《操作系统》实验指导书

实验三——OpenHarmony操作系统实验

北京邮电大学网络空间安全学院

2024年 5 月

目录

实验三： OpenHarmony操作系统实验

1 实验类别

综合实验

2 实验内容和实验目的

使用QEMU硬件模拟器运行开源鸿蒙OpenHarmony4.x操作系统

3 实验学时

4学时（课后时间完成）

4 实验组人数

每组 1-3 人，合作完成操作系统平台搭建、操作系统安装、编译源码进行内核升级、验证操作基本模块管理并撰写实验报告。

5 实验设备环境

一台装有并安装VMWare软件的计算机。

6 教学要点与学习难点

在课程教学中，涵盖了通用操作系统、Linux系统结构、国产操作系统结构等相关知识以及操作系统的基本原理，但是学生对于操作系统的运行细节，代码编译以及Linux常用命令缺乏感性认识，理解不足。在本实验中，学生通过安装安装虚拟机、编译openHarmony操作系统、安装QEMU模拟器等，了解行内核模块编程、内存管理、进程管理、中断实验、设备管理等教学知识点，以加深对课堂上基础知识的理解。

7 实验步骤提示

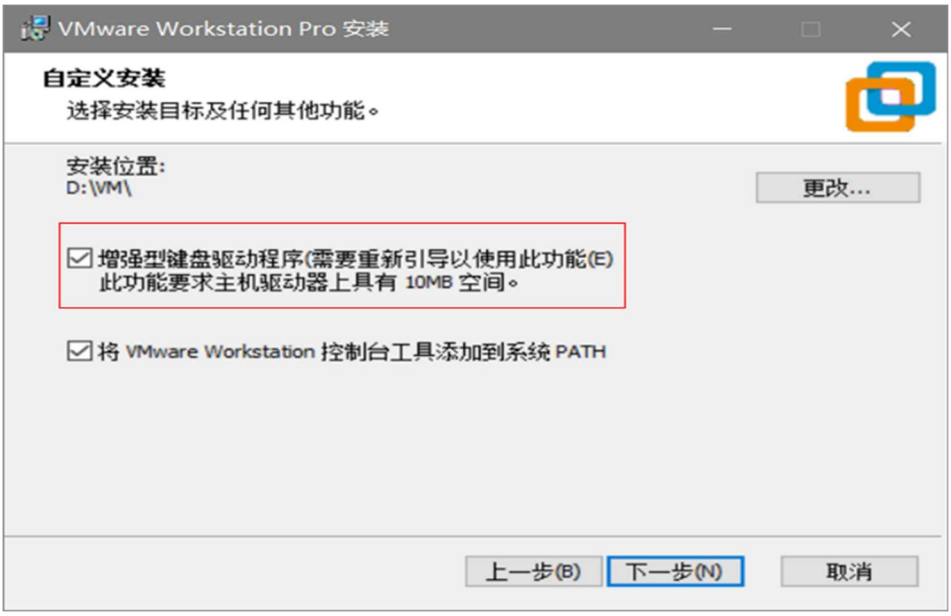
7.1 安装VMWare WorkStation

1） 版本：VMWare WorkStation 16 Pro

2） 下载地址：https://www.vmware.com/cn/products/workstation-pro/workstation-pro-evaluation.html

3） 采用重新编译源代码的方式将内核更新至最新版。

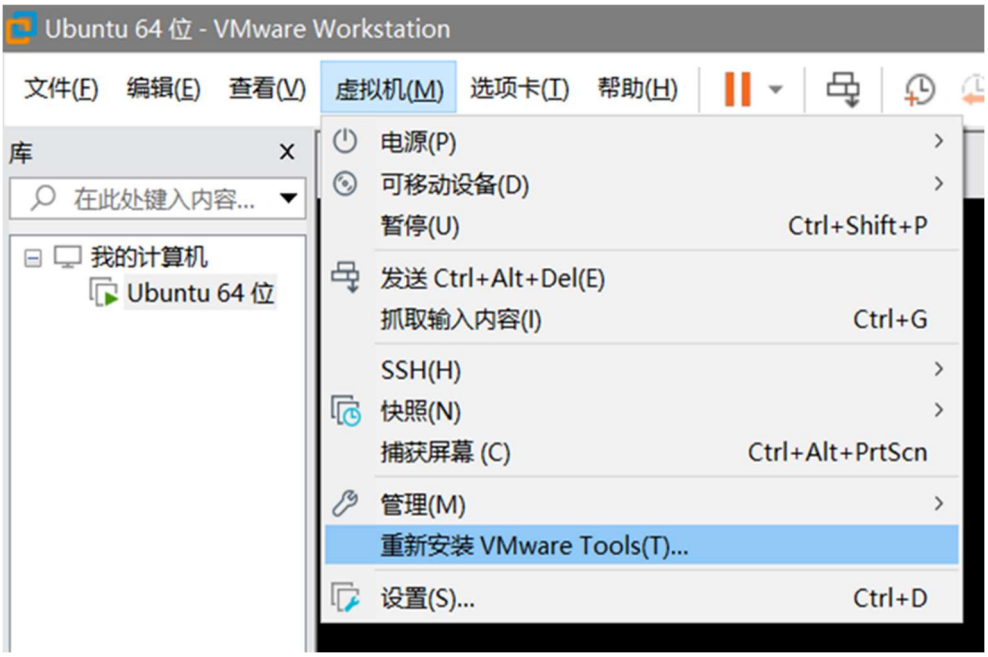
VMWare Workstation 下载界面



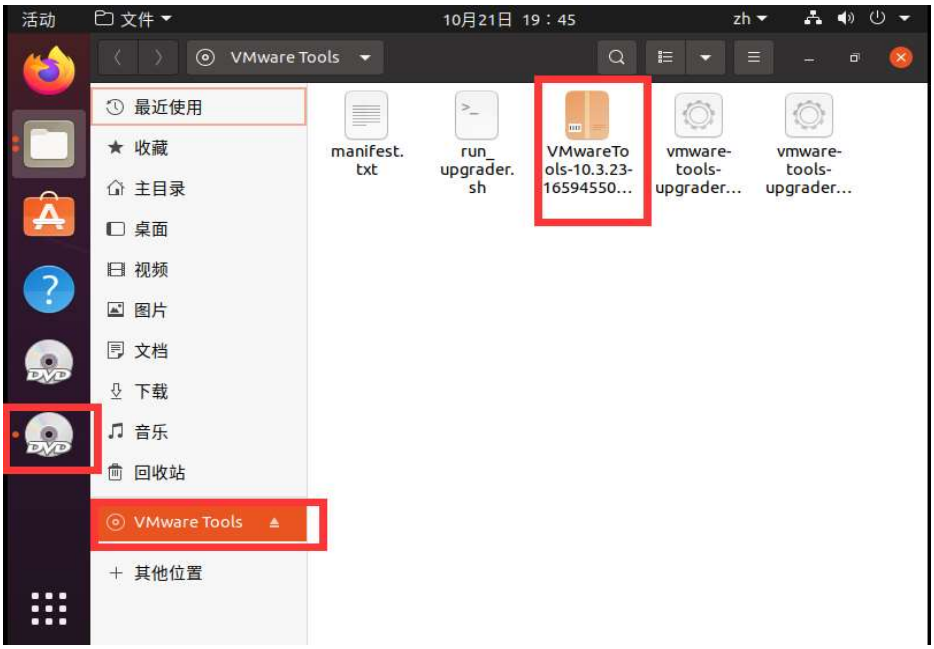
安装过程中的选项

4） 激活许可证：ZF3RO-FHED2-M80TY-8QYGC-NPKYF；YF390-0HF8P-M81RQ-2DXQE-M2UT6；ZF71R-DMX85-08DQY-8YMNC-PPHV8

7.2 安装VMWare Tools

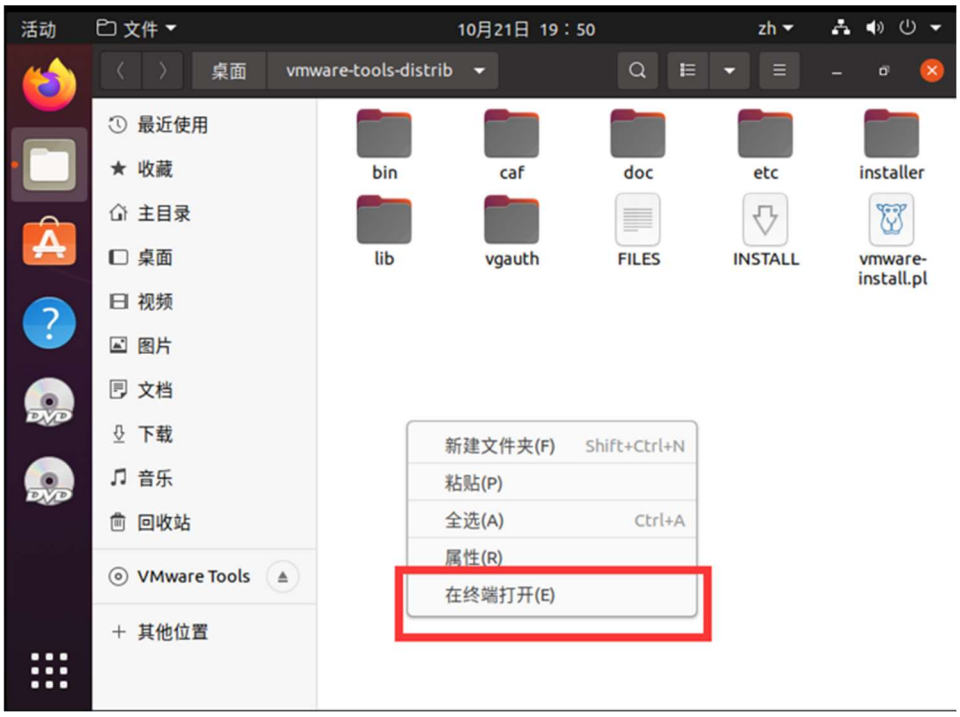


安装VMWare tools



复制压缩包至桌面

提取压缩包内容



在终端打开文件夹

```
wenjie@wenjie-virtual-machine:~/桌面/vmware-tools-distrib$ sudo su
[sudo] wenjie 的密码:
root@wenjie-virtual-machine:/home/wenjie/桌面/vmware-tools-distrib# ./vmware-install.pl
```

输入安装命令

```
1. Manually start /usr/bin/vmware-user
2. Log out and log back into your desktop session
3. Restart your X session.

Enjoy,

--the VMware team
```

安装成功后重启电脑或者虚拟机软件

7.3 在Vmware中安装ubuntu 20.04系统

1） 推荐使用华为镜像的ubuntu操作系统，即ubuntu-20.04.6-desktop-amd64.iso文件地址为：

repo.huaweicloud.com/ubuntu-releases/20.04/

2） 由于操作系统源码较大，请预留足够的空间，建议给ubuntu分配100G以上的磁盘空间。文档参考：zh-cn/device-dev/quick-start/quickstart-ide-env-ubuntu.md • OpenHarmony/docs - Gitee.com

7.4 安装鸿蒙QEMU硬件模拟器

1） 鸿蒙QEMU的下载地址为device_qemu: Hardware platforms emulation by QEMU | QEMU模拟不同的硬件单板 (gitee.com)。当然一些厂商也为自己的硬件设备开发了相应的QEMU，相应的地址为OpenHarmony/vendor_ohemu - 码云 - 开源中国 (gitee.com)，同学们可以按需采用。

7.5 下载编译鸿蒙系统源码，并生成操作系统镜像

1） 下载源码

下载方式1：完整版本的openHarmony系统较大（30G左右），请安装ubuntu操作系统时预留足够的空间。最新版的操作系统源码如下：
https://repo.huaweicloud.com/harmonyos/os/4.1-Release/code-v4.1-Release.tar.gz。
下载完成会，解压到相应的目录下，进行编译。

下载方式2（建议采用）：使用repo+ssh直接从gitee下载，具体步骤详见zh-cn/device-dev/get-code/sourcecode-acquire.md • OpenHarmony/docs - Gitee.com

注意：repo 在python2 运行不了，请改为python3 就解决。

如果采用方式2下载，在Ubuntu系统下，下载完整代码，采用repo进行初始化时，可以用
repo init -u git@github.com:openharmony/manifest.git -b master --no-repo-verify
如果采用方式2下载，可以选择下载部分代码，例如，采用的chipset时qemu，并且只针对轻量系统（mini系统），则可以用
repo init -u https://gitee.com/openharmony/manifest -b master -m chipsets/qemu.xml -g ohos:mini
具体指令请参考：https://blog.csdn.net/qq582880551/article/details/136330246

2） 编译源码

根据文档，搭建docker编译环境zh-cn/device-dev/get-code/gettools-acquire.md • OpenHarmony/docs - Gitee.com根据系统类型，执行行营的编译指令。
•轻型系统：python3 build.py -p qemu_mini_system_demo@ohemu
•小型系统：python3 build.py -p qemu_small_system_demo@ohemu

注意：编译工具和开发板都分为轻型、小型、标准，必须属于同一种类型，否则编译时各种报错。

编译完成后会生成操作系统镜像OHOS_Image，供QEMU加载使用。

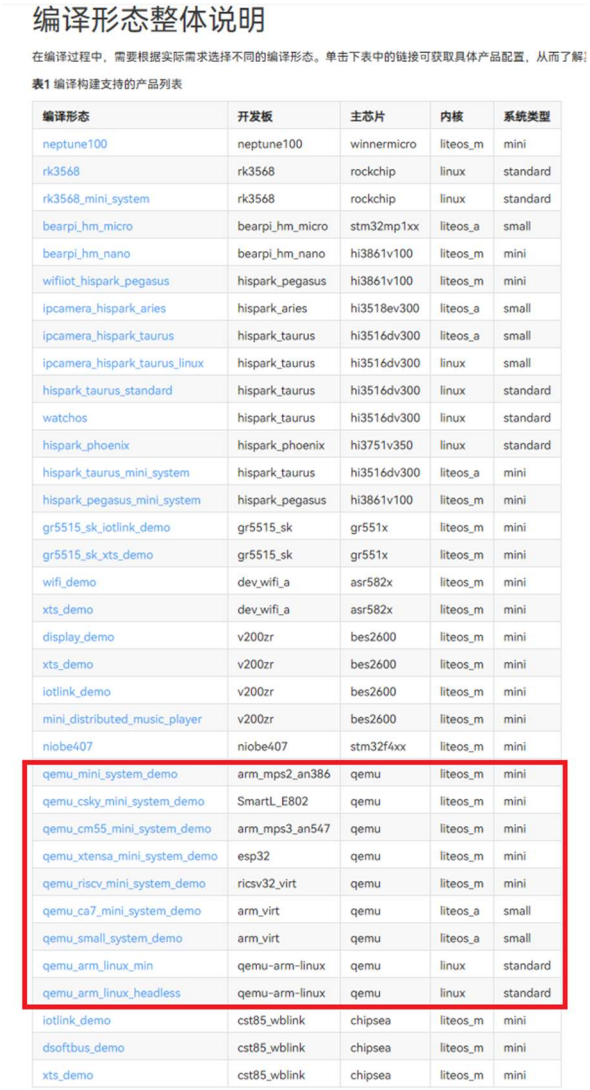
7.6 使用QEMU加载并运行生成的操作系统镜像

1) 例如，如果选择x86_64_virt平台，使用QEMU启动操作系统的过程请参考：https://gitee.com/openharmony/device_qemu/tree/master/x86_64_virt/linux

8 实验细则

1) 因为运行openHarmony操作系统需要相应的硬件设备，因此，本实验采用QEMU硬件模拟器模拟硬件设备，并将自己手动编译好的openHarmony操作系统image“安装到”QEMU所模拟出的硬件设备上运行。鸿蒙QEMU的下载地址为：device_qemu: Hardware platforms emulation by QEMU | QEMU模拟不同的硬件单板 (gitee.com)。

2) 目前QEMU模拟器支持arm_mps2_an386、arm_virt、esp32、riscv32_virt、x86_64_virt、SmartL_E802等开发板的模拟，因此同学们可以选择上述任意一款开发版对应的QEMU作为操作系统的硬件使用，详见下图。



编译形态整体说明

在编译过程中，需要根据实际需求选择不同的编译形态。单击下表中的链接可获取具体产品配置，从而了解

表1 编译构建支持的产品列表

编译形态	开发板	主芯片	内核	系统类型
neptune100	neptune100	winnermicro	liteos_m	mini
rk3568	rk3568	rockchip	linux	standard
rk3568_mini_system	rk3568	rockchip	linux	standard
bearpi_hm_micro	bearpi_hm_micro	stm32mp1xx	liteos_a	small
bearpi_hm_nano	bearpi_hm_nano	hi3861v100	liteos_m	mini
wifiiot_hispark_pegasus	hispark_pegasus	hi3861v100	liteos_m	mini
ipcamera_hispark_aries	hispark_aries	hi3518ev300	liteos_a	small
ipcamera_hispark_taurus	hispark_taurus	hi3516dv300	liteos_a	small
ipcamera_hispark_taurus_linux	hispark_taurus	hi3516dv300	linux	small
hispark_taurus_standard	hispark_taurus	hi3516dv300	linux	standard
watchos	hispark_taurus	hi3516dv300	linux	standard
hispark_phoenix	hispark_phoenix	hi3751v350	linux	standard
hispark_taurus_mini_system	hispark_taurus	hi3516dv300	liteos_a	mini
hispark_pegasus_mini_system	hispark_pegasus	hi3861v100	liteos_m	mini
gr5515_sk_iotlink_demo	gr5515_sk	gr551x	liteos_m	mini
gr5515_sk_xts_demo	gr5515_sk	gr551x	liteos_m	mini
wifi_demo	dev_wifi_a	asr582x	liteos_m	mini
xts_demo	dev_wifi_a	asr582x	liteos_m	mini
display_demo	v200zr	bes2600	liteos_m	mini
xts_demo	v200zr	bes2600	liteos_m	mini
iotlink_demo	v200zr	bes2600	liteos_m	mini
mini_distributed_music_player	v200zr	bes2600	liteos_m	mini
niobe407	niobe407	stm32f4xx	liteos_m	mini
qemu_mini_system_demo	arm_mps2_an386	qemu	liteos_m	mini
qemu_csky_mini_system_demo	SmartL_E802	qemu	liteos_m	mini
qemu_cm55_mini_system_demo	arm_mps3_an547	qemu	liteos_m	mini
qemu_xtensa_mini_system_demo	esp32	qemu	liteos_m	mini
qemu_riscv_mini_system_demo	ricsv32_virt	qemu	liteos_m	mini
qemu_ca7_mini_system_demo	arm_virt	qemu	liteos_a	small
qemu_small_system_demo	arm_virt	qemu	liteos_a	small
qemu_arm_linux_min	qemu-arm-linux	qemu	linux	standard
qemu_arm_linux_headless	qemu-arm-linux	qemu	linux	standard
iotlink_demo	cst85_wblink	chipsea	liteos_m	mini
dsoftbus_demo	cst85_wblink	chipsea	liteos_m	mini
xts_demo	cst85_wblink	chipsea	liteos_m	mini

3) 请同学们不要在自己的物理主机上做实验（为了保护同学们的电脑），同学们可以下载VMware虚拟机，然后安装ubuntu 20.04操作系统，所有的实验都在ubuntu 20.04中完成。

8.1 实验说明

1) 采用虚拟机 Vmware 完成 Ubuntu 操作系统的安装，安装操作系统后，（1）请创建一个可以彰显自己小组信息的用户名（可以选择一位同学的姓名和学号组合，例如图 1 中的 bupt-os-zhangsan20108021234），否则只能获得该实验步骤的一半分数；

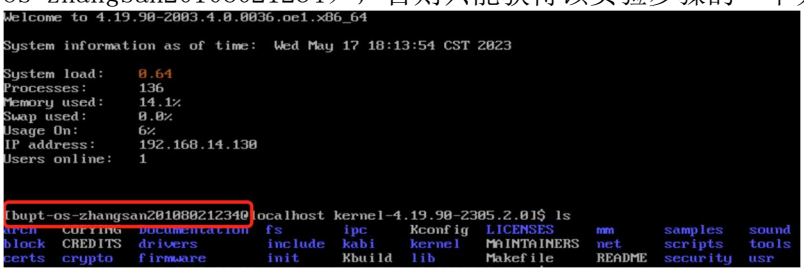


图1. 用户名示例

2) 本实验全部采用QEMU，因此无需指定特定的硬件系统，但需要在实验报告中指明（1）所采用QEMU版本；（2）所要编译的系统是轻型、小型或标准系统。

3) 鸿蒙操作系统有多种编译方式，可以不采用docker的方式进行编译。无论采用哪种编译方式，都要进写一下详细的实验步骤。

8.2 输出成果

1) 小组实验报告：实验报告中必须包含上述实验的细致步骤（实现步骤越详细越好）及其相应的指令，关键步骤的截图也请写到实验报告中；

2) 小组实验介绍PPT：实验中遇到各种问题，以及最终小组成员是如何解决的（最好将对应的资料和网页链接写出来）。

8.3 注意事项

必须在文档和PPT中写出每个成员的分工、实验过程（时间和对应的结果）、问题及解决方案。

8.4 参考资料

[1] OpenAtom OpenHarmony教育资源仓：OpenAtom OpenHarmony项目教育资源仓是可自由访问的，开放许可的培训课程、技术手册、解决方案、成功案例等的集合，可用于OpenHarmony教学，学习和评估以及研究目的。 (gitee.com)

[2] 基于Ubuntu20.04搭建OpenHarmony v3.0.6的qemu仿真环境_openharmony qemu-CSDN博客

[3] Openharmony鸿蒙内核编译及qemu运行过程问题记录 openharmony 编译内核-CSDN博客

[4] 3 开源鸿蒙OpenHarmony4.1源码下载、编译，生成OHOS_Image可执行文件的最简易流程_openharmony-4.1-release下载-CSDN博客

[5] 运行在Qemu上的鸿蒙内核Liteos-m - 哔哩哔哩 (bilibili.com)

[6] 在qemu上体验芯来RISC-V处理器运行鸿蒙LiteOS-M内核__专栏_RISC-V MCU中文社区 (riscv-mcu.com)

[7] arm_virt/liteos_a/README_zh.md • OpenHarmony/device_qemu - Gitee.com

[8] 华为开源镜像站_软件开发服务_华为云 (huaweicloud.com)

[9] zh-cn/device-dev/kernel/kernel-overview.md • OpenHarmony/docs - Gitee.com

9 附录

9.1 鸿蒙操作系统基本介绍

OpenHarmony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 > 组件”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组件。OpenHarmony技术架构如下所示：

docs.openharmony.cn/pages/v4.0/zh-cn/OpenHarmony-Overview_zh.md/

1) 内核层

- 内核子系统：采用多内核（Linux 内核或者 LiteOS）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：驱动框架（HDF）是系统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。

2) 一次开发，多端部署

OpenHarmony提供用户程序框架、Ability框架以及UI框架，能够保证开发的应用在多终端运行时保证一致性。一次开发、多端部署。

多终端软件平台API具备一致性，确保用户程序的运行兼容性。

- 支持在开发过程中预览终端的能力适配情况（CPU/内存/外设/软件资源等）。

- 支持根据用户程序与软件平台的兼容性来调度用户呈现。

3) 统一OS，弹性部署

OpenHarmony通过组件化和组件弹性化等设计方法，做到硬件资源的可大可小，在多种终端设备间，按需弹性部署，全面覆盖了ARM、RISC-V、x86等各种CPU，从百KiB到GiB级别的RAM。

9.2 内核概述

1) 内核简介

用户最常见到并与之交互的操作系统界面，其实只是操作系统最外面的一层。操作系统最重要的任务，包括管理硬件设备，分配系统资源等，我们称之为操作系统内在最重要的核心功能。而实现这些核心功能的操作系统模块，业界一般称之为操作系统“内核”。

2） 实现原理

操作系统是位于应用和硬件之间的系统软件，向上提供易用的程序接口和运行环境，向下管理硬件资源。内核位于操作系统的下层，为操作系统上层的程序框架提供硬件资源的并发管理。

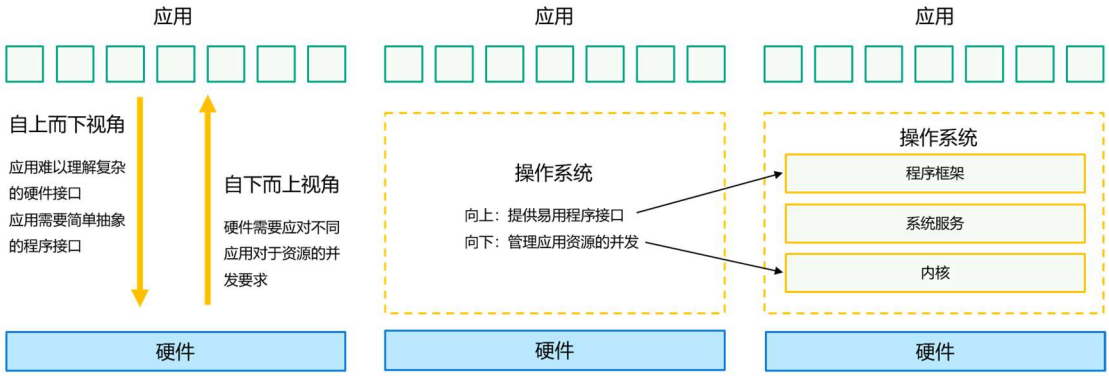


图 1　操作系统架构

3） 多内核架构和基本组成

业界的内核有很多，但无论是什么内核，基本上有几个最重要的组成单元是每个内核均要具备的，分别是：

- 负责持久化数据，并让应用程序能够方便的访问持久化数据的“文件系统”。
- 负责管理进程地址空间的“内存管理”。
- 负责管理多个进程的“进程管理”或者“任务管理“。
- 负责本机操作系统和另外一个设备上操作系统通信的“网络”。

OpenHarmony 采用了多内核结构，支持 Linux 和 LiteOS，开发者可按不同产品规格进行选择使用。Linux 和 LiteOS 均具备上述组成单元，只是实现方式有所不同。多个内核通过 KAL（Kernel Abstraction Layer）模块，向上提供统一的标准接口。

内核子系统位于 OpenHarmony 下层。需要特别注意的是，由于 OpenHarmony 面向多种设备类型，这些设备有着不同的 CPU 能力，存储大小等。为了更好的适配这些不同的设备类型，内核子系统支持针对不同资源等级的设备选用适合的 OS 内核，内核抽象层（KAL，Kernel Abstract Layer）通过屏蔽内核间差异，对上层提供基础的内核能力。

图 2　OpenHarmony 架构图

4)　不同内核适配的系统及设备类型

OpenHarmony 按照支持的设备可分为如下几种系统类型：

- 轻量系统（mini system） 面向 MCU 类处理器例如 Arm Cortex-M、RISC-V 32 位的设备，硬件资源极其有限，支持的设备最小内存为 128KiB，可以提供多种轻量级网络协议，轻量级的图形框架，以及丰富的 IOT 总线读写部件等。可支撑的产品如智能家居领域的连接类模组、传感器设备、穿戴类设备等。

- 小型系统（small system） 面向应用处理器例如 Arm Cortex-A 的设备，支持的设备最小内存为 1MiB，可以提供更高的安全能力、标准的图形框架、视频编解码的多媒体能力。可支撑的产品如智能家居领域的 IP Camera、电子猫眼、路由器以及智慧出行域的行车记录仪等。

- 标准系统（standard system） 面向应用处理器例如 Arm Cortex-A 的设备，支持的设备最小内存为 128MiB，可以提供增强的交互能力、3D GPU 以及硬件合成能力、更多控件以及动效更丰富的图形能力、完整的应用框架。可支撑的产品如高端的冰箱显示屏。

- OpenHarmony 针对不同量级的系统，使用了不同形态的内核。轻量系统、小型系统可以选用 LiteOS；小型系统和标准系统可以选用 Linux。其对应关系如下表：

表 1　系统关系对应表

系统级别	轻量系统	小型系统	标准系统
LiteOS-M	√	×	×
LiteOS-A	×	√	√

Linux	×	√	√

9.3 LiteOS-M

OpenHarmony LiteOS-M 内核是面向 IoT 领域构建的轻量级物联网操作系统内核，具有小体积、低功耗、高性能的特点，其代码结构简单，主要包括内核最小功能集、内核抽象层、可选组件以及工程目录等，分为硬件相关层以及硬件无关层，硬件相关层提供统一的 HAL（Hardware Abstraction Layer）接口，提升硬件易适配性，不同编译工具链和芯片架构的组合分类，满足 AIoT 类型丰富的硬件和编译工具链的拓展。

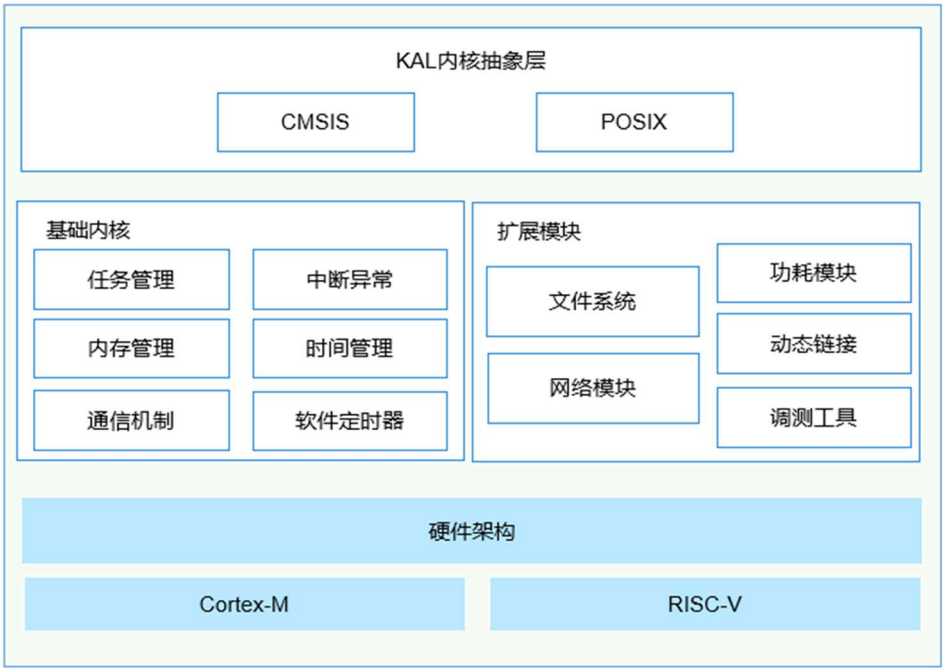


图 3 LiteOS-M 架构图

9.4 LiteOS-A

OpenHarmony 轻量级内核是基于 IoT 领域轻量级物联网操作系统 Huawei LiteOS 内核演进发展的新一代内核，包含 LiteOS-M 和 LiteOS-A 两类内核。LiteOS-M 内核主要应用于轻量系统，面向的 MCU（Microprocessor Unit）一般是百 K 级内存，可支持 MPU（Memory Protection Unit）隔离，业界类似的内核有 FreeRTOS 或 ThreadX 等；LiteOS-A 内核主要应用于小型系统，面向设备一般是 M 级内存，可支持 MMU（Memory Management Unit）隔离，业界类似的内核有 Zircon 或 Darwin 等。

为适应 IoT 产业的高速发展，OpenHarmony 轻量级内核不断优化和扩展，能够带给开发者友好的开发体验和统一开放的生态系统能力。轻量级内核 LiteOS-A 重要的新特性如下：

- 新增了丰富的内核机制：
    - 新增虚拟内存、系统调用、多核、轻量级 IPC（Inter-Process Communication，进程间通信）、DAC（Discretionary Access Control，自主访问控制）等机制，丰富了内核能力；

- 为了更好的兼容软件和开发者体验，新增支持多进程，使得应用之间内存隔离、相互不影响，提升系统的健壮性。

- 引入统一驱动框架 HDF（Hardware Driver Foundation）

  引入统一驱动框架 HDF，统一驱动标准，为设备厂商提供了更统一的接入方式，使驱动更加容易移植，力求做到一次开发，多系统部署。

- 支持 1200+标准 POSIX 接口

  更加全面的支持 POSIX 标准接口，使得应用软件易于开发和移植，给应用开发者提供了更友好的开发体验。

- 内核和硬件高解耦

  轻量级内核与硬件高度解耦，新增单板，内核代码不用修改。

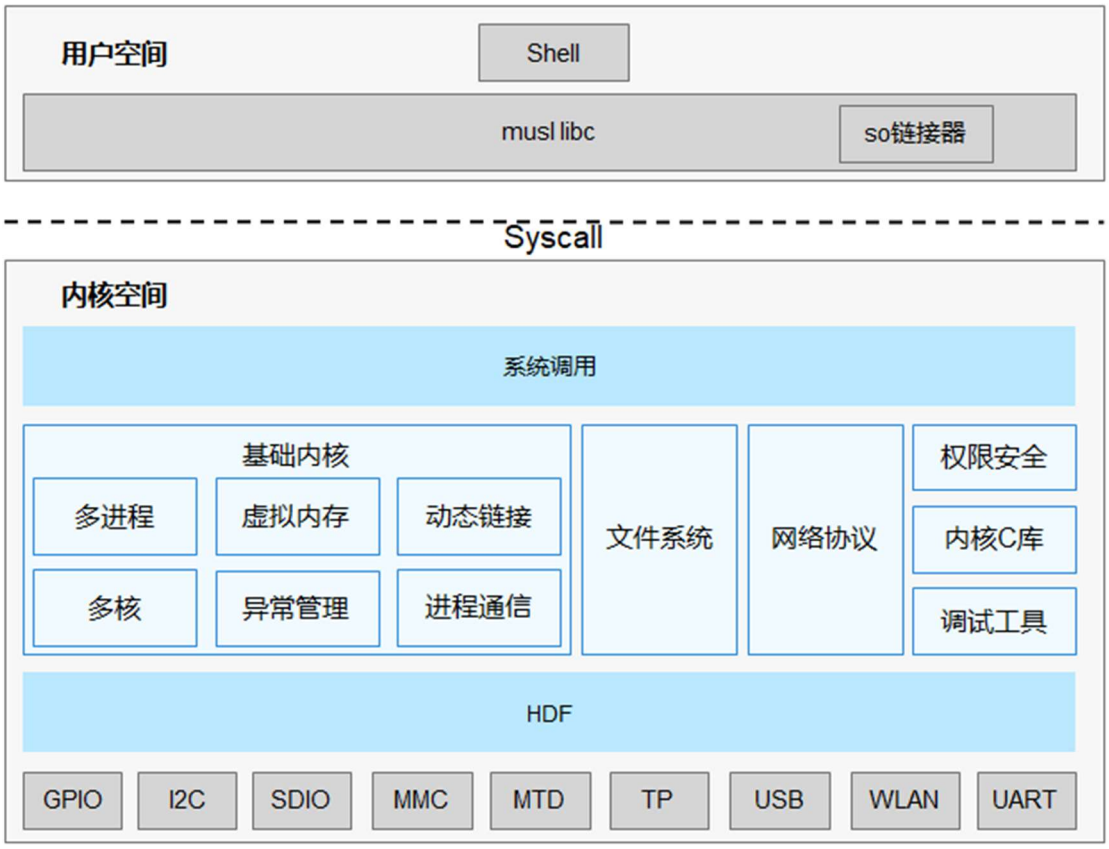


图 4　OpenHarmony LiteOS-A 内核架构图

9.5　Linux

1)　linux内核概述

OpenHarmony 的 Linux 内核基于开源 Linux 内核 LTS　**4.19.y / 5.10.y**　分支演进，在此基线基础上，回合 CVE 补丁及 OpenHarmony 特性，作为 OpenHarmony Common Kernel 基线。针对不同的芯片，各厂商合入对应的板级驱动补丁，完成对 OpenHarmony 的基线适配。

Linux 社区 LTS 4.19.y 分支信息请查看 kernel 官网。

Linux 社区 LTS 5.10.y 分支信息请查看 kernel 官网。

内核的 Patch 组成模块，在编译构建流程中，针对具体芯片平台，合入对应的架构驱动代码，进行编译对应的内核镜像。所有补丁来源均遵守 GPL-2.0 协议。

2) 内核增强特性

OpenHarmony 针对 linux 内核在 ESwap(Enhanced Swap)、关联线程组调度和 CPU 轻量级隔离做了增强。

3) Enhanced SWAP特性

ESwap 提供了自定义新增存储分区作为内存交换分区的能力，并创建了一个常驻进程 zswapd 将 ZRAM 压缩后的匿名页加密换出到 ESwap 存储分区，从而能完全的空出一块可用内存，以此来达到维持 Memavailable 水线的目标。同时，配合这个回收机制，在整个内存框架上进行改进，优化匿名页和文件页的回收效率，并且使两者的回收比例更加合理以避免过度回收导致的 refault 问题造成卡顿现象。

4) 关联线程组调度

关联线程组(related thread group)提供了对一组关键线程调度优化的能力，支持对关键线程组单独进行负载统计和预测，并且设置优选 CPU cluster 功能，从而达到为组内线程选择最优 CPU 运行并且根据分组负载选择合适的 CPU 调频点运行。

5) CPU轻量级隔离

CPU 轻量级隔离特性提供了根据系统负载和用户配置来选择合适的 CPU 进行动态隔离的能力。内核会将被隔离 CPU 上的任务和中断迁移到其他合适的 CPU 上执行，被隔离的 CPU 会进入 ilde 状态，以此来达到功耗优化的目标。同时提供用户态的配置和查询接口来实现更好的系统调优。

6) 使用指导

1. 合入 HDF 补丁 在 kernel/linux/build 仓中，按照 kernel.mk 中 HDF 的补丁合入方法，合入不同内核版本对应的 HDF 内核补丁：

   ```
   $(OHOS_BUILD_HOME)/drivers/hdf_core/adapter/khdf/linux/patch_hdf.sh $(OHOS_BUILD_HOME)
   $(KERNEL_SRC_TMP_PATH) $(KERNEL_PATCH_PATH) $(DEVICE_NAME)
   ```

2. 合入芯片平台驱动补丁 以 Hi3516DV300 为例：

   在 kernel/linux/build 仓中，按照 kernel.mk 中的芯片组件所对应的 patch 路径规则及命名规则，将对应的芯片组件 patch 放到对应路径下：

   ```
   DEVICE_PATCH_DIR :=
   $(OHOS_BUILD_HOME)/kernel/linux/patches/${KERNEL_VERSION}/$(DEVICE_NAME)_patch
   DEVICE_PATCH_FILE := $(DEVICE_PATCH_DIR)/$(DEVICE_NAME).patch
   ```

3. 修改自己所需要编译的 config 在 kernel/linux/build 仓中，按照 kernel.mk 中的芯片组件所对应的 patch 路径规则及命名规则，将对应的芯片组件 config 放到对应路径下：

   ```
   KERNEL_CONFIG_PATH :=
   $(OHOS_BUILD_HOME)/kernel/linux/config/${KERNEL_VERSION}DEFCONFIG_FILE :=
   $(DEVICE_NAME)_$(BUILD_TYPE)_defconfig
   ```

须知： 由于 OpenHarmony 工程的编译构建流程中会拷贝 kernel/linux/linux-*.*的代码环境后进行打补丁动作，在使用 OpenHarmony 的版本级编译命令前，需要 kernel/linux/linux-*.*原代码环境。

根据不同系统工程，编译完成后会在out目录下的kernel目录中生成对应实际编译的内核，基于此目录的内核，进行对应的config修改，将最后生成的.config文件cp到config仓对应的路径文件里，即可生效。