第 1 章
鸿蒙系统简介

要了解一个新事物，就需要去追溯它的过去，去畅想它的未来。本书的主角是鸿蒙操作系统，对于这个国产操作系统、民族之光，我们有必要用一章的篇幅，从多个方面来详细讲解一下其特点，阐述我们对其的热爱及对未来的信心。

1.1 智能手机操作系统

1.1.1 智能手机操作系统发展历史

时代变迁，新技术替代旧技术是技术发展的趋势。我们所处的时代，技术正在发生翻天覆地的变化：新能源正在取代传统能源，电动汽车正在取代传统燃油车，5G 网络正在取代 4G 网络，新一代操作系统正在取代智能手机操作系统。这里所说的“新一代操作系统”，其实就是指的鸿蒙这样的万物互联系统。我们以智能手机操作系统的发展为例，慢慢引入对鸿蒙系统的讨论。

2007 年以前，以诺基亚为代表的功能手机还是全球手机的领导者，经典的板砖造型和开机铃声，让人记忆深刻。当时的操作系统还是 Series30/40 等功能机操作系统。

2007 年 1 月，iPhone 横空出世。由于诺基亚手机的应用程序比不上 iPhone 的应用程序，加上诺基亚的固执，所以其迅速被时代抛弃，全球市场份额快速衰减。

2008 年 9 月，Android 1.0 发布：此后的两年，Google 迅速发布了多个 Android 版本。智能手机操作系统以迅雷不及掩耳之势被 iOS 和 Android 两大系统占据。

在智能手机的竞争中，微软的发力较晚，直到 2010 年，微软才正式发布了 Windows Phone 7.0 系统。比 iOS 和 Android 足足落后了两年多。在这两年中，iOS 和 Android 羽翼已日渐丰满。毫无疑问，相比 Android 和 iOS，Windows Phone 是一个非常尴尬的存在。与 Android 相比，Windows Phone 的闭源路线无法战胜 Android 系统的开源路线。与 iOS 相比，在友好甚至惊艳的用户体验上，Windows Phone 从来都不是其对手。还有一点最为致命，就是生态，数以千万计的研发人员，不愿为了转入 Windows Phone 系统而熟悉一种新的开发方式。

2011 年初，诺基亚和微软同时发现了危机，两大巨头宣布合作，诺基亚官方宣布放弃塞班（Symbian）品牌，转而投入微软的怀抱，全面支持 Windows Phone 系统，不过为时已晚。Android 和 iOS 的生态已经发展起来，想要颠覆它们的生态系统，实在太难。多年后，2019 年 12 月，微软表示将暂停 Windows 10 Mobile 的更新，这意味着微软将放弃移动操作系统。作为份额仅占 0.1% 的手机操作系统，Windows Mobile 将彻底地退出历史舞台。

与此同时，2019 年 8 月 9 日，华为在东莞举行华为开发者大会，正式发布鸿蒙系统，鸿蒙时代慢慢开启。在介绍鸿蒙系统之前，我们来研究一下 Android 和 iOS 操作系统走过的路。

1.1.2 智能手机操作系统的开放与封闭之争

在智能手机操作系统发展的过程中，特别是 Android、iOS、Windows Phone 的竞争

过程中，操作系统的开放与封闭问题非常值得思考。

“开放”与“封闭”本身是一对反义词，也代表着当今世界智能设备中举足轻重的两款操作系统的态度：Android 选择的是开放，iOS 选择的是封闭。由于这两个系统几乎是同时跨入智能手机这个风口的，所以无论是开放还是封闭，这两个操作系统都展现出了巨大的商业价值。目前其市场价值不分伯仲。

1. 开放战略

开放即开放源代码，开放生态合作伙伴，开放应用商店等，Google 即选择了开放这条路。开放有以下一些优势：

- 商业风险低。开源软件的生命周期长，维护者多，不用担心开源软件的 bug 没人修改，而如果闭源，会担心闭源软件的公司倒闭。开源软件中的 bug 如果实在没人修改，那么自己修改即可。以 Android 为例，维护者有众多国际一线公司，首先是 Google 自身，其次有华为、高通、小米、Sony 等。对于 Android 来说，基本上都是大型的公司在共享代码，而个人贡献者相对较少，原因是担心出了问题，找不到个人贡献者来修改。由于基本上是国内外顶尖科技公司在贡献代码，所以其代码质量非常高。
- 开源软件能保证好的品质。相对于大多数闭源产品，开源产品的社区一般有众多免费的设计、编码、测试、维护人员，他们以满腔热血在工作，不计报酬。因为同一份代码被多个人阅读审查，所以一旦发现 bug，整个社区也会积极修复。
- 安全性相对更高。开源软件中一般很少有病毒，因为很少有人把病毒放在开源软件中，让成百上千的研发人员发现。同时，由于参与研发的人员多，能够更快地发现软件的安全性 bug。
- 成本低。开源社区集合了大量的软件开发者，很多开发者是不需要工资的，开源工作者都是默默地付出劳动成果，为理想而战。

2. 封闭战略

封闭一般是指封闭源代码，即封闭部分接口和权限。例如，2014 年之前的 iOS 系统，是不允许安装第三方输入法的，对于在计算机上用惯了讯飞、搜狗输入法的人来说，iOS 自带的输入法使用起来确实不便，这就是封闭的坏处。但是封闭也是有好处的，例如，有几款输入法被 App Store 下架，就是因为其收集了大量用户文字输入信息，用于分析用户画像，提供给第三方电商平台，导致很多用户的数据被泄露，这就是开放导致的一些问题。

封闭有以下一些优势：

- 有助于保持品牌优势。一直以来，iOS 系统都是较为封闭的。iOS 系统只能用于自身的硬件中，其他厂商不能使用。这种封闭性，让 iOS 有更一致的体验，在很长一段时间内，iOS 的流畅度都远远高于 Android 系统，良好的用户体验对于树立品牌形象有很大的帮助。
- 满足核心需求。苹果只满足用户的核心需求，它找到一个卖点，以此聚集用户

打造基础生态圈，再通过基础生态圈形成辐射影响，用户的体验和习惯是可以被影响和引导的。

- 安全性更高。众所周知，iOS 的病毒是少于 Android 的。这是因为 iOS 的应用是通过 App Store 审核的，苹果的审核是很严格的，要求很高，恶意或病毒程序是通不过审核的。

1.2 鸿蒙系统的发展历史

在华为“2012 诺亚方舟实验室”专家座谈会上，任正非提出要做终端操作系统，以防患于未然，要在“断了我们粮食的时候，备份系统要能用得上”，而这就是“鸿蒙”操作系统的起点。鸿蒙系统主要经历了以下几个发展阶段：

- 2012 年，华为开启了操作系统“B 计划”，开始规划自有操作系统“鸿蒙”，不被卡脖子是华为的重要战略。
- 2016 年 5 月，华为消费者 BG 软件部开始立项研发“分布式操作系统 1.0 版本”，这就是鸿蒙系统的雏形。
- 2017 年 5 月，华为消费者 BG 软件部研发完成“分布式操作系统 1.0 版本”，并开始研发 2.0 版本。
- 2018 年初，任正非听取华为消费者 BG 业务汇报时，非常认可自研操作系统。
- 2018 年 5 月，华为消费者 BG 投资委员会投票同意“分布式操作系统”的研发，“分布式操作系统”正式成为 BG 核心项目。
- 2019 年 5 月，经过三年研发，“分布式操作系统”已开始成熟，为了更好地推广，更名为“鸿蒙”。传说在盘古开天辟地之前，世界是一团混沌状，那个时代称作“鸿蒙时代”，后来该词也常被用来泛指远古时代。鸿蒙系统寓意着它将开启一个新的伟大的物联网系统时代。
- 2019 年 8 月，鸿蒙系统 1.0 发布。作为首批试点，荣耀智慧电视中搭载了该系统。同时，余承东表示，鸿蒙系统实行开源战略。
- 2020 年 9 月，鸿蒙系统 2.0 发布。该系统逐步开源，电视、手机、车机等设备可以使用鸿蒙系统。在小型设备领域，鸿蒙系统支持 128K 到 128MB 的设备，并首先对这些设备进行了开源。
- 2020 年 12 月，鸿蒙系统面向移动开发者，推出鸿蒙 Beta 版。
- 2021 年 4 月，鸿蒙系统扩大开源范围，向内存 128MB 到 4GB 的设备开源，华为 Mate X2 手机开始使用鸿蒙系统。
- 2021 年 10 月，鸿蒙系统计划向 4GB 内存以上的设备开源，手机厂商可以修改系统源代码以适应自己手机的特殊需求，这和 Android 走的是同一条路线。

到成书为止，鸿蒙系统已经发布了两个版本，分别是鸿蒙系统 1.0 和鸿蒙系统 2.0。下面对这两个版本做简要的介绍。

1.2.1　鸿蒙系统 1.0 介绍

2019 年 8 月 9 日，华为在东莞松山湖举行了一年一度的开发者大会，正式发布操作系统鸿蒙系统 1.0，发布会上宣布荣耀智慧屏、荣耀智慧屏 Pro 都搭载了鸿蒙系统。

鸿蒙系统是一款全场景分布式系统，可按需扩展，实现更广泛的系统安全，主要用于物联网，特点是低时延。分布式系统是一种面向未来的操作系统，华为定义的分布式操作系统有以下几个特性：

- 多终端之间的能力共享，互为外设。例如 A 终端能操作 B 终端的摄像头，鸿蒙的前辈们是不具备这个激动人心的功能的。
- 系统与硬件解耦，弹性部署。鸿蒙系统支持 128KB 到 4GB 的设备，从微型设备到大型设备都支持。可以根据资源情况，定制内核，增加或删除一些模块。鸿蒙系统支持低资源设备，这完全符合物联网设备的特性。
- 应用一次开发，多端部署。鸿蒙实现了应用跨终端运行，一次开发，可以将同一程序部署到不同的运行鸿蒙系统的终端上，如手机、电视、车载系统中。

鸿蒙系统 1.0 实现了模块化解耦，对应不同设备可弹性部署。弹性部署是指可以根据硬件的情况，对系统进行裁剪，最小化部署到硬件设备上。鸿蒙系统 1.0 已经构建了完整的四层架构，第一层是内核，第二层是服务层，第三层是开发框架层，第四层是应用层，为后续鸿蒙系统 2.0 的开发打下了良好的基础。

1.2.2　鸿蒙系统 2.0 介绍

2020 年 9 月 10 日，华为发布了鸿蒙系统 2.0。鸿蒙系统 2.0 重点在分布式架构上做了大量的升级优化。在分布式软总线、分布式数据管理、分布式安全等方面进行了重大的更新，开发者能够从这些更新中获得更多的开发能力。在推出鸿蒙系统 2.0 的同时，华为也联合了国内众多知名的厂商，如美的、九阳、老板等智能家居公司，在其生产的智能家居上搭载鸿蒙系统。若大家购买了这些产品，就能够很方便地使用相关功能。例如，可以在冰箱的屏幕中，观看手机视频软件播放的视频，完全实现分布式的视频通信；可以通过手机，便捷地操作空调等。

1.2.3　鸿蒙系统与物联网

鸿蒙系统是新一代的物联网操作系统，这是其最大的特点。鸿蒙系统从立项之初就不是为了手机而生的，而是为了物联网。从这个角度分析，鸿蒙系统并不是为了取代 Android 和 iOS 系统而开发的，而是一个全新的物联网系统。为什么现在需要一个全新并统一的物联网操作系统呢？这就要从物联网说起了。

1. 物联网的两层含义

从物联网的概念中，我们能够发现物联网有两层含义。第一，物联网仍然是基于互联网的，是使用互联网的基础设施，是互联网的扩展。第二，物联网的创新是将人与人的连接，扩展到人与物、物与物的连接，任何物之间都可以进行信息交换，即所谓的万物互联。

传统互联网中的信息交换一般通过 OSI 参考模型（七层网络模型）来处理，物联网则不一样。物联网的通信可能会通过多种设备，如蓝牙、红外、ZigBee、光通信、RFID 等技术，采用的协议栈多种多样，设备与设备间互联非常困难，需要互相兼容，才可能通信，所以这对操作系统的要求非常高。而目前没有这样的操作系统。鸿蒙系统最大的优势就是其轻量化的内核，能较容易地支持各种协议、各种大小的设备，让各种协议、设备高效地互联起来。

2. 物联网时代鸿蒙系统的巨大优势

操作系统无法通过复制商业模型、投资巨额资金成功，必须在市场萌芽期快速占据市场，才可能立于不败之地。桌面时代微软的 Windows 如此，智能手机时代的 Android、iOS 系统也如此。所以摆在鸿蒙系统面前的主要难题是生态问题——不能和 Android、iOS 抢生态，因为突破 Android、iOS 的生态异常困难，只能去打造一个新的生态，即物联网操作系统生态。

目前，智能手机操作系统已经发展到了成熟期，新意越来越少，操作系统的风口正在从智能手机操作系统转向物联网操作系统。相较于 Android、iOS 来说，鸿蒙系统在物联网上是做得最好的，原因是 Android、iOS 本身并不是为物联网设计的操作系统，而鸿蒙系统从诞生开始，就是一个物联网操作系统。

软件银行集团董事长孙正义在演讲中说，物联网将会引领下一轮技术爆炸，就像历史上寒武纪爆发形成了无数新物种一样，用不了多久，到 2035 年，联网的物联网设备数量就将达到 1 万亿，届时，平均每个人有 100 个设备联网。

在这些设备中，每一个个体都和手机有很大的差异，它们的成本、标准化都各不相同，这便需要一个新的操作系统来支撑如此异构的硬件设备。目前，鸿蒙系统正是这样一个系统。

鸿蒙系统不是以往操作系统的一种重复，而是一种全新的操作系统。如果顺利，属于华为鸿蒙系统的物联网时代即将到来。

1.3 鸿蒙系统的特点

计算机、手机、平板电脑、物联网设备的操作系统生态一般由硬件、操作系统、应用程序组成，如图 1-1 所示。

图 1-1　操作系统生态

操作系统控制和管理着计算机的硬件和软件资源，合理地对各类作业进行调度。操作系统对下拥有控制硬件的能力，对上为应用程序提供底层支撑，为应用提供访问硬件，如内存、磁盘、鼠标、键盘、相机等各种外设的能力。

从系统设计来看，一个操作系统的体系架构一般分为用户态和内核态。内核态中运行着系统内核。内核处于一个操作系统最核心的地位，其本质上是一种系统软件，主要控制计算机系统的硬件资源，为应用软件提供运行的支撑环境，如图 1-2 所示。

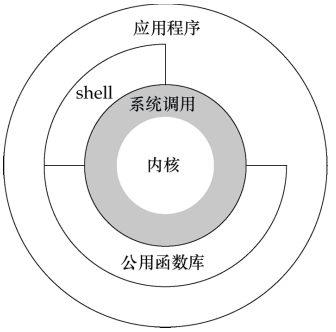

图 1-2　操作系统体系架构

内核是通过 API 函数向外提供功能的，术语叫作系统调用。系统调用函数是操作系统提供功能的最小功能单位，系统调用函数一般有：

- 进程管理函数。
- 文件管理函数。
- 系统控制函数。
- 内存管理函数。
- 网络管理函数。
- 用户管理函数。

系统调用函数高度抽象、粒度非常细，一般一个操作系统中的系统调用函数较为稳定，不会经常变化函数签名。系统调用函数的个数也非常少，如 Linux 一般约有 250 个，FreeBSD 约有 320 个。Android 基于 Linux，所以系统调用函数和 Linux 几乎一致。

系统调用函数都属于细粒度的操作函数，要实现一些复杂的功能，就需要组合多个系统调用函数来实现，这就形成了一些公用函数库。

此外，在图形界面还未流行前，出现了一种 shell 命令行程序，用户主要通过 shell 来和系统交互。

了解了内核的一些功能，下面主要对内核的分类做一些介绍，同时引出鸿蒙系统使用什么类型的内核。

1.3.1 内核特点简介

一个操作系统最核心的部分是其内核，内核就是操作系统的最主要代码，也可以简单地认为内核就是指操作系统。内核分为宏内核和微内核。下面对两种内核进行简要的说明。

1. 宏内核

宏内核，又称单内核，是一种传统的内核结构。宏内核定义所有核心的程序都是在内核空间运行的，宏内核中的所有服务都是以特权模式执行的。宏内核要求所有服务模块都在同一个地址空间运行，所有服务使用共享的地址空间、内存空间。

宏内核从设计之初，就重点考虑了效率问题。宏内核模块由于使用的是同一个地址空间，所以它们之间能够更直接地通信，通信效率更高，从而运行效率也更高。

但是宏内核也有天生的缺陷，其稳定性相对较差。由于各个模块运行在同一地址空间，所以，如果某个模块出现 bug，就会导致整个系统出现异常。因此，如果宏内核设计及开发的质量不佳，就容易导致稳定性差的问题出现。

UNIX、Linux、iOS 系统是宏内核。

2. 微内核

微内核是提供操作系统核心功能的内核精简版本，它的核心设计原则是：能多小就多小。

微内核提供一组“最基本”的服务，如进程调度、进程间通信、存储管理、I/O 设备管理。其他服务如文件管理、网络服务等，并不是一个微内核必须有的。例如一个蓝牙设备，就不需要网络服务，仅仅只需要文件管理服务。

除了“最基本”的服务外，在有需求的情况下，微内核怎么支持其他复杂的服务呢？其他服务模块可以通过接口和微内核交流，从而扩展系统的功能。

Windows、鸿蒙系统是微内核。

3. 鸿蒙系统微内核的优势

鸿蒙系统是微内核。Android 系统（采用 Linux 内核）和 iOS 系统都是宏内核。宏内核将所有的模块放在一个内核中，如文件系统、内存管理、网络管理、磁盘管理、设备驱动等。其中，设备驱动是内核代码量最大的模块。为了支持众多系统，一般编译的时候，会将常用驱动编译到内核中，从而让内核镜像更大。

随着操作系统的功能越来越多，内核也越来越庞大，宏内核面临的挑战突显出来，主要有两个：

- 宏内核操作系统代码量非常庞大。Linux 3.3 的内核约 1500 万行，且每年以 100 万行的速度增长。虽然全球有约 8 万人在参与 Linux 内核的开发和测试，但是因为有如此大的代码量及系统复杂度，所以 Linux 内核潜在的问题也相当多，并且修改起来非常复杂。

- 大量的驱动程序、模块集中在一个内核中，导致内核模块之间的关联性强，可扩展性差。内核之间的模块互相依赖，导致在修改某个模块时，可能会影响其他模块。在多硬件适配的工作中，某些服务要匹配不同的硬件，要做大量的适配工作，导致服务不稳定，经常变化。这一点在物联网硬件中表现得尤为突出。物联网设备类型繁多，变化及组合方式多样，内核要适应这些设备，就需要针对不同的物联网设备做大量的适配工作，这也是物联网宣传了很多年，但是目前流行程度还是相对较低的原因之一。

鸿蒙系统微内核完全克服了宏内核的这些缺点，将内核做到足够精简。微内核中只保留核心的进程管理、内存管理、进程间通信，其他的非核心服务放到非内核中去执行，如文件系统、网络协议、一些驱动程序等。这样做的好处在于：

- 高扩展。由于很多服务不在内核态，所以服务开发者可以自行开发、裁剪服务，并直接在用户态运行服务，对内核不会有任何影响。这种方式实现了高扩展。
- 可靠性高。微内核的代码量非常少，可靠性高。用户态的服务不会影响内核态的服务，从而实现高可靠性。
- 安全性高。微内核由于只实现了核心功能，代码量非常少，所以可以通过多次评审，不断地进行黑白盒安全测试，确保内核的安全性。
- 可维护性高。内核态运行稳定，基本不需要维护。 运行在用户态的服务，可以正常启动与停止，即使出现问题，也不会导致内核崩溃。

1.3.2　鸿蒙系统分布式技术特性

除了鸿蒙系统微内核的优势，鸿蒙系统还解决了很多实际场景中的痛点，可以将软件与软件、软件与硬件、软件与资源快速共享起来。为了实现这些场景，鸿蒙系统引入了以下几个关键技术：

- 分布式软总线。
- 分布式设备虚拟化。
- 分布式数据管理。
- 分布式任务调度。

下面对这几项技术进行详细阐述。

1. 分布式软总线

分布式软总线是解决物联网硬件设备之间无缝通信的总线架构，类似于企业服务总线（ESB）。分布式软总线为设备之间的互联互通提供了统一的分布式通信能力，各种设备（如手机、平板电脑、智能穿戴、智慧屏、车机、智慧家居）能通过注册到分布式软总线，进行快速的数据、命令交互。分布式软总线为设备与设备之间无感发现和零等待传输创造了技术条件，设备注册到分布式软总线中，无须关心彼此的协议，协议互相兼容由分布式软总线解决。

分布式软总线可以解决很多目前体验不好的问题，以智能家居场景为例：

- 午睡的时候，可以用手机控制智能窗帘的外设开关：碰一下表示窗帘关闭，碰两下表示窗帘打开。这样就不需要在手机上打开控制窗帘的 App，再找到控制窗帘开关的按钮，这个过程可能需要 20 秒钟，而使用手机碰一下外设开关可以让整个流程非常自然快速。智能窗帘硬件也不用自己设计操作系统，直接使用鸿蒙系统即可，这样可以极大地节省软件研发成本。
- 一回家，你的智能穿戴设备就可以接入家庭软总线中，各个设备之间不用任何设置，即可互相快速通信。例如，正在工作的扫地机器人，可以识别你的手机，当发现主人回家时，机器人可以自己回到充电桩中。智能电灯发现主人回来了，可以自动打开灯光。这一切都不需要人为的任何操作。

2. 分布式设备虚拟化

分布式设备虚拟化平台可以将不同的设备连接起来，形成一个整体。例如，在一个手机上可以访问其他手机的摄像头，从而实现设备的资源融合，多种不同的设备共同形成一个超级虚拟终端，如图 1-3 所示。

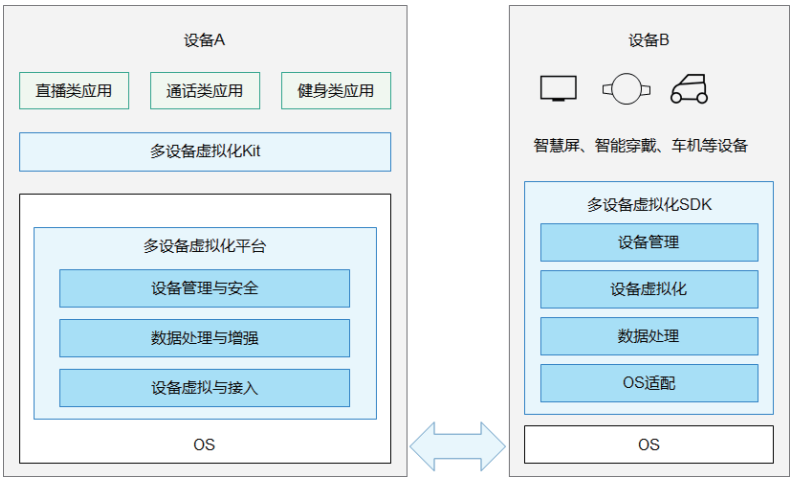

图 1-3　分布式设备虚拟化

这里以视频通话和游戏场景来说明分布式设备虚拟化的作用。

- 视频通话场景：在做家务时突然接听到视频电话，但使用手机不方便，这时可以将手机与华为智慧屏连接，并将智慧屏的屏幕、摄像头与音箱虚拟化为手机的本地资源，替代手机自身的屏幕、摄像头、听筒与扬声器，实现一边做家务、一边通过智慧屏和音箱来视频通话。
- 游戏场景：在智慧屏上玩游戏时，可以将手机虚拟化为遥控器，借助手机的重力传感器、加速度传感器、触控能力，为玩家提供更便捷、更流畅的游戏体验。

3. 分布式数据管理

分布式数据管理基于分布式软总线的能力，实现应用程序数据和用户数据的分布式

管理。所谓分布式数据管理，是指应用程序数据和用户数据不与某台设备绑定，数据可以在跨设备间进行访问。开发者不用关心数据存储在哪台设备上，直接调用相应的模块就能访问到数据，如图 1-4 所示。

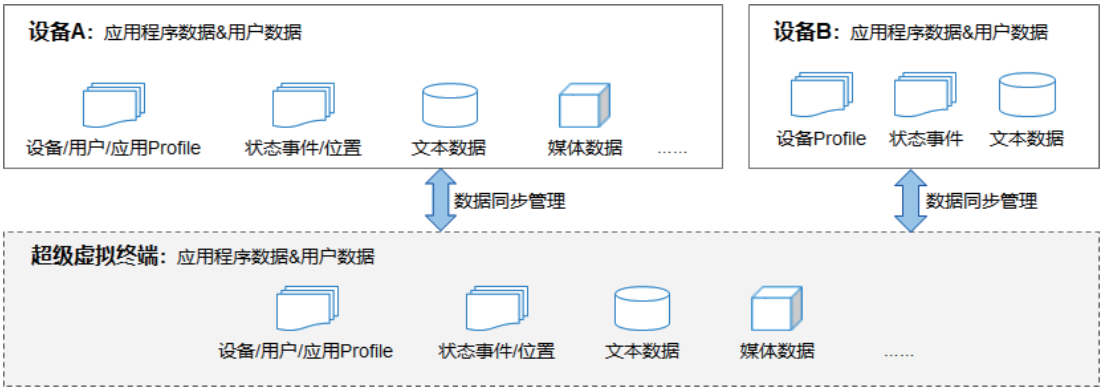

图 1-4　分布式数据访问

从图 1-4 中可以看出，设备 A 中有设备 / 用户 / 应用 Profile、状态事件 / 位置、文本数据、媒体数据等数据，设备 B 中有设备 Profile、状态事件、文本数据。设备 A 和设备 B 都将数据同步到超级虚拟终端，这样超级虚拟终端就拥有设备 A 和设备 B 的所有数据，并且可以实时同步更新。这就像设备 A 和设备 B 使用同一块磁盘一样，任何一个设备对磁盘的更改，在其他设备中都能查看到。

这里有一个概念——超级虚拟终端（super virtual device），又称超级终端，其通过分布式网络技术将多个终端 / 设备的能力进行整合，形成相互协同的工作场景。

例如，在搭载鸿蒙系统的手机上，可以直接向附近的计算机、平板电脑、手表、窗帘等推送相关的数据，从而控制其他设备实现相应功能，这也是华为为鸿蒙系统“多端协同，多端流转”提供的一个重要能力。

超级虚拟终端将会带来十分强大的生态体验，能够将几乎所有搭载鸿蒙系统的电子设备融为一体，打造完全无缝的使用体验，让设备与设备之间的互联更为高效。

那么其他系统能否互联实现同样的功能呢？从某种程度来说是可以部分实现的，只是无法达到鸿蒙系统流畅的体验，同时硬件成本非常高昂。

以 Android 为例，窗帘必须搭载一套成本相对较高的硬件，才能安装 Android 系统，从而实现窗帘与手机之间无缝的集成互联。如果窗帘的硬件较弱，无法安装 Android 系统，那么窗帘和 Android 手机就必须通过一些成本较低的蓝牙进行通信，并且 Android 手机上需要安装能接收和控制窗帘的 App。从体验上来说，这个连接过程比较烦琐，响应速度也会变慢一些，无法做到无感操作。

4. 分布式任务调度

分布式任务调度表示任务调度可以跨设备进行，任务可以通过远程调用，执行在其他设备上，从而实现一些只有其他设备才具有的能力。

分布式任务调度有以下使用场景：

- 长任务场景：在手机上正在处理一个视频文件，需要 20 分钟。可以把这个处理任务的信息发送到手表上，随时在手表上查看视频处理信息。
- 导航场景：如果用户驾车出行，上车前，在手机上规划好导航路线；上车后，导航自动迁移到车机和车载音箱；下车后，导航自动迁移回手机。如果用户骑车出行，在手机上规划好导航路线，骑行时手表可以接续导航。

1.4 鸿蒙系统的分层架构

在软硬件开发中，为了让一个复杂的系统能够顺利地实现，常用的方法是分层设计。有一个大家熟知的例子，就是网络七层协议。鸿蒙系统也是分层设计的，从下到上由 4 层组成，分别是：内核层、系统服务层、框架层和应用层（见图 1-5）。

图 1-5　鸿蒙系统的技术架构

下面对这 4 层进行详细介绍。

1.4.1 内核层

内核层是操作系统的核心，主要是管理硬件资源，同时为外层应用提供系统调用。为了管理各种硬件（如显卡、声卡），需要有驱动子系统。为了管理进程、线程、内存等资源，需要有内核子系统。

鸿蒙系统采用多内核设计，支持针对不同硬件资源受限设备选用适合的内核。目前，鸿蒙系统包含 Linux 内核、LiteOS 内核。 Linux 内核主要兼容 Android。由于 Android 是依赖于 Linux 内核的，所以为了支持 Android，必须要内置 Linux 内核。华为对 Linux 内核进行了优化裁剪，去掉了很多不需要的组件， 同时添加了鸿蒙的一些新特性，如分布

式框架、分布式数据存储等。

LiteOS 内核主要支持小型物联网设备，可以根据硬件资源进行裁剪；LiteOS 内核还具有低功耗的特点，非常适合价格便宜的微小硬件设备。

驱动子系统包含了常用的驱动程序，其主要模块包括硬件驱动框架（Hardware Driver Foundation，HDF）。HDF 用于统一对外设进行管理，访问、修改、控制外设的状态。同时，HDF 还提供驱动开发的框架。驱动开发是开发工作中难度较大的部分，需要对操作系统及硬件接口及原理有较多的了解，才能写出一个完整的、正确的驱动。驱动开发的框架可以简化驱动的开发，让程序员不需要学习太复杂的驱动原理，只需要在框架留出的代码桩处写代码，即可完成一个简单的驱动程序。

1.4.2　系统服务层

系统服务层主要提供系统的基础服务，为上层框架提供底层系统调用支持。系统服务层包含了大量的基础服务组件，如分布式软总线、分布式数据管理、分布式任务调度等。其中方舟多语言运行时子系统，能够支持多种编程语言的编译及运行，为系统支持多语言提供帮助。

1.4.3　框架层

框架层主要为应用开发者服务，是系统对应用开放的接口。为了丰富开发生态，鸿蒙系统提供了多种语言的接口，目前支持 Java/C/C++/JS 等语言。同时，框架层还提供了 UI 框架、用户程序框架、Ability 框架以及操作硬件服务的开发组件，开发者可以通过这些组件快速实现特定功能，从而加速应用的开发。

1.4.4　应用层

应用层包括系统应用和非系统应用。系统应用包括鸿蒙桌面、控制栏、系统设置、电话、短信等。非系统应用是指第三方开发者开发的，可以通过应用商店下载安装的应用，如 QQ、微信、抖音、板栗看板等。本书主要围绕框架层、应用层，对软件开发进行讲解。

1.5　小结

现在，你应该感觉很平静，甚至充满了成就感，因为你开始逐步了解了鸿蒙系统是什么，以及鸿蒙系统能给世界带来什么。鸿蒙以其先进的分布式系统设计理念，让万物互联更为容易。在不久的将来，软硬件生态会逐步跟上，到那时候，我们在日常生活中能体验到以下场景：每天下班回家，再也不用在指纹锁中录入指纹，只需要带上智能

手表，轻轻地敲两下门，智能手表就可以通过 NFS 自动和电子锁进行沟通，确认是谁回家了。通过超级虚拟终端技术，智能手表将电子锁当作自己的外设，从而自动操作了电子锁。这种超级虚拟终端技术，是一种对用户透明，能让万物无感互联的令人兴奋的技术突破。

当智能家居感应到我们回到家时，空调会自动打开，窗帘会自动打开，冰箱的显示屏上也会提醒你晚餐可以吃什么，并根据你的喜好，在食物不够的时候，提前在电商平台下单；热水壶能将水调整到你喜欢的温度；LED 灯泡也能自动识别到你回家，为你营造出你喜欢的环境光。

这些场景在很多科幻电影中以及在我们的脑海中都出现过，只是互联万物的鸿蒙系统让这种可能离我们更近，未来可期。