

实验案例一：鸿蒙设备开发环境的搭建

实验案例一：鸿蒙设备开发环境的搭建

- 一、实验简介
- 二、实验内容及要求
 - 1. 任务一：配置环境，进入ohos
- 三、实验原理
- 四、实验步骤
 - 1. Ubuntu环境搭建
 - (1)、安装库和工具
 - (2)、获取源码
 - (3)、执行prebuilts
 - (4)、安装编译工具
 - 2.编译
 - 3.运行
 - 4.常见问题
- 五、参考资料

一、实验简介

OpenHarmony 是鸿蒙设备开发所使用的系统，其是开放原子开源基金会孵化及运营的开源项目，由 HarmonyOS 操作系统拆分而来。OpenHarmony 与 HarmonyOS 之间类似于 AOSP(Android Open Source Project) 与 Android 系统的关系。OpenHarmony 是 HarmonyOS 的开源部分，HarmonyOS 是 OpenHarmony 的商业版本。在当前的鸿蒙设备开发工作之中，也主要基于 OpenHarmony 系统进行。

本次实验通过在 OpenHarmony 平台上进行，学习鸿蒙操作系统的特性与开发方法。后续实验均基于 OpenHarmony 源码开展。本次实验的目标是完成鸿蒙设备开发环境的搭建，为后续实验工作奠定基础。

二、实验内容及要求

本次实验旨在完成 OpenHarmony 设备开发环境的搭建与配置。实验所需系统及工具如下：

- **VMware Workstation Player**：由于实验需在 Linux 操作系统环境下进行，若主机运行 Windows 系统，可通过虚拟机安装 Linux。VMware Workstation Player 是 VMware 提供的免费虚拟化软件，亦可满足本次实验需求。
- **Linux发行版**：Linux 发行版由 Linux 内核、GNU 工具、附加软件及软件包管理器组成。常见发行版包括 Ubuntu、Debian、Fedora、CentOS 等。本实验提供的命令与安装步骤以 **Ubuntu 20.04** 为基础，其他版本在具体操作上可能存在差异。
- **OpenHarmony v4.1**：当前经过测试能够顺利编译与运行 qemu 模拟器的 OpenHarmony 版本。

由于 OpenHarmony 的官方文档在不断更新，部分安装步骤与文档说明可能存在差异。配置过程中可根据实际情况进行适当调整。如若在安装过程中遇到问题，请先查看第4节是否有遇到问题的解决方案。

1. 任务一：配置环境，进入ohos

本次实验仅要求完成设备环境的搭建和配置，需要完成 OpenHarmony 的本地环境搭建，并基于 qemu 进入 ohos 系统。

三、实验原理

如图1所示，OpenHarmony作为一款面向全场景的开源分布式操作系统，采用组件化设计使得设备开发者可基于目标硬件能力自由选择系统组件进行集成。系统功能按照“系统 > 子系统 > 组件”逐级展开，可以根据实际需求裁剪非必要的组件。

OpenHarmony主要包含三大技术特性，分别为(1).**硬件互助，资源共享**。(2).**一次开发，多端部署**。(3).**统一OS，弹性部署**。后续实验内容将围绕这三项特性展开。

近年来，OpenHarmony 的开发进展迅速，已完成多个版本迭代。本次实验主要基于OpenHarmony 4.1版本进行，其源码体量约为 **50 GB**，对本地存储空间有较高要求。需要注意的是，随着版本更新，官方文档及代码目录结构可能发生较大调整。尽管当前实验版本与最新版本在实现细节上可能存在差异，但核心理念与系统特性保持一致。

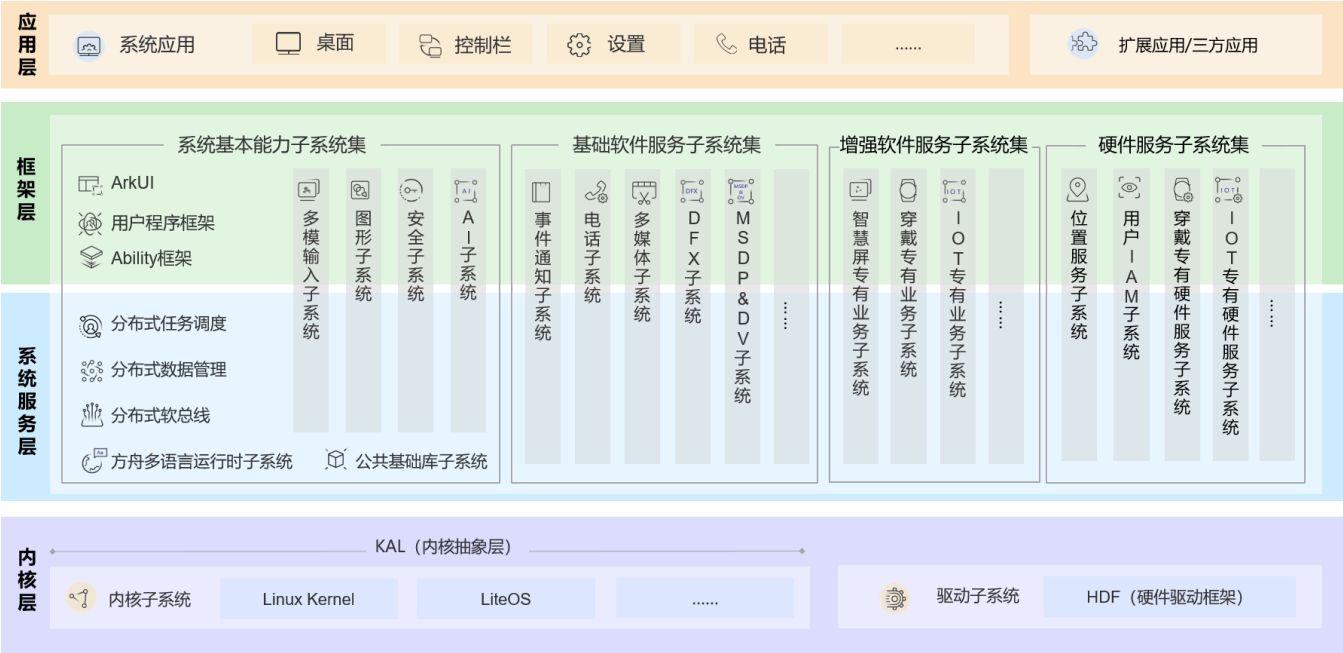


图1.OpenHarmony技术架构[1]

四、实验步骤

1. Ubuntu环境搭建

(1)、安装库和工具

使用命令行进行设备开发时，可以通过以下步骤安装编译OpenHarmony需要的库和工具。相应操作在**Ubuntu20.04**环境中进行。

1. 运行如下命令安装后续操作所需的库和工具

```
sudo apt update
sudo apt install g++ gcc
sudo apt install g++-multilib gcc-multilib
sudo apt install qemu-system-arm qemu
sudo apt install binutils binutils-dev git git-lfs gnupg flex bison gperf build-essential
zip curl zlib1g-dev gcc-multilib g++-multilib lib32ncurses5-dev x11proto-core-dev libx11-
dev lib32z1-dev ccache libgl1-mesa-dev libxml2-utils xsltproc unzip m4 bc gnutls-bin
python3.8 python3-pip ruby genext2fs device-tree-compiler make libffi-dev e2fsprogs pkg-
config perl openssl libssl-dev libelf-dev libdwarf-dev u-boot-tools mtd-utils cpio doxygen
liblz4-tool openjdk-8-jre gcc g++ texinfo dosfstools mtools default-jre default-jdk
libncurses5 apt-utils wget scons python3.8-distutils tar rsync git-core libxml2-dev lib32z-
dev grsync xxd libglib2.0-dev libpixmap-1-dev kmod jfsutils reiserfsprogs xfsprogs squashfs-
tools pcmciautils quota ppp libtinfo-dev libtinfo5 libncurses5-dev libncursesw5 libstdc++6
gcc-arm-none-eabi vim ssh locales libxinerama-dev libxcursor-dev libxrandr-dev libxi-dev
libc6-dev-i386
```

1. 将Python 3.8设置为默认Python版本。

查看Python 3.8的位置：

```
which python3.8
```

将Python和Python3切换为Python 3.8：

```
sudo update-alternatives --install /usr/bin/python python {Python 3.8 路径} 1    #{Python
3.8 路径}为上一步查看的Python 3.8的位置
sudo update-alternatives --install /usr/bin/python3 python3 {Python 3.8 路径} 1    #{Python
3.8 路径}为上一步查看的Python 3.8的位置
```

(2)、获取源码

本次实验使用 **OpenHarmony-4.1-Release** 版本，需自行下载并获取对应的源代码。

镜像站点获取路径：<https://repo.huaweicloud.com/openharmony/os/4.1-Release/code-v4.1-Release.tar.gz>

(3)、执行prebuilts

解压源码之后在源码根目录下执行prebuilts脚本，安装编译器及二进制工具

```
bash build/prebuilts_download.sh
```

(4)、安装编译工具

hb为OpenHarmony所使用的编译构建命令行工具。

安装hb：

1. 在源码根目录运行如下命令，以安装hb工具。

```
python3 -m pip install --user build/hb
```

1. 设置环境变量

```
vim ~/.bashrc
```

将以下命令拷贝到.bashrc文件的最后一行，保存并退出。

```
export PATH=~/.local/bin:$PATH
```

执行如下命令更新环境变量。

```
source ~/.bashrc
```

1. 在源码目录执行 `hb help`，界面打印以下信息即表示安装成功：

```
[OHOS INFO] -----
-----
[OHOS INFO] usage: hb build [option]
[OHOS INFO]
[OHOS INFO] optional arguments:
[OHOS INFO]   -h, --help            show this help message and exit
[OHOS INFO]   --target-cpu {arm,arm64,x86_64,x64,mipsel,riscv64}
[OHOS INFO]                        Default: ''. Help:Specifies the desired cpu architecture
for the build, each may support different cpu architectures, run 'hb set --all' to list
product all
[OHOS INFO]                        supported cpu architectures
[OHOS INFO]   --target-os {android,ios}
...

```

注意：

可采用以下命令卸载hb：

```
pip3 uninstall ohos-build
```

若安装hb的过程中遇到问题，请参见常见问题进行解决。

1. 切换Linux的shell环境至bash

```
sudo dpkg-reconfigure dash
```

选择 `no` 选项

```
ls -l /bin/sh
```

运行命令后应该得到如下结果

```
lrwxrwxrwx 1 root root 4 Feb 20 21:42 /bin/sh -> bash
```

2.编译

在下载OpenHarmony源代码的根目录依次执行如下操作构建编译源码：

1. 进入源码根目录后，执行命令：

```
hb set
```

2. 根据提示选择 OpenHarmony 的系统等级，选择 **small** 系统（即小型系统内核 **LiteOS-A**）：

```
OHOS which os_level do you need? (Use arrow keys)
mini
> small
standard
```

3. 随后，系统会要求选择编译的目标产品，选择 **qemu_small_system_demo**：

```
OHOS which product do you need? qemu_small_system_demo
```

4. 完成选择后，执行以下命令开始编译：

```
hb build -f
```

-f选项意味着全代码编译。在某些情况下构建系统意识不到源代码已经变化，可能会忽略文件的修改，而没有对其重新编译。导致最终结果出错。

3.运行

若编译没有出错，则对应生成的文件应当在out/arm_virt/qemu_small_system_demo/目录。由于 OpenHarmony 在版本更新中调整了部分代码结构，而 **qemu-run** 脚本未同步更新，需对源码根目录下的 **qemu-run** 脚本进行修改。请将脚本中第 **18-21 行** 的内容替换为以下代码：

```
board=$(cat out/ohos_config.json | grep -oP '(?<="board": ")[^"]*' | sed -e 's/\\//g')
kernel_type=$(cat out/ohos_config.json | grep -oP '(?<="kernel": ")[^"]*' | sed -e
's/\\//g')
product=$(cat out/ohos_config.json | grep -oP '(?<="product": ")[^"]*' | sed -e 's/\\//g')
product_path=$(cat out/ohos_config.json | grep -oP '(?<="product_path": ")[^"]*' | sed -e
's/\\//g')
```

修改后运行如下命令进入qemu环境中

```
./qemu-run -f
```

-f命令用于重建smallmmc.img,否则如若修改代码重新编译，之后在qemu运行中也仍然会使用更早之前的smallmmc.img,导致结果没有变化。

如若一切正常，输出结果应当如下所示：

```
waiting VNC connection on: 5920 ...(Ctrl-C exit)
```

```
*****welcome*****
```

```
Processor   : Cortex-A7
Run Mode    : UP
GIC Rev     : GICv2
build time  : Apr  7 2024 21:32:27
Kernel     : Huawei LiteOS 2.0.0.37/debug
```

```
*****
```

```
main core booting up...
cpu 0 entering scheduler
dev random init ...
mem dev init ...
DevMmzRegister...
Date:Apr  7 2024.
Time:21:31:57.
net init ...
```

```
*****
```

```
DeviceManagerStart start ...
[ERR][KProcess:SystemInit]virtio-mmio ID=1 device not found
SoftbusLwipMonitorInit init success...
DeviceManagerStart end ...
OsMountRootfs start ...
warning: /dev/mmcblk0 lost, force umount ret = -6
OsMountRootfs end ...
virtual_serial_init start ...
virtual_serial_init end ...
system_console_init start ...
system_console_init end ...
OsUserInitProcess start ...
OsUserInitProcess end ...
[ERR]Unsupported API tcgetpggrp
OHOS:/$
```

提示：输入命令Ctrl+a,再按x键即可退出qemu。

至此即完成了本次实验环境配置的所有内容。

4.常见问题

1. 运行命令 `./qemu-run` 出现如下错误

```
DeviceManagerStart start ...
[ERR][KProcess:SystemInit]virtio-mmio ID=1 device not found
SoftbusLwipMonitorInit init success...
DeviceManagerStart end ...
OsMountRootfs start ...
[ERR][KProcess:SystemInit]Cannot find bootargs!
[ERR][KProcess:SystemInit]parse bootargs error!
[ERR][KProcess:SystemInit]Cannot find root![ERR][KProcess:SystemInit]mount rootfs error!
```

```
OsMountRootfs end ...
virtual_serial_init start ...
virtual_serial_init end ...
system_console_init start ...
system_console_init end ...
OsUserInitProcess start ...
OsUserInitProcess end ...
[ERR][Init:thread0][VFS]lookup failed, invalid path err = -22
```

可能由于 `/vendor/ohemu/qemu_small_system_demo/qemu_run.sh` 脚本文件制作 `smallmmc.img` 文件过程中出现错误，导致生成的 `smallmmc.img` 文件残缺，因此使用该文件运行 `qemu` 会出现错误。遇到此类问题可尝试在命令 `./qemu-run` 后加上 `-f` 标志，让脚本重新生成 `smallmmc.img` 磁盘映像文件。或者手动删除 `out/smallmmc.img`

2. 运行命令 `./qemu-run -f` 出现如下错误

```
Error: Partition(s) 1 on /dev/loop590 have been written, but we have been unable to inform
the kernel of the change, probably because it/they are in use. As a result, the old
partition(s) will remain in use. You should reboot now before making further changes.
```

此部分错误由于执行 `parted` 命令进行分区后内核没有意识到硬盘分区的改变，但实际上操作已经完成。该错误其实并不影响 `smallmmc.img` 制作，但是脚本文件 `qemu_run.sh` 默认为严格模式执行，导致没有执行之后的代码，因此制作出的 `smallmmc.img` 残缺。解决方法如下：

1. 打开 `/vendor/ohemu/qemu_small_system_demo/qemu_run.sh` 文件，注释其中第16行代码以关闭脚本文件的严格模式

```
#set -e
```

2. 将第98行代码修改如下即可正常运行：

```
if [ $existed == false ]; then
    sudo parted -s /dev/loop590 -- mklabel msdos mkpart primary fat32 10MiB 30MiB # 修改
添加
    sudo parted -s /dev/loop590 -- mkpart primary fat32 30MiB 80MiB # 修改
添加
    sudo parted -s /dev/loop590 -- mkpart primary fat32 80MiB -1s # 修改
添加
fi
```

五、参考资料

- [1]. OpenHarmony技术架构: https://gitee.com/openharmony/docs/blob/master/zh-cn/OpenHarmony-Overview_zh.md
- [2]. OpenHarmony仓库: <https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/quick-start/Readme-CN.md>
<https://gitee.com/openharmony/docs/blob/master/zh-cn/device-dev/quick-start/quickstart-pkg-prepare.md>
- [3]. OpenHarmony参考文档: <https://docs.openharmony.cn/pages/v4.1/zh-cn/device-dev/device-dev-guide.md>