

****

软 件 学 院

《实用操作系统》Project1

**题　　目 向鸿蒙Liteos中加入一个自定义的系统调用**

**姓　　名 陈澄**

**学　　号 32420212202930**

**班　　级 软工三班**

**实验时间 2023/10/1**

**2023 年 10 月 1 日**

# 项目环境

主机：Windows11

虚拟机：VMware Workstation 17Player，Ubuntu18.04

开发板：MAX6ULL MINI

传输工具：VMware共享文件夹

信道传输工具：MobaXterm

# 项目内容

1.向鸿蒙Liteos中加入一个自定义的系统调用

2.测试新加入的自定义系统调用

# 项目步骤

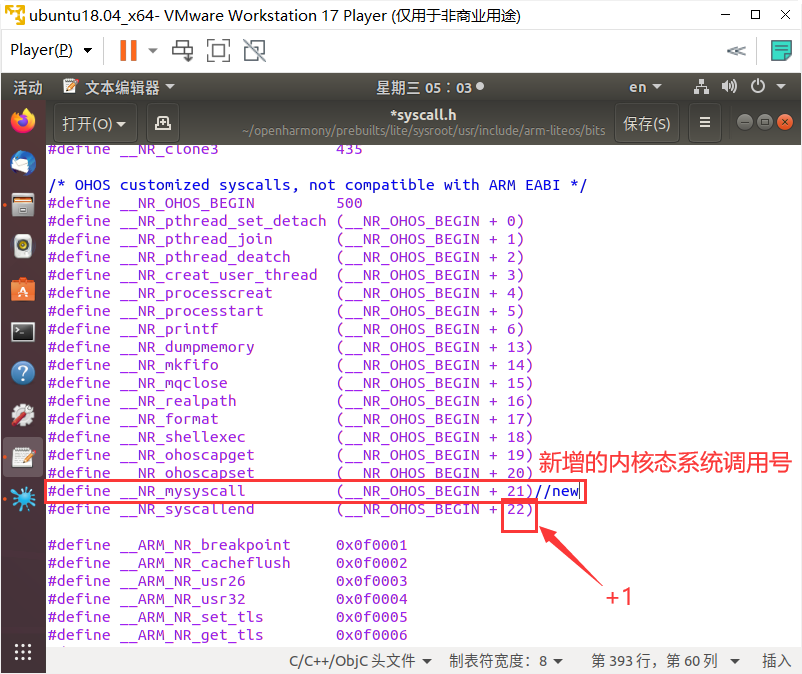
1.向syscall.h文件中，新增一个自定义系统调用号

打开/home/book/openharmony/prebuilts/lite/sysroot/usr/include

/armliteos/bits/syscall.h

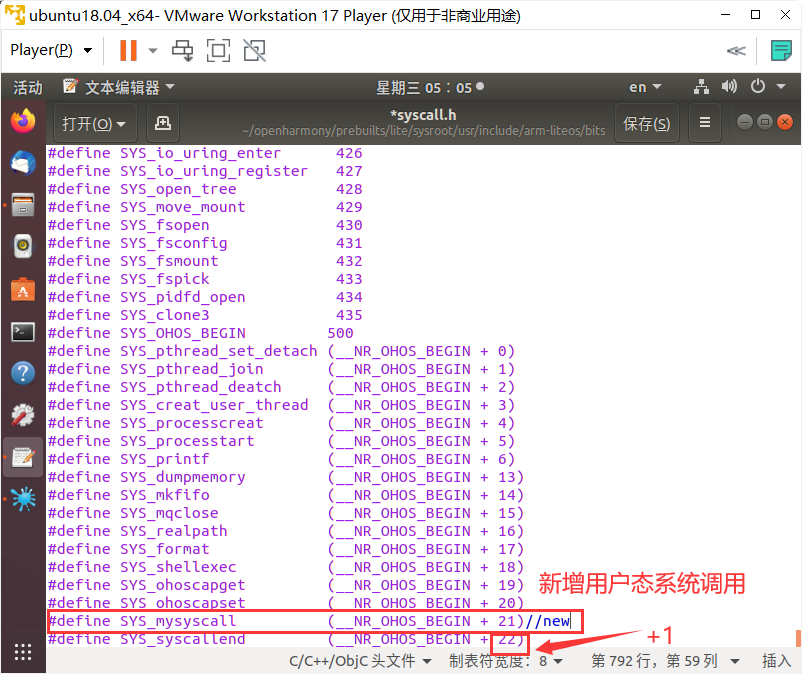
在中间新增一个内核态的系统调用号\_\_NR\_mysyscall

并且将结束的系统调用号+1用于系统调用号的边界判断



在结尾新增一个用户态的系统调用号\_\_SYS\_mysyscall

并且将结束的系统调用号+1用于系统调用号的边界判断

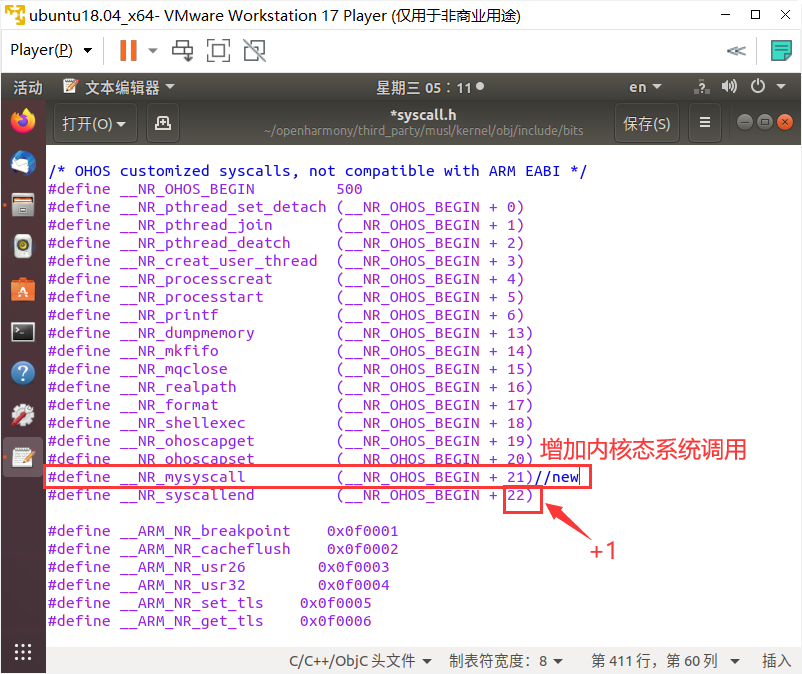


打开/home/book/openharmony/third\_party/musl/kernel/obj/

include/bits/syscall.h

在结尾新增一个内核态的系统调用号\_\_NR\_mysyscall

并且将结束的系统调用号+1用于系统调用号的边界判断

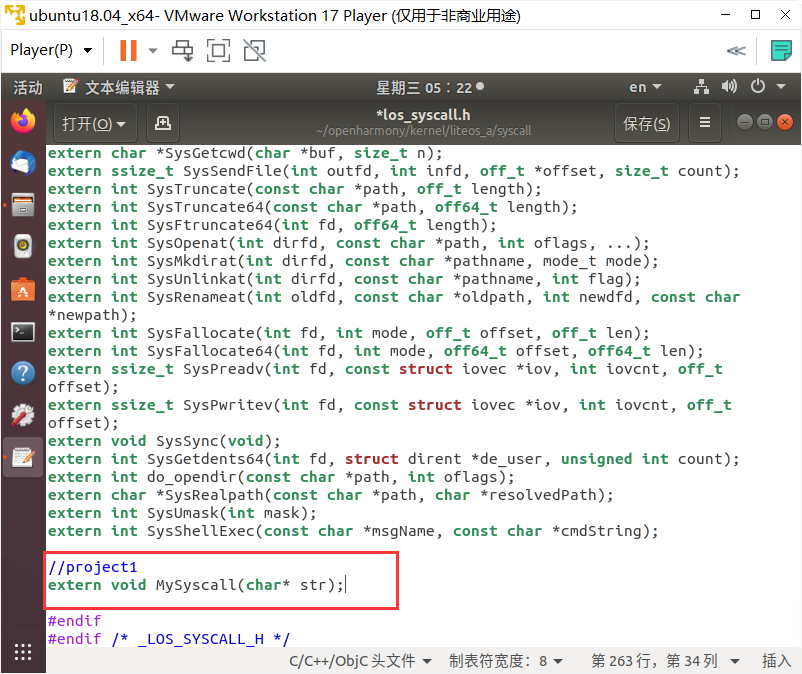


2.在los\_syscall.h文件中，新增系统调用的函数声明

打开/home/book/openharmony/kernel/liteos\_a/syscall/

los\_syscall.h

新增系统调用函数 MySyscall声明。当发生对应的系统调用时，该函数将被执行。

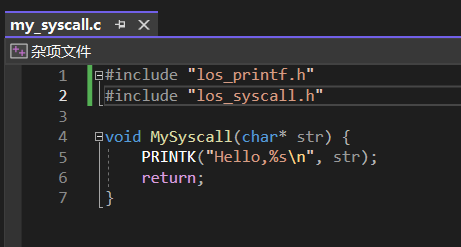


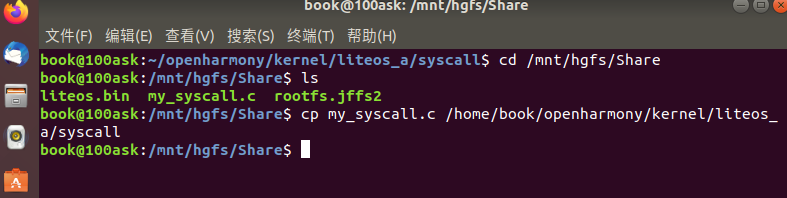
3.在los\_syscall.h文件的同级目录下，新增系统调用的函数实现

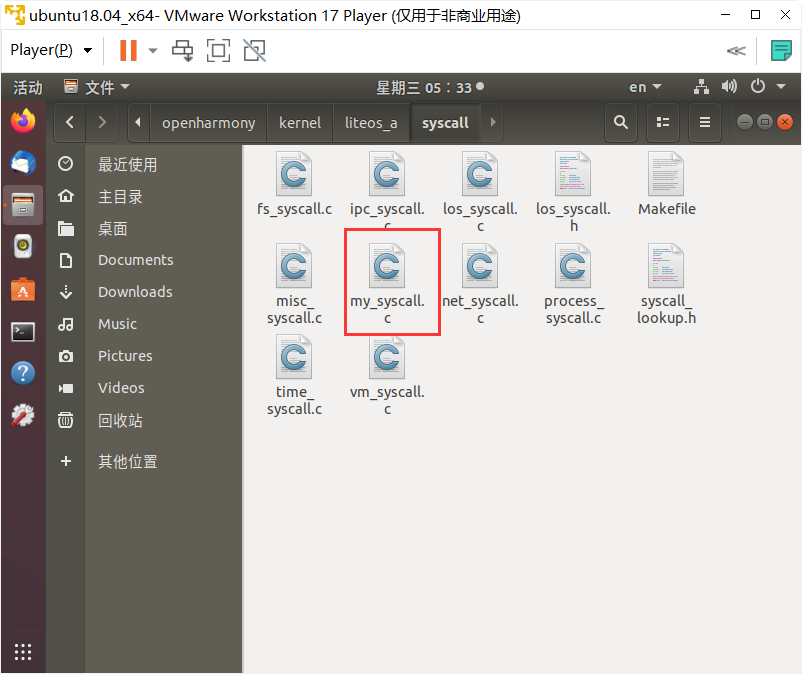
在主机创建一个MySyscall.c文件

传入一个字符串并输出用于简单测试

保存后传输到虚拟机的los\_syscall.h文件的同级目录下





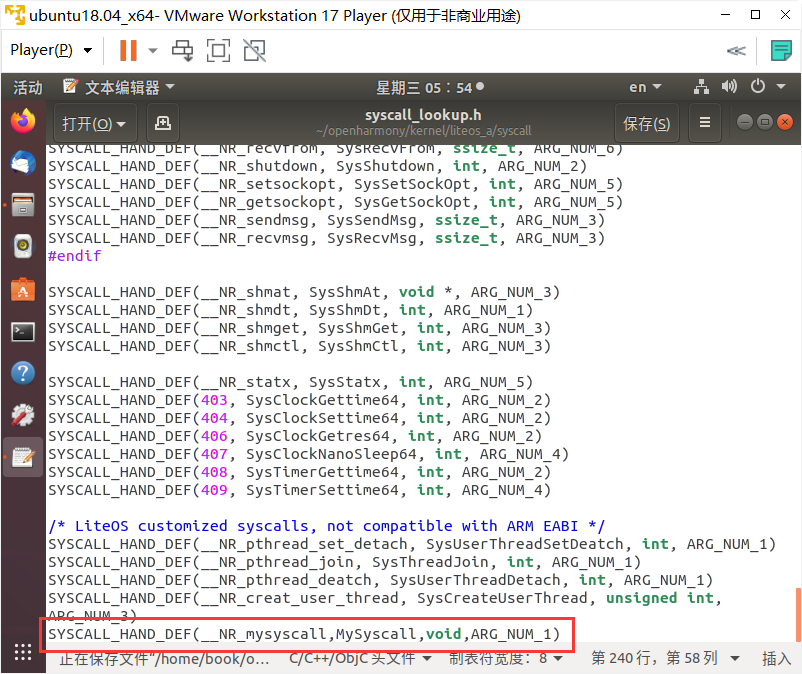


4.在syscall\_lookup.h文件中，新增系统调用号和系统调用函数的映射关系

打开/home/book/openharmony/kernel/liteos\_a/syscall/

syscall\_lookup.h

新增系统调用号\_\_NR\_mysyscall和系统调用函数MySyscall之间的映射关系



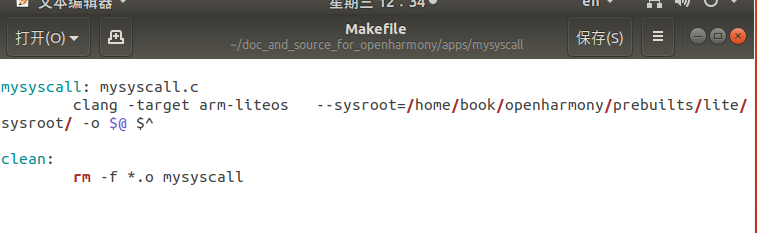
5.设计测试软件

新建测试软件mysyscall.c

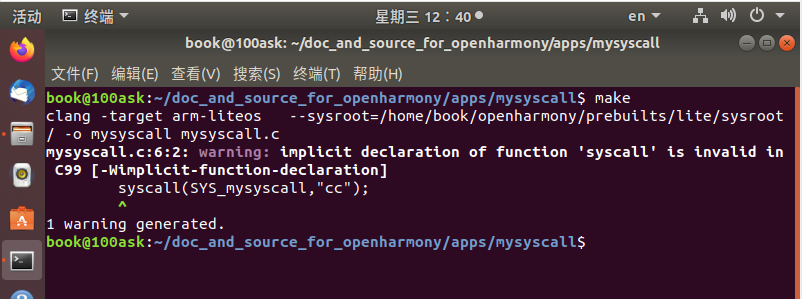


直接在主函数中调用新增的用户态系统调用，并传入一个字符串”cc”

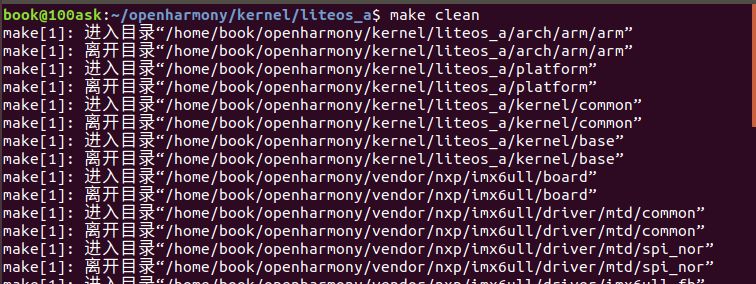
修改Makefile文件用于编译mysyscall.c

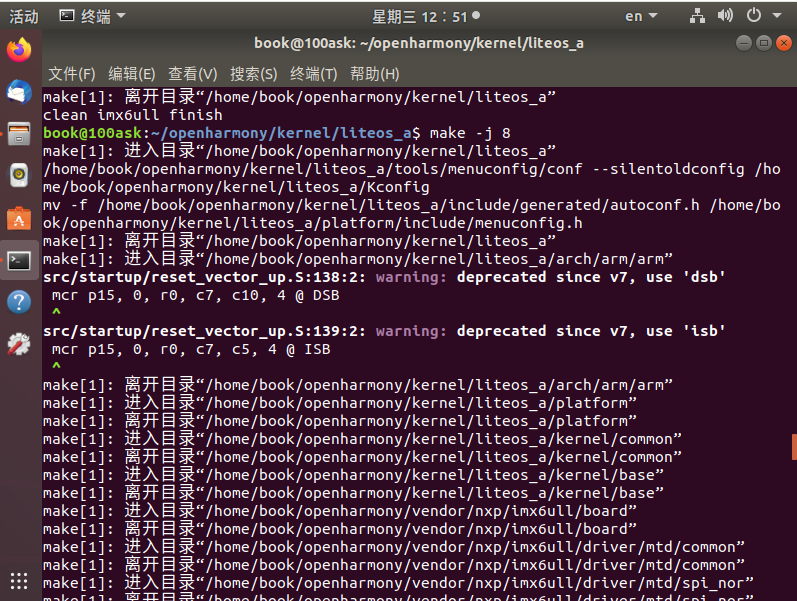


进行编译



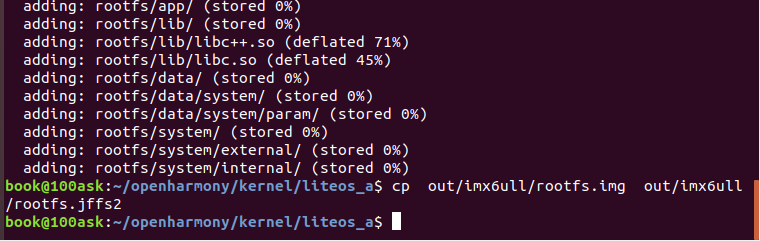
6.进入/openharmony/kernel/liteos\_a重新编译系统内核





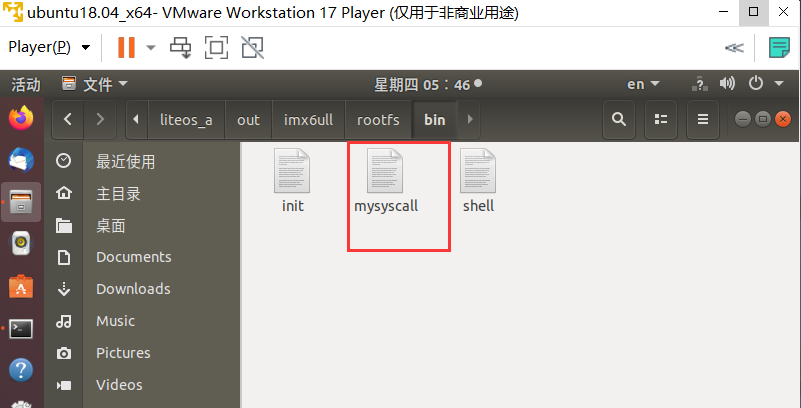
制作rootfs.jffs2



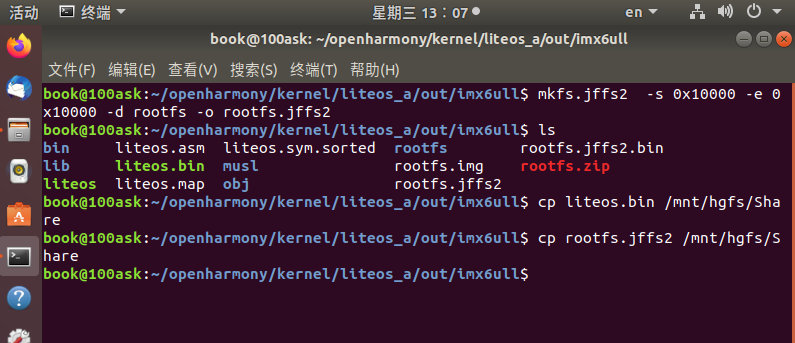


将mysyscall复制到/openharmony/kernel/liteos\_a/out/imx6ull

/bin



重新制作rootfs.jffs2将得到的liteos.bin与rootfs.jffs2复制到共享文件夹里

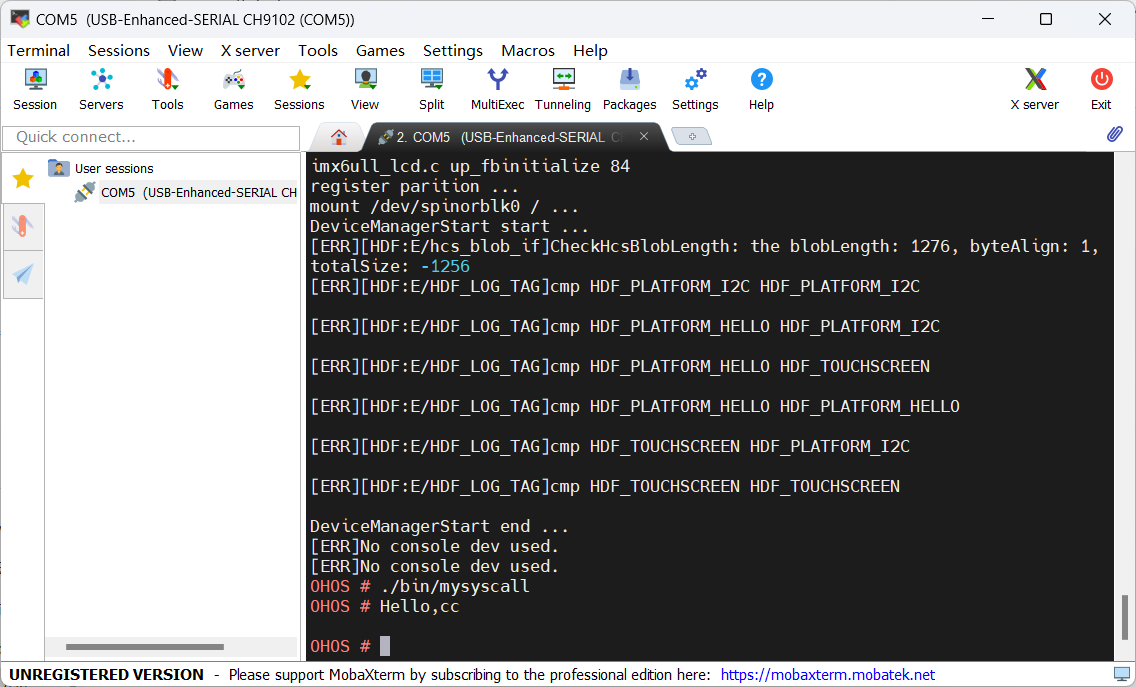


7.测试结果

通过烧写软件将其烧写到开发板

执行mysyscall

运行结果如图，成功打印，测试成功



# 实验遇到的问题及其解决方法

无

# 我的体会

通过本次实验，学会了如何对鸿蒙liteos的内核进行更改，学会了增加内核态和用户态的系统调用，掌握了编写测试程序对更改进行验证的方法，对鸿蒙内部结构的理解更加深刻了。