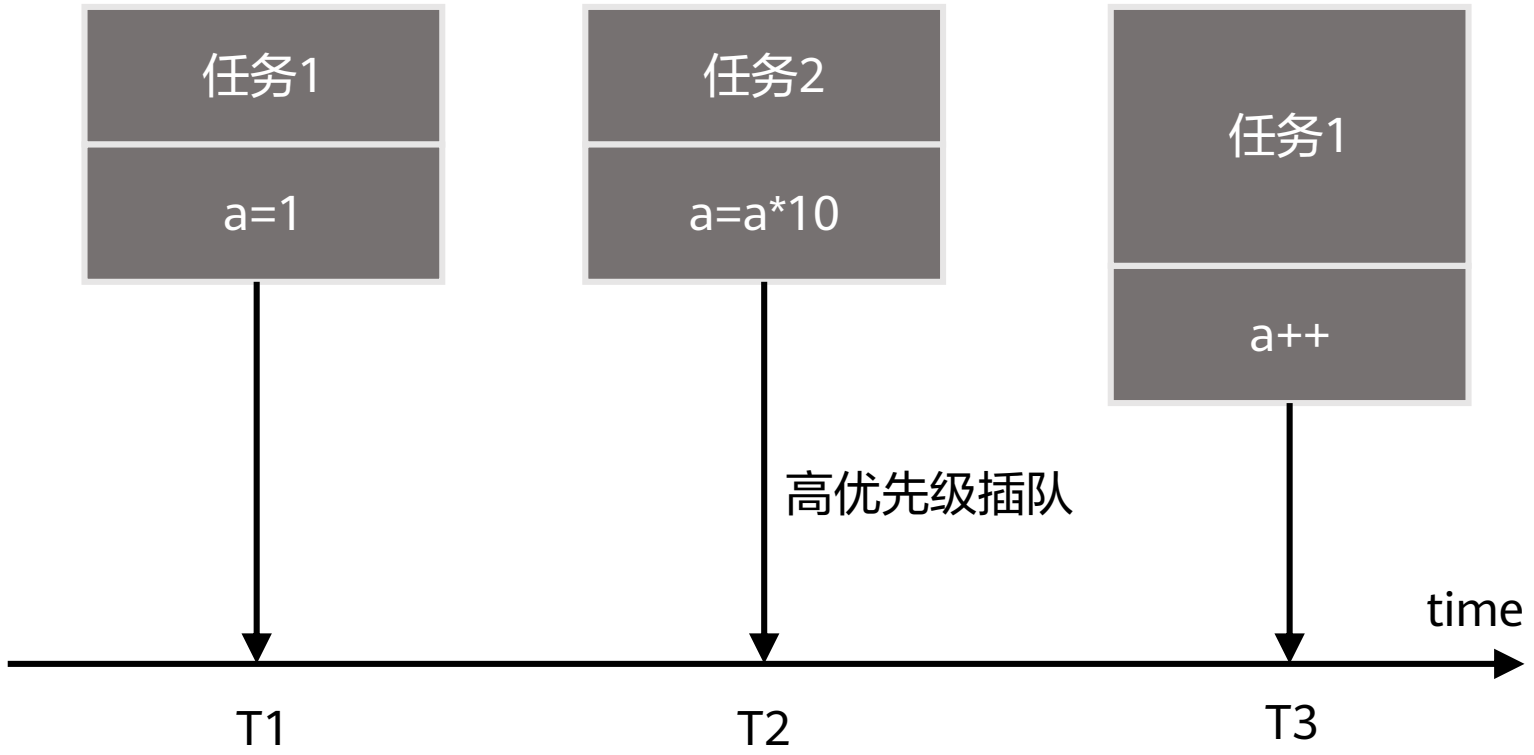
进程状态迁移 - 进程插队场景

- 考虑**单核CPU**的情况，在进程1的执行过程中，进程2插队执行。
- 进程1的状态流转：Init →Ready →Running →Ready →Running →Zombies。
- 进程2的状态流转：Init →Ready →Running →Zombies。



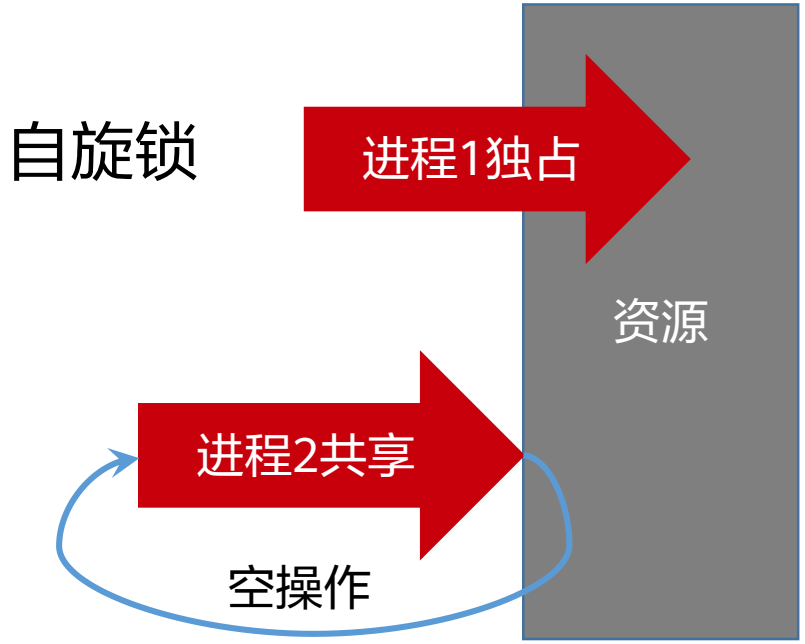
讨论：高优先级的进程一定能抢占低优先级进程么？

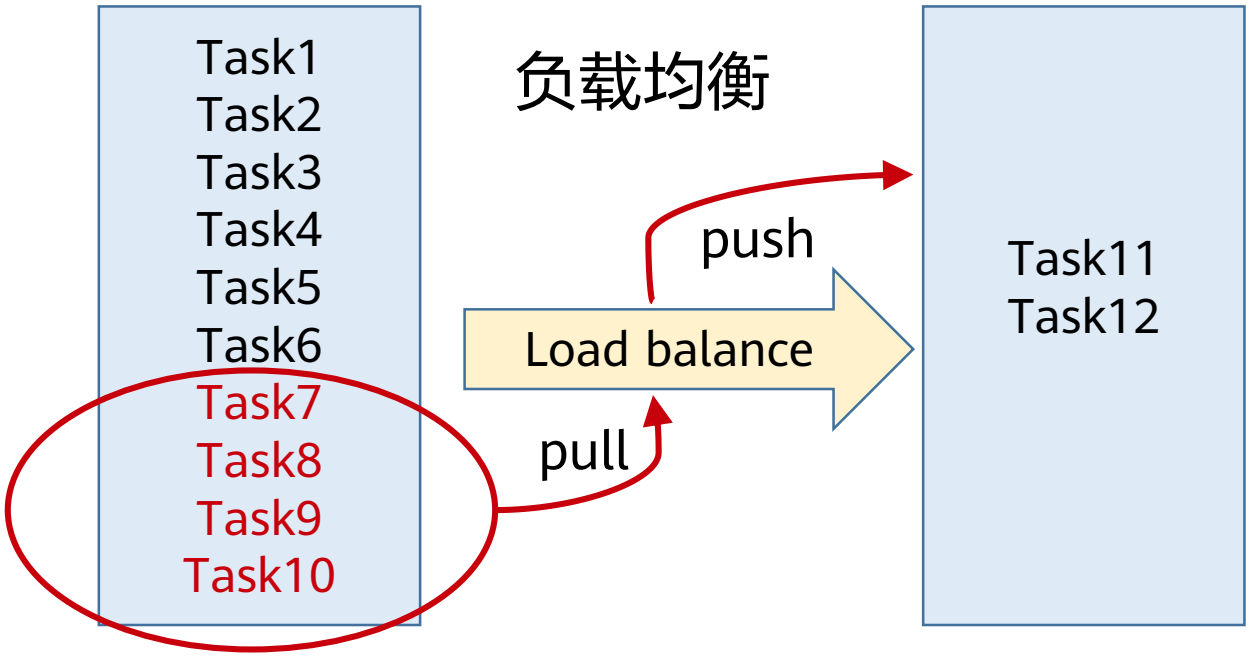
- 如果任何时候高优先级进程都会抢占低优先级进程，则可能出现异常。
- 需要一种“保护机制”，确保程序的执行结果不出现异常，这种机制称为“锁”。



进程的防冲突与负载均衡

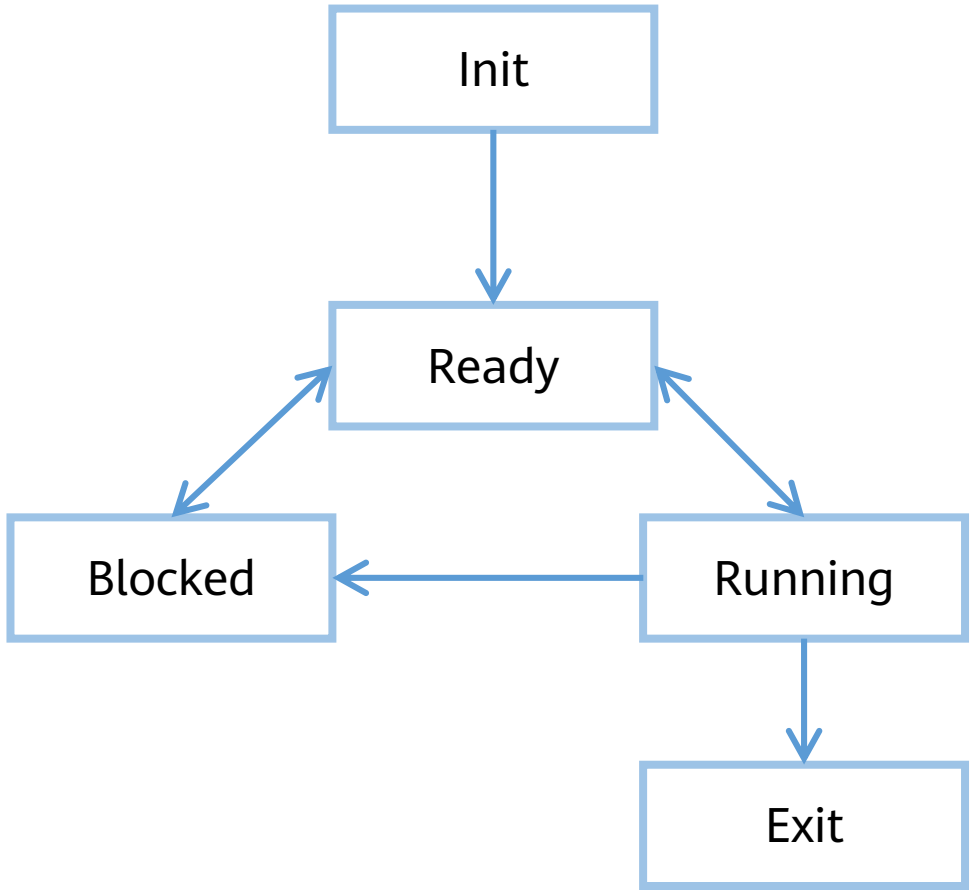
- 对于多核CPU而言，HarmonyOS通过自旋锁实现进程间防冲突，任务的重新分配实现多核处理器之间的负载均衡。





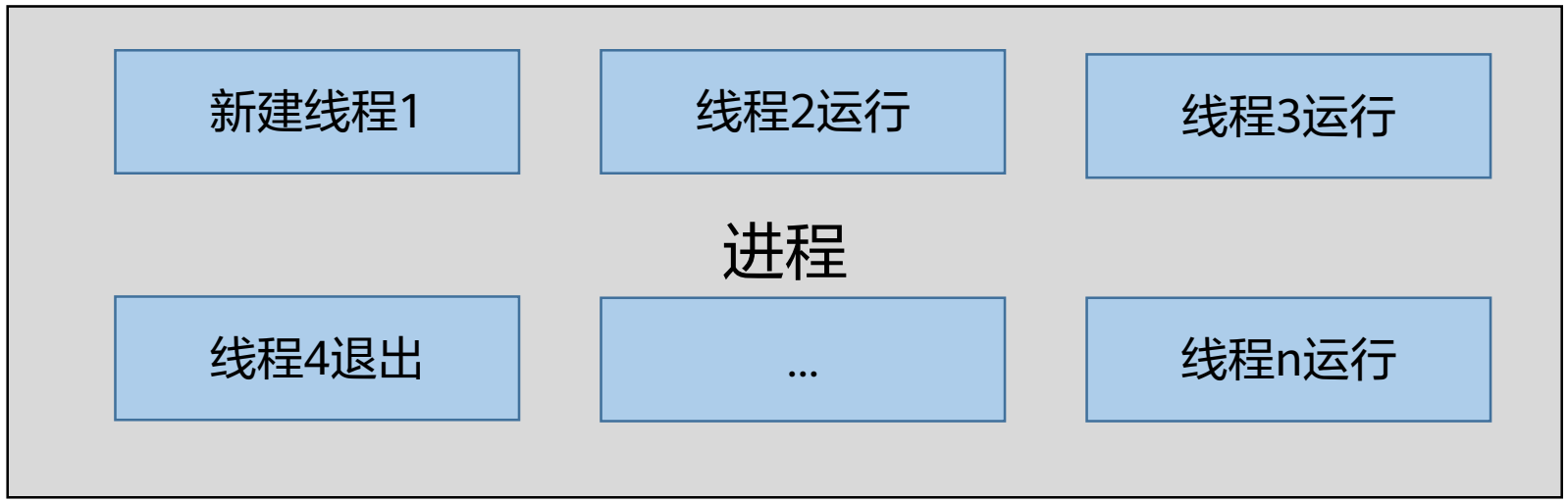
线程状态迁移 - 线程插队场景

- 对于**同一个进程**，在线程1的执行过程中，优先级更高的线程2插队执行。
- 线程1：Init →Ready →Running →Ready →Running →Exit。
- 线程2：Init →Ready →Running →Exit。



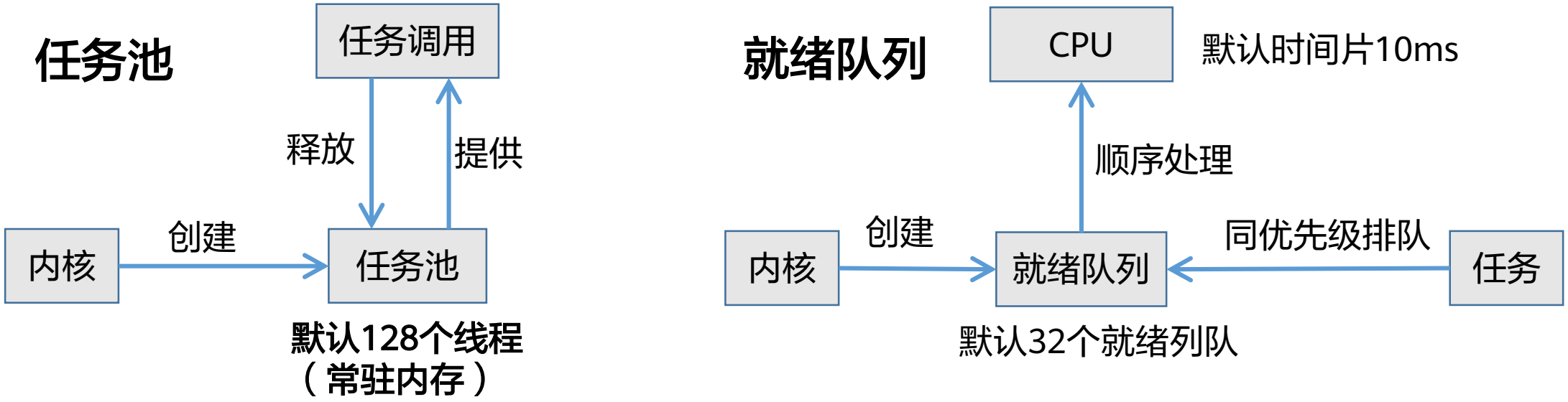
讨论：对于线程数量非常多的场景，如何管理

- 在一个进程中，可能同时运行着多个线程，这些线程会频繁的产生，运行，退出。有什么合适的策略能够管理这些线程，减少CPU资源的浪费?
- 对于如此众多的线程数量，如何保证线程之间的“公平”，减少线程的平均等待时间?



HarmonyOS 内核的线程管理

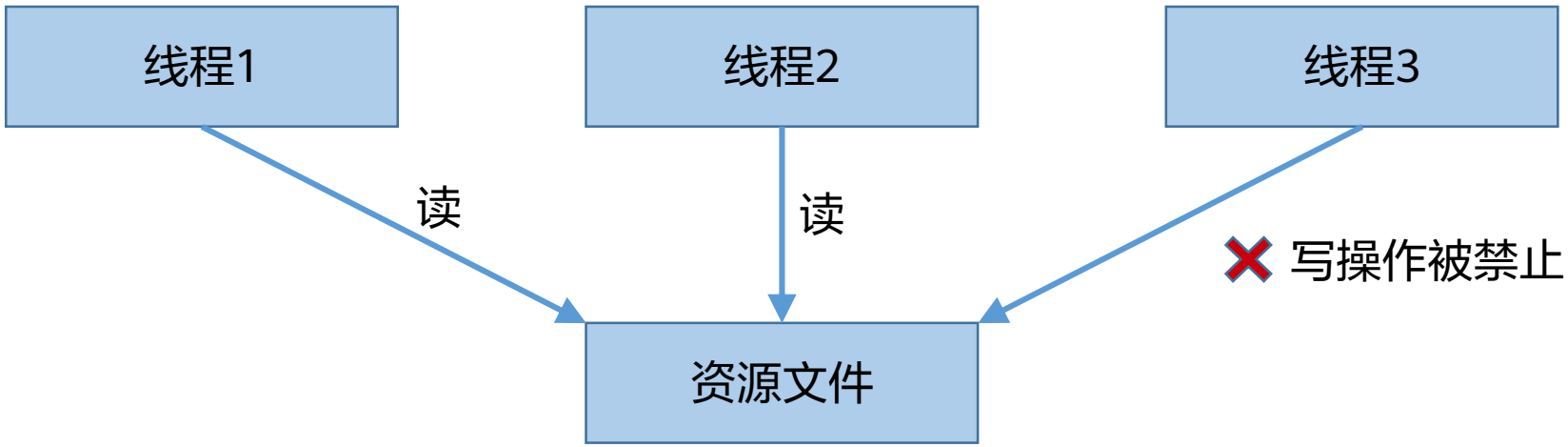
- 对于多线程的场景，HarmonyOS内核管理线程靠**任务池**和**就绪队列**，执行靠**调度算法**。



调度算法：HarmonyOS内核中的线程采用抢占式调度机制，同时支持SCHED_RR和SCHED_FIFO调度策略。

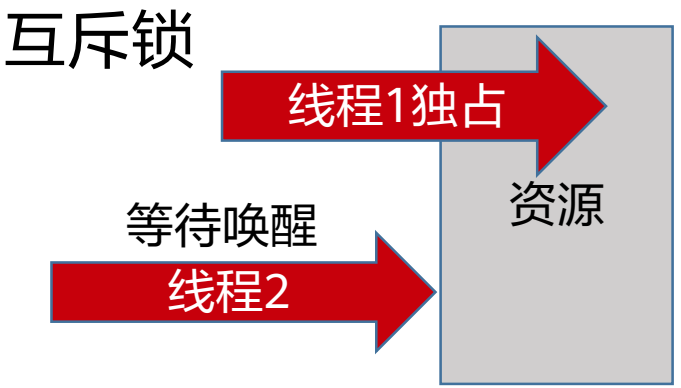
讨论：HarmonyOS 线程常用的锁机制有什么？

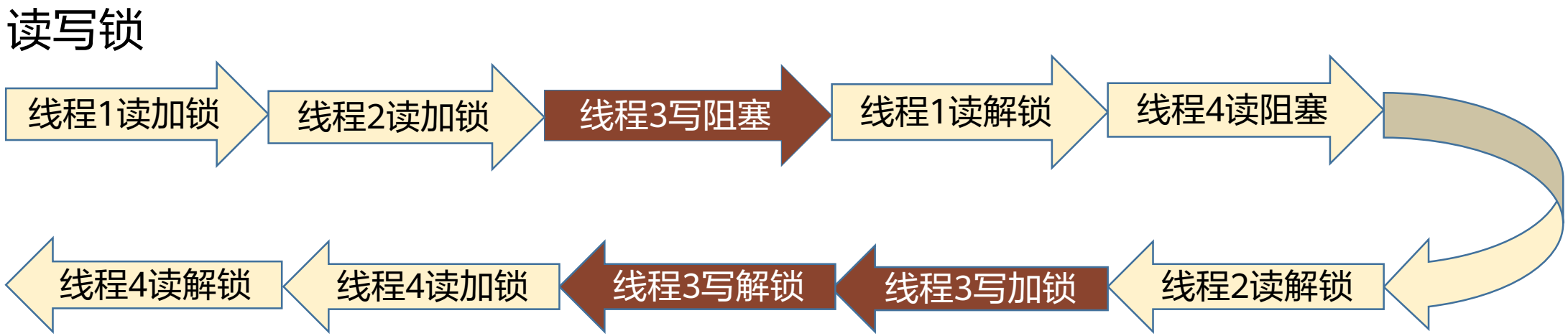
- 在同一个进程，同一个CPU上，未得到资源的线程应当做什么？
- 文件的读取是不会改变文件的内容，是否可以在保证文件内容不被篡改的情况下，同时让多个线程同时读取文件？



HarmonyOS 线程常用的两种锁

- **互斥锁**：未得到资源的线程将不会占用CPU资源。
- **读写锁**：也称“共享-独占锁”适用于读比写多的场景。





思考题

1. (判断题)HarmonyOS在多核设备上采用互斥锁来实现进程的防冲突。(　　)

   A. 正确

   B. 错误

2. (判断题)Zombies表示进程运行结束。(　　)

   A. 正确

   B. 错误

思考题

3. (多选题)在HarmonyOS中，常用的两种线程锁是什么？（　　）

A. 互斥锁

B. 读写锁

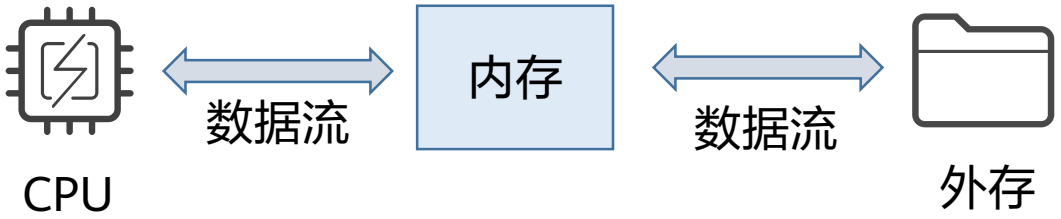
C. 递归锁

D. 悲观锁

目录

HarmonyOS 的内存管理

- Hi3861芯片使用LiteOS-M内核，无MMU（ Memory Management Unit，内存管理单元），内存管理依赖应用程序的malloc方法与free方法。
- Hi3516，Hi3518芯片使用LiteOS-A内核，有MMU。



- HarmonyOS的内存分为四个区域：**代码区**，**静态区**，**栈区**，**堆区**。
- HarmonyOS将内存分为**用户空间**与**内核虚拟空间**，通过**MMU对虚拟内存进行管理调度**。

HarmonyOS 的网络机制

- 网络模块实现了TCP/IP协议栈基本功能，提供标准的POSIX socket接口。
- 当前系统使用LwIP提供网络能力。

头文件	接口	功能
sys/socket.h	int accept(int socket, struct sockaddr *address, socklen_t *address_len)	接受连接
sys/socket.h	int bind(int s, const struct sockaddr *name, socklen_t namelen)	socket与IP地址绑定
sys/socket.h	int shutdown(int socket, int how)	关闭连接
sys/socket.h	int connect(int s, const struct sockaddr *name, socklen_t namelen)	连接到指定的目的IP
sys/socket.h	int socket(int domain, int type, int protocol)	创建socket
sys/socket.h	ssize_t recv(int socket, void *buffer, size_t length, int flags)	接收socket上收到的数据

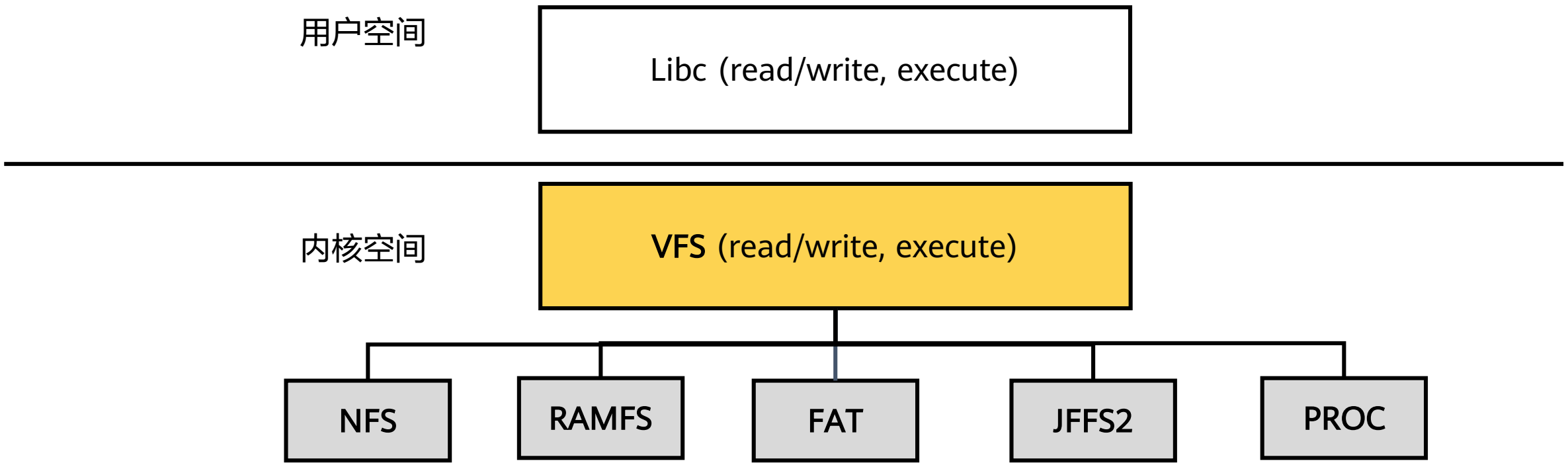
HarmonyOS 文件系统概述

- HarmonyOS文件系统包括：VFS（Virtual File System，虚拟文件系统），NFS（Network File System，网络文件系统），RAMFS，FAT（File Allocation Table，文件配置表），JFFS2（Journalling Flash File System Version 2，日志文件系统）。

文件系统	功能特点概述
VFS	虚拟文件系统。为用户提供统一的类Unix文件操作接口
NFS	网络文件系统。通过网络让不同的主机之间共享文件
RAMFS	基于RAM的动态文件系统。作为一种缓冲机制，减少存储器的读写损耗，提高读写效率
FAT	文件配置表。用于可移动存储介质，使设备与桌面系统有较好兼容性
JFSS2	日志文件系统。主要对NOR_FLASH闪存的文件管理

VFS 介绍

- Virtual File System（虚拟文件系统），它不是一个实际的文件系统，而是一个异构文件系统之上的**软件粘合层**，为用户提供统一的**类Unix文件操作接口**。



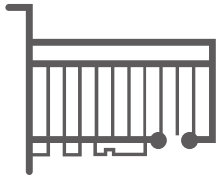
HarmonyOS NFS 特性

- 在HarmonyOS中，创建NFS目录与文件默认使用777权限，当前NFS可支持TCP与UDP两种传输层协议，**默认使用TCP**。

- 补充：777权限，三个数字分别对应三种用户：文件所有者，群组用户，其他用户。每一个数字可以表示执行=1，写=2，读=4权限的情况。

HarmonyOS FAT 简介

- 在HarmonyOS中，文件配置表有FAT12，FAT16，FAT32这三种，它将硬盘分为MBR（Master Boot Record，主引导分区），DBR（Dos Boot Record，操作系统引导记录区），FAT，DIR（Directory，根目录区），DATA（数据区）五个区。
- FAT特性：支持多种介质，尤其在可移动存储设备上使用广泛。FAT具有**代码量和资源占用小、可裁切、支持多种物理设备、读写指针不分离**的特性。



SD卡

移动硬盘

U盘

思考题

1. (判断题)MMU可以让两个不同的进程共用同一个物理内存地址。(　　)

   A. 正确

   B. 错误

2. (判断题)NFS默认使用的传输协议是UDP。(　　)

   A. 正确

   B. 错误

思考题

3. (单选题)Malloc函数申请的内存位于那个区？（ ）

A. 代码区

B. 静态区

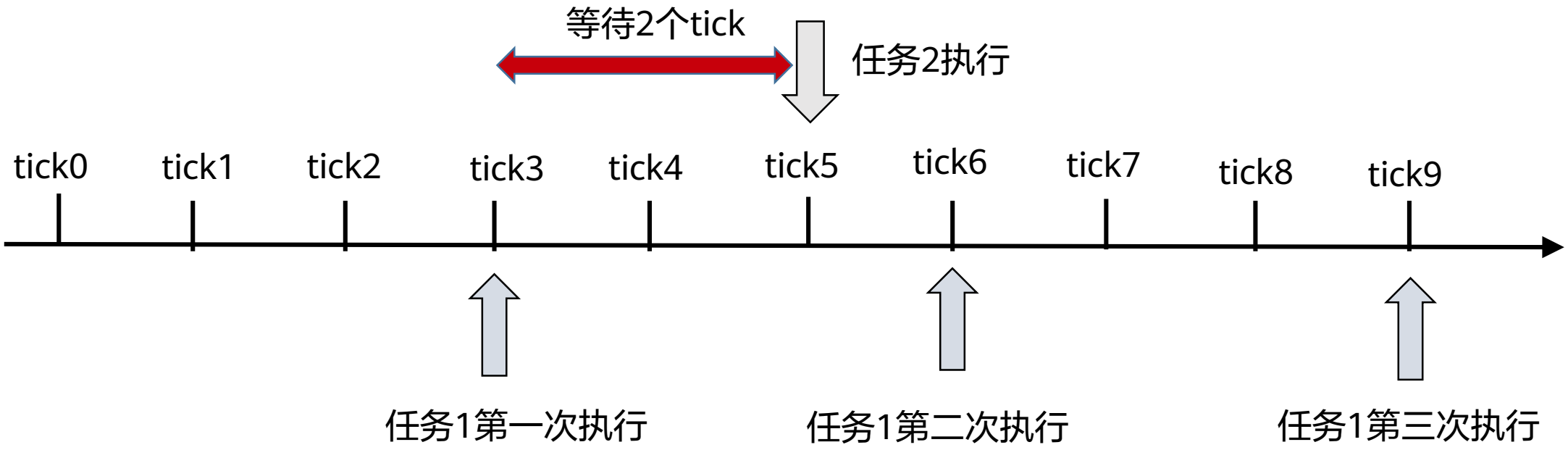
C. 栈区

D. 堆区

目录

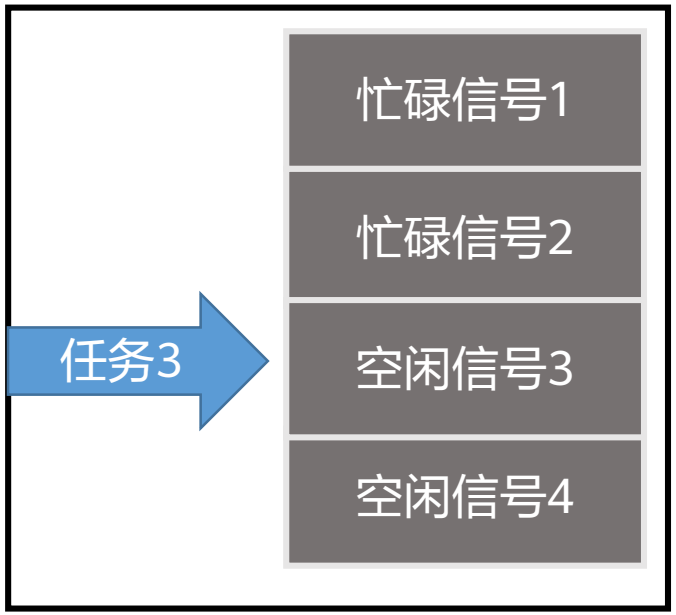
软件定时器

- 软件定时器的功能：制定任务**循环执行的周期**，用作任务的**延迟执行**。
- 在HarmonyOS内，swtmrTask.usTaskPrio = 0，软件定时器拥有最高的线程优先级。

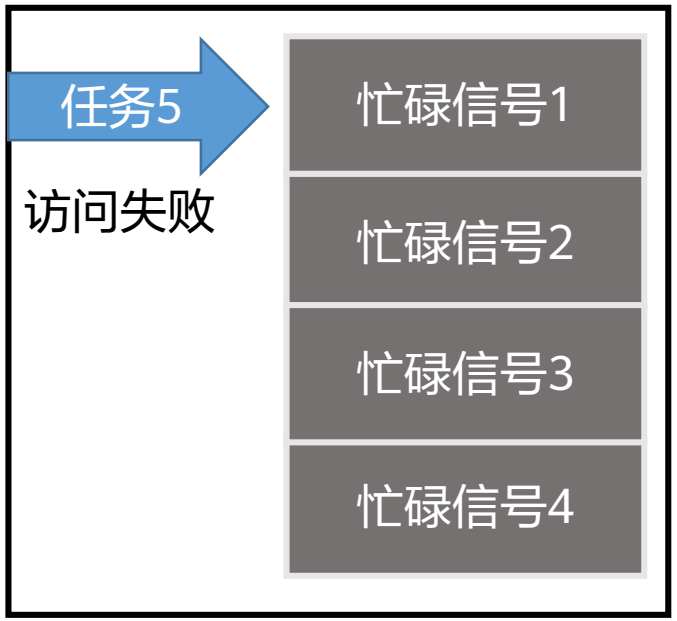


信号量

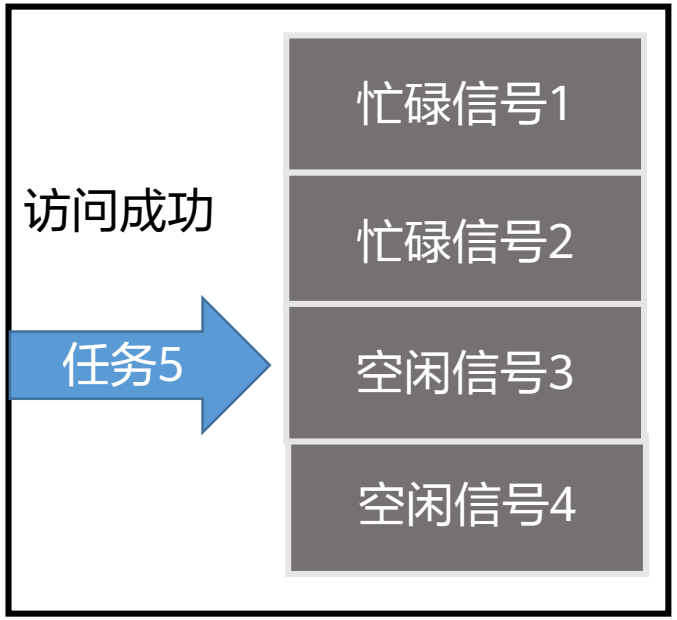
- 信号量可以**实现任务间通信**，它可以实现**任务间同步或共享资源的互斥访问**。
- HarmonyOS的信号量最大个数1024，使用信号量的任务队列是一个双向链表。



多个任务同时访问一份资源



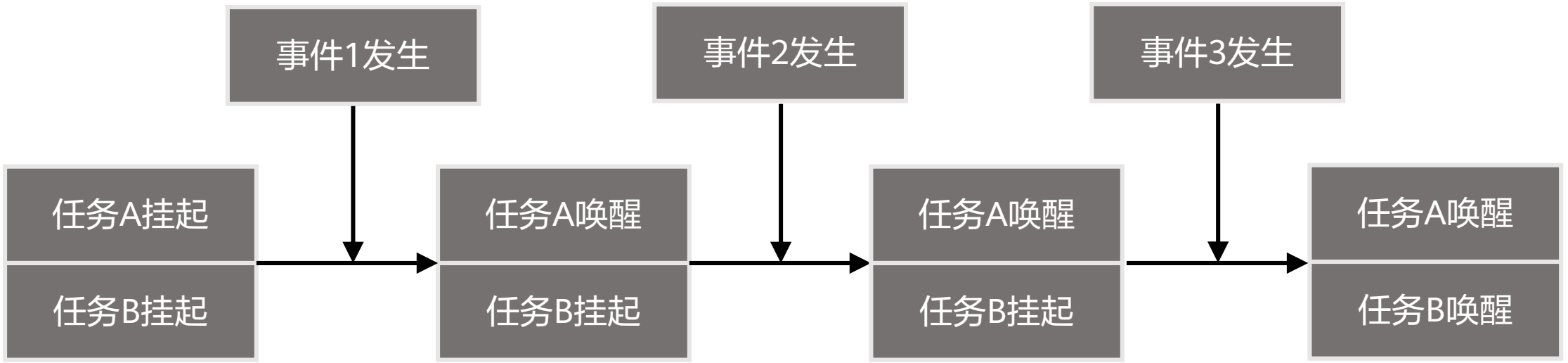
访问量不能超过信号量上限



信号量被释放，后续任务可用

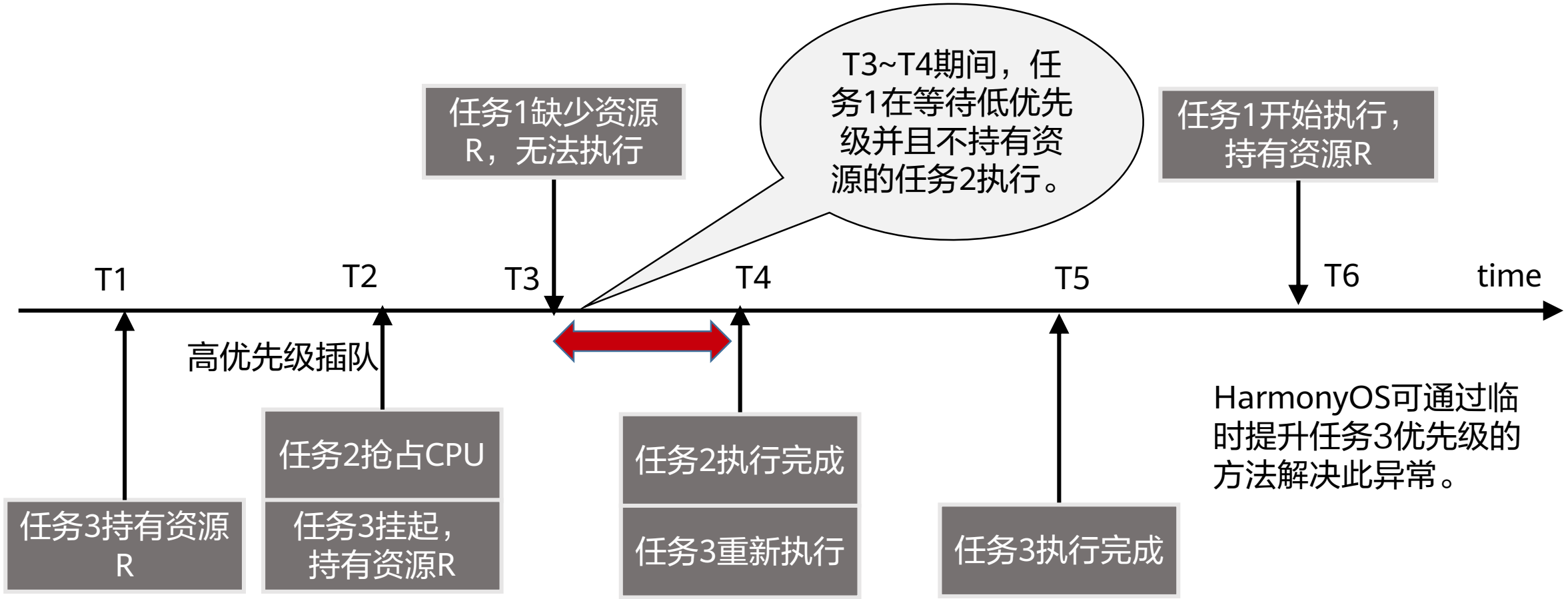
事件运行机制

- 事件Event是一种任务间通信机制，常用于任务之间的同步。
- 事件模型：一对多模型与多对多模型。



锁的优先级翻转异常与解决

- 假设有三个任务，任务1高优先级，任务2中优先级，任务3低优先级。



HarmonyOS 的消息队列

- 消息队列常用于消息的**异步处理**，消息的**缓冲**。对于任务间通信的场景，消息发送方与消息接受方实现**解耦**。
- HarmonyOS消息队列最大消息内容1KB，POSIX最大消息内容64字节；最大消息数量256个，POSIX最大消息数量16个。

思考题

1. (判断题)互斥锁是用于保证数据操作一致性的机制。(　　)

   A. 正确

   B. 错误

2. (判断题)消息队列可以实现异步消息处理，缓冲消息，对于任务间通信可以实现发送发与接收方的解耦。(　　)

   A. 正确

   B. 错误

思考题

3. (多选题)软件定时器的两个主要功能是什么？(　　)

A. 制定任务循环执行的周期

B. 用于任务的延迟执行

C. 区分不同任务的优先级

D. 给资源文件加锁

本章总结

- 本章内容是HarmonyOS设备开发的内核基础知识，描述了HarmonyOS内核的基础内容，涵盖进程与线程的关系与调度策略，HarmonyOS设备的内存管理，网络模块与文件系统等知识点。

学习推荐

- HarmonyOS官网社区：
  - https://www.harmonyos.com/cn/home/
- HarmonyOS应用开发文档：
  - https://developer.harmonyos.com/cn/home/
- HarmonyOS设备开发文档：
  - https://device.harmonyos.com/cn/home/
- OpenHarmony开源地址：
  - https://gitee.com/openharmony
- 华为人才在线：
  - https://e.huawei.com/cn/talent/#/

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

驱动基础

HUAWEI

前言

- 操作系统的一个关键作用，就是对硬件设备进行管理，无论是IO设备、CPU算力、内存资源这些都需要由硬件来承接，而驱动的作用就是可以使得硬件设备能够正常运转。
- HarmonyOS面向万物互联时代，而万物互联涉及到了大量的硬件设备，而这些硬件的离散度很高，它们的性能差异与配置差异都很大，所以这要求使用一个更灵活、功能更强大、能耗更低的驱动框架。
- 本章节将介绍有关驱动的基础知识，同时还将介绍HDF驱动框架，使开发者能够对不同的驱动进行统一的管理。

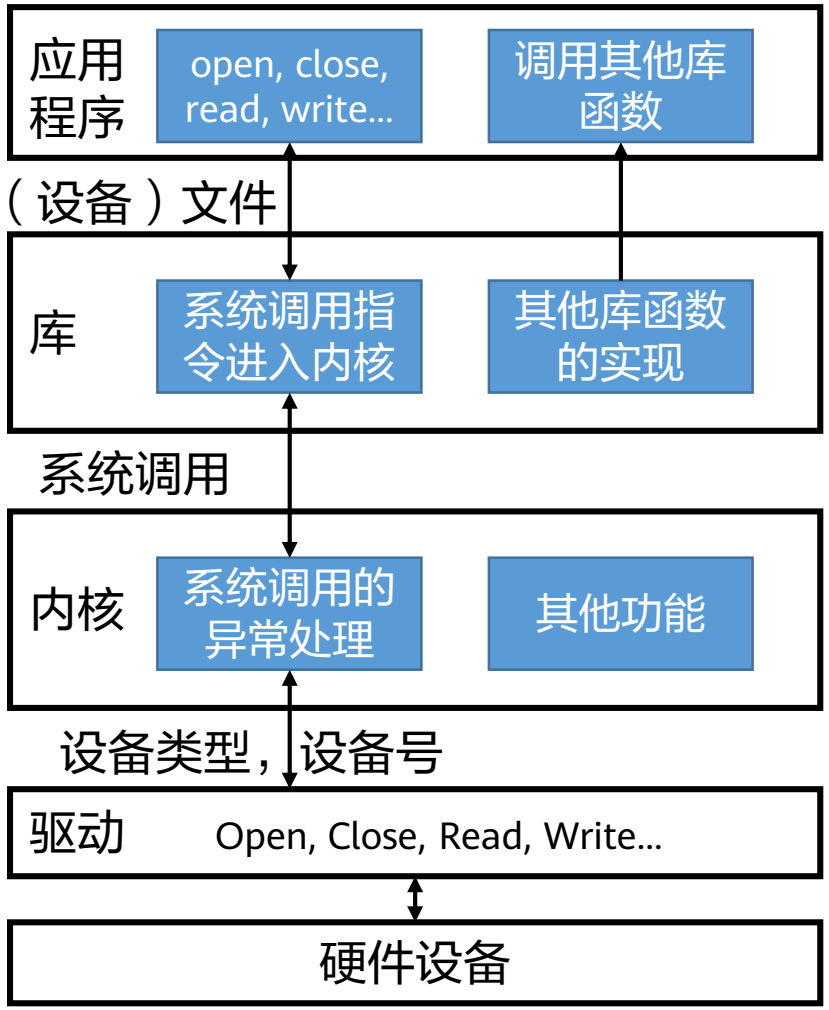
目标

- 学完本章内容后，您将能够：
  - 知晓驱动的概念；
  - 理清并区分HDF驱动框架的功能及其提供的能力；
  - 掌握不同平台驱动的基础知识并进行区分。

目录

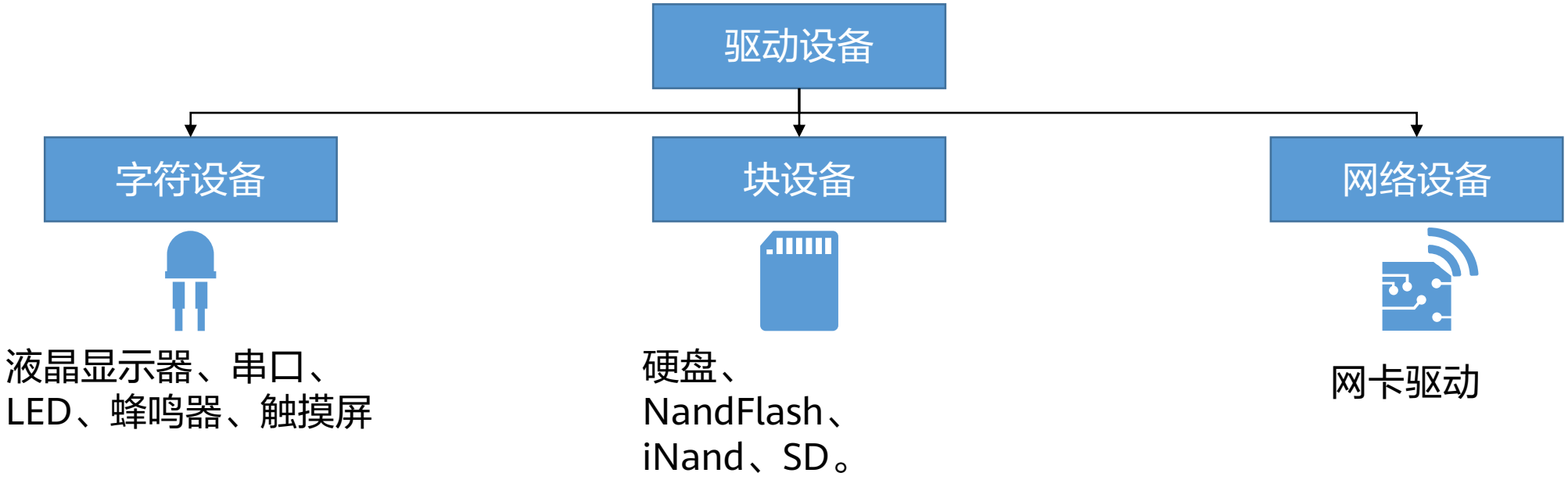
驱动概述

- 驱动程序全称**设备驱动程序**，是添加到操作系统中的特殊程序，其中包含有关硬件设备的信息。
- 驱动程序能使计算机与相应的设备进行通信。
- 不同的操作系统，硬件的驱动程序也各不相同。



设备驱动分类

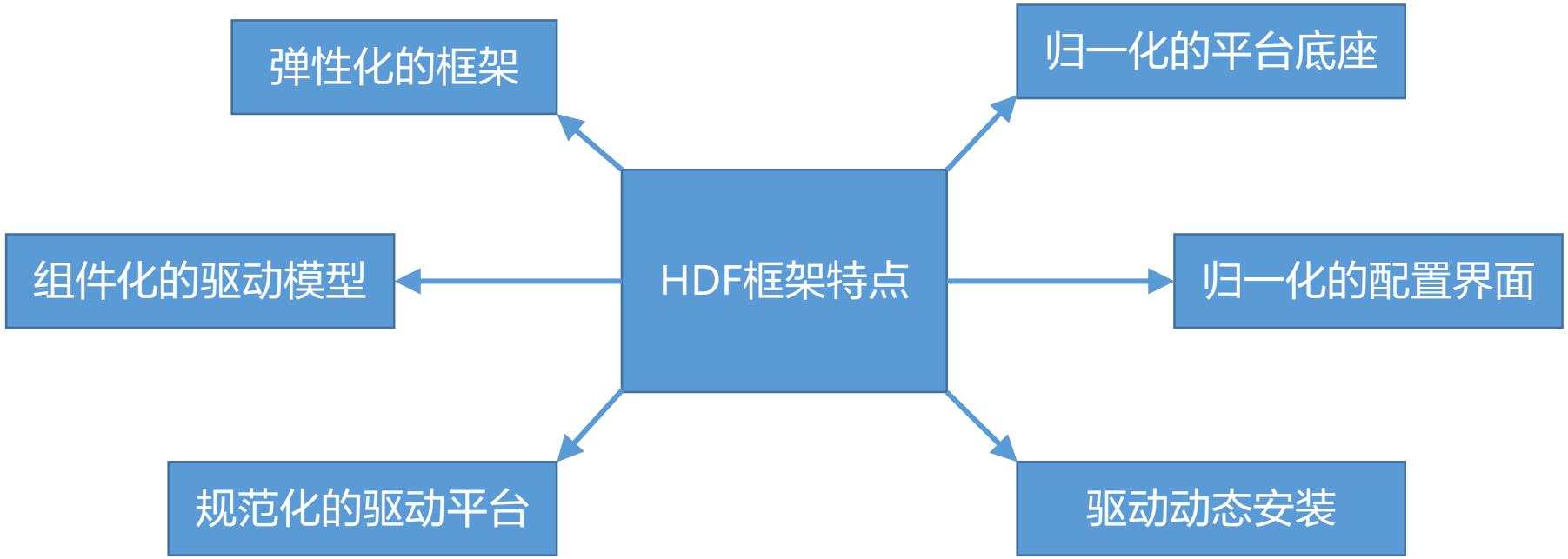
- 根据设备读写操作的特征差异，设备驱动大致上可以分为三类：
  - **字符设备**：即“字节设备”，软件操作设备时，以字节为单位进行。
  - **块设备**：设备的块大小是设备本身设计时定义好的，软件无法更改。
  - **网络设备**：网络设备是专为网卡设计的驱动模型，主要用于支持API中socket相关函数的工作。



目录

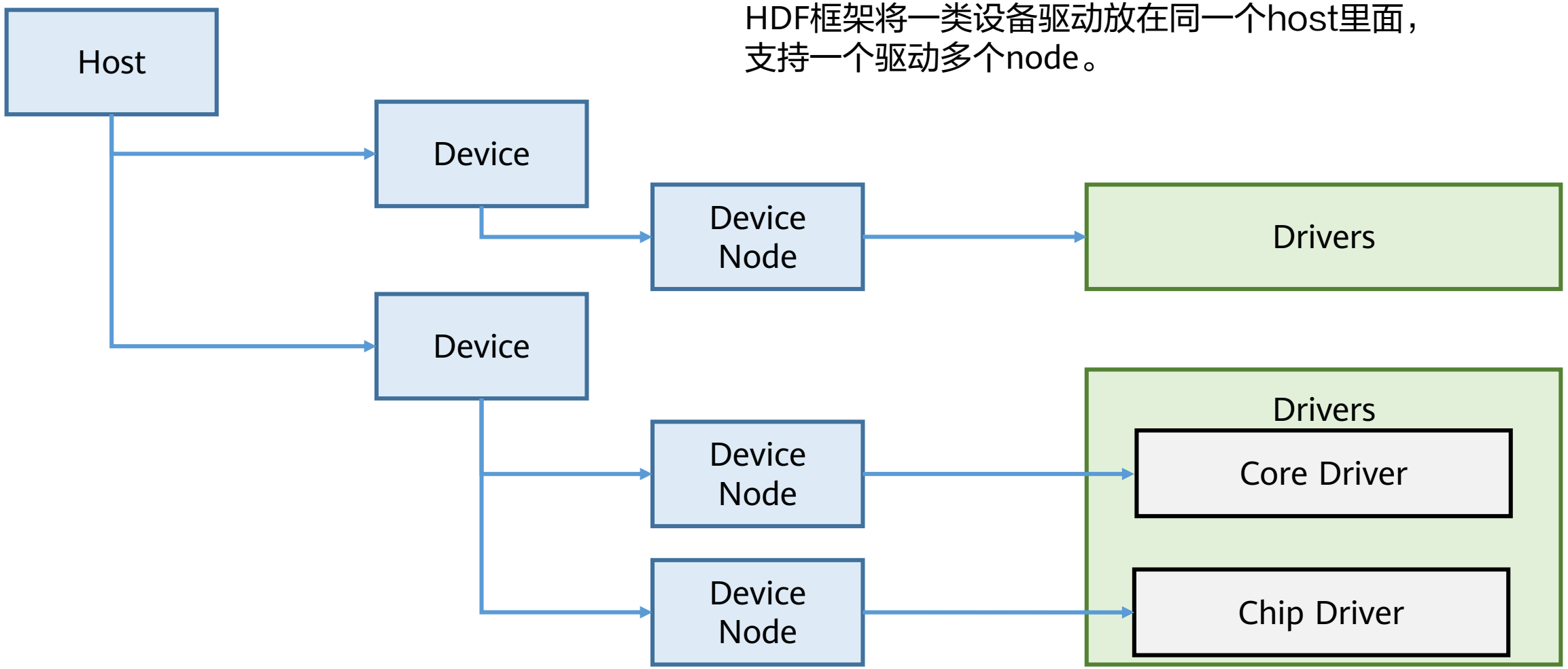
HDF 驱动模型

- HDF (Hardware Driver Foundation) 硬件驱动框架，为驱动开发者提供驱动框架能力，包括**驱动加载、驱动服务管理**和**驱动消息机制管理**。旨在构建统一的驱动架构平台，为驱动开发者提供更精准、更高效的开发环境，力求做到**一次开发，多系统部署**。



HDF 驱动模型图解

HDF框架将一类设备驱动放在同一个host里面，
支持一个驱动多个node。



驱动框架能力 - 驱动加载

- 按需加载：HDF框架定义了驱动按需加载方式的策略，是由配置文件中的preload字段来控制，preload字段的取值范围以及含义如下：

字段	取值	含义
PRELOAD	0	DEVICE_PRELOAD_ENABLE, 系统启动过程中默认加载；
	1	当系统支持快启的时候，则在系统系统完成之后再加载这一类驱动，否则和0含义相同
	2	DEVICE_PRELOAD_DISABLE, 系统启动过程中默认不加载，支持后续动态加载，当用户态获取驱动服务时，如果驱动服务不存在时，HDF框架会尝试动态加载该驱动；
	OTHER	DEVICE_PRELOAD_INVALID, 非法参数；

- 按序加载：配置文件中的priority（取值范围为整数0到200）是用来表示host和驱动的优先级。驱动的加载顺序，优先根据host的priority决定，其次是同一个host内驱动的priority值。

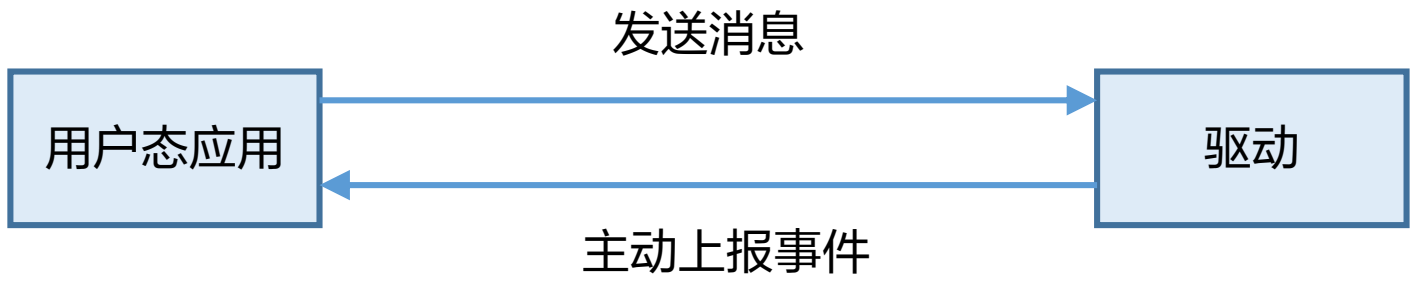
驱动框架能力 - 驱动服务管理

- 驱动服务是HDF驱动设备**对外提供能力的对象**，由HDF框架统一管理。
- 驱动服务管理主要包含驱动服务的**发布**和**获取**。
- HDF框架定义了驱动对外发布服务的策略，是由配置文件中的policy字段来控制，policy字段的取值范围以及含义如下：

字段	取值	含义
POLICY	0	SERVICE_POLICY_NONE, 驱动不提供服务；
	1	SERVICE_POLICY_PUBLIC, 驱动对内核态发布服务；
	2	SERVICE_POLICY_CAPACITY, 驱动对内核态和用户态都发布服务；
	3	SERVICE_POLICY_FRIENDLY, 驱动服务不对外发布服务，但可以被订阅；
	4	SERVICE_POLICY_PRIVATE, 驱动私有服务不对外发布服务，也不能被订阅；
	OTHER	SERVICE_POLICY_INVALID, 错误的服务策略

驱动框架能力 - 驱动消息机制管理

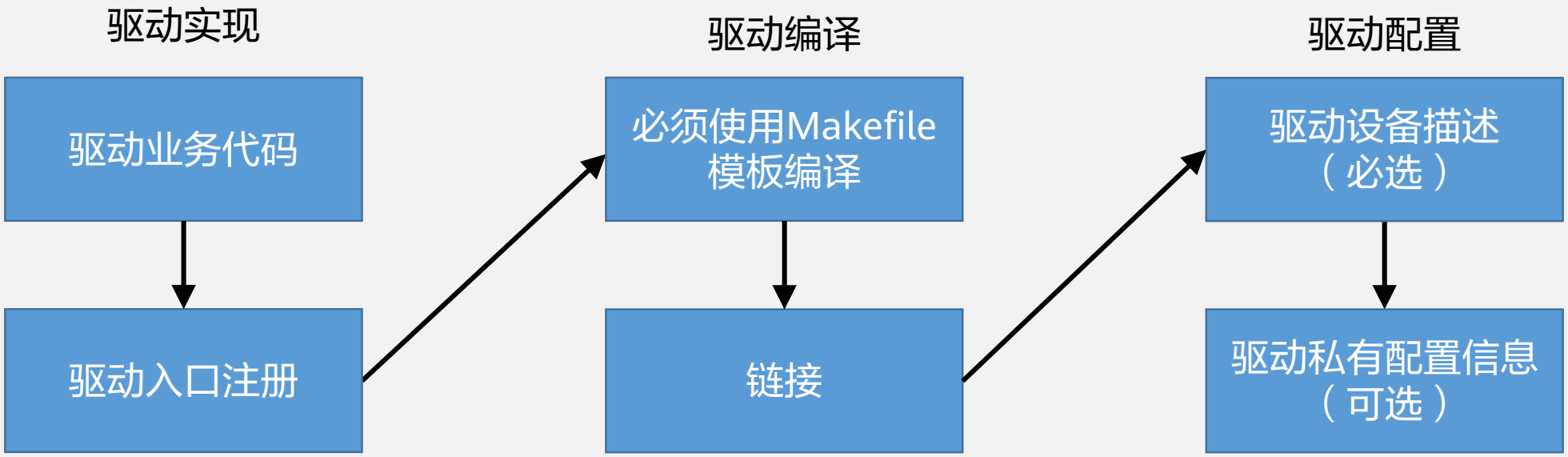
- 使用场景：
  - 当用户态应用和内核态驱动需要交互时，可以使用HDF框架的消息机制来实现。
- 消息机制的功能主要有以下两种：
  - 用户态应用发送消息到驱动。
  - 用户态应用接受驱动主动上报事件。



目录

驱动开发

- 基于HDF框架进行驱动的开发主要分为两个主要部分：驱动实现和驱动配置。



驱动业务代码

```
#define HDF_LOG_TAG "sample_driver"   // 打印日志所包含的标签，如果不定义则用默认定义的HDF_TAG标签
// 驱动对外提供的服务能力，将相关的服务接口绑定到HDF框架
int32_t HdfSampleDriverBind(struct HdfDeviceObject *deviceObject){
    HDF_LOGD("Sample driver bind success");
    return 0;
}
// 驱动自身业务初始的接口
int32_t HdfSampleDriverInit(struct HdfDeviceObject *deviceObject){
    HDF_LOGD(“Sample driver Init success”);
    return 0;
}
// 驱动资源释放的接口
void HdfSampleDriverRelease(struct HdfDeviceObject *deviceObject){
    HDF_LOGD("Sample driver release success");
    return;
}
```

驱动入口注册

```
// 定义驱动入口的对象，必须为HdfDriverEntry（在hdf_device_desc.h中定义）类型的全局变量
struct HdfDriverEntry g_sampleDriverEntry = {
    .moduleVersion = 1,
    .moduleName = "sample_driver",
    .Bind = HdfSampleDriverBind,
    .Init = HdfSampleDriverInit,
    .Release = HdfSampleDriverRelease,
};


// 调用HDF_INIT将驱动入口注册到HDF框架中，在加载驱动时HDF框架会先调用Bind函数,再调用Init函数加载该驱动，当Init调用异常时，HDF框架会调用Release释放驱动资源并退出。
HDF_INIT(g_sampleDriverEntry);
```

驱动编译

- 驱动代码的编译必须要使用HDF框架提供的Makefile模板进行编译：

```
include $(LITEOSTOPDIR)/../../drivers/adapter/lite/khdf/lite.mk #导入hdf预定义内容，必需
MODULE_NAME :=    #生成的结果文件
LOCAL_INCLUDE :=  #本驱动的头文件目录
LOCAL_SRCS :=     #本驱动的源代码文件
LOCAL_CFLAGS： =  #自定义的编译选项
include $(HDF_DRIVER) #导入模板makefile完成编译
```

- 编译结果文件链接到内核镜像，示例如下：

```
LITEOS_BASELIB +=  -lxxx  #链接生成的静态库
LIB_SUBDIRS    +=         #驱动代码Makefile的目录
```

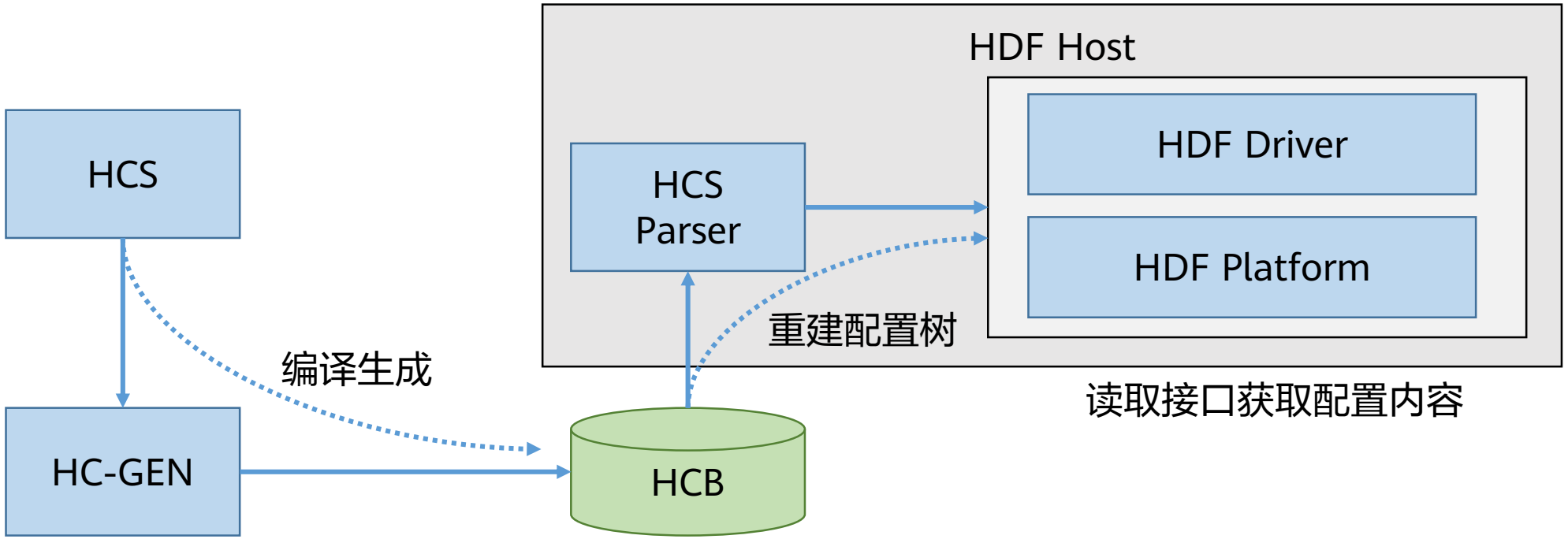
驱动设备描述

- HDF框架加载驱动所需要的信息来源于HDF框架定义的驱动设备描述，因此基于HDF框架开发的驱动必须要在HDF框架定义的device_info.hcs配置文件中添加对应的设备描述，驱动的设备描述填写如下所示：

```
sample_host :: host{
        hostName = "host0";    // host名称，host节点是用来存放某一类驱动的容器
        priority = 100;        // host启动优先级（0-200），值越大优先级越低，建议默认配100，优先级相同则不保证host的加载顺序
        device_sample :: device {        // sample设备节点
            device0 :: deviceNode {      // sample驱动的DeviceNode节点
                policy = 1;              // policy字段是驱动服务发布的策略
                priority = 100;          // 驱动启动优先级（0-200），值越大优先级越低，建议默认配100，优先级相同则不保证device的加载顺序
                preload = 0;             // 驱动按需加载字段
                permission = 0664;       // 驱动创建设备节点权限
                moduleName = "sample_driver";   // 驱动名称，该字段的值必须和驱动入口结构的moduleName值一致
                serviceName = "sample_service";    // 驱动对外发布服务的名称，必须唯一
                deviceMatchAttr = "sample_config"; // 驱动私有数据匹配的关键字，必须和驱动私有数据配置表中的match_attr值相等
```

配置管理

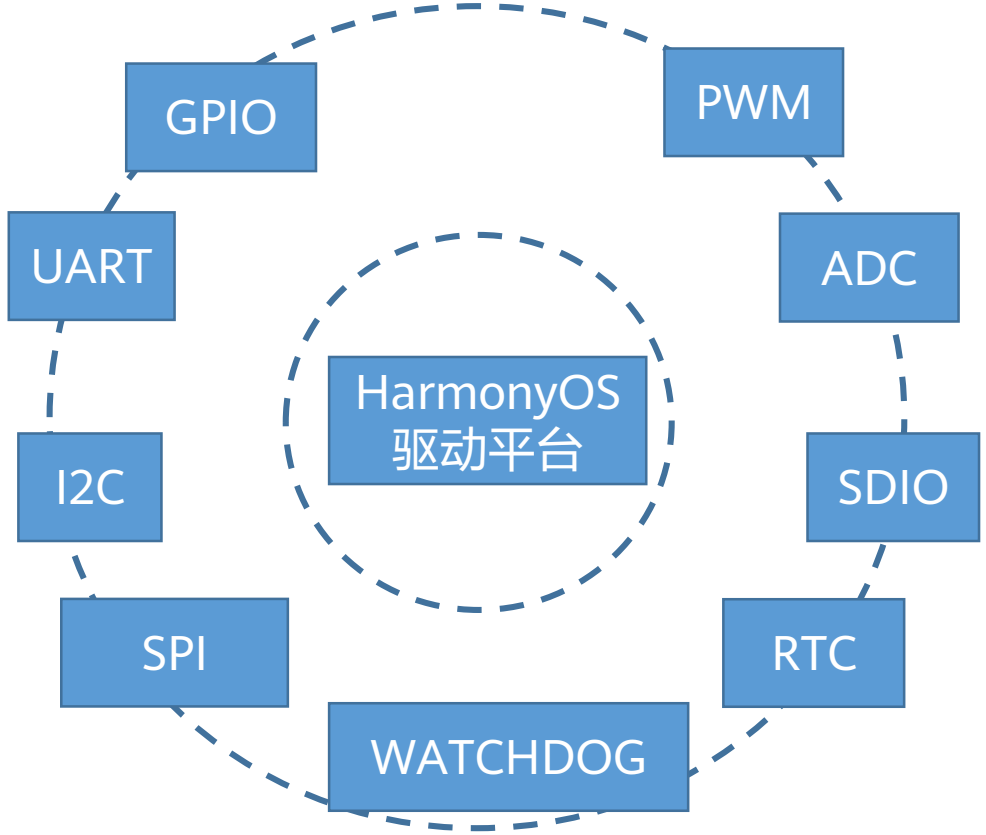
- HCS (HDF Configuration Source)是HDF驱动框架的配置描述源码，内容以Key-Value（键值对）为主要形式。配置代码与驱动代码解耦，便于配置管理。
- HC-GEN (HDF Configuration Generator) 是HCS配置转换工具。



目录

驱动平台总览

- HarmonyOS提供统一的驱动平台，用于实现串口控制、数据通信、时间控制、CPU复位、模数转换、脉冲宽度调制等功能。

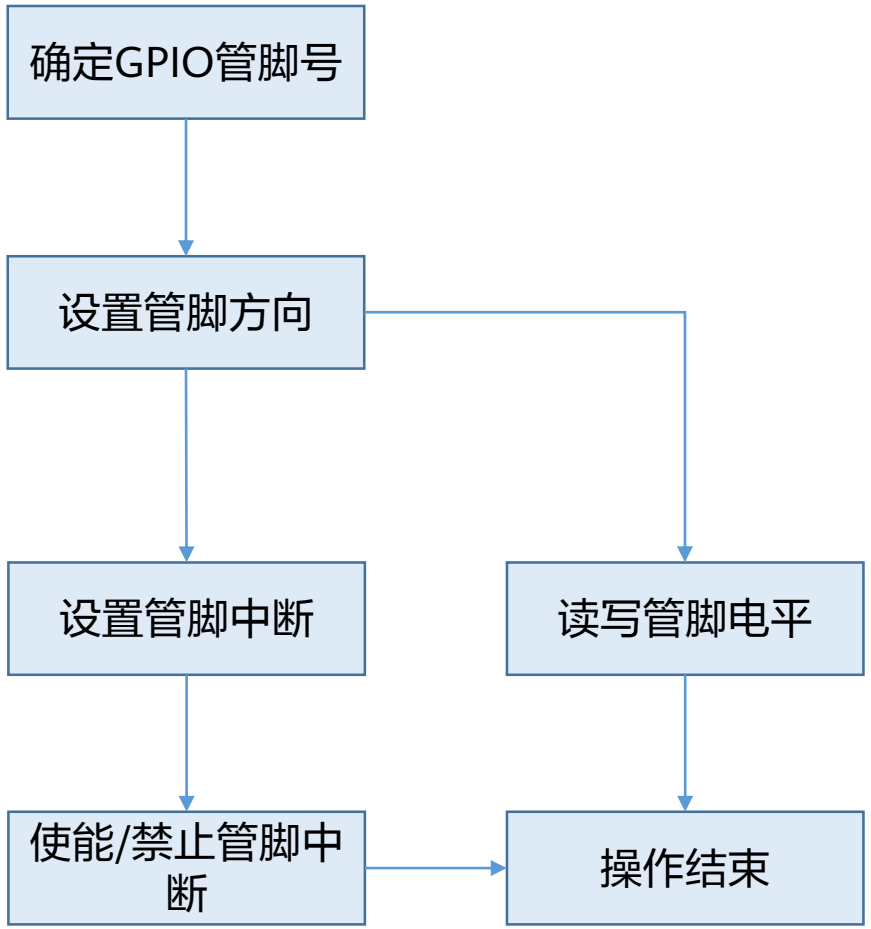


驱动平台 GPIO 简介

- GPIO（General-purpose input/output）即**通用型输入输出**。通俗地说，**GPIO口就是一些引脚**，可以通过它们输出高低电平或者读入引脚的高低电平状态。GPIO控制器通过分组的方式管理所有GPIO管脚，每组GPIO有一个或多个寄存器与之关联，通过**读写寄存器**完成对GPIO管脚的操作。
- GPIO是芯片上一根能完成多种功能的管脚，用户可以通过GPIO口和硬件进行**数据交互**（如UART），**控制硬件工作**（如LED，蜂鸣器等），**读取硬件的工作状态信号**（如中断信号）等。

Pin	管脚名称	类型	驱动(mA)	电压(V)	描述
2	GPIO_00	I_{SPU}/O	1	3.3/1.8	复用信号0：GPIO_00 复用信号1：UART1_TXD 复用信号2：SPI1_CLK 复用信号3：PWM3_OUT 复用信号4：I2C_SDA 复用信号5：RTC_OSC_32K 复用信号6：RTC32K_XOUT 复用信号7：保留

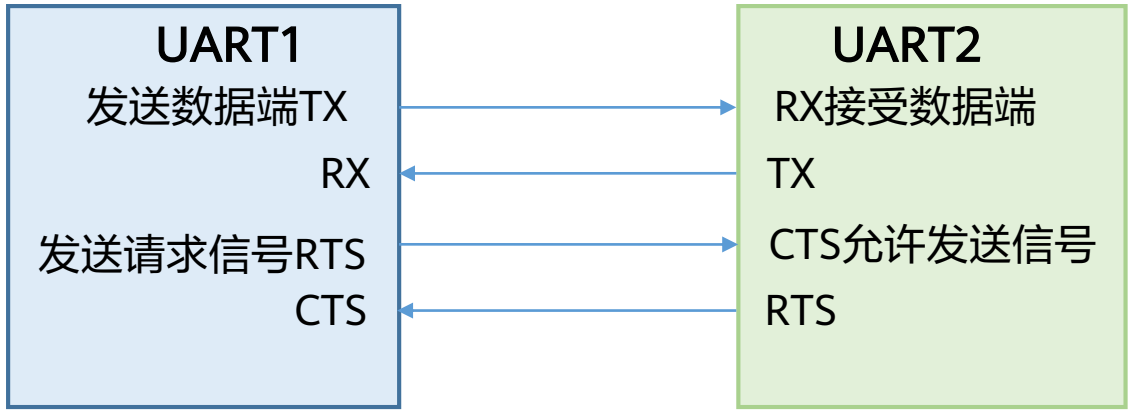
GPIO 接口说明



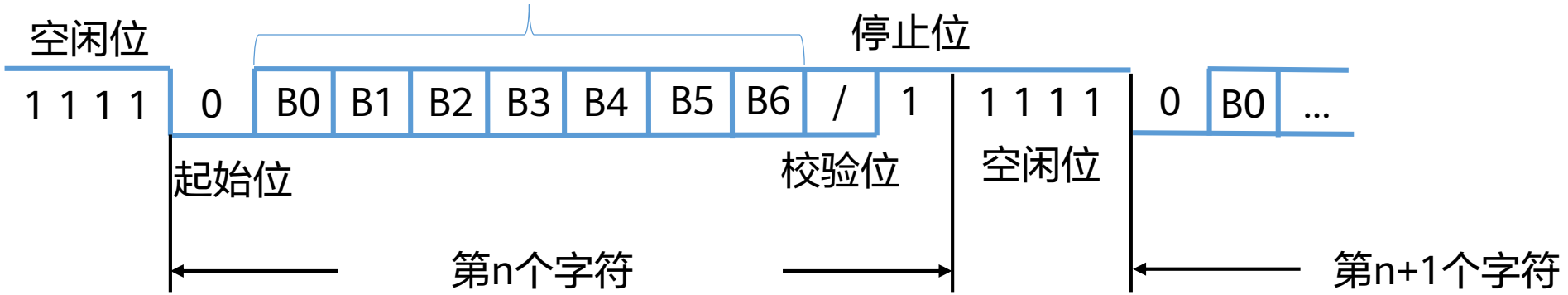
功能分类	接口名	描述
GPIO读写	GpioRead	读管脚电平值
	GpioWrite	写管脚电平值
GPIO配置	GpioSetDir	设置管脚方向
	GpioGetDir	获取管脚方向
GPIO中断设置	GpioSetIrq	设置管脚对应的中断服务函数
	GpioUnSetIrq	取消管脚对应的中断服务函数
	GpioEnableIrq	使能管脚中断
	GpioDisableIrq	禁止管脚中断

UART 简介

- UART是通用**异步收发传输器**(Universal Asynchronous Receiver/Transmitter)的缩写，是通用串行数据总线，用于**异步通信**。该总线双向通信，可以实现**全双工传输**。



4线UART设备连接示意



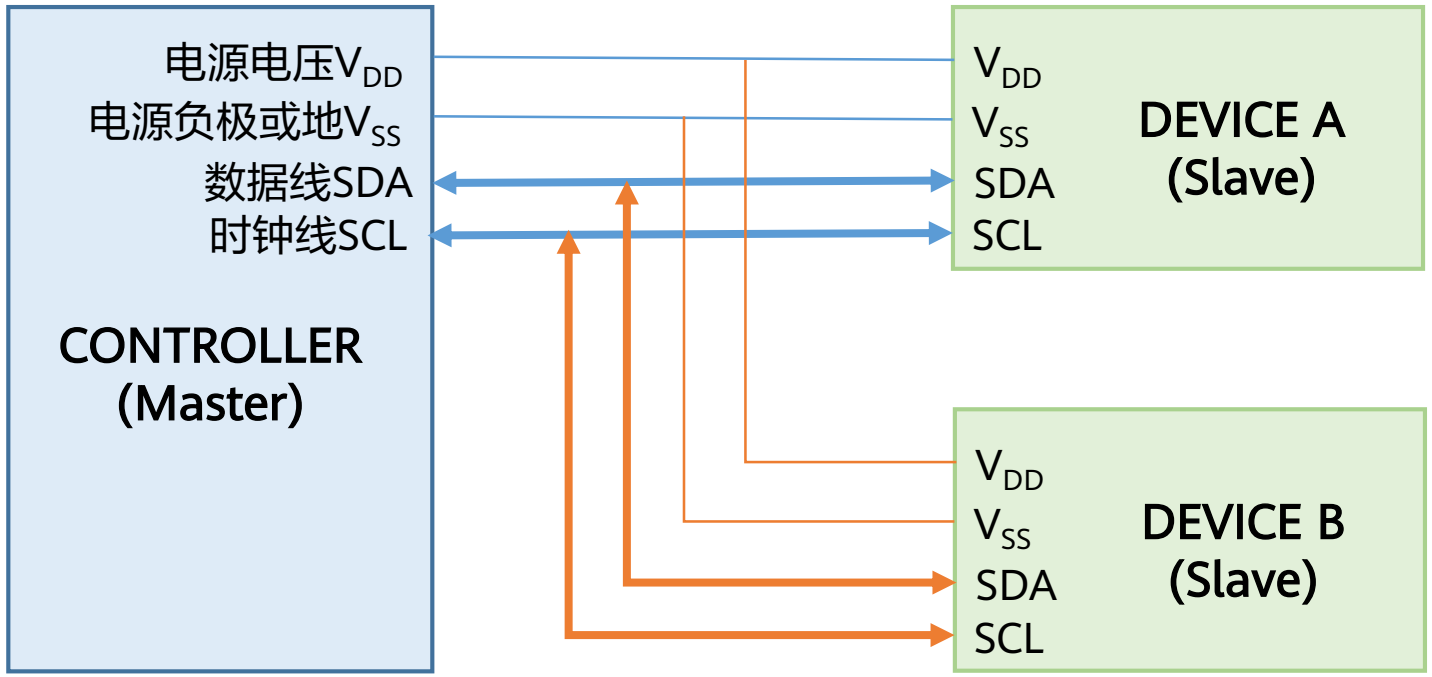
UART 接口说明

- UART的所有接口仅限内核态使用，不支持在用户态使用。

功能分类	接口名	描述
UART获取/释放设备句柄	UartOpen	UART获取设备句柄
	UartClose	UART释放设备句柄
UART读写接口	UartRead	从UART设备中读取指定长度的数据
	UartWrite	向UART设备中写入指定长度的数据
UART获取/设置波特率接口	UartGetBaud	UART获取波特率
	UartSetBaud	UART设置波特率
UART获取/设置设备属性	UartGetAttribute	UART获取设备属性
	UartSetAttribute	UART设置设备属性
UART设置传输模式	UartSetTransMode	UART设置传输模式

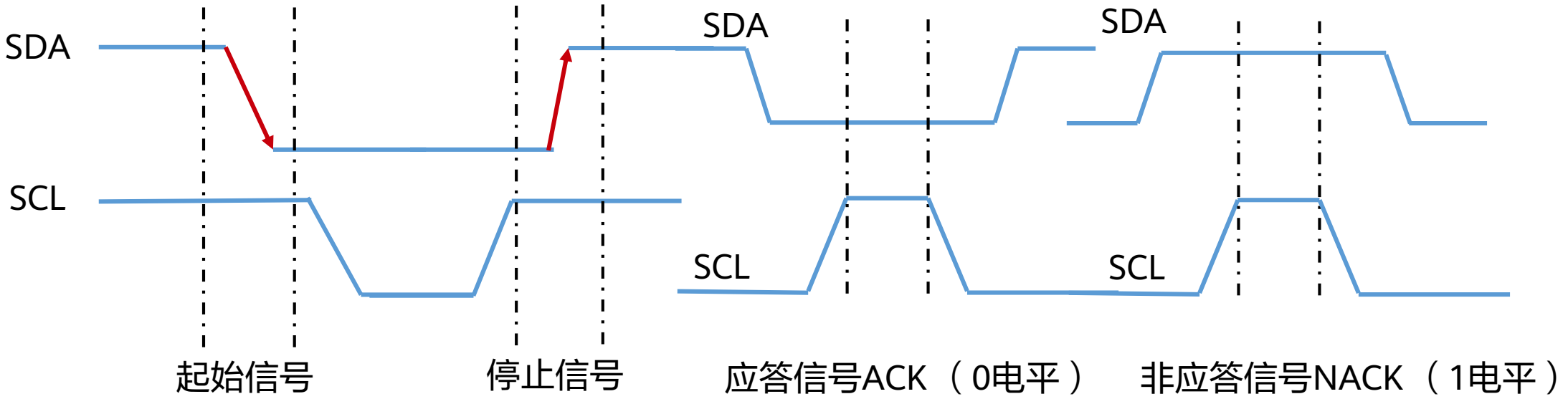
I2C 简介

- I2C (Inter Integrated Circuit) 集成电路间总线是由Philips公司开发的一种简单、双向二线制同步串行总线。
- I2C以主从方式工作，通常有一个主设备和一个或者多个从设备，主从设备通过SDA（SerialData）串行数据线以及SCL (SerialClock) 串行时钟线两根线相连。



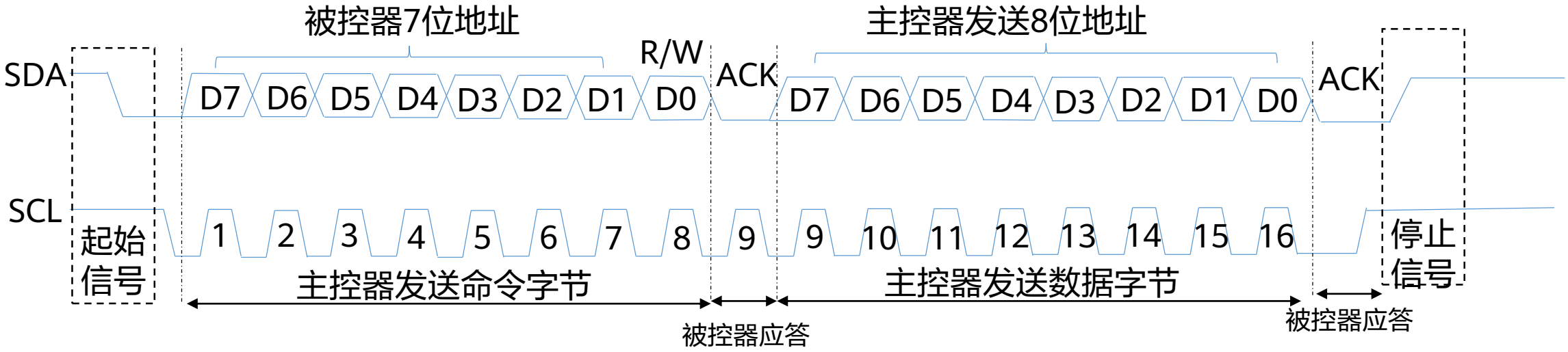
I2C 通信时序

- I2C时序主要有四个元素组成：起始信号，终止信号，应答(0)，非应答(1)。



I2C 数据传输

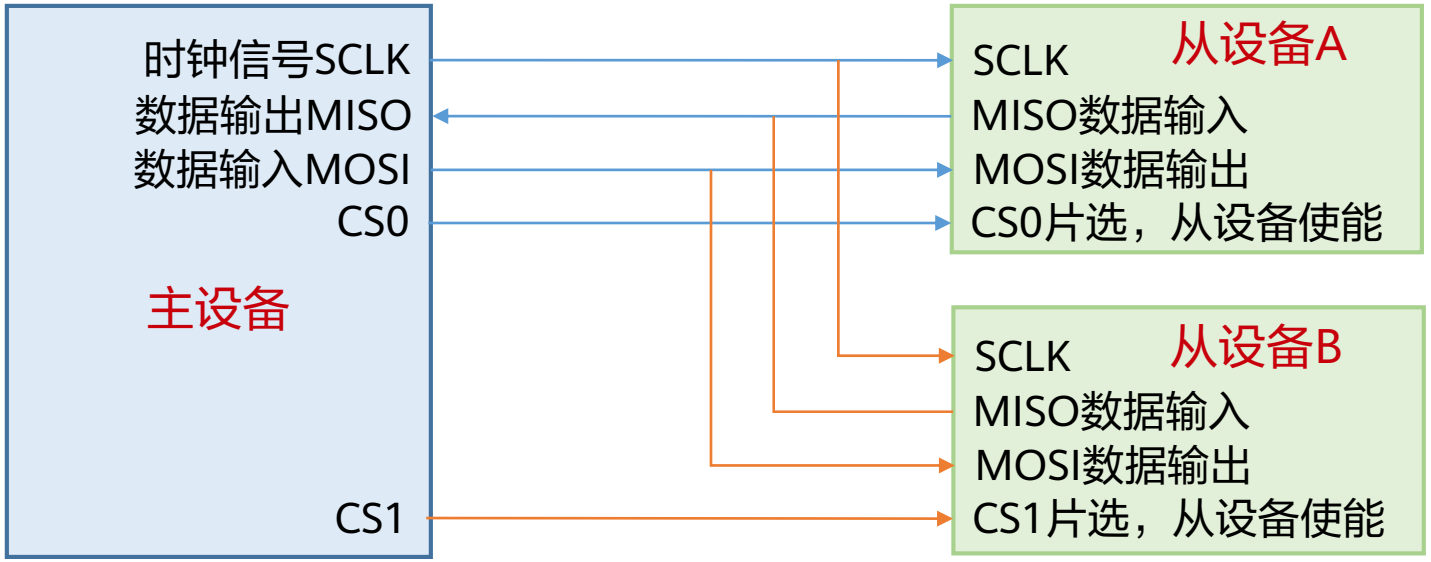
- I2C数据的传输必须以一个起始信号作为开始条件，以一个结束信号作为传输的停止条件。数据传输以字节为单位，高位在前，逐个bit进行传输。



SCL个数	1	7	1	1	8	1	1
SDA	起始信号	从设备地址	写操作	ACK信号	数据信号	ACK信号	停止信号
SDA被主机操作	主机写	主机写	主机写		主机写		主机写
SDA被从机操作				从机写		从机写	

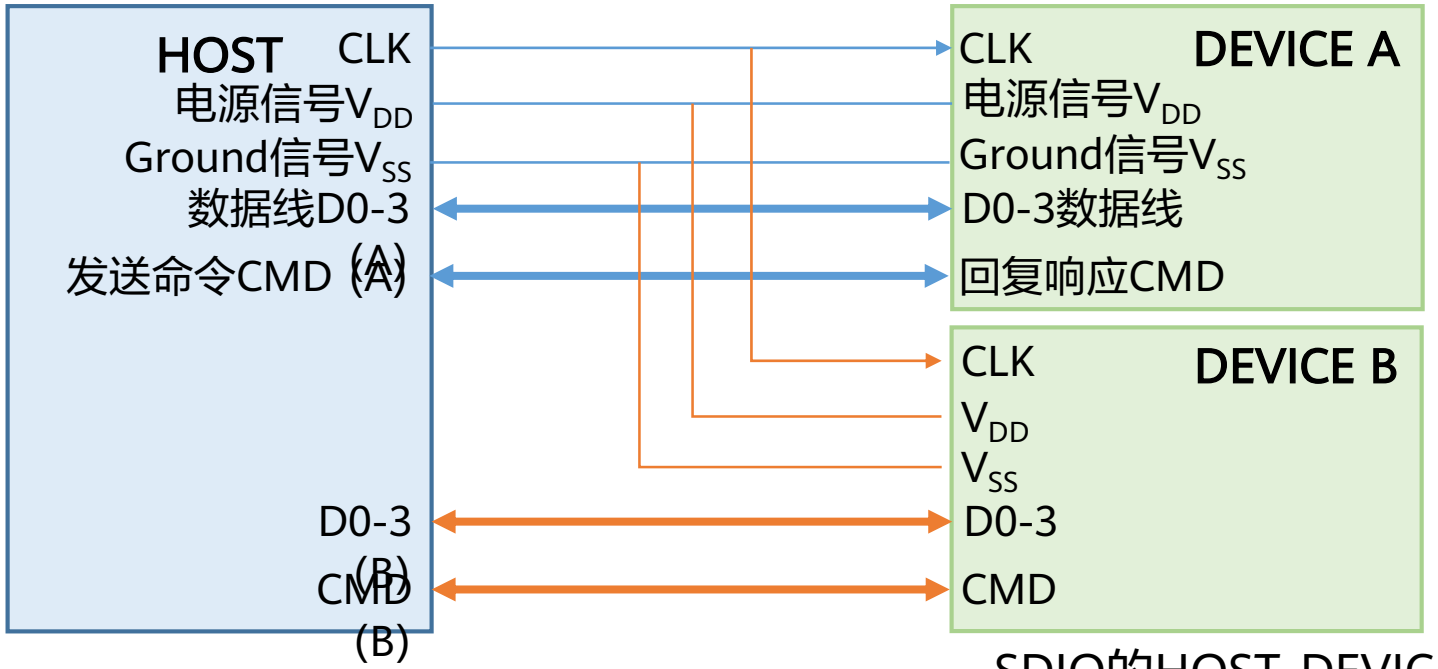
SPI 简介

- SPI是串行外设接口 (Serial Peripheral Interface) 的缩写，是一种高速的，全双工，同步的通信总线。
- SPI是由Motorola公司开发，用于在主设备和从设备之间进行通信，常用于与闪存、实时时钟、传感器以及模数转换器等进行通信。
- 一个主设备和两个从设备的连接示意图所示，从设备A和从设备B共享主设备的SCLK、MISO和MOSI三根引脚，从设备A的片选CS0连接主设备的CS0，从设备B的片选CS1连接主设备的CS1。



SDIO 简介

- SDIO是安全数字输入输出接口(Secure Digital Input and Output)的缩写，是从SD内存卡接口的基础上演化出来的一种外设接口。SDIO接口兼容以前的SD内存卡，并且可以连接支持SDIO接口的设备。
- SDIO常用于移动端设备例如手机的外设开发，使得设备外接外设更加容易，常见的SDIO外设有WLAN设备、GPS、CAMERA、蓝牙。



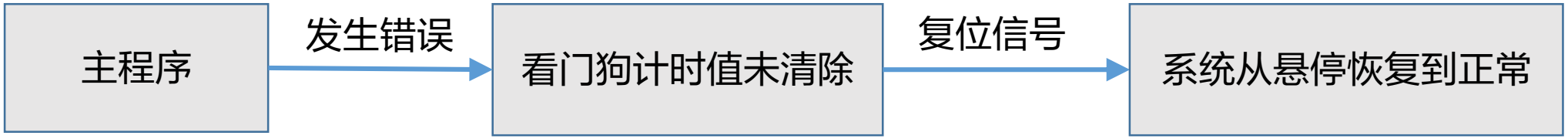
SDIO的HOST-DEVICE连接示意图

RTC 简介

- RTC(real-time clock)为操作系统中的实时时钟设备，为操作系统提供**精准的实时时间**和**定时报警功能**。当设备下电后，通过外置电池供电，RTC继续记录操作系统时间；设备上电后，RTC提供实时时钟给操作系统，确保断电后系统时间的连续性。
- 以STM32F103芯片为例，其内部有独立供电的实时时钟和单独的振荡电路，无需外接时钟芯片。

WATCHDOG

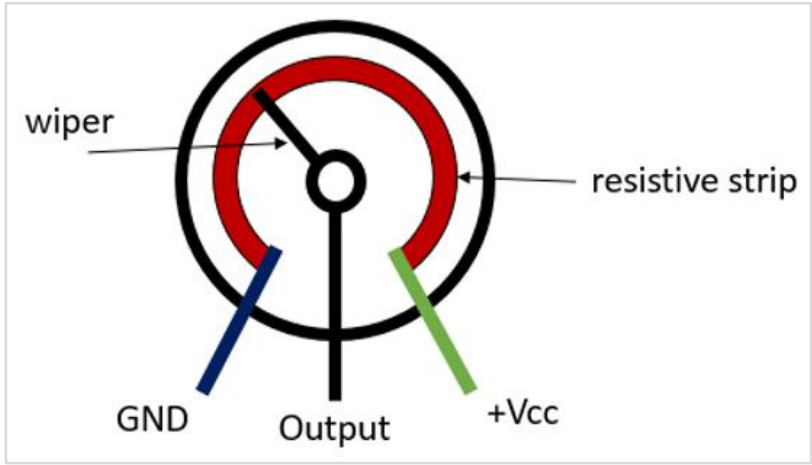
- 看门狗（watchdog)，又叫看门狗计时器（watchdog timer)，是一种硬件的计时设备，当系统的主程序发生某些错误时，导致未及时清除看门狗计时器的计时值，这时看门狗计时器就会对系统发出复位信号，使系统从悬停状态恢复到正常运作状态。
- 看门狗可以看成是一种**特殊的定时器**，只不过定时时间到了之后**不只是产生中断**，还可以**复位CPU**。



ADC

- ADC（Analog to Digital Converter）**模数转换器**。现实生活中的所有属性（如温度、湿度、光照强度等）都是连续的，即为**模拟信号**；而单片机或电子计算机所能识别的信号都是离散的数字信号。此时，若是需要使用现实世界中的各种属性，就需要一种设备将模拟信号转换为数字信号，它就是模数转换器。
- 模数转换一般要经过**采样**、**量化**和**编码**这几个步骤。

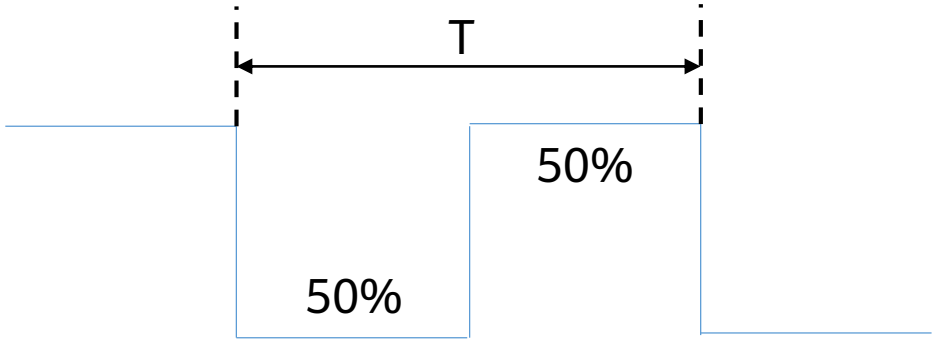
2-bit	电压值
00	0V
01	1.1V
10	2.2V
11	3.3V



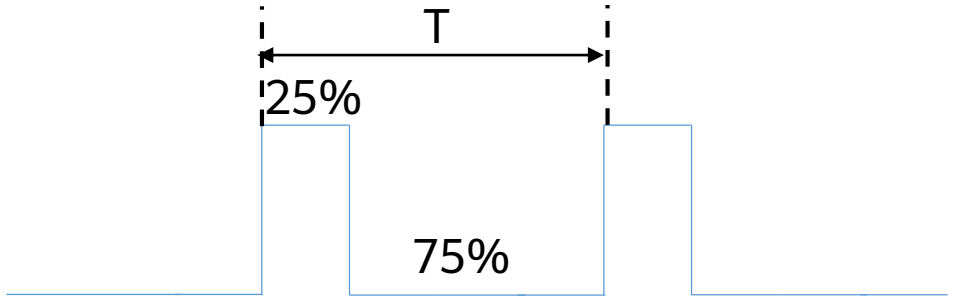
模拟电压表

PWM

- PWM (Pulse Width Modulation)又叫**脉冲宽度调制**，它是通过对一系列脉冲的宽度进行调制，**等效出所需要的波形**（包含形状以及幅值），对模拟信号电平进行数字编码，也就是说通过调节占空比的变化来调节信号、能量等的变化。占空比就是指在一个周期内，信号处于高电平的时间占据整个信号周期的百分比，例如方波的占空比就是50%。



占空比为50%的方波



占空比为25%的方波

$$占空比 = \frac{高电平时间}{周期时间} * 100\%$$

思考题

1. (判断题)I2C时序由四个元素组成：起始信号，终止信号，应答(0)，非应答(1)。(　　)

   A. 正确

   B. 错误

2. (判断题)ADC数模转换器可以将离散的数字信号转化为连续的模拟信号。(　　)

   A. 正确

   B. 错误

思考题

3. (单选题)根据设备读写操作的特征差异，设备驱动大致上可以分为几类？（ ）

A. 2

B. 3

C. 4

D. 5

本章总结

- 通过本章节的学习，您可以知晓驱动设备的概念以及区分不同种类的驱动设备，同时还了解了HarmonyOS中HDF驱动框架的使用及开发步骤。
- 除此之外，您还了解到了不同平台驱动的基础知识及其工作原理。同时还可以对不同的串口类驱动，时间类驱动，模数转换类驱动进行区分。

学习推荐

- HarmonyOS官网社区：
  - https://www.harmonyos.com/cn/home/
- HarmonyOS应用开发文档：
  - https://developer.harmonyos.com/cn/home/
- HarmonyOS设备开发文档：
  - https://device.harmonyos.com/cn/home/
- OpenHarmony开源地址：
  - https://gitee.com/openharmony
- 华为人才在线：
  - https://e.huawei.com/cn/talent/#/

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

基础子系统开发

HUAWEI

前言

- HarmonyOS将一些常用的功能划分为独立的子系统，以便于更好的实现相关功能。基础的子系统包括：编译构建子系统，分布式远程启动子系统，公共基础子系统，OTA升级子系统，启动恢复子系统，软总线子系统。
- 本章将介绍硬件设备开发中常用的子系统，使开发者能对HarmonyOS的子系统有一个基础的了解。

目标

- 学完本章内容后，您将能够：
  - 了解HarmonyOS子系统整体构成；
  - 熟悉设备开发常用的数个子系统；
  - 掌握基础子系统的各项特性。

目录

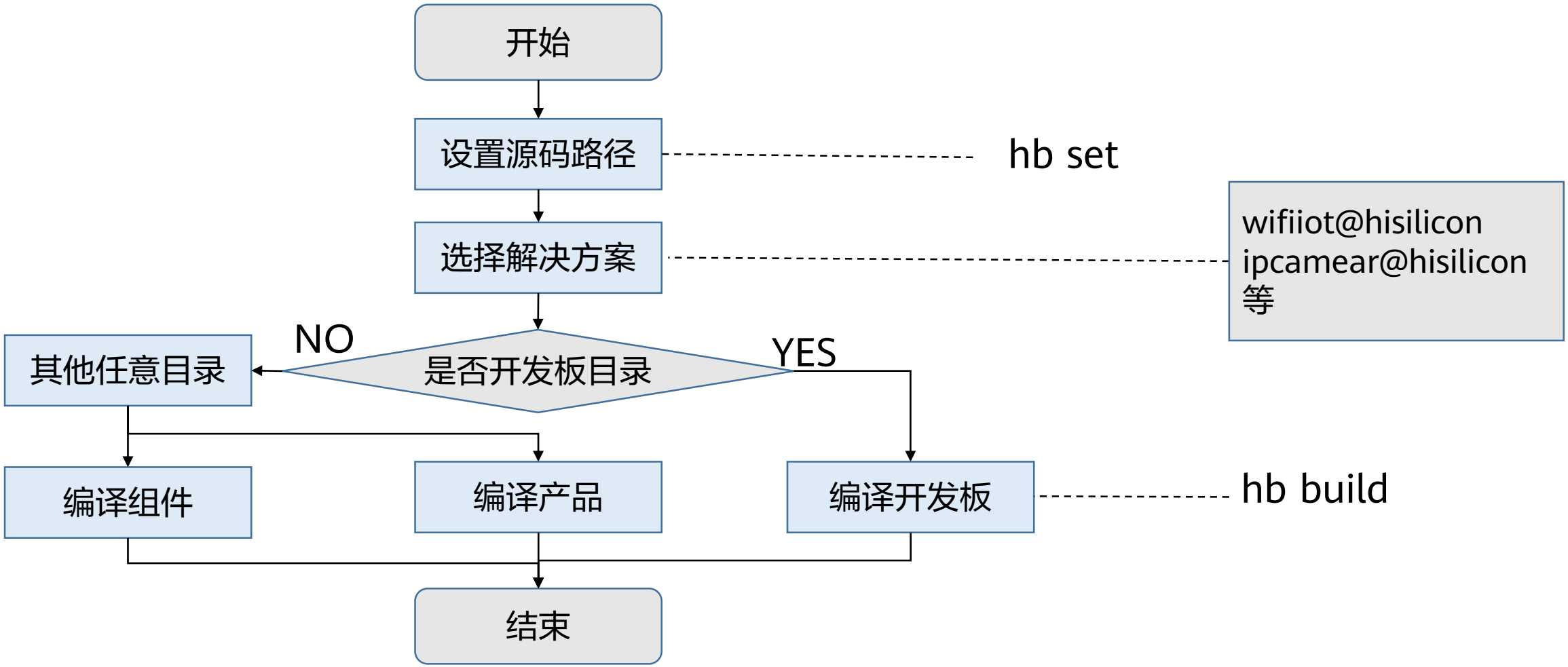
编译构建子系统是什么?

- 编译构建子系统是基于gn和ninja，支持OpenHarmony组件化开发的**编译框架**，主要提供以下功能：
  - **构建已有产品**。
  - **独立构建芯片厂商源码**。
  - **独立构建单个组件**。
- 在开发编译构建前，应了解如下基本概念：
  - **组件**：可复用的软件单元，它可包含源码、配置文件、资源文件和编译脚本等。
  - **Ninja**：ninja是一个专注于速度的小型构建系统。
  - **Gn**：Generate ninja的缩写，用于产生ninja文件。

Ninja 和 Gn 概述

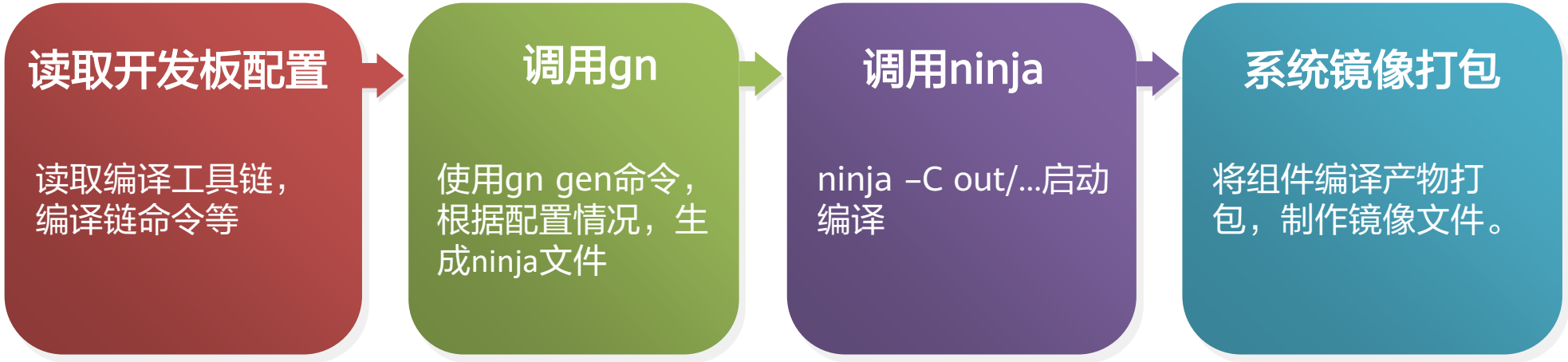
- Ninja是一个致力于速度的小型编译系统（类似于Make编译工具）。
- Ninja主要有两个特点：
  - 可以通过其他高级的编译系统生成其输入文件。
  - 它的设计就是为了**更快的编译**。
- Ninja核心是由C/C++编写的，同时有一部分辅助功能由python和shell实现，可以利用Ninja的开源代码进行各种**个性化的编译定制**。
- Gn：Generate ninja，用于产生ninja文件。

编译构建流程



设置与编译

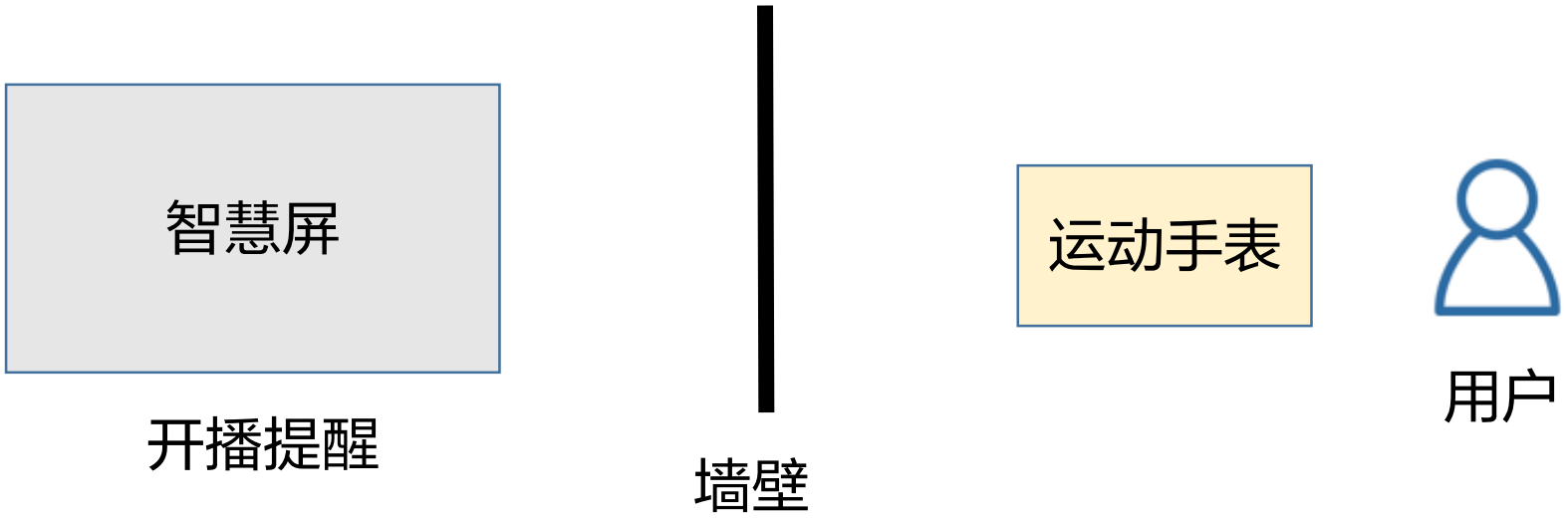
- hb set命令功能：设置HarmonyOS源码目录和要编译的产品。
- hb build命令功能：编译产品、开发板或者组件。
- 解决方案编译实现过程：



目录

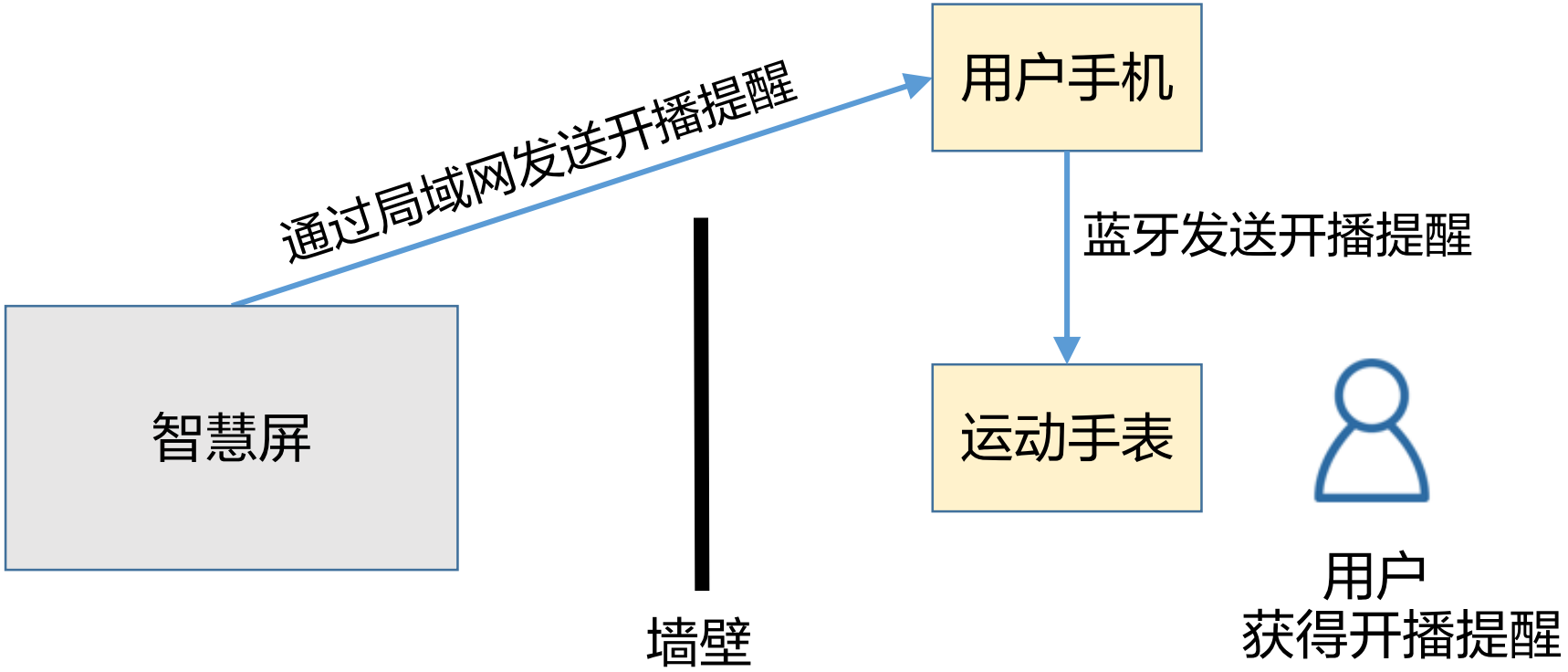
分布式远程启动场景案例

- 现有如下场景：某用户想看某节目，于是在智慧屏上点击了节目开播后提醒，之后用户去了另一个房间，智慧屏无法直接提醒用户收看节目。
- 场景条件：智慧屏幕可连接局域网，运动手表只能通过蓝牙连接，用户的手机在房间内充电，用户希望通过运动手表获得提醒。



分布式远程启动场景案例

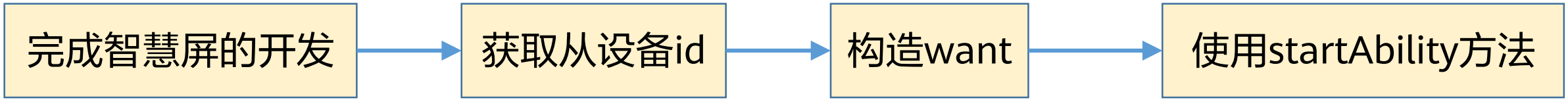
- 解决方案：通过分布式软总线的方式，让智慧屏的消息传达到用户的运动手表上，智慧屏远程启动运动手表，给用户做开播提醒。



分布式远程启动是什么？

- 分布式远程启动由分布式任务调度模块执行，它通过**主从设备服务代理机制**，在HarmonyOS操作系统上建立起分布式服务平台，支持主设备（搭载HarmonyOS的智慧屏设备）启动从设备（IP Camera、运动手表等小内存HarmonyOS设备）FA（Feature Ability）的能力。
- **基本概念**：
  - **FA**：代表有界面的Ability，用于与用户进行交互。
  - **远程启动**：即跨设备启动FA，与本地启动FA相对应。

远程启动步骤



want参数：远端设备id，包名，ability类名。

```
// 启动远程设备FA
Want want = new Want();     // 封装启动远端FA的Want
// 使用之前获取的设备ID，并指定FA信息
ElementName name = new
ElementName(remote_device_id,”com.Huawei.remote_package_name”,”remote_class_name”);
want.setElement(name);        // 将待启动的FA信息添加到Want中
want.setFlags(Want.FLAG_ABILITYSLICE_MULTI_DEVICE);      // 设置分布式标记，若不设置则无法使用分布式能力
startAbility(want);      // 按照Want启动指定FA，Want参数命名以实际开发平台的API为准
```

- 支持主设备侧远程启动从设备侧FA，不支持从设备远程启动主设备FA。
- 远程启动前必须确保设备间分布式组网成功（需要在同一网段内，可互相ping通），否则无法远程启动。
- 当前只支持拥有共同公钥信息的主从设备间FA（即主从设备的FA使用相同华为证书）的拉起。

思考题

1. （判断题）分布式远程启动可以使得一个设备远程启动另一个设备的Ability。（　　）

   A. 正确

   B. 错误

2. （判断题）分布式远程启动使用的方法是startAbility方法。（　　）

   A. 正确

   B. 错误

思考题

3. （多选题）远程启动FA的时候，需要指定FA的哪些参数？（　　）

A. 设备id

B. 应用包名

C. 要启动的ability类名

D. 应用的安装时间

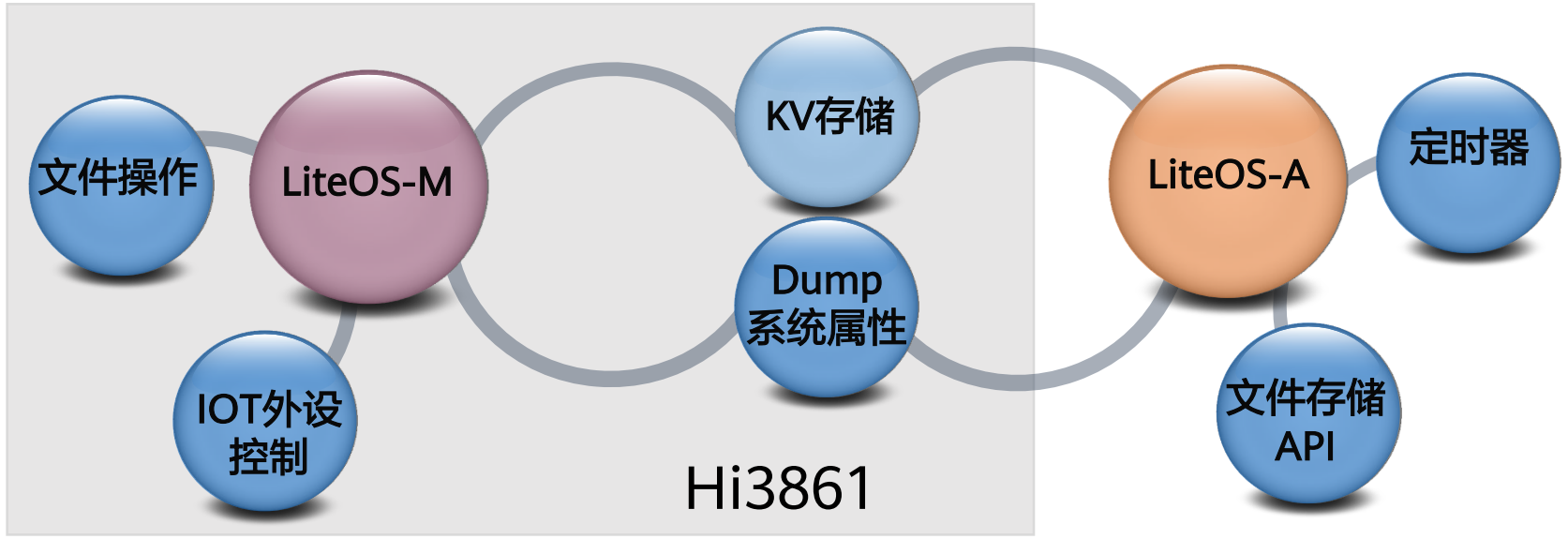
目录

公共基础使用场景

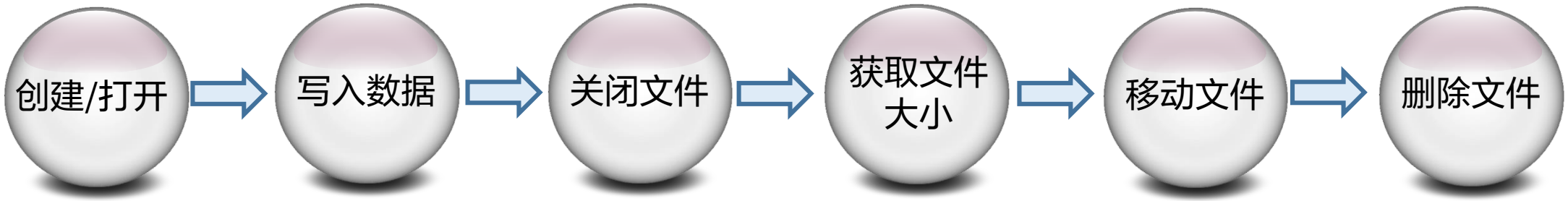
- 案例需求：某开发人员希望能在Hi3861，Hi3516，Hi3518开发板上都实现一个功能类似的应用程序，其中都会涉及到相同的文件操作。开发人员希望减少自己的代码开发量，通用的操作可以直接调用现成的库函数。
- 解决方案：在HarmonyOS操作系统内将常见的文件操作，数据库操作都实现并封装成库函数，用以开发者方便快捷的调用，避免“重复造轮子”。

公共基础是什么?

- 公共基础库存放HarmonyOS通用的基础组件。这些基础组件可被HarmonyOS各业务子系统及上层应用所使用。
- 公共基础库在不同平台上提供对应的能力:



常见文件操作接口介绍



接口名称	主要功能
int UtilsFileOpen(const char* path, int oflag, int mode)	打开或创建文件
int UtilsFileWrite(int fd, const char *buf, unsigned int len)	向文件写入特定大小的数据
int UtilsFileClose(int fd)	关闭文件
int UtilsFileStat(const char *path, unsigned int *fileSize)	获取文件大小
int UtilsFileMove(const char* src, const char* dest)	将源文件移动到指定目标文件
int UtilsFileRead(int fd, char *buf, unsigned int len)	删除指定文件

文件操作使用案例

- 常见的文件操作包括：创建或打开文件，写入文件，关闭文件，删除文件，获取文件大小，移动文件。

```
// 文件打开与文件写入特定大小的数据
char fileName[] = “testfile”;
static const char def[] = “utils_file_operation implement.”;
int fd = UtilsFileOpen(fileName, O_RDWR_FS | O_CREAT_FS | O_TRUNC_FS, 0);
int ret = UtilsFileWrite(fd, def, strlen(def));
printf(“file handle = %d\n write ret = %d\n”, fd , ret);
// 读取特定长度的文件数据
char buf[64] = {0};
int readLen = UtilsFileRead(fd, buf, 64);
printf(“read len = %d : buf = %s\n”, readLen, buf);// 使用ret = UtilsFileClose(fd)将文件关闭。
// 获取文件大小
ret = UtilsFileStat(fileName, &fileLen);
printf(“file size = %d\n”, fileLen);
// 文件删除
ret = UtilsFileDelete(fileName);
printf("delete ret = %d\n", ret);
```

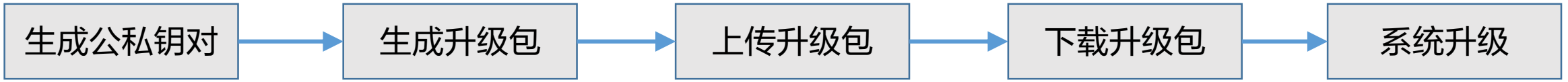
远程升级使用场景

- 案例场景：现有一批开发板上的应用程序都需要升级刷新，传统方式需要开发人员将每一块开发板挨个重新烧录。
- 业务痛点：效率低下，需要开发人员在现场进行重复机械的工作。
- 解决思路：如果开发板能够连上网络，能否通过网络连接的方式远程控制一批开发板同时升级?

升级服务是什么？

- **OTA(Over the Air)**提供对设备远程升级的能力，可以让各种设备（如IP摄像头等），轻松支持远程升级能力。
- 目前HarmonyOS仅支持全量包升级，暂不支持差分包升级。
  - 全量包升级是将新系统全部内容做成升级包，进行升级。
  - 差分包升级是将新老系统的差异内容做成升级包，进行升级。

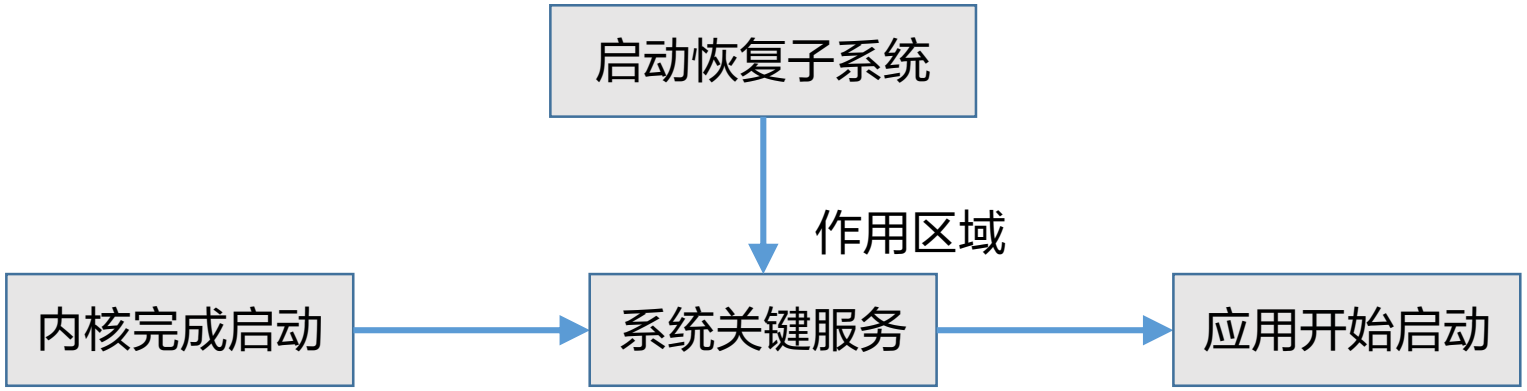
OTA升级步骤：

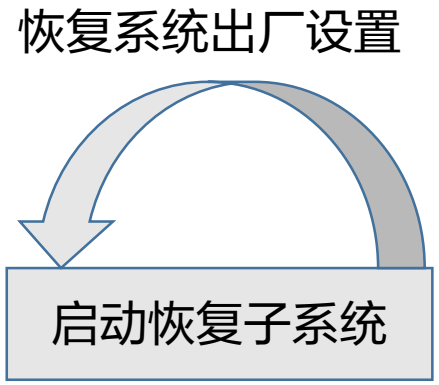


目录

启动恢复是什么？

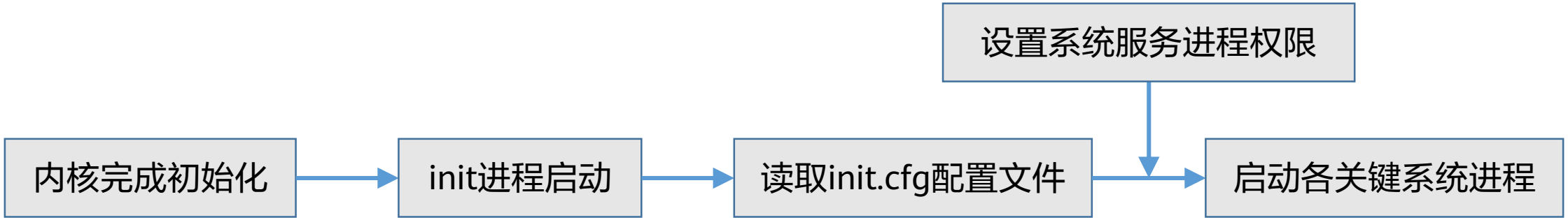
- 启动恢复子系统负责从内核启动之后到应用启动之前的**系统关键服务进程的启动过程**以及**设备恢复出厂设置**的功能。
- 涉及以下组件： Init启动引导组件，Appspawn启动引导组件， Bootstrap服务启动组件，Syapara系统属性组件， Startup启动组件。





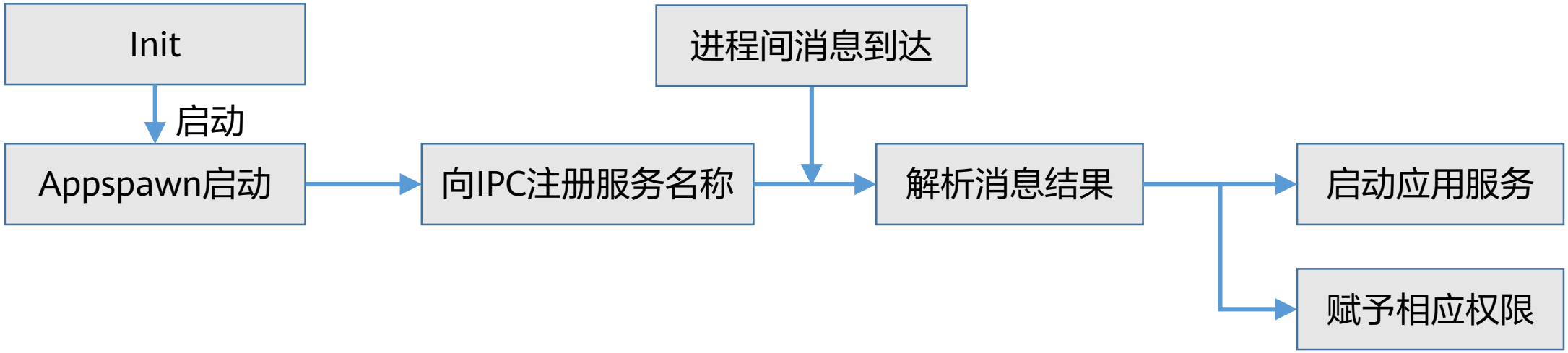
Init启动引导组件

- Init启动引导组件：
  - 主要功能：Init启动引导组件负责在系统启动阶段**启动关键服务进程**，若用户需要新增随开机自启动的系统服务，可将新增服务加入配置文件init.cfg中。
  - 运行机制：Init启动引导组件对应的进程为init进程，是内核完成初始化后启动的第一个用户态进程。**init进程启动之后，读取init.cfg配置文件**，根据解析结果，执行相应命令并依次启动各关键系统服务进程，在启动系统服务进程的同时设置其对应权限。



Appspawn启动引导组件

- Appspawn启动引导组件：
  - 主要功能：负责**接收用户程序框架的命令，孵化应用进程，设置新进程的权限，并调用应用程序框架的入口函数**。
  - 运行机制：Appspawn被Init启动后，向IPC（Inter-Process Communication跨进程通信）框架注册服务名称，之后等待接收进程间消息，根据消息解析结果启动应用服务并赋予其对应权限。

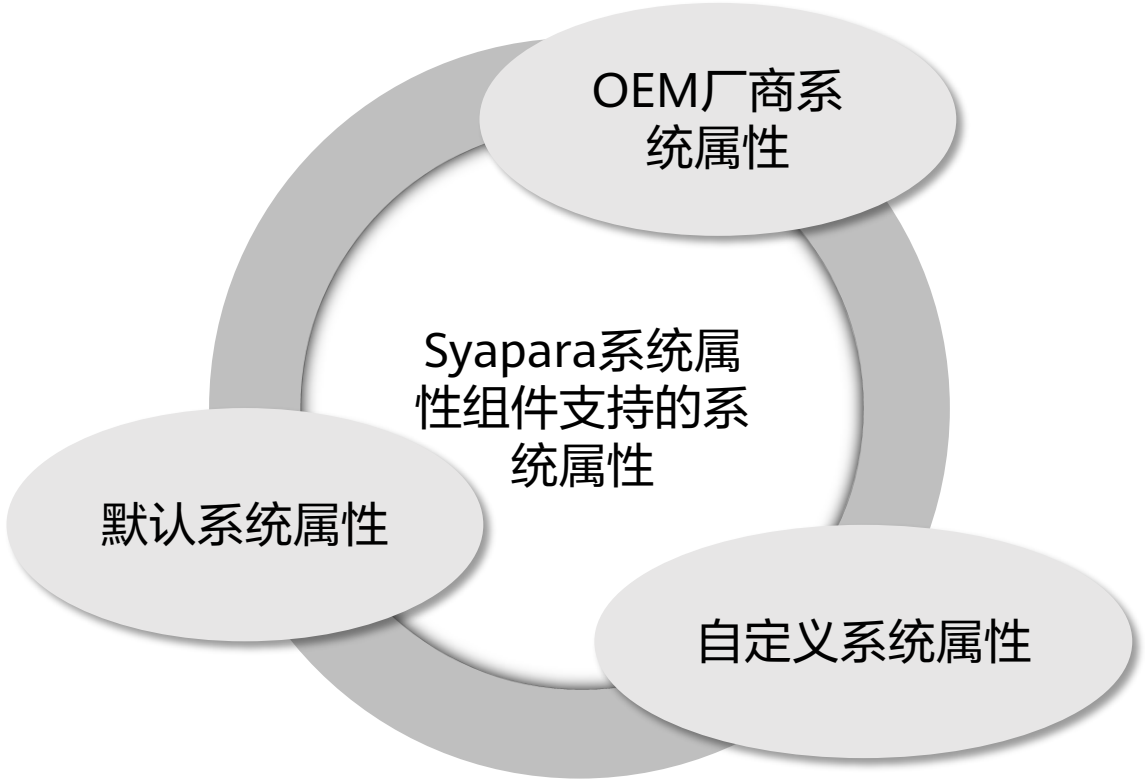


Bootstarp服务启动组件

- 主要功能：
  - 提供各服务和功能的启动入口标识。在SAMGR（系统服务框架子系统）启动时，会调用boostrap标识的入口函数，并启动系统服务。
- 主要特点：
  - **实现了服务的自动初始化**，即服务的初始化函数无需显式调用，而是将其使用宏定义的方式申明，就会在系统启动时自动被执行。
- 实现原理：
  - 将服务启动的函数通过宏申明之后，放在预定义好的zInit代码段中，系统启动的时候调用OHOS_SystemInit接口，遍历该代码段并调用其中的函数。

Syapara系统属性组件

- Syapara系统属性组件：
  - 主要功能：负责提供**获取与设置操作系统相关的系统属性**。
  - OEM厂商部分仅提供默认值，具体值需OEM产品方按需进行调整。



思考题

1. （判断题）启动恢复子系统在内核启动之后才发挥作用。(　　)

   A. 正确

   B. 错误

2. （判断题）大型系统支持的系统属性主要包括系统信息与设备信息。(　　)

   A. 正确

   B. 错误

思考题

3. （单选题）哪个启动引导组件负责在系统启动阶段启动关键服务进程？（　　）

A. Init启动引导组件

B. Appspawn启动引导组件

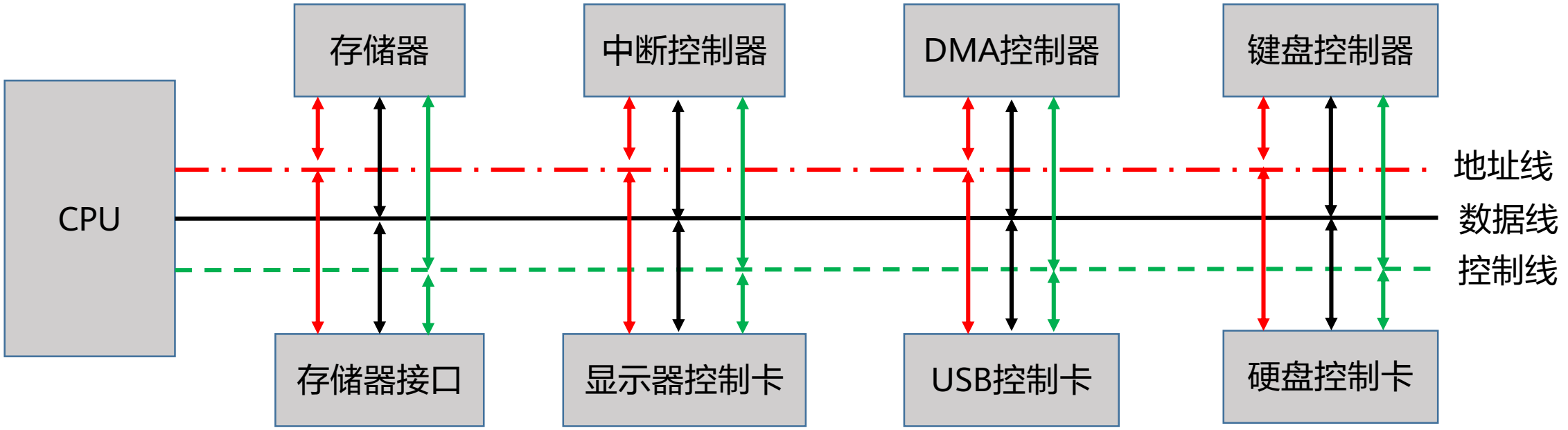
C. Bootstarp启动引导组件

D. Syapara启动引导组件

目录

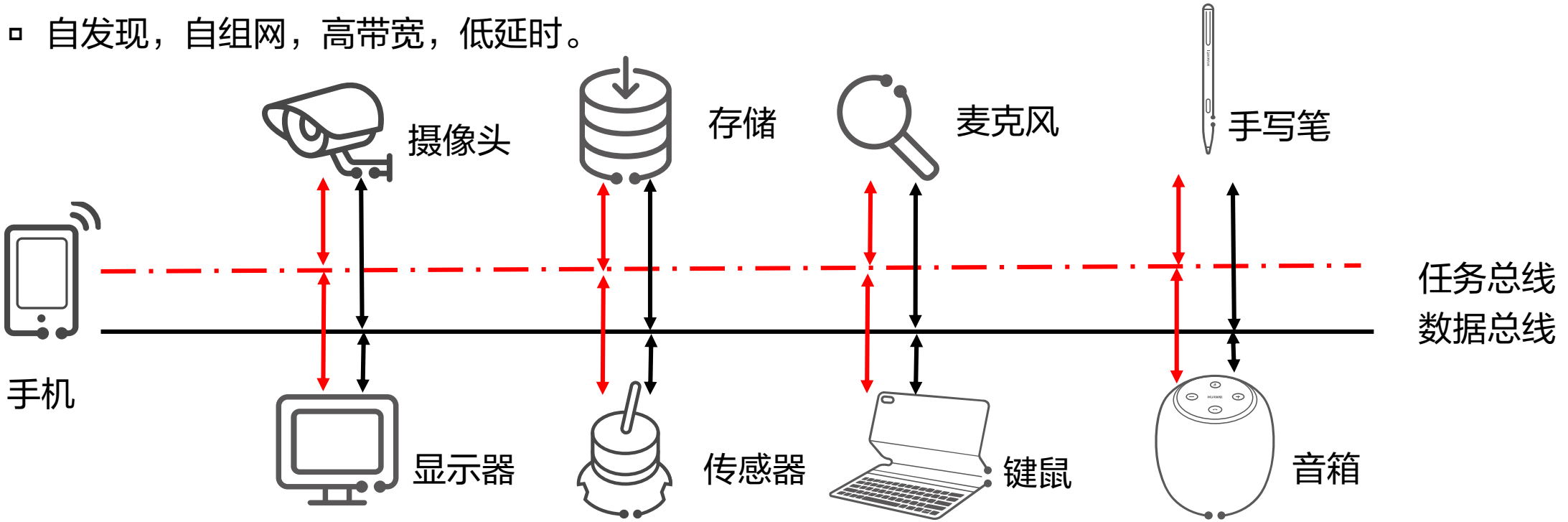
传统总线结构

- 传统的总线是一种内部结构，它连通着CPU，内存，输入输出设备，主机的各个部件共用这条通信通路。



软总线结构

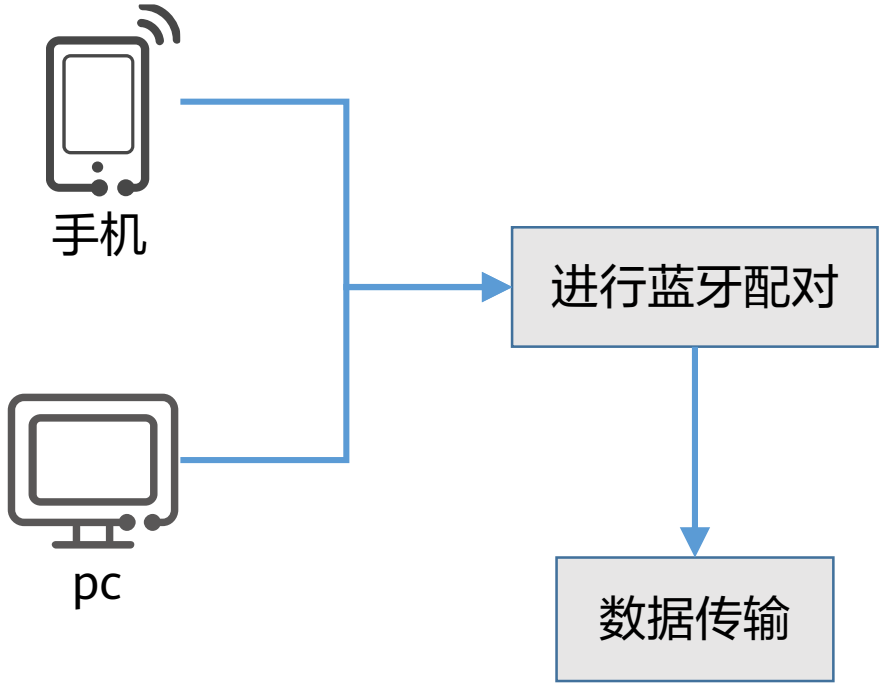
- 分布式软总线：
  - 连通的是1+8+N的独立设备。1个手机，8种设备（车机，音箱，耳机，手表/手环，平板，大屏，pc，AR/VR），N种IoT设备。
  - 自发现，自组网，高带宽，低延时。



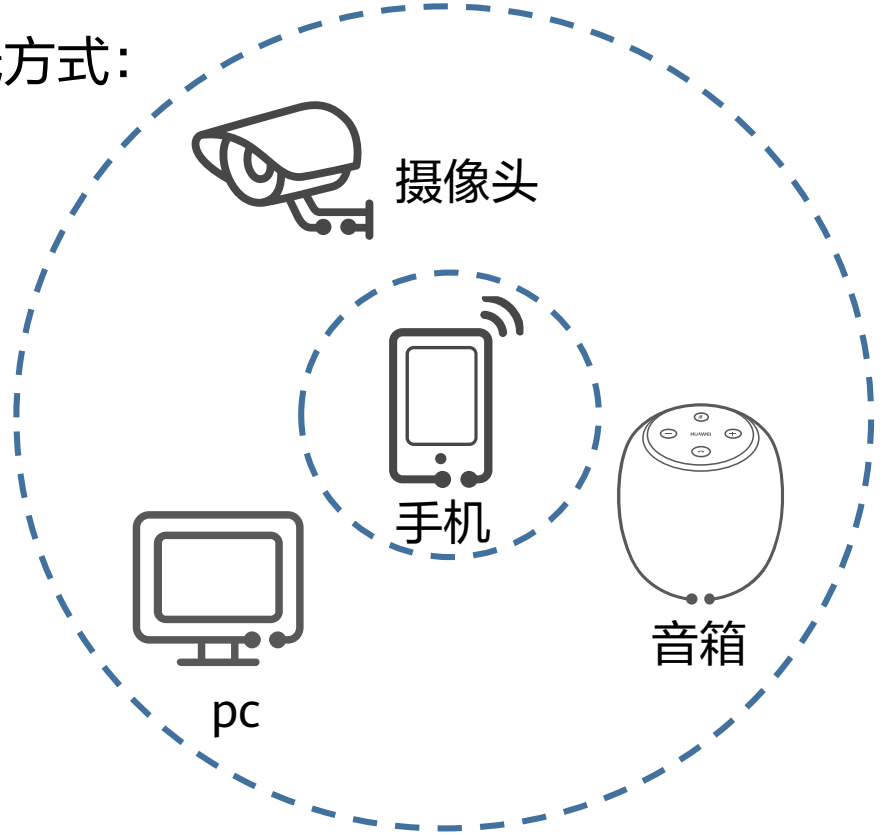
传统连接方式与软总线连接方式对比

- 软总线的一项重要优势就是具备“自发现”能力。

传统方式：



软总线方式：



思考题

1. (判断题)软总线可以实现1+8+N个设备的连接。(　　)

   A. 正确

   B. 错误

2. (判断题)软总线的一项重要优势就是具备“自发现”能力。(　　)

   A. 正确

   B. 错误

思考题

3. (多选题)软总线的数据传输具有哪三种特性？(　　)

A. 高带宽

B. 高时延

C. 高可靠

D. 低时延

本章总结

- 通过本章的学习，您可以了解HarmonyOS的基础子系统组成，熟悉常用的子系统构成。

学习推荐

- HarmonyOS官网社区：
  - https://www.harmonyos.com/cn/home/
- HarmonyOS应用开发文档：
  - https://developer.harmonyos.com/cn/home/
- HarmonyOS设备开发文档：
  - https://device.harmonyos.com/cn/home/
- OpenHarmony开源地址：
  - https://gitee.com/openharmony
- 华为人才在线：
  - https://e.huawei.com/cn/talent/#/

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

拓展子系统开发

HUAWEI

前言

- HarmonyOS除了基础子系统之外，为了一些特殊的使用场景，划分了拓展子系统。拓展子系统包括：图形图像子系统，媒体子系统，AI框架子系统，Sensor服务框架与用户服务框架，安全子系统，测试子系统，DFX与XTS。
- 本章将对拓展子系统做初步介绍，使开发者能对HarmonyOS的子系统有一个全面的了解。

目标

- 学完本章内容后，您将能够：
  - 了解HarmonyOS的扩展子系统构成；
  - 了解设备开发中安全与测试子系统；
  - 对AI框架、Sensor服务框架与用户服务框架有初步了解。

目录

图形图像子系统主要功能

- HarmonyOS的图形图像子系统主要功能：
  - 提供了基础UI组件和容器类组件。包括button（按钮）、image（图片）、label（标签）、list（列表）、animator（动画）、scroll view（滚动条）、swipe view（滑动视图）、font（字体）、clock（时钟）、chart（图表）、canvas（画布）、slider（滑块）、layout（布局）等。
  - 提供截屏、导出组件树的能力。
  - 模块内部实现组件渲染、动画、输入事件分发等功能。

UI组件与布局

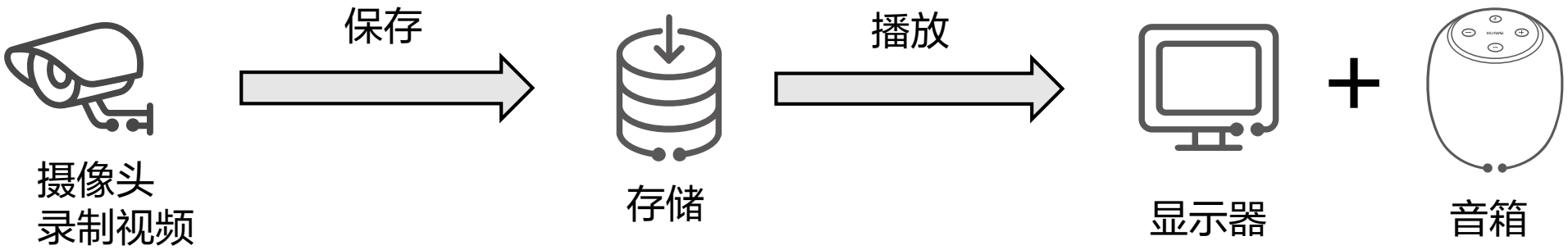
- **UI组件：**
  - 实现各种控件，如按钮、文本、进度条等各种基本控件。
  - 提供界面切换、图片序列帧等复杂控件。
- **布局：**
  - 实现栅格布局、灵活布局（如居中、左对齐、右对齐）。
  - 布局为一次性布局。布局函数每运行一次，会计算一次控件的位置，但是控件位置由其他方式改变时（如拖动），其他相关联的控件位置不会自动发生变化，需要重新调用一次布局函数。

动画与输入事件

- **动画：**
  - 根据tick（时钟滴答）事件，由Task Manager周期性调用回调函数处理属性变化，然后**触发刷新重新绘制**组件，达到组件动画效果。
  - 提供动画的开始/停止、暂停/恢复、创建/销毁等各种操作 ，用于实现动画效果。
- **Input事件：**
  - Input事件包括触摸屏触摸输入事件和物理按键输入事件，GUI（Graphical User Interface，图形用户界面）引擎每运行一次，Input Manager会读取一次所有注册的硬件设备的输入，转化为各种事件供UI控件使用。
- **渲染：**
  - 2D图形渲染实现线、矩形、三角形、弧线的绘制操作。
  - 图像渲染实现各种类型图片的绘制API，如RGB565、RGB888、ARGB8888、PNG、JPG格式。
  - 字体渲染支持矢量字体（通过数学曲线描述的字形）的实时绘制。

媒体子系统开发场景

- 某开发人员希望能开发一套摄像和视频播放的应用。基础功能需要满足视频拍摄和视频播放以及视频播放过程的控制。

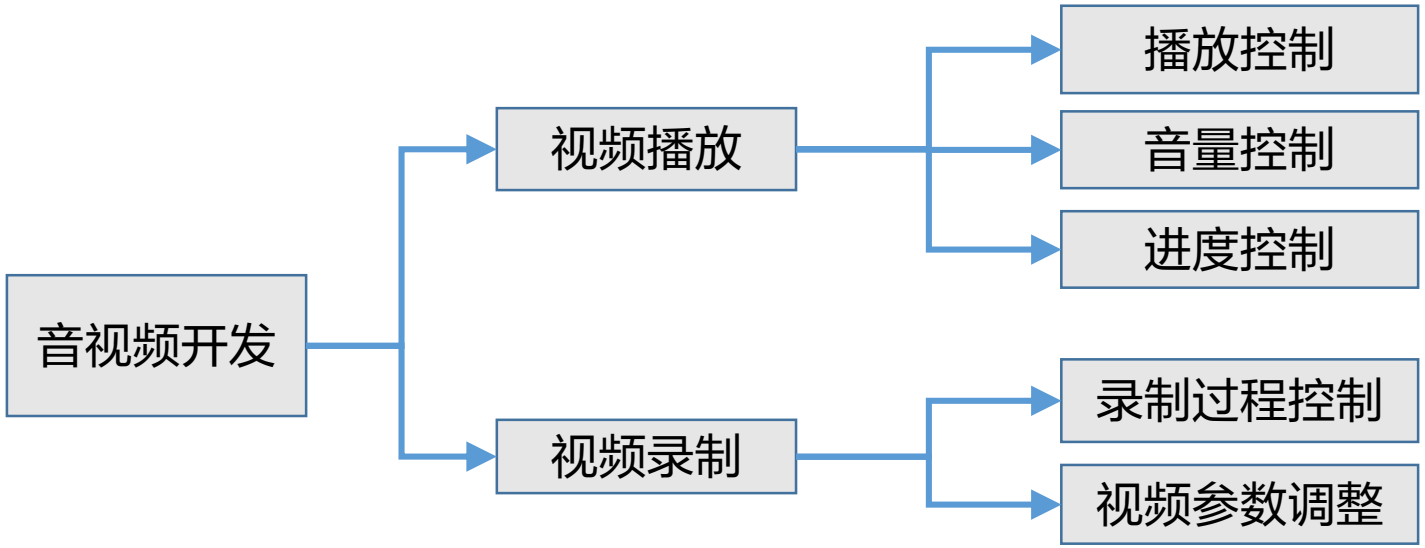


媒体子系统 - 相机开发

- 相机是HarmonyOS多媒体进程提供的服务之一，提供了相机的录像、预览、拍照功能，支持多用户并发取流。
- 基本概念：
  - 视频帧：每一张图片数据称为一帧，这样的一帧称为视频帧。
  - 视频流：一系列图片数据按照固定时间间隔排列形成的数据流。
  - 帧速率(FPS: Frames Per Second)：视频播放每秒钟刷新图片的速度，或是视频每秒的帧数。
  - 分辨率每一帧的图片信息都是由像素点组成的，分辨率描述了一张图片中像素点的个数。例如1920x1080(1080P)。

媒体子系统 - 音视频开发

- HarmonyOS音视频包括音视频播放和录制。
  - HarmonyOS音视频播放模块：包括音视频文件和音视频流播放、音量和播放进度控制功能。
  - HarmonyOS录制模块：提供音视频录制相关的功能，包括设置录制视频画面尺寸、音视频编码码率、编码器类型、视频帧率、音频采样率、录制文件输出格式相关功能。



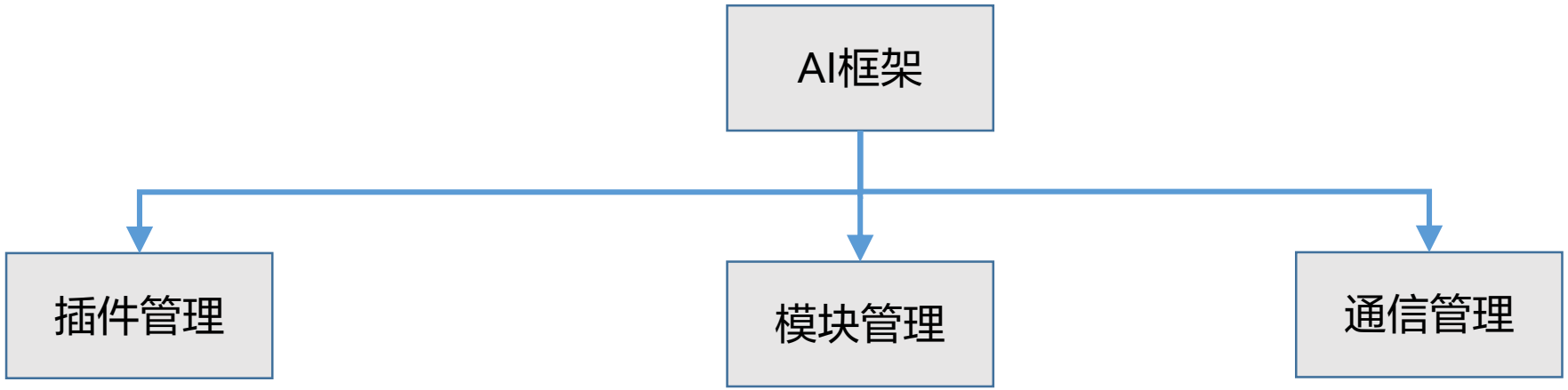
音视频开发基本概念

- **流媒体技术**：把连续的影像和声音信息进行编码处理后放在网络服务器上，让浏览者一边下载、一边观看与收听，而不需要等整个多媒体文件下载完成就可以即时观看、收听的技术。
- **码率**：数据传输时单位时间内传送的数据位数，常用单位是kbps即千位每秒。
- **采样率(Hz)**：每秒从连续信号中提取并组成离散信号的采样个数。采样率越高，声音的还原就越真实自然。

目录

AI框架是什么？

- AI框架定义：HarmonyOS提供原生的分布式AI能力的子系统。
- AI框架功能：提供统一的AI引擎框架，实现算法能力快速插件化集成。
- AI框架的组成：

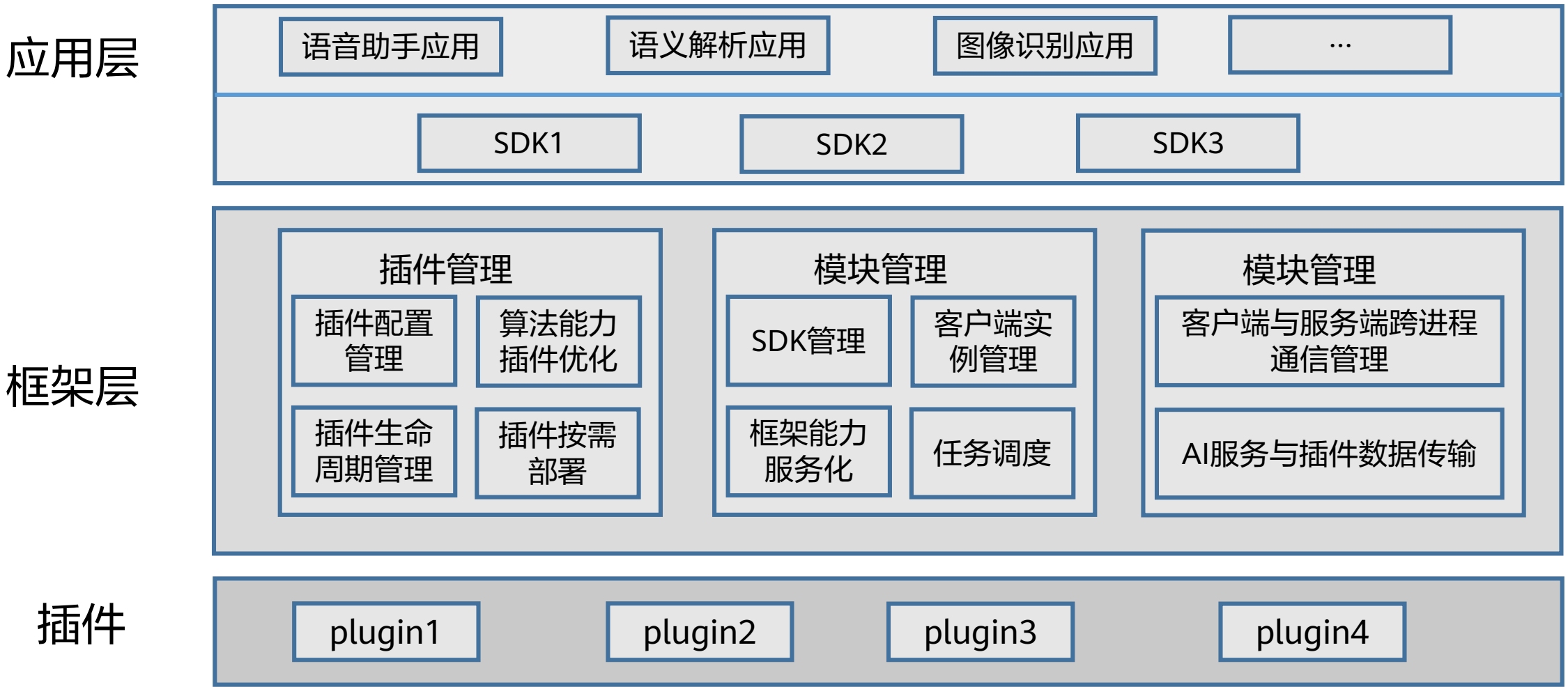


- 实现插件的生命周期管理
- 实现插件按需部署
- 快速集成AI能力插件

- 任务调度
- 管理客户端的实例

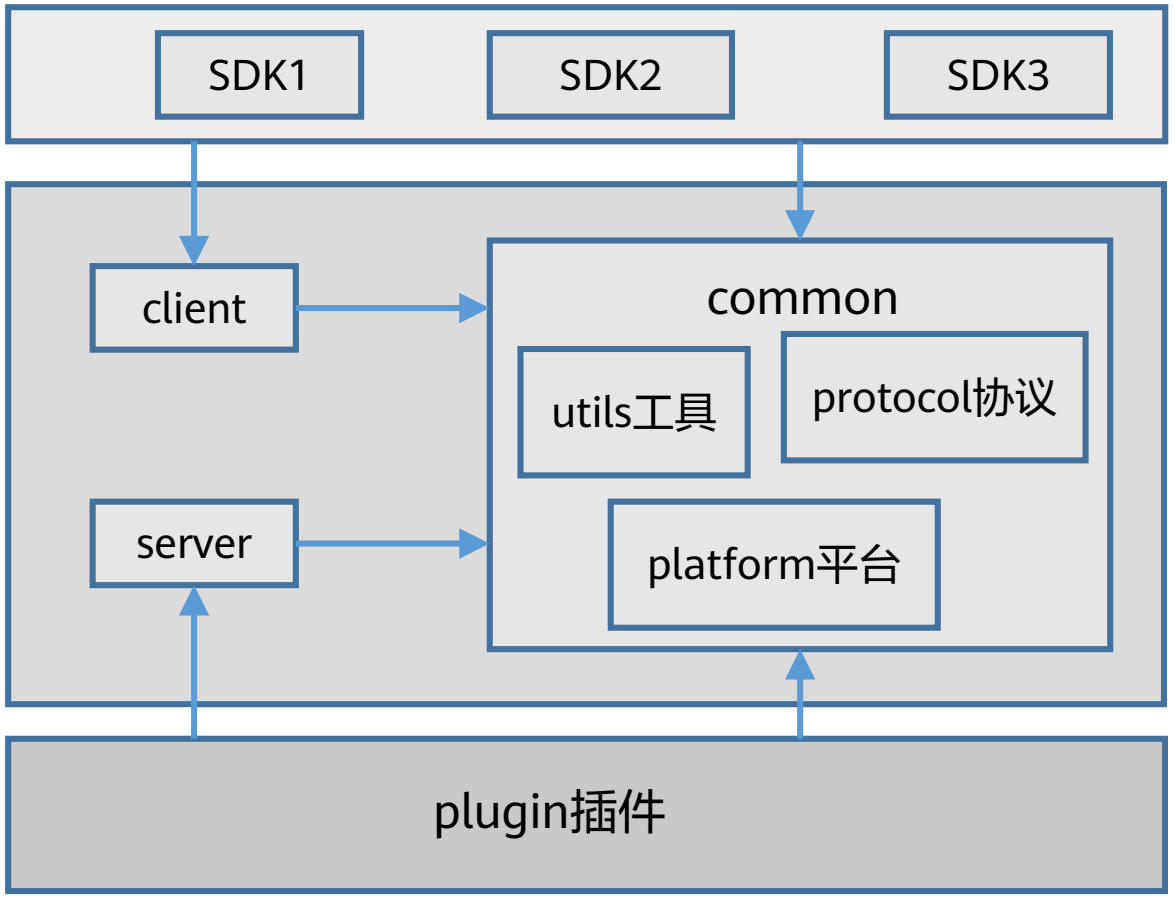
- 客户端与服务端的跨进程通信
- 引擎与插件之间的数据传输

AI框架详细结构



AI框架运行机制

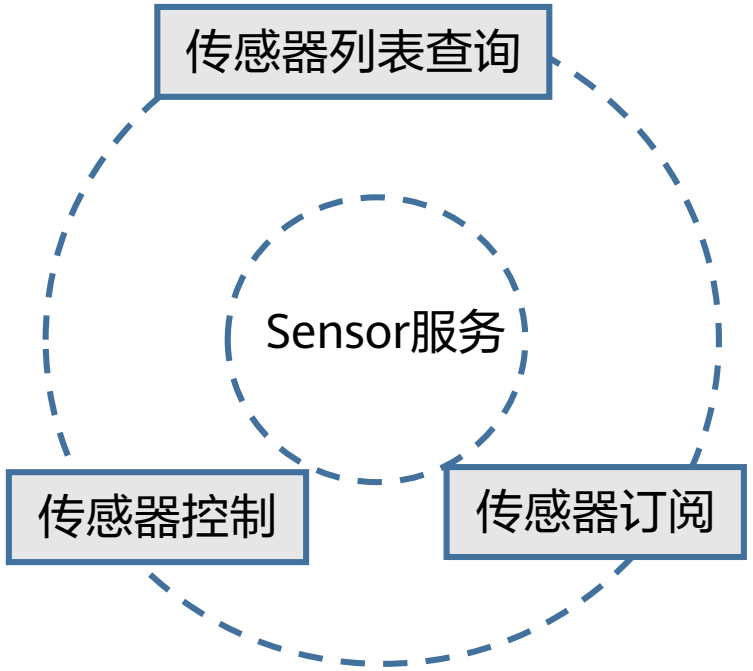
- AI引擎框架包含client、server和common三个主要模块。
  - client提供server端连接管理功能；
  - server提供插件加载以及任务管理等功能，各plugin实现由server提供的插件接口，完成插件接入；
  - common提供与平台相关的操作方法、引擎协议以及相关工具类，供其他各模块调用。



目录

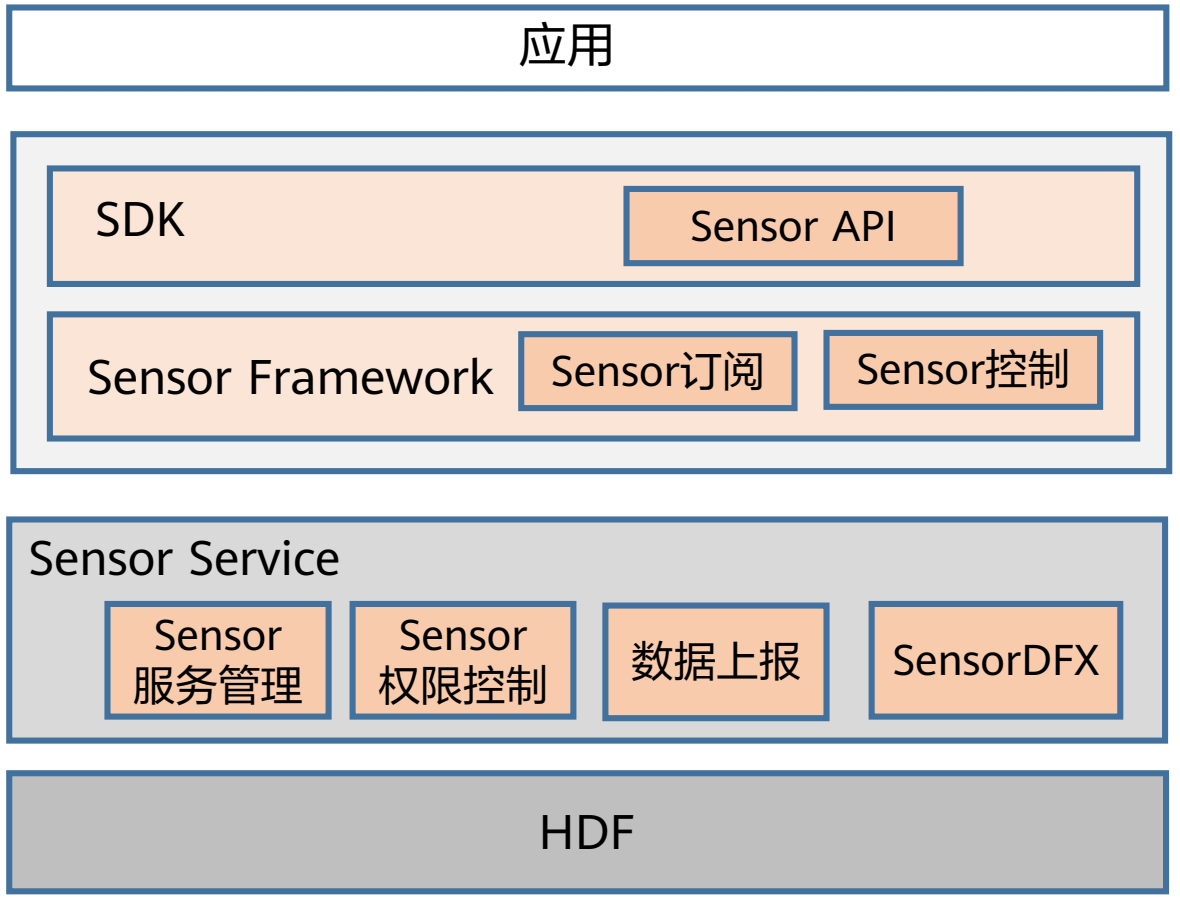
Sensor服务是什么?

- Sensor是一个轻量级传感器服务框架。
- Sensor提供了轻量级传感器服务基础框架，可以使用该框架接口实现**传感器列表查询**、**传感器控制**、**传感器订阅去订阅**等功能。



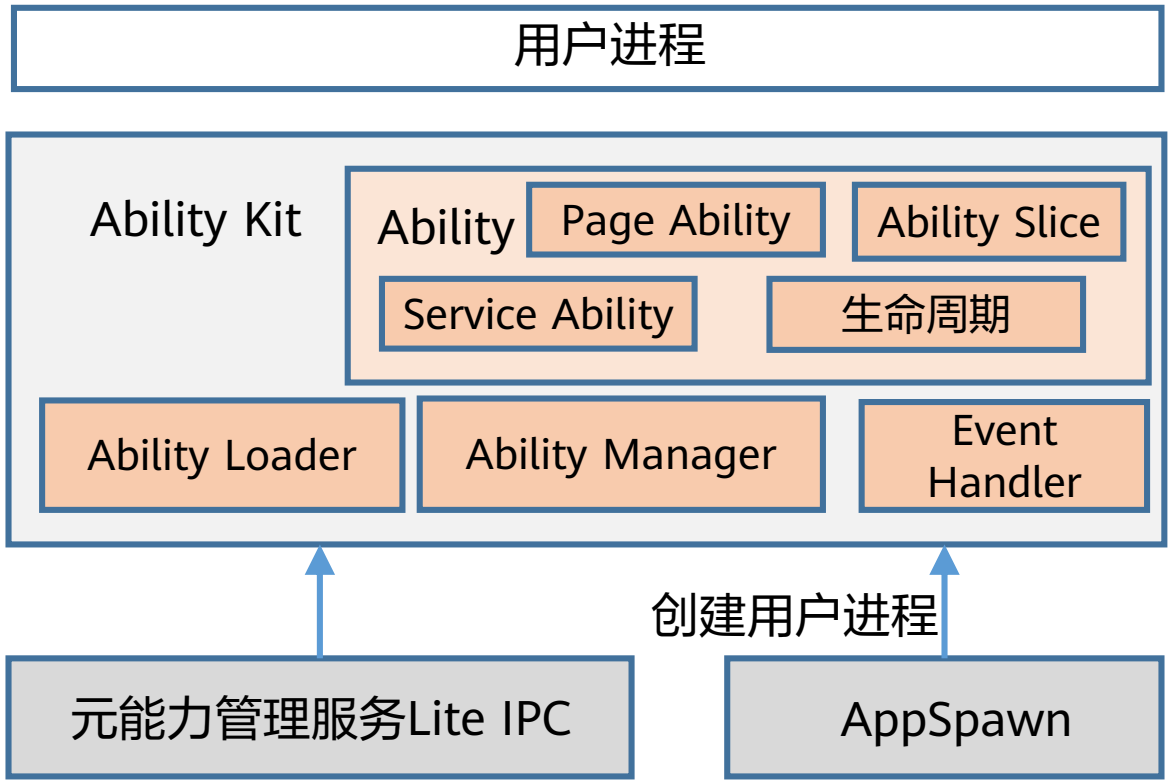
Sensor服务框架结构

- 轻量级传感器服务框架如图所示：
  - Sensor API：提供传感器的基础API。主要包含查询传感器的列表、订阅/取消传感器数据、执行控制命令等，简化应用开发。
  - Sensor Framework：主要实现传感器的订阅管理、数据通道的创建、销毁等，实现与传感器服务层的通信。
  - Sensor Service：主要实现HDF层数据接收、解析、分发，对设备传感器的管理，数据上报管理以及传感器权限管控等。



用户程序框架结构

- **Ability**：系统调度应用的最小单元，是能够完成一个独立功能的组件，一个应用可以包含一个或多个Ability。Ability分为两种类型： Page类型的Ability和Service类型的Ability。
  - **Page类型的Ability**：带有界面，为用户提供人机交互的能力。
  - **Service类型的Ability**：不带界面，为用户提供后台任务机制。



目录

软件安全的重要性

- 如果用户使用的软件缺乏必要的安全保障，那么会发生什么？
- 场景：某用户甲想要将运动手环A的体温数据Data1发送到手机上。

- 异常场景：
  - 场景1：用户乙将运动手环A的数据Data1发送到手机上。
  - 场景2：用户甲将冰箱的温度Data2发送到手机上。
  - 场景3：用户甲将运动手环A的数据Data1发送到电视上。

HarmonyOS 安全子系统

- HarmonyOS安全子系统目前提供给开发者的安全能力主要包含应用可信、权限管理、设备可信。

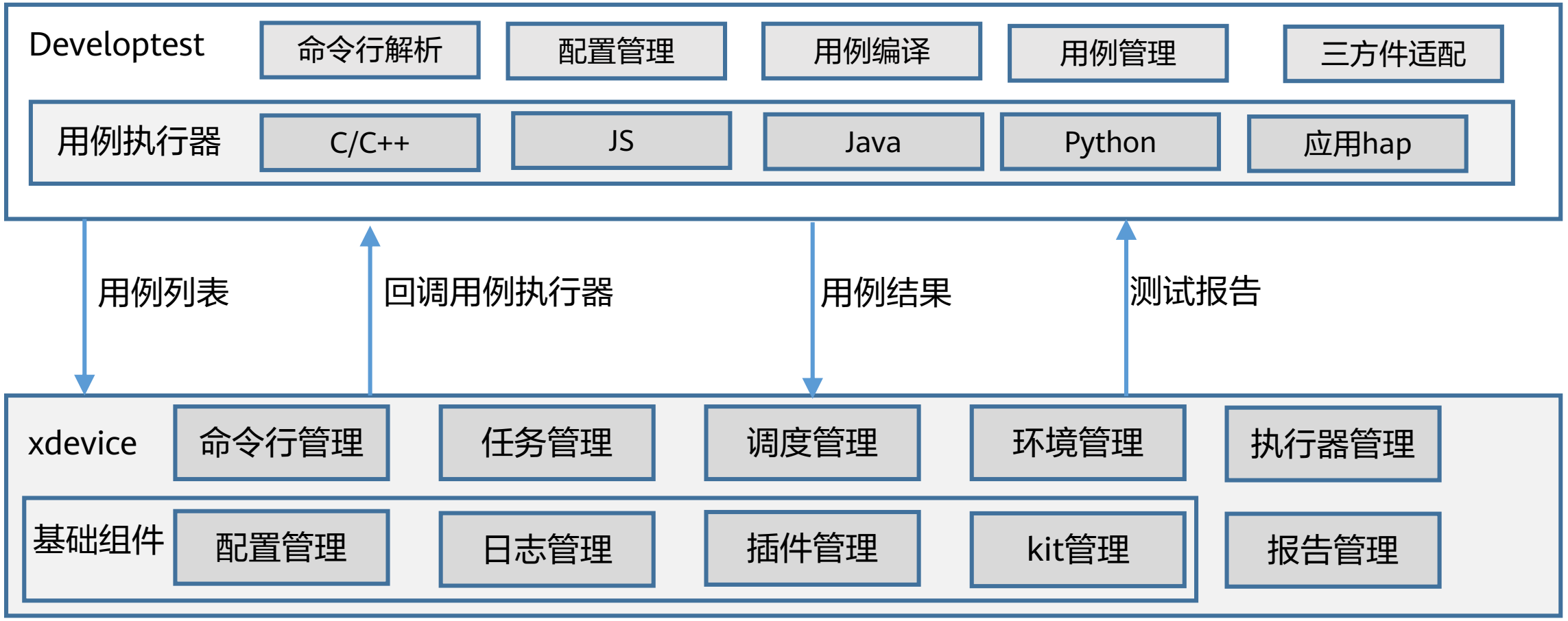
安全子系统限制

- 仅支持以下三类应用的验签：应用市场调试应用、应用市场发布应用、HarmonyOS自签名应用的验签。
- 若对应用市场调试应用验签，则本机UDID需要在描述文件包含的UDID列表中。
- 待上架应用无法验签通过。
- 验签组件提供的接口都位于security_interfaces_innerkits_app_verify仓app_verify_pub.h中，仅支持系统应用开发者调用。
- 可信设备群组管理接口，目前只对系统签名权限才可以使用。

测试子系统是什么？

- 测试子系统提供基于Python开发的一键式的**开发者自测试平台**，支持跨平台使用以及三方测试框架拓展，主要包括测试用例编译、测试用例管理、测试用例调度分发、测试用例执行、测试结果收集、测试报告生成、测试用例模板、测试环境管理等模块。
- **约束与限制：**
  - 功能使用范围：开发自测试平台仅支持代码级的测试用例开发和验证，如单元测试，模块测试。
  - 规格限制：当前测试框架的适用范围仅支持白盒测试。
  - 操作限制：一台测试设备上仅支持启动单个测试平台。

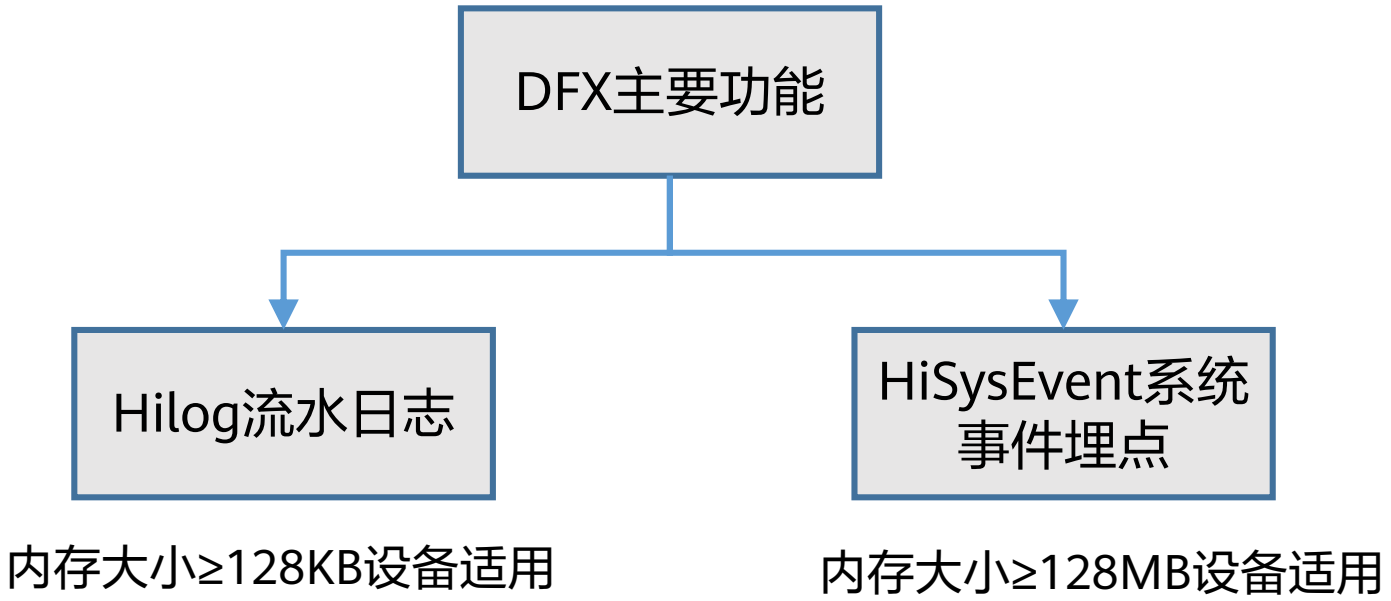
测试子系统架构



目录

DFX 是什么?

- DFX(Design for X)是为了提升质量属性的软件设计。包含两个内容主要:
  - DFR（ Design for Reliability，可靠性 ）。
  - DFT（ Design for Testability，可测试性 ）。

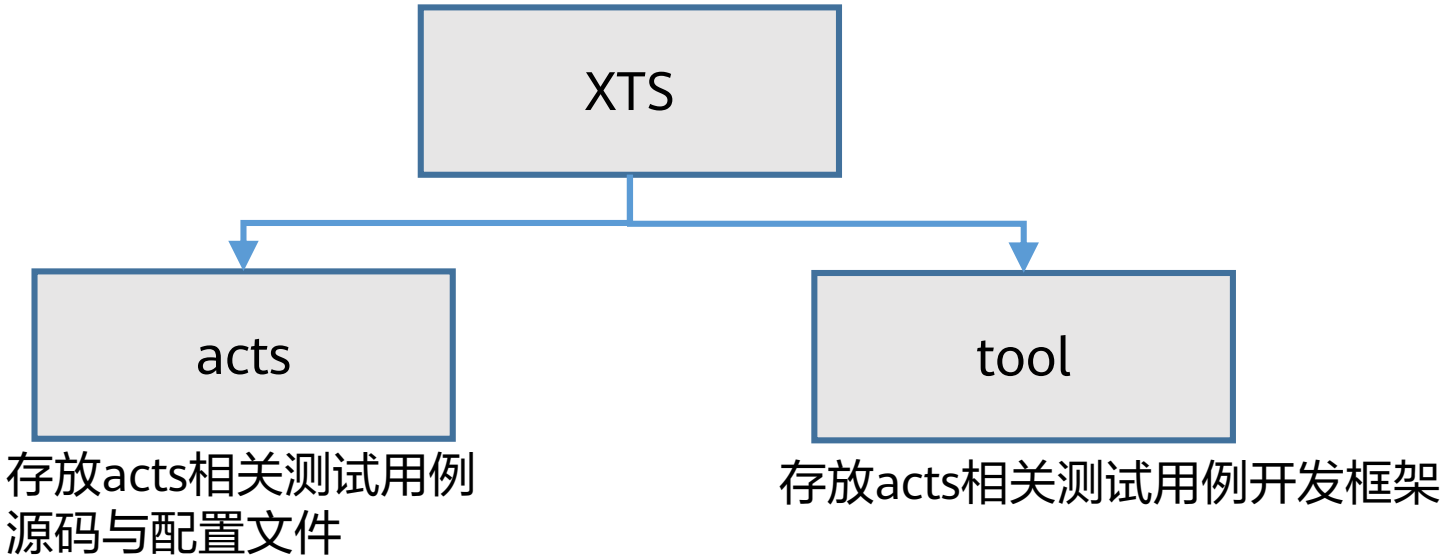


DFX 的几个基本概念

- **流水日志：**
  - 流水日志是系统运行过程中产生的一些日志信息，用于开发人员了解系统或应用运行过程、状态。
- **分布式跟踪：**
  - 在一个分布式系统中，一次业务的发起往往会经历多个软件模块，通过进程内、进程间、设备间的通信接口进行控制和数据传递。
  - DFX提供的分布式跟踪框架可以便于开发人员对这类复杂流程的问题跟踪定界。
- **线程卡死故障：**
  - 线程在运行过程中，如果进入死循环，或者陷入内核态，将无法响应正常的业务请求，且无法自己恢复。
  - 通过watchdog机制，在易于发生卡死的流程中插入检测点，对于卡死故障可进行故障恢复和日志采集。

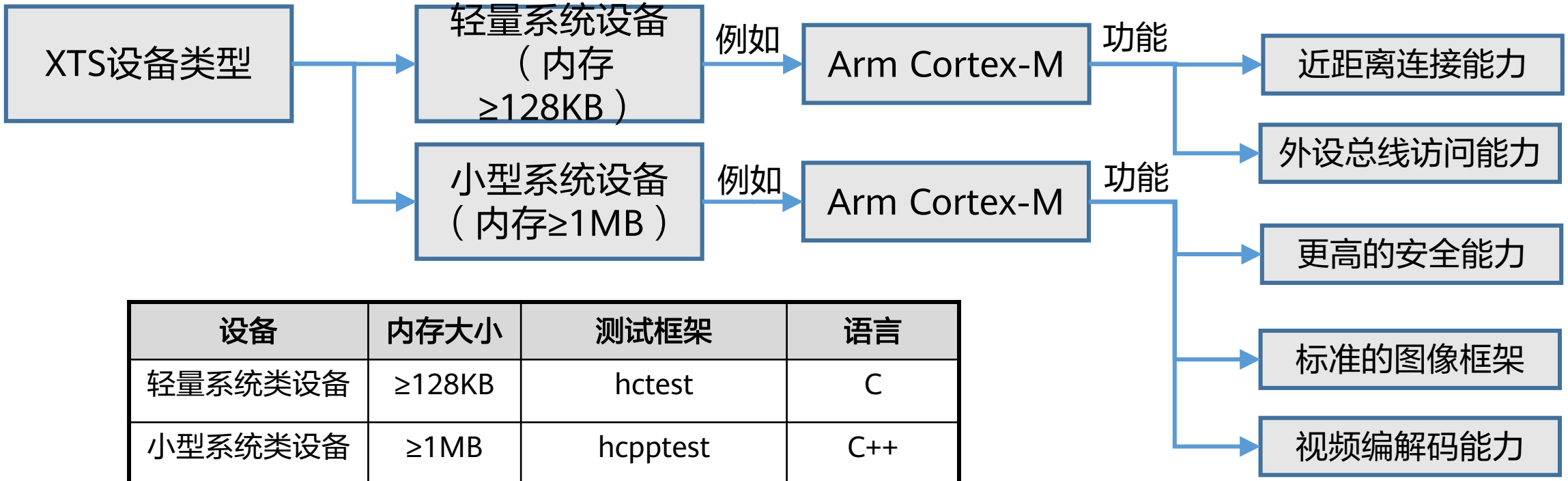
XTS 是什么?

- XTS子系统是**HarmonyOS生态认证测试套件的集合**，当前版本可使用acts(application compatibility test suite)应用兼容性测试套件。
- XTS主要功能：帮助终端设备开发者尽早明确软件与HarmonyOS的兼容性。



XTS 设备类型

- HarmonyOS支持以下两种设备类型：



设备	内存大小	测试框架	语言
轻量系统类设备	≥128KB	hctest	C
小型系统类设备	≥1MB	hcpptest	C++
标准系统类设备	≥128MB	HJUnit、hcpptest	Java、C++
大型系统类设备	≥1GB	HJUnit、hcpptest	Java、C++

思考题

1. (判断题)软总线可以实现1+8+N个设备的连接。(　　)

   A. 正确

   B. 错误

2. (判断题)HarmonyOS可以让正确的人，用正确的设备，正确的使用数据。(　　)

   A. 正确

   B. 错误

思考题

3. (多选题)HarmonyOS目前支持哪些验签？（　　）

A. 应用市场调试应用的验签

B. 应用市场发布应用的验签

C. 应用市场撤销应用的验签

D. HarmonyOS自签名应用的验签

本章总结

- 通过本章的学习，你可以了解HarmonyOS的扩展子系统组成，熟悉扩展子系统的相关概念。

学习推荐

- HarmonyOS官网社区：
  - https://www.harmonyos.com/cn/home/
- HarmonyOS应用开发文档：
  - https://developer.harmonyos.com/cn/home/
- HarmonyOS设备开发文档：
  - https://device.harmonyos.com/cn/home/
- OpenHarmony开源地址：
  - https://gitee.com/openharmony
- 华为人才在线：
  - https://e.huawei.com/cn/talent/#/

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

功能调测

HUAWEI

前言

- HarmonyOS设备开发的重要步骤之一，就是对已经初步开发完成的设备进行功能调测，在功能调测中主要使用到了Shell命令。

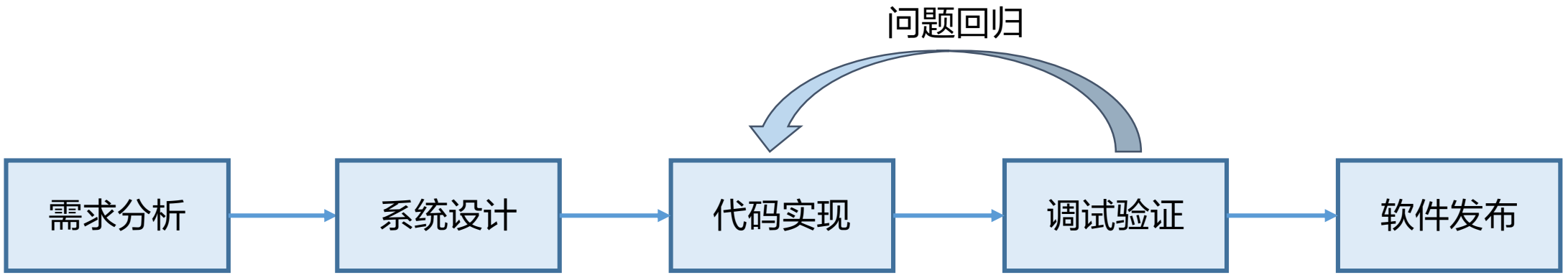
目标

- 学完本课程后，您将能够：
  - 知晓Shell命令的基本概念；
  - 掌握Shell命令如何用于编程；
  - 掌握Shell命令的基本使用方法。

目录

软件调测的价值与意义

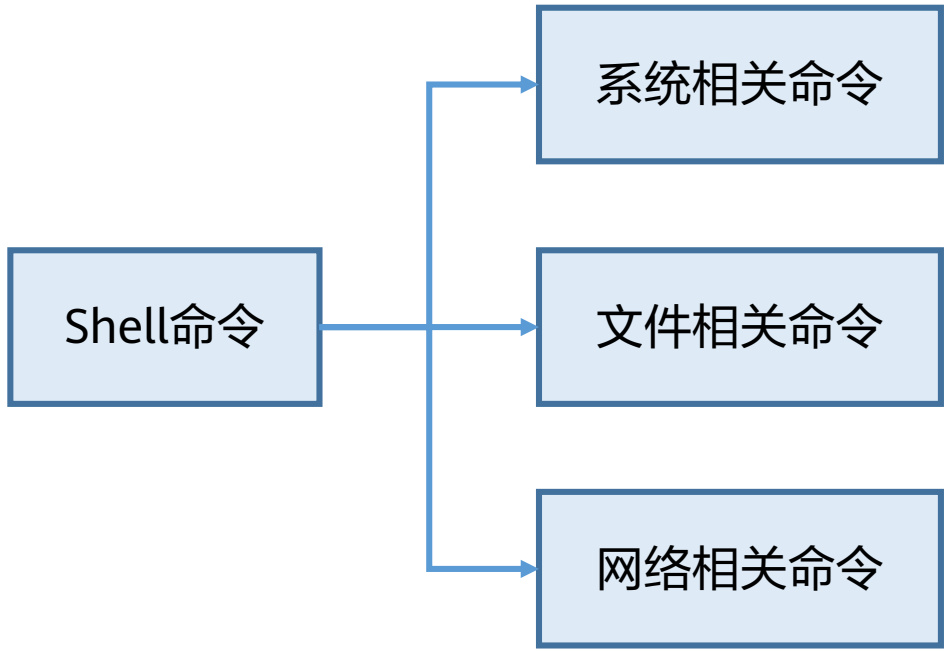
- 在硬件设备开发完成之后，需要对开发完成的软件进行测试验证，确保软件的功能正确。



- Shell命令是HarmonyOS设备开发常用的调测工具。

Shell 命令的常用功能

- HarmonyOS内核提供的Shell支持调试常用的基本功能，包含系统、文件、网络和动态加载相关命令。
- 同时HarmonyOS内核的Shell支持添加新的命令，可以根据需求来进行定制。



系统任务、内核信号量、系统软件定时器、CPU占用率、当前中断。

文件的创建，移动，属性查看等功能。

查询IP、测试网络连接、设置开发板的AP和station模式等相关功能。

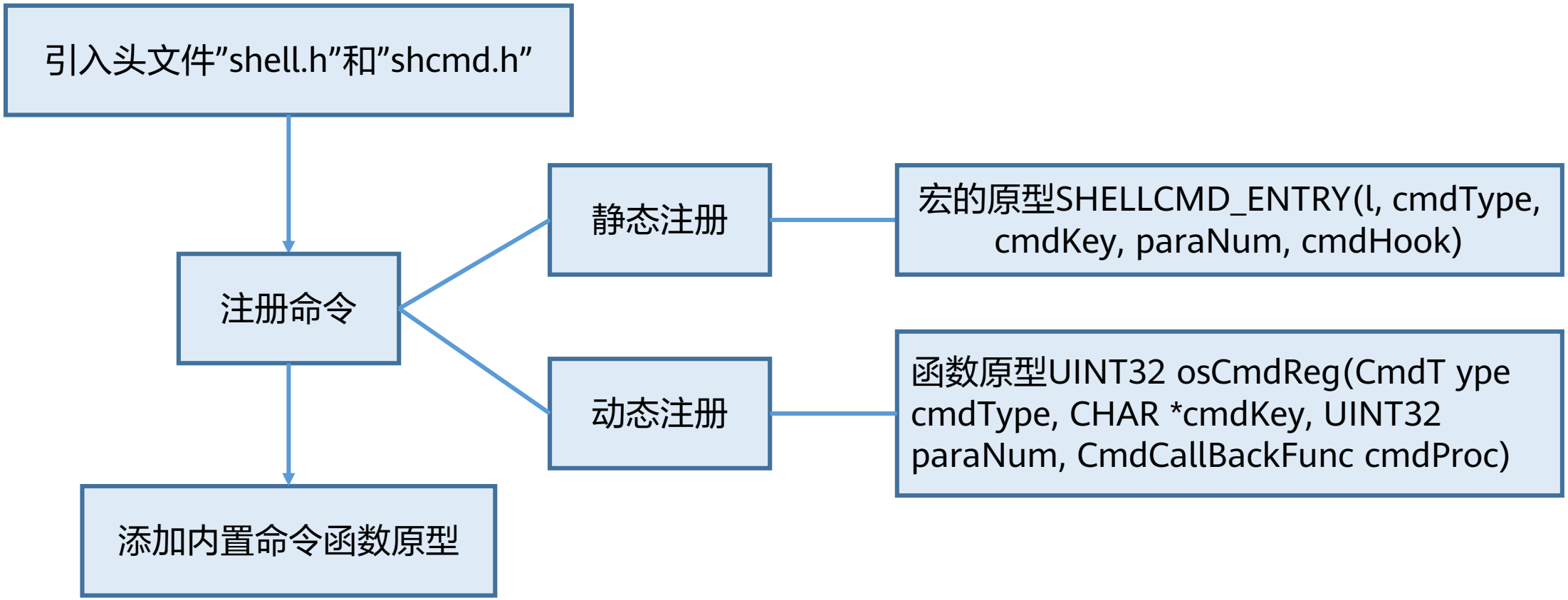
Shell 命令注意事项

- Shell功能支持Shell命令、文件名及目录名的Tab键联想补全。
- Shell端工作目录与系统工作目录是分开的，即通过Shell端cd、pwd等命令是对Shell端工作目录进行操作，通过chdir、getcwd等命令是对系统工作目录进行操作，两个工作目录相互之间没有联系。
- 在使用网络Shell指令前，需要先调用tcpip_init函数完成网络初始化并完成telnet连接后才能起作用，内核默认不初始化tcpip_init。
- 不建议使用Shell命令对/dev目录下的设备文件进行操作。

目录

Shell 命令编程实例

- 新增Shell命令的典型开发流程如下：



静态注册参数介绍

- 宏的原型为：SHELLCMD_ENTRY(l, cmdType, cmdKey, paraNum, cmdHook)。
- 例如：

```
SHELLCMD_ENTRY(ls_shellcmd, CMD_TYPE_EX, “ls”, XARGS, (CMD_CBK_FUNC)osShellCmdLs)
```

参数	含义
l	静态注册全局变量名（注意：不与系统中其他symbol重名）。
cmdType	命令类型。 CMD_TYPE_EX不支持标准命令参数输入，会把用户填写的命令关键字屏蔽掉；CMD_TYPE_STD支持的标准命令参数输入，输入的字符会通过命令解析后被传入。
cmdKey	命令关键字，函数在Shell中访问的名称。
paraNum	调用的执行函数的入参最大个数，默认值XARGS(0xFFFFFFFF)。
cmdHook	命令执行函数地址，是命令实际执行函数。

静态注册方式实例

- 定义一个新增命令所要调用的执行函数cmd_test。

```
#include "shell.h"
#include "shcmd.h"
int cmd_test(void)
{
printf("hello everybody!\n");
return 0;
}
```

- 新增命令项：

```
SHELLCMD_ENTRY(test_shellcmd, CMD_TYPE_EX, "test", 0, (CMD_CBK_FUNC)cmd_test);
```

- 在链接选项中添加链接该新增命令项参数：
  - 在liteos_tables_ldflags.mk文件的LITEOS_TABLES_LDFLAGS项下添加-utest_shellcmd。

静态注册方式实例

- 重新编译代码：

```
make clean
make
```

- 用help命令查看当前系统所有的注册命令，可以发现test命令已经注册。

```
OHOS # help
******************shell commands:*************************
arp       cat       cd        chgrp     chmod     chown      cp        cpup
date      dhclient  dmesg     dns       format    free       help      hwi
Ifconfig  ipdebug   kill      log       ls        lsfd       memcheck            mkdir
mount     netstat   oom       partinfo  partition           ping      ping6     pwd
reset     rm        rmdir     sem       statfs    su         swtmr     sync
systeminfo          task      telnet    test      tftp       touch     umount    uname
watch     writeproc
```

动态注册参数介绍

- 函数原型UINT32 osCmdReg(CmdType cmdType, CHAR *cmdKey, UINT32 paraNum, CmdCallBackFunc cmdProc)。
- 例如：`osCmdReg(CMD_TYPE_EX, “ls”, XARGS, (CMD_CBK_FUNC)osShellCmdLs)`
- osCmdReg参数详解：

参数	含义
cmdType	命令类型。 CMD_TYPE_EX不支持标准命令参数输入，会把用户填写的命令关键字屏蔽掉；CMD_TYPE_STD支持的标准命令参数输入，输入的字符会通过命令解析后被传入。
cmdKey	命令关键字，函数在Shell中访问的名称。
paraNum	调用的执行函数的入参最大个数，默认值XARGS(0xFFFFFFFF)。
cmdHook	命令执行函数地址，是命令实际执行函数。

动态注册方式实例

- 在用户应用函数中调用osCmdReg函数动态注册命令。

```
#include "shell.h"
#include "shcmd.h"
int cmd_test(void)
{
printf("hello everybody!\n");
return 0;
}
void app_init(void)
{
....
....
osCmdReg(CMD_TYPE_EX, "test", 0,(CMD_CBK_FUNC)cmd_test);
....
}
```

动态注册方式实例

- 重新编译代码：

```
make clean
make
```

- 用help命令查看当前系统所有的注册命令，可以发现test命令已经注册，执行效果与静态注册一致。

```
OHOS # help
******************shell commands:*************************
arp       cat       cd        chgrp     chmod     chown      cp        cpup
date      dhclient  dmesg     dns       format    free       help      hwi
Ifconfig  ipdebug   kill      log       ls        lsfd       memcheck            mkdir
mount     netstat   oom       partinfo  partition           ping      ping6     pwd
reset     rm        rmdir     sem       statfs    su         swtmr     sync
systeminfo          task      telnet    test      tftp       touch     umount    uname
watch     writeproc
```

目录

系统命令

- cpup，用于查询cpu占用率。命令格式：cpup [mode] [taskID]。
  - 例如：

```
OHOS # cpup 1 5

pid 5 CpuUsage in 1s: 0.1
```

- date，用于查询与设置系统日期时间。例如：

```
OHOS # date +%Y--%m--%d
2020--03--20
```

- exec，用于执行用户态程序的功能。例如：exec helloworld。
- hwi，用于显示中断信息。例如：

```
OHOS # hwi
 InterruptNo      Count      Name
        32:         0:      rtc_alarm
        37:       247:
        39:         4:      uart_pl011
        62:         0:      MMC-SD
        63:        13:      MMC-SD
        65:        42:      ETH
        71:         0:
```

系统命令

- kill，发送特定信号给指定进程。命令格式：kill [signo / -signo] [pid]。
  - 例如：

```
OHOS # kill 14 7
OHOS # 01-01 00:27:08.783 3 16 I 00000/(null): Send signal(14) to pidNo = 7!
```

- log，用于修改&查询日志配置。命令格式：log level [levelNum]。
  - 例如：

```
OHOS # log level 4
Set current log level INFO
```

- memcheck，检查动态申请的内存块是否完整，是否存在内存越界造成节点损坏。
  - 例如：

```
OHOS # memcheck
system memcheck over, all passed!
```

- reset命令用于重启设备。

文件命令

- cat用于显示文本内容。例如：

```
OHOS # cat hello-harmony.txt
OHOS # Hello HarmonyOs ; )
```

- cd用于切换目录，cd ..可以回退到上一目录。
- chmod用于修改文件操作权限。命令格式：chmod [mode] [pathname]。
- cp用于拷贝文件。命令格式：cp [source file] [destination file]。
- ls用于显示目录内容，命令格式：ls [path]，path为空表示当前目录。
- mkdir命令用来创建一个目录，命令格式：mkdir [directory]。
- mount命令用来将设备挂载到指定目录。例如：

```
OHOS # mkdir /bin1/vs/sd
OHOS # mount /dev/mmcblk0p0 /bin1/vs/sd vfat
mount ok
```

文件命令

- pwd命令用来显示当前路径，例如：

```
OHOS # pwd
/bin/vs
```

- rm命令用来删除文件或文件夹，命令格式：rm [-r] [dirname / filename]，-r为可选参数，如果需要删除目录则需要此参数。
- rmdir只能用于删除目录，且一次操作只能删除一个空目录。
  - 例如：

```
OHOS # ls
Directory /bin/vs:
drwxr-xr-x 0        u:0     g:0     dir
OHOS # rmdir dir
OHOS # ls
Directory /bin/vs:
```

- touch命令用来在指定的目录下创建一个不存在的空文件，如果是操作已经存在的文件，则不会更新时间戳。命令格式：touch [filename]。

```
OHOS # touch file.c
OHOS # ls
Directory /bin/vs:
-rw-r--r-- 0        u:0     g:0     file.c
```

网络命令

- arp用于查询ARP缓冲，ARP缓冲存储了IP与MAC的对应表。
  - 例如：

```
OHOS # arp
Address                  HWaddress                Iface       Type
192.168.1.1              00:0E:C6:58:B6:50        eth0        static
```

- dhclient设置和查看dhclient的参数。例如：

```
OHOS # dhclient -sv MFSI
dhclient: set vendor info [MFSI] success
OHOS # dhclient eth0
set dns option
OHOS # set dns option
dhclient -gd 0
dns[0]: 192.168.1.100
OHOS # dhclient -gd 1
dns[1]: 192.168.1.101
```

- dns用于查看和设置单板dns服务器地址。

网络命令

- netstat用于检验本设备各端口信息。例如：

```
OHOS # netstat
========== total sockets 128 ======  unused sockets 122 ==========
Proto    Recv-Q      Send-Q      Local Address           Foreign Address         State
tcp      0           0           192.168.1.10:343        192.168.1.1:2049        ESTABLISHED
tcp      0           0           192.168.1.10:53933      0.0.0.0:0               LISTEN
tcp      0           0           192.168.1.10:53932      0.0.0.0:0               LISTEN
```

- ping用于检验网络连接情况。例如：

```
OHOS # ping 192.168.1.10

[0]Reply from 192.168.1.10: time<1ms TTL=255
[1]Reply from 192.168.1.10: time<1ms TTL=255
[2]Reply from 192.168.1.10: time<1ms TTL=255
[3]Reply from 192.168.1.10: time<1ms TTL=255
--- 192.168.1.10 ping statistics ---
4 packets transmitted, 4 received, 0 loss
```

思考题

1. (判断题)Shell可用于HarmonyOS设备开发的功能调测。(　　)

   A. 正确

   B. 错误

2. (判断题)Shell命令可以通过动态注册和静态注册两种方式完成注册。(　　)

   A. 正确

   B. 错误

思考题

3. (多选题)Shell常用的文件操作命令有哪些？(　　)

A. reset

B. mkdir

C. rm

D. ping

本章总结

- 通过本章的学习，您可以知晓HarmonyOS设备开发功能调测中需要使用到的Shell工具概念以及Shell命令如何用于设备调测。

学习推荐

- HarmonyOS官网社区：
  - https://www.harmonyos.com/cn/home/
- HarmonyOS应用开发文档：
  - https://developer.harmonyos.com/cn/home/
- HarmonyOS设备开发文档：
  - https://device.harmonyos.com/cn/home/
- OpenHarmony开源地址：
  - https://gitee.com/openharmony
- 华为人才在线：
  - https://e.huawei.com/cn/talent/#/

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

HarmonyOS 移植

HUAWEI

前言

- 适用于不同场景的物联网设备可能基于各种芯片架构，而开发者期望能使用一个统一的操作系统对所有设备进行控制。
- HarmonyOS可以运行在不同体系架构的处理器和开发板上。一般情况下，操作系统的编写者通常不会一次性完成整个操作系统的代码，而是会把一部分与具体硬件设备相关的代码作为抽象的接口保留出来，这样可以保证整个操作系统的可移植性。

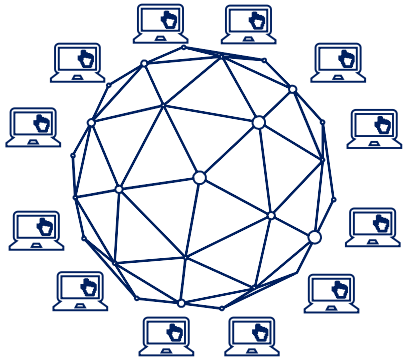
目标

- 学完本课程后，您将能够：
  - 描述移植的概念；
  - 了解移植的步骤；
  - 熟悉第三方库的移植过程。

目录

为什么要移植操作系统?

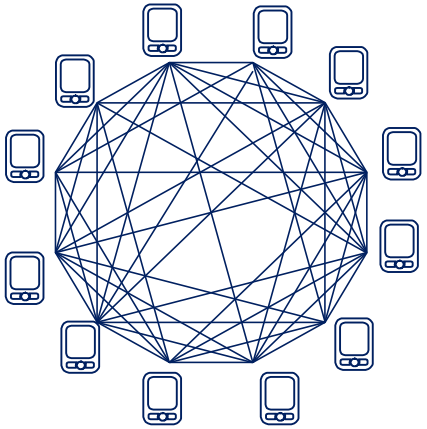
- 移植可以有效的降低开发的适配成本。



10亿连接

互联网时代
x86架构电脑

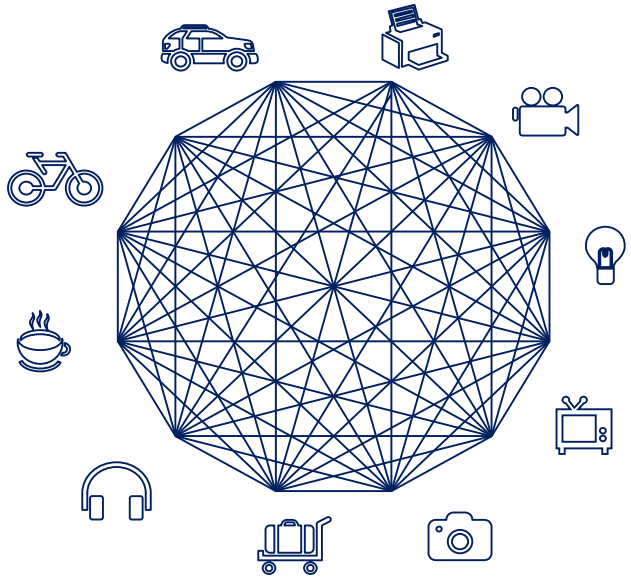
Windows



100亿连接

移动互联网时代
ARM架构手机

Android/iOS

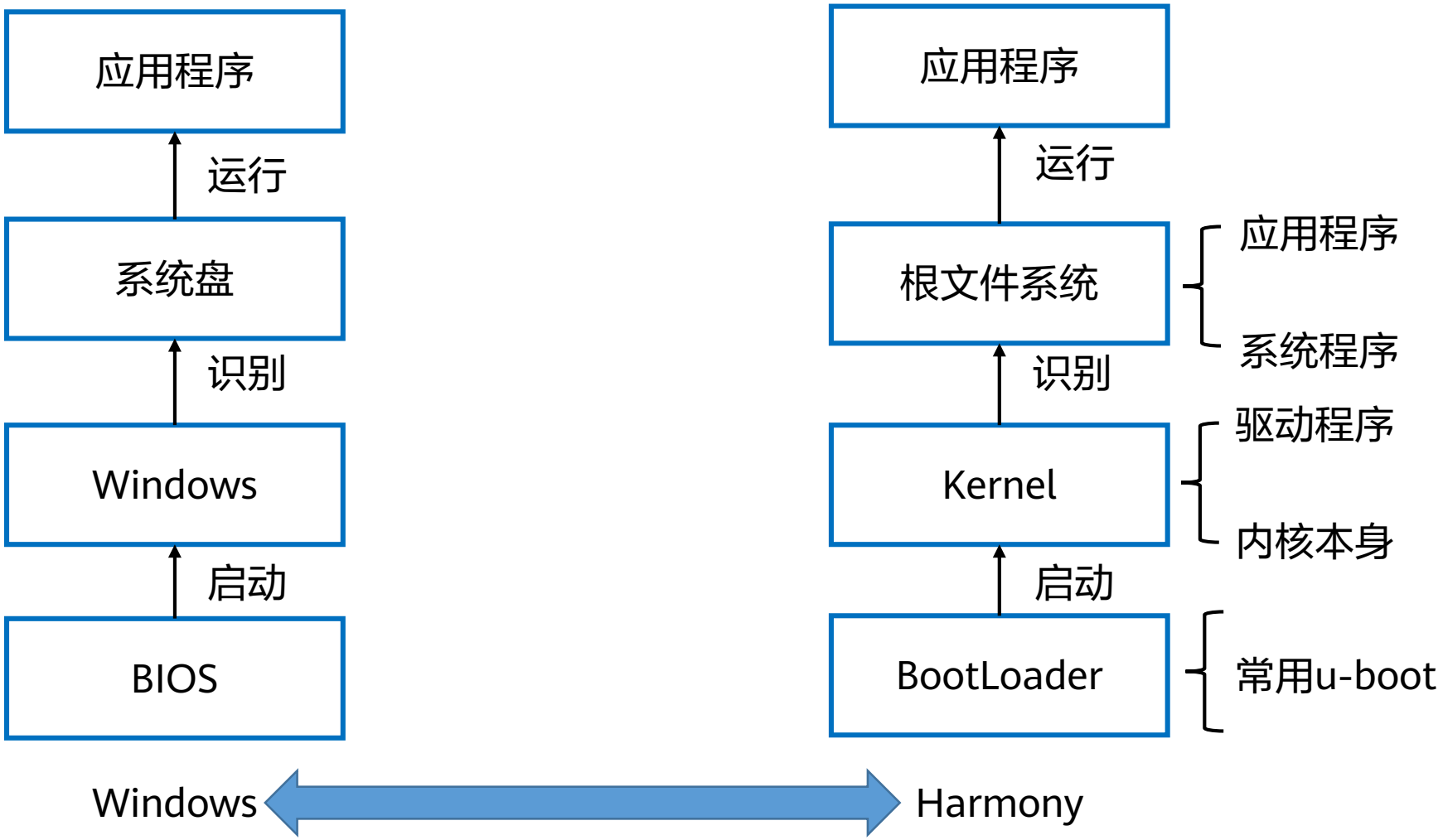


1000亿连接

物联网时代
x86/ARM/DSP/MIPS/FPGA...架构终端

HarmonyOS

嵌入式软件系统的组成：Windows VS HarmonyOS



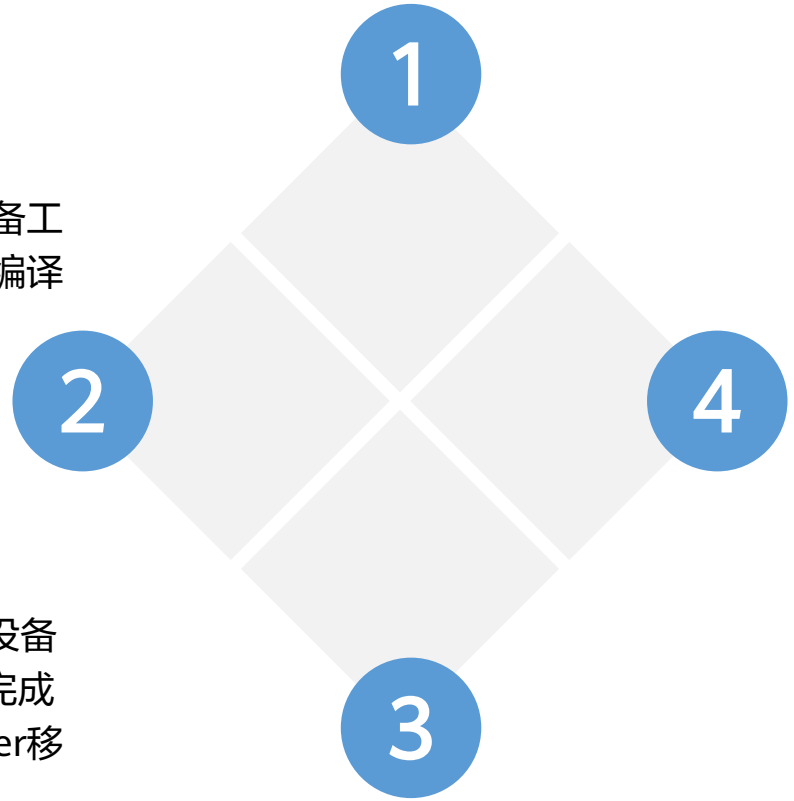
操作系统移植步骤

环境准备

移植系统前需要进行一系列准备工作，包括下载源码、建立交叉编译环境等。

BootLoader移植

引导加载程序(BootLoader)是设备上电后运行的第一个程序。在完成环境准备后，需要对BootLoader移植。



编译内核

在完成BootLoader的移植后，需要对操作系统内核进行配置和编译，如有必要，还需要对源码做一定的修改。

根文件系统制作

制作根文件系统，并将程序、库、配置文件等文件放入根文件系统中进行调用。如有需要，用户数据和驱动程序也可放入根文件系统

移植场景对比

移植场景	是否有完整芯片适配	是否有完整系统适配	是否有第三方库需要移植
第三方库移植	是	是	是
板级系统移植	是	否	/
内核移植	否	/	/

目录

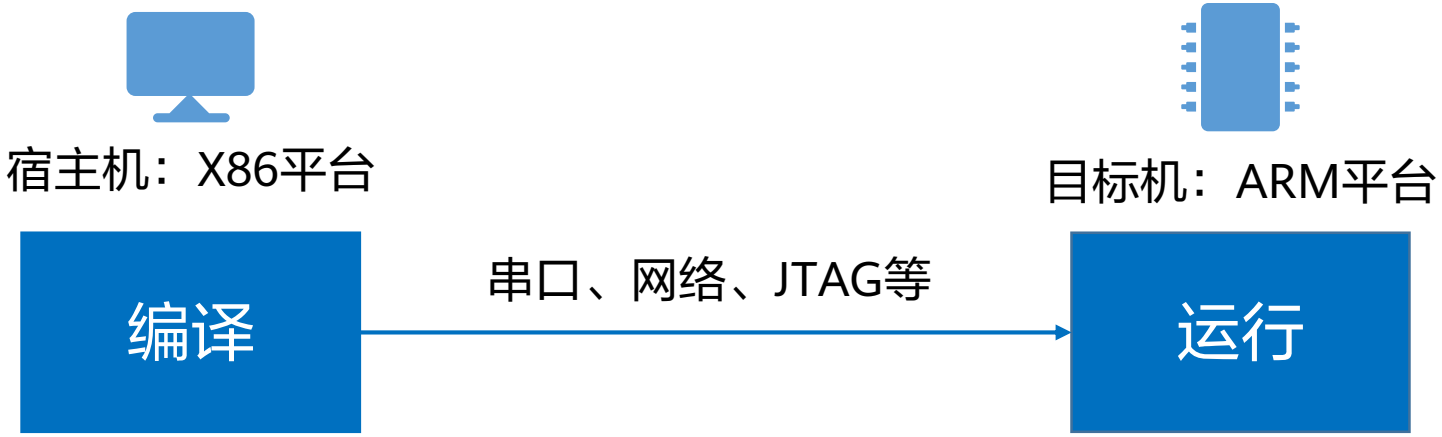
编译环境简介：本地编译

- 对于常见的软件开发，尤其是非嵌入式开发，基本都采用本地编译的方式。本地编译是指在当前的PC中直接编译出来程序或者库文件，这个程序或者库文件可以直接在当前的环境中运行。
- 所谓的**本地编译**，就是在当前目标平台中完成了编译与运行，在这个过程中不需要考虑跨设备与跨平台的情况。

本地编译流程

编译环境简介：交叉编译

- **交叉编译**是一个和本地编译相对应的概念。
- 交叉编译是在一种平台上编译，编译出来的程序或者库文件放到别的平台上运行。编译的环境和运行的环境不一样，这称之为**交叉**。
- 交叉编译这个概念主要与**嵌入式开发**有关。

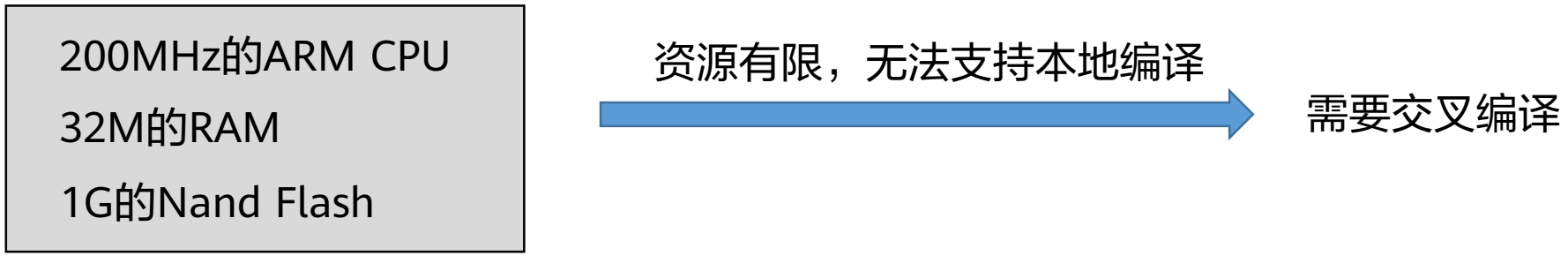


交叉编译流程

为何要有交叉编译?

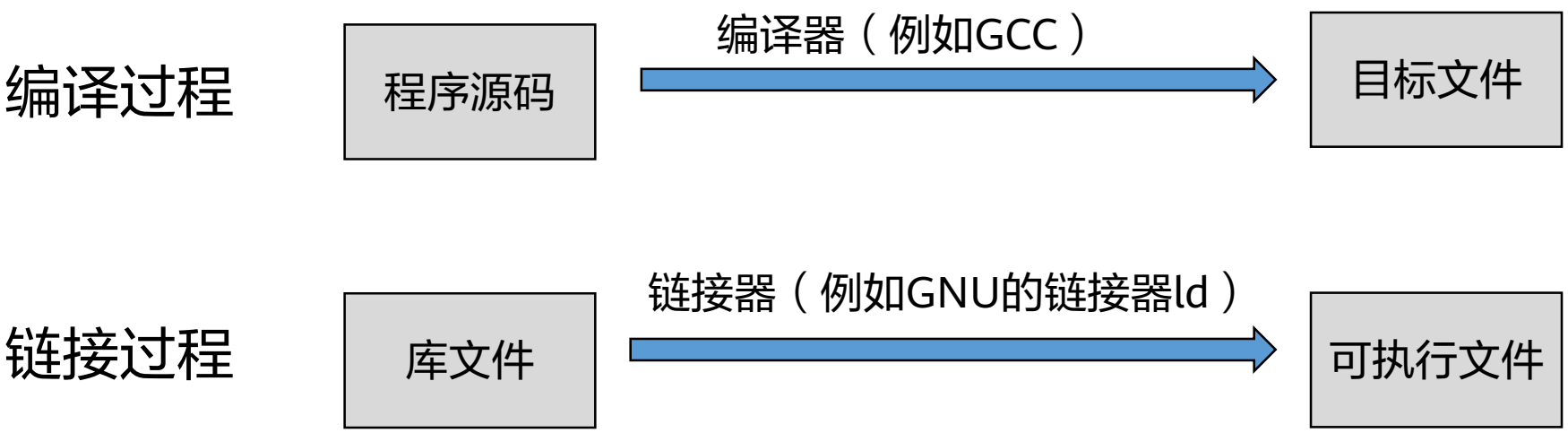
- 之所以要有交叉编译的主要原因是：**嵌入式系统中的资源有限**。
- 对于需要进行交叉编译的嵌入式操作系统而言，因为各种资源都十分有限，难以进行本地编译。

例：

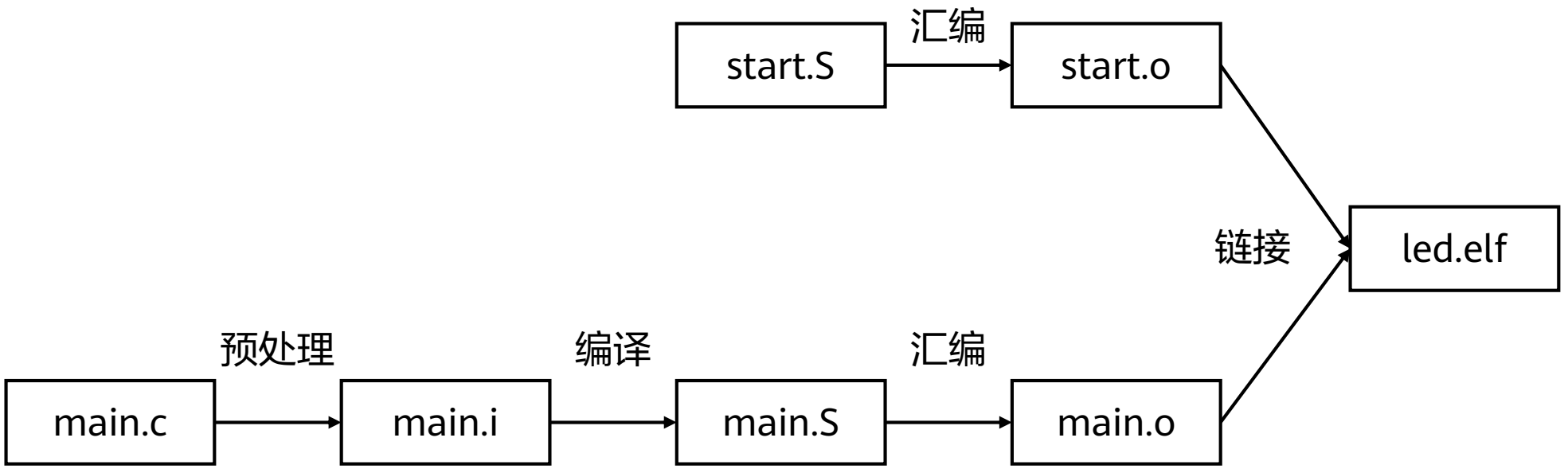


交叉工具链简介

- 交叉工具链是为了**达成交叉编译的目的**以及**生成可运行的程序或库文件**所使用的工具链。
- 交叉工具链内部的执行过程和逻辑主要包含了两个方面：**编译与链接**。



程序编译步骤



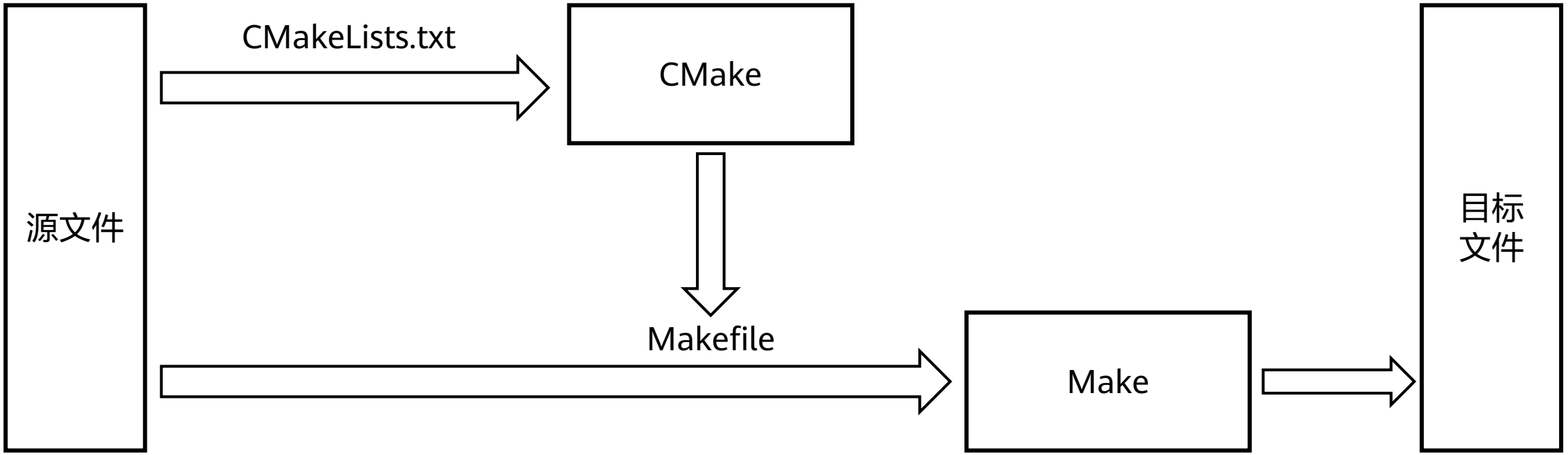
编译构建工具 - Make

- 一个C语言源文件从源文件到目标文件的过程叫做编译，但是一个项目中可能存在多个源文件。这就涉及到多个文件的编译顺序问题，程序**构建**过程就是安排文件的编译先后顺序。
- Make是一种构建工具，它属于GNU项目。Make命令执行时，需要一个**Makefile**文件告诉make命令如何去编译和链接程序。在执行make命令的过程中，make会自动查找Makefile文件并执行，这就避免了开发者一步步的手动编译文件。

编译构建工具 - CMake

- 虽然Make和Makefile简化了手动构建的过程，但是编写Makefile文件仍然有点麻烦，为了进一步的简化编译构建的工作，就有了CMake工具。
- CMake工具可以通过CMakeLists.txt文件生成Makefile文件，CMakeList.txt的编写难度较Makefile文件的编写有显著的降低。

Make 与 CMake 的关系



目录

库移植概述 - 什么是库?

- 库的本质是可执行代码的二进制形式，库文件可以在编译时**由编译器直接链接到可执行程序中**，也可以在运行时由操作系统的运行环境根据需要动态加载到内存中。
- 一组库可以形成了一个发布包，具体发布多少个库完全由库的提供商决定。
- 现实中**每个程序都要依赖很多基础的底层库**，不可能每个人的代码都从零开始，因此库的存在意义非同寻常。
- 在C语言中，C标准库是一组C内置函数、常量和头文件，例如<stdio.h>，<stdlib.h>，<math.h>等等。这个标准库可以作为C程序员的参考手册。

CMake 方式库移植 - 源码目录结构

- 以完成双精度进制双向转换的double-conversion库为例，其部分源码目录结构如下：

名称	描述
double-conversion/cmake/	CMake组织编译使用到的模板
double-conversion/double-conversion/	源文件目录
double-conversion/msvc/	/
double-conversion/test/	测试用例源文件
double-conversion/BUILD	/
double-conversion/CMakeLists.txt	CMake方式顶层编译组织文件
double-conversion/COPYING	/
double-conversion/Makefile	/
double-conversion/README.md	/
...	

CMake 方式库移植 - 移植思路

- 移植思路：
  - 通过修改工具链，交叉编译该三方库，生成HarmonyOS平台的可执行文件，最后再通过GN调用CMake的方式添加到HarmonyOS工程中。

设置交叉编译	测试	添加库至工程
CMake方式可通过指定工具链进行交叉编译，修改并编译该库，生成HarmonyOS平台的可执行文件。	搭建HarmonyOS环境并执行测试用例完成测试。	拷贝已经能够成功交叉编译的库到HarmonyOS的third_party目录，为了不修改要移植的三方库目录下的BUILD.gn文件，再添加一层目录放置新增的gn转CMake编译适配文件，最后再构建编译。

Makefile 方式库移植 - 源码目录结构

- 以第三方库XML解析器yxml为例，其部分源码目录结构如下：

名称	描述
yxml/bench/	benchmark相关代码
yxml/test/	测试输入输出文件，及测试脚本
yxml/Makefile	编译组织文件
yxml/.gitattributes	/
yxml/yxml.c	/
yxml/yxml.c.in	/
yxml/yxml.h	/
yxml/yxml.md	/
yxml/yxml-states	/
...	

Makefile 方式库移植 - 移植思路

- 移植思路：
  - Makefile方式的库移植思路与CMake类似，主要的区别在于CMake方式通过修改CMakeLists.txt设置工具链，而Makefile方式需要自行手动修改Makefile文件进行工具链设置，除此之外，在将库添加至工程时创建的适配文件BUILD.gn和condig.gni与CMake方式有区别，其他完全一致。

设置交叉编译	测试	添加库至工程
设置Makefile的交叉编译工具链，修改并编译该库，生成HarmonyOS平台的可执行文件。	yxml库测试步骤与double-conversion库基本一致，搭建HarmonyOS环境并执行测试用例完成测试。	拷贝已经能够成功交叉编译的库到HarmonyOS的third_party目录，为了不修改要移植的三方库目录下的BUILD.gn文件，再添加一层目录放置新增的gn转CMake编译适配文件，最后再构建编译。

思考题

1. (判断题)由于HarmonyOS目前不支持市面上所有的芯片及开发板，所以需要对操作系统进行移植以供市面上的设备使用。(　　)

   A. 正确

   B. 错误

2. (判断题)一般嵌入式开发所使用的编译方式为本地编译。(　　)

   A. 正确

   B. 错误

思考题

3. (单选题)下列选项中，关于操作系统移植步骤的描述正确的是哪一项？(　　)

A. 环境准备 --> 内核移植 --> BootLoader移植 --> 根文件系统制作

B. 环境准备 --> BootLoader移植 --> 内核移植 --> 根文件系统制作

C. BootLoader移植 --> 环境准备 --> 内核移植 --> 根文件系统制作

D. 环境准备 --> 根文件系统制作 --> 内核移植 --> BootLoader移植

本章总结

- 通过本章节的学习，您可以了解到移植的概念、原因以及场景，同时还可以知晓与移植相关的编译、构建等基础知识。
- 除此之外，本章节还介绍了如何对第三方库进行移植。

学习推荐

- HarmonyOS官网社区：
  - https://www.harmonyos.com/cn/home/
- HarmonyOS应用开发文档：
  - https://developer.harmonyos.com/cn/home/
- HarmonyOS设备开发文档：
  - https://device.harmonyos.com/cn/home/
- OpenHarmony开源地址：
  - https://gitee.com/openharmony
- 华为人才在线：
  - https://e.huawei.com/cn/talent/#/

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI

附录1：思考题参考答案

HUAWEI

参考答案 (1)

- 1-2 (P36) 参考答案：
  - 1. ABCD。HarmonyOS系统主要分为内核层、系统服务层、框架层、应用层。
  - 2. AB。HarmonyOS的UI框架支持JAVA和JS语言。
- 1-4 (P55) 参考答案：
  - 1. A。通过HarmonyOS的分布式数据管理技术，能够让开发者轻松实现全场景、多设备下的数据存储、共享和访问。
  - 2. BCD。HarmonyOS支持根据硬件形态和需求、硬件资源情况和功能需求、编译链关系来实现弹性部署。

参考答案 (2)

- 2-1 (P70-71) 参考答案：
  - 1. A。代码编译在linux环境下进行。
  - 2. B。DevEco Studio是应用开发的IDE。
  - 3. B。设备开发常用语言是C或者C++。
- 2-2 (P82-83) 参考答案：
  - 1. A。CMSIS是ARM公司为Cortex芯片设计的一种标准。
  - 2. A。CMSIS与POSIX都可以增强软件的可移植性，降低开发难度。
  - 3. ABD。在CMSIS架构中，MCU层包含Cortex（ARM处理器），系统定时器与调试与追踪接口。

参考答案 (3)

- 3-1 (P106-107) 参考答案：
  - 1. B。采用的是自旋锁。自旋锁与互斥锁最大的区别就在于未持有锁的进程是否会持续等待持有锁的进程释放资源，自旋锁会持续等待，互斥锁直接sleep。
  - 2. A。Zombies表示僵尸态，是进程的退出态。实际上僵尸态的进程并不是进程完全退出，而是此时还留有一个僵尸进程，它不持有任何资源，知识保留了一个在进程列表中的位置，它的父进程会回收此僵尸进程，如果僵尸进程的父进程退出了，僵尸进程会演变为孤儿进程，被Init进程回收。
  - 3. AB。互斥锁与读写锁是HarmonyOS常用的两种线程锁。
- 3-2 (P115-116) 参考答案：
  - 1. A。共用同一个物理地址，但是快表上的编号不同，两个应用进程之间无法感受到对方的存在。
  - 2. B。NFS默认使用TCP。
  - 3. D。malloc函数申请的内存在堆区。

参考答案 (4)

- 3-3 (P123-124) 参考答案：
  - 1. A。互斥锁是用于保证数据操作一致性的机制。
  - 2. A。消息队列可以实现异步消息处理，缓冲消息，对于任务间通信可以实现发送发与接收方的解耦。
  - 3. AB。软件定时器的两个主要功能是制定任务循环执行的周期和用于任务的延迟执行。
- 4-3 (P162-163) 参考答案：
  - 1. A。I2C时序主要有四个元素组成：起始信号，终止信号，应答(0)，非应答(1)。
  - 2. B。ADC主要功能是将模拟信号转换为数字信号。
  - 3. B。驱动设备可分为字符设备驱动，块设备驱动，网络设备驱动。

参考答案 (5)

- 5-2 (P180-181) 参考答案：
  - 1. A。分布式远程启动可以使得一个设备远程启动另一个设备。
  - 2. A。分布式远程启动使用的方法是startAbility方法。
  - 3. ABC。远程启动FA需要构造want，构成want的参数有远端设备id、包名、ability类名。
- 5-4 (P196-197) 参考答案：
  - 1. A。启动恢复子系统在内核启动之后，在应用启动之前，负责系统的关键进程的启动，此外还有设备恢复出厂设置的功能。
  - 2. A。大型系统支持的系统属性主要包括系统信息与设备信息。
  - 3. A。init启动引导组件负责在系统启动阶段启动关键服务进程。

参考答案 (6)

- 5-5 (P202-203) 参考答案：
  - 1. A。软总线可以实现1+8+N个设备的连接。
  - 2. A。软总线的一项重要优势就是具备“自发现”能力。
  - 3. ACD。软总线的数据传输具有高带宽，低时延，高可靠的特性。
- 6-5 (P237-238) 参考答案：
  - 1. A。软总线可以实现1+8+N个设备的连接。
  - 2. A。HarmonyOS的安全目标是让正确的人，用正确的设备，正确的使用数据。
  - 3. ABD。没有应用市场撤销应用的验签。

参考答案 (7)

- 7-3 (P267-268) 参考答案：
  - 1. A。Shell可用于HarmonyOS设备开发的功能调测。
  - 2. A。Shell命令可以通过动态注册和静态注册两种方式完成注册。
  - 3. BC。mkdir用于创建文件夹，rm用于删除文件或文件夹。
- 8-2 (P292-293) 参考答案：
  - 1. A。由于HarmonyOS目前不支持市面上所有的芯片及开发板，所以需要对操作系统进行移植以供市面上的设备使用。
  - 2. B。一般嵌入式开发所使用的编译方式不是本地编译。
  - 3. B。首先进行环境准备，之后进行BootLoader移植，再进行内核移植，最后完成跟系统制作。

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI