实验案例一：鸿蒙设备开发环境的搭建

一、实验简介

OpenHarmony是鸿蒙设备开发所使用的系统，其是由开放原子开源基金会孵化及运营的开源项目，由HarmonyOS操作系统拆分而来的一部分。假设将HarmonyOS系统类比于Android系统，那么OpenHarmony就相当于AOSP(Android Open Source Project)。也就是HarmonyOS的的开源部分，HarmonyOS则是OpenHarmony的商业版本。同时在当前的鸿蒙设备开发之中，也都是基于OpenHarmony系统来进行的。

在同学们本次学习鸿蒙操作系统的过程当中，我们就通过在OpenHarmony上的实验来学习鸿蒙操作系统的特性以及开发方法。在本次以及以后的设备相关实验之中，我们都将基于OpenHarmony源码进行实验，同学们需要在本次的实验之中根据文档资料完成鸿蒙设备开发环境的搭建，也是为之后进一步的实验做好准备。

二、实验内容及要求

本次实验为OpenHarmony的设备开发环境搭建实验，需要同学们完成相关环境的搭建与配置。本次所需要下载与使用到的系统和工具包括有：

- **VMware Workstation Player**：本次实验需要使用Linux操作系统进行，若电脑为Windows系统，则需要使用虚拟机来安装。VMware Workstation Player为VMware的免费虚拟机产品，可以在本次实验中使用。
- **Linux发行版**：Linux发行版是由Linux内核、GNU工具、附加软件和软件包管理器组成的操作系统。主流的Linux发行版有Ubuntu、Debian、Fedora、Centos等，本次实验提供的命令以及安装步骤主要是针对**Ubuntu 20.04**，其它不同版本的操作系统安装细节可能会有些许差别。
- **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。

由于目前OpenHarmony的官方文档仍然不够完备，因此我们的安装过程中有部分步骤与文档所述不一致，同学们在配置环境过程中可根据情况自行调整。**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：完成环境配置，进入ohos系统

本次实验仅要求同学们完成设备环境的搭建和配置，需要同学们搭建环境之后，使用qemu进入ohos系统并提交截图。

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，其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某些非必要的组件。

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。**在之后的实验之中，同学们也会不同程度接触体会到这三大特性。

近些年之中，OpenHarmony系统开发进展迅速，已历经多个版本的迭代。本次实验主要基于OpenHarmony 4.1版本进行，其系统源代码大小大致50g左右，可以说对同学们的电脑硬盘空间是一个巨大挑战。不过在OpenHarmony版本更新的过程之中，无论是参考文档，还是代码的目录结构都可能经历大规模的修改，虽然无法保证当前实验所使用版本和最新版本之间的一致性，但是其中的思想、特性不会产生根本变化。

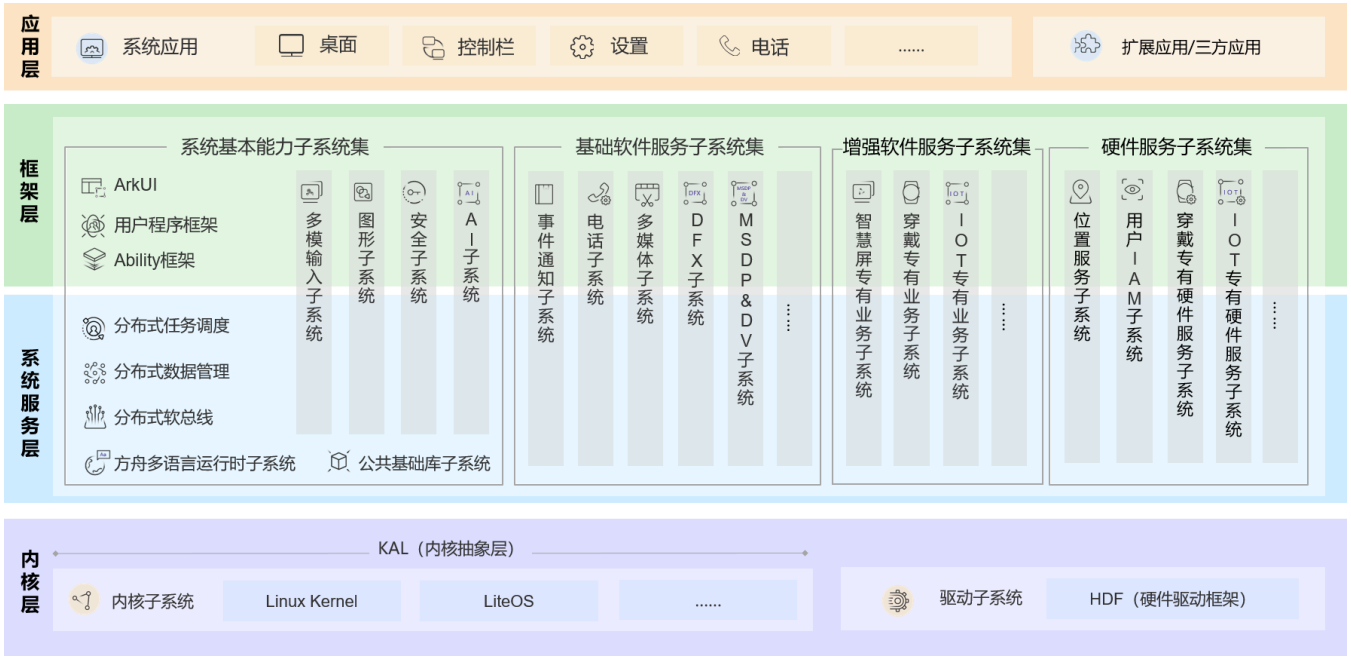

图1.OpenHarmony技术架构[1]

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. 用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 libpixman-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

本次实验所使用的版本为OpenHarmony-4.1-Release,需要同学们自行下载获取OpenHarmony源码。

> 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是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安装hb。

```
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

在下载的OpenHarmony源代码的根目录依次执行如下操作构建编译源码

在已经获取的源码根目录，输入如下代码

```
hb set
```

之后选择OpenHarmony的编译系统等级,选择small系统，即小型系统内核Lite-OS-A

```
OHOS Which os_level do you need?  (Use arrow keys)
   mini
 › small
   standard
```

之后应当会出现要求选择要编译的产品，请选择`qemu_small_system_demo`选项,结果如下：

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

选择完产品之后，即可执行如下命令编译产品：

```
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版本更新，代码结构有进行修改，但是关于`qemu-run`脚本文件没有同步更新。需要先修改根目录下的qemu-run脚本文件，将其中第18-21行代码修改为如下内容:

```
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')
```

最后即可使用如下命令进入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 tcgetpgrp
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。解决方法只需要打开`/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
```

并修改第98行代码为如下内容即可正常运行：

```
    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