# 操作系统实验二

本实验包含**三**个实验**，**即《自制OS实验》**、**《openEuler实验》**、《**OpenHarmony操作系统实验**》**，**大家**任选其一完成即可。

《自制OS实验》主要面向底层，从零开始写一个通用型的操作系统，可以帮助同学们从底层深刻理解操作系统的底层工作原理。此实验的资料比较丰富，同学们可以参考提供的链接。

《openEuler实验》主要是面向国产操作系统openEuler的实验，考察同学们对操作系统框架的整体理解以及实操能力。该实验要求同学们能够从零安装openEuler操作系统，采用重新编译源码的方式将内核更新至最新版，并且完成一些基础的实验。

《OpenHarmony实验》主要是面向国产操作系统OpenHarmony的实验，因为OpenHarmony较为复杂，所以本实验手册附录部分该系统做了简要的介绍，同学们也可以自行到官方网站上搜索相应的材料。本实验考察同学们对操作系统框架的整体理解以及实操能力，要求同学们能够对OpenHarmony操作系统有基本的理解，并且可以编译OpenHarmony系统源码，最后在QEMU硬件模拟器中模拟OpenHarmony操作系统的运行。

## 考核标准

1. 该实验满分12分.
2. 《自制OS实验》中，能用U盘或在虚拟机上完成开机启动，并实现任意一个功能，并提交完整输出成果，即可得到合格成绩，即12\*60%；在此基础上，每多完成一个功能，可获12\*20%成绩。
3. **《**openEuler实验**》**中，完成openEuler操作系统的安装得合格成绩，即12\*60%；完成内核更新（源代码更新方式）得12\*80%；在此基础上，每多完成一个功能，可获12\*10%成绩。
4. 《OpenHarmony实验》中，完成OpenHarmony操作系统的编译（得到OHOS image镜像文件）得合格成绩，即12\*60%；采用QEMU成功运行OHOS得12\*100%；
5. 实验可由1～3人组队完成，得分成绩按照文档和PPT中每个成员的分工百分比分配，所以同学们在提交的文档中，一定要说明自己的工作量。例如：

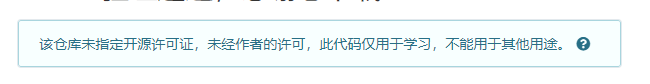
姓名：xxx 班级：xxx 学号：xxx 分工：60%

姓名：xxx 班级：xxx 学号：xxx 分工：40%

## 自制OS实验

1. **实验说明**：通过查找资料，小组完成一个自制OS，要求能够用U盘或在虚拟机上完成启动，并实现一个（含）以上自行拟定的功能，例如：启动后输出显示预先设定的字符串；读出硬盘数据；开启内存分页机制；内存管理操作；进程操作；线程操作；锁。
2. **输出成果：**设计文档、测试文档、源代码、小组实验报告；实验介绍PPT。
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（时间和对应的成果）、问题及解决方案。
4. **参考资料：**

[1][自制os 极简教程1：写一个操作系统有多难](https://www.cnblogs.com/flashsun/p/13969254.html)，<https://www.cnblogs.com/flashsun/p/13969254.html>

代码参考：<https://gitee.com/sunym1993/flash-4th-os.git>

[2]自己制作一个简单的操作系统，<https://www.cnblogs.com/Cherrison-Time/p/11590693.html>

## openEuler实验

1. **实验说明：**
   1. 采用虚拟机Vmware完成openEuler操作系统的安装，安装版本为20.03-TLS，iso镜像从<https://mirror.iscas.ac.cn/openeuler/openEuler-20.03-LTS/ISO/x86_64/openEuler-20.03-LTS-x86_64-dvd.iso> 镜像源下载；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；（2）请执行uname -a指令，并将截图放到实验报告中；（3）请执行getconf PAGESIZE指令，查看openEuler的分页大小。

A picture containing text, screenshot

Description automatically generated

图1.用户名示例

* 1. 采用重新编译源代码的方式将内核更新至最新版。具体而言，下载最新版本的openEuler内核源码，见https://gitee.com/openeuler/kernel/releases 页面中提交ID为fc6966431495e208fd9372cb5924ca4484455368的内核源码zip包或者tar.gz包，详见图2。

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图2. 内核源码

* 1. 重新编译后，安装内核，然后执行uname -a指令，并将截图放到实验报告
  2. 更新内核至最新后，参考<https://blog.csdn.net/qq_46744173/article/details/122199462>，完成基础操作系统实验，例如内核模块编程、内存管理、进程管理、中断实验、设备管理等。
  3. 参考

1. **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）以及必要的截图；（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）；。
2. **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（时间和对应的结果果）、问题及解决方案。
3. **参考资料：**

[1] <https://blog.csdn.net/hiascend/article/details/128579458>

[2] [https://blog.csdn.net/m0\_56602092/article/details/118604262?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_baidulandingword~default-0-118604262-blog-128579458.235^v36^pc\_relevant\_anti\_vip&spm=1001.2101.3001.4242.1&utm\_relevant\_index=3](https://blog.csdn.net/m0_56602092/article/details/118604262?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-118604262-blog-128579458.235%5ev36%5epc_relevant_anti_vip&spm=1001.2101.3001.4242.1&utm_relevant_index=3)

[3] <http://mirrors.aliyun.com/openeuler/openEuler-20.03LTS/?spm=a2c6h.25603864.0.0.e5ae71a8Fexn4l>

[4] <https://www.openeuler.org/zh/mirror/list/>

[5] <https://zhuanlan.zhihu.com/p/423624818>

[6] <https://blog.csdn.net/xuyuefei1988/article/details/8635539>

[7] <https://blog.csdn.net/alwaysbefine/article/details/123363999>

[8] <https://blog.csdn.net/weixin_56483756/article/details/124185212>

[9] <https://blog.csdn.net/xcjyxy2021/article/details/123341308>

[10] <https://blog.csdn.net/qq_45945548/article/details/120557041>

[11] <https://blog.csdn.net/weixin_45467056/article/details/112930163>

[12] <https://blog.csdn.net/qq_36393978/article/details/117816425>

## OpenHarmony操作系统实验

## 实验内容

使用QEMU硬件模拟器运行开源鸿蒙OpenHarmony4.x操作系统

## 实验步骤提示

### （1）安装Vmware WorkStation的安装

### （2）在Vmware中安装ubuntu 20.04系统

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/](https://repo.huaweicloud.com/ubuntu-releases/20.04/)

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](https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/quick-start/quickstart-ide-env-ubuntu.md)

### （3）安装鸿蒙QEMU硬件模拟器

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)](https://gitee.com/openharmony/device_qemu)。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)](https://gitee.com/openharmony/vendor_ohemu/tree/master)，同学们可以按需采用。

### （4）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](https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/get-code/sourcecode-acquire.md)

**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**](https://blog.csdn.net/qq582880551/article/details/136330246)

1. **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](https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/get-code/gettools-acquire.md)

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。

### （5）使用QEMU加载并运行生成的操作系统镜像

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>

**注意事项：**

（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)](https://gitee.com/openharmony/device_qemu)

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。

表格

中度可信度描述已自动生成

（2）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。

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；

A picture containing text, screenshot

Description automatically generated

图1. 用户名示例

* 1. 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
  2. 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。

1. **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）；。
2. **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（时间和对应的结果果）、问题及解决方案。
3. **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)](https://gitee.com/openatom-university/openharmony-oer)

[2] [基于Ubuntu20.04搭建OpenHarmony v3.0.6的qemu仿真环境\_openharmony qemu-CSDN博客](https://blog.csdn.net/a171232886/article/details/128716627)

[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](https://blog.csdn.net/m0_37637511/article/details/124023977)

[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](https://blog.csdn.net/qq582880551/article/details/136396307)

[5] [运行在Qemu上的鸿蒙内核Liteos-m - 哔哩哔哩 (bilibili.com)](https://www.bilibili.com/read/cv9720120/)

[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)](https://www.riscv-mcu.com/column-topic-id-462.html)

[7] [arm\_virt/liteos\_a/README\_zh.md · OpenHarmony/device\_qemu - Gitee.com](https://gitee.com/openharmony/device_qemu/blob/HEAD/arm_virt/liteos_a/README_zh.md)

[8] [华为开源镜像站\_软件开发服务\_华为云 (huaweicloud.com)](https://mirrors.huaweicloud.com/home)

[9] [zh-cn/device-dev/kernel/kernel-overview.md · OpenHarmony/docs - Gitee.com](https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/kernel/kernel-overview.md)

# 附录

### 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：

图形用户界面, 表格

描述已自动生成

[docs.openharmony.cn/pages/v4.0/zh-cn/OpenHarmony-Overview\_zh.md/](https://docs.openharmony.cn/pages/v4.0/zh-cn/OpenHarmony-Overview_zh.md/)

#### **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。

#### 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。

#### 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。

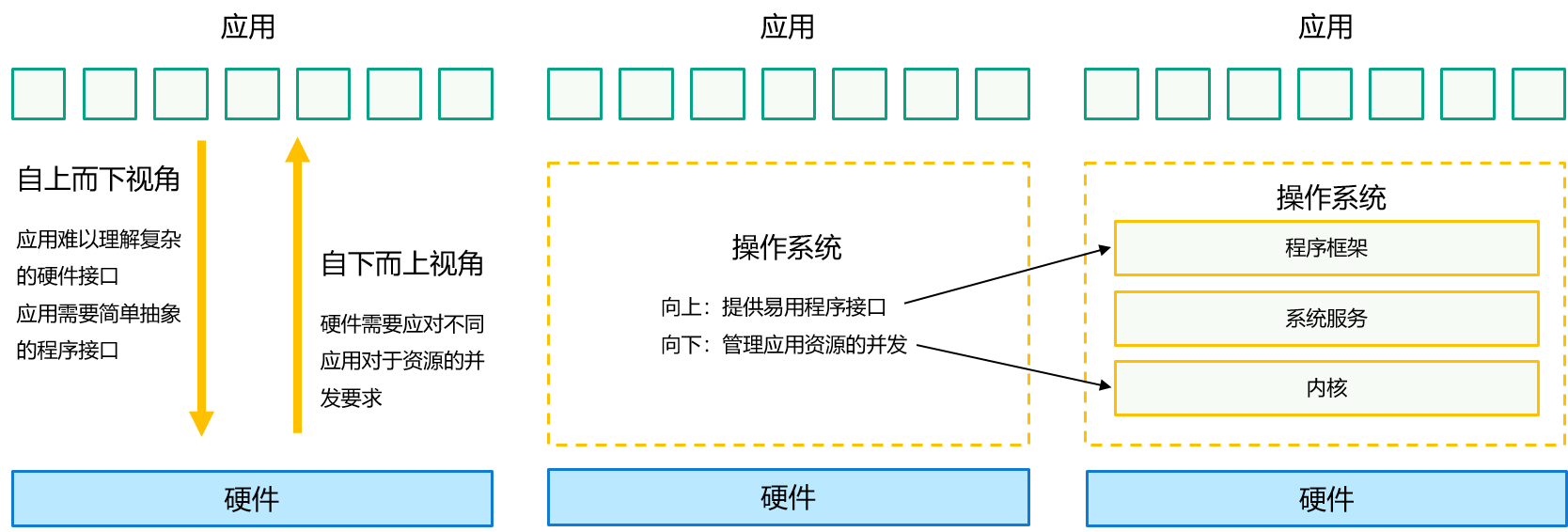
### 内核概述

#### 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”。

#### 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

#### 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

#### 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：

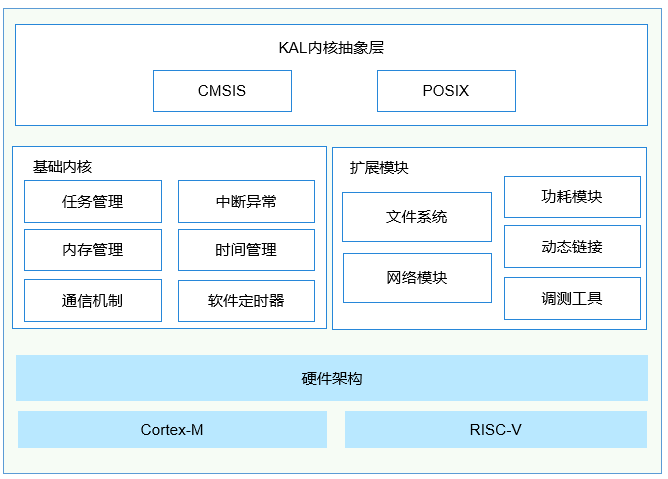
**表1** 系统关系对应表

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

## LiteOS-M

### 内核架构

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

#### 使用指导

LiteOS-M使用指导请参见LiteOS-M[内核概述](https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/kernel/kernel-mini-overview.md)的“使用说明”章节。

## LiteOS-A

#### 内核架构

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）

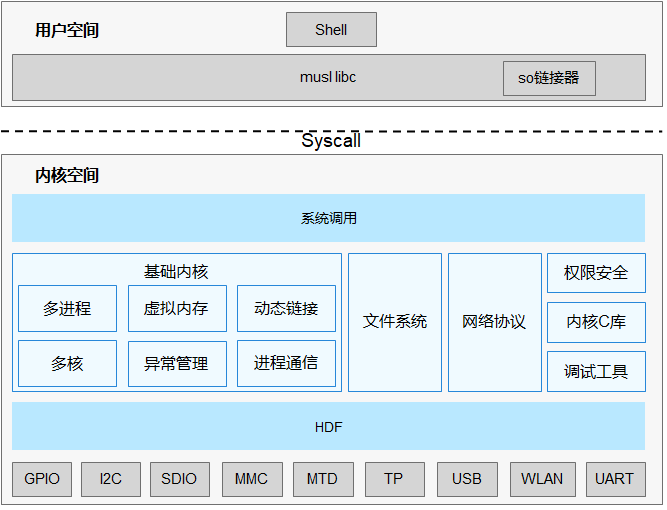
引入统一驱动框架HDF，统一驱动标准，为设备厂商提供了更统一的接入方式，使驱动更加容易移植，力求做到一次开发，多系统部署。

* 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

#### 使用指导

LiteOS-A使用指导请参见LiteOS-A[内核概述](https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/kernel/kernel-small-overview.md)的“使用说明”章节。

## Linux

#### 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](https://gitee.com/link?target=https%3A%2F%2Fgit.kernel.org%2Fpub%2Fscm%2Flinux%2Fkernel%2Fgit%2Fstable%2Flinux.git%2Flog%2F%3Fh%3Dlinux-4.19.y)。

Linux社区LTS 5.10.y分支信息请查看[kernel官网](https://gitee.com/link?target=https%3A%2F%2Fgit.kernel.org%2Fpub%2Fscm%2Flinux%2Fkernel%2Fgit%2Fstable%2Flinux.git%2Flog%2F%3Fh%3Dlinux-5.10.y)。

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。

#### 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。

#### 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](https://gitee.com/link?target=https%3A%2F%2Fwww.kernel.org%2Fdoc%2Fhtml%2Flatest%2Fadmin-guide%2Fblockdev%2Fzram.html)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。

##### **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。

##### **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。

#### 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)

1. 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

1. 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。