

**软件课程设计(III)**

**课程项目报告**

**班 级 9191062301**

**组队人员 孙傲歆-919106840333**

**指导教师 赵庆玲**

**2023年1月**

# **目 录**

[**目 录** 2](#_Toc121496927)

[一、概述 3](#_Toc121496928)

[1.1 实验概述 3](#_Toc121496929)

[1.2 实验环境 3](#_Toc121496930)

[二、openEuler虚拟机的安装 3](#_Toc121496931)

[2.1 x86架构虚拟机的安装 3](#_Toc121496932)

[2.2 aarch64架构虚拟机的安装 6](#_Toc121496933)

[三、内核编程实验 7](#_Toc121496934)

[3.1 实验介绍 7](#_Toc121496935)

[3.2 实验目的 7](#_Toc121496936)

[3.3 内核的编译与安装 7](#_Toc121496937)

[3.4 字符打印内核模块 11](#_Toc121496938)

[四、系统编程环境实验 12](#_Toc121496939)

[4.1 实验介绍 12](#_Toc121496940)

[4.2 实验目的 13](#_Toc121496941)

[4.3 x86架构下实验 13](#_Toc121496942)

[4.4 aarch64架构下实验 14](#_Toc121496943)

[五、总结 14](#_Toc121496944)

[5.1 实验过程中遇到的问题及解决方案 14](#_Toc121496945)

[5.2 心得体会 15](#_Toc121496946)

一、概述

1.1 实验概述

本实验是基于华为自主研发的操作系统——openEuler进行的操作系统内核编程实验。

本次实验我们没有使用华为云平台，而是自己于虚拟机上搭建openEuler操作系统。并在自己搭建的虚拟机上进行本次课程要求的全部实验。实验一共包含两个部分：**一是内核编程实验**，要求安装openEuler操作系统内核并编写一个简单的内核模块以验证安装是否成功；**二是系统编程环境实验**，本实验不属于内核编程实验，而是应用层实验，主要目的是对比x86架构和Kunpeng架构（实际上就是aarch64架构）的不同。

实验报告将依次介绍：虚拟机的安装过程、内核编程实验及系统编程环境实验。最后还将列出我们在实验中遇到的一些问题和解决方案以及经过本次实验我们的一些心得体会。

1.2 实验环境

* **本地主机操作系统：**windows 11
* **远程主机操作系统：**openEuler 20.03
* **远程连接工具**：MobaXtrem Home Edition v22.0
* **虚拟机：**Oracle VM VirtualBox & QEMU

二、openEuler虚拟机的安装

2.1 x86架构虚拟机的安装

**步骤1：新建虚拟机**

打开VirtualBox软件。在VirtualBox的管理器上点击＂新建＂即可开始创建虚拟机。**如下图所示：**



接下来便是对虚拟机进行基础配置，包括：虚拟电脑名称和系统类型、内存大小、虚拟硬盘设置等等，最后虚拟机的配置**如下图所示**：

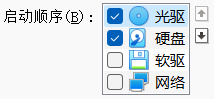


以上的配置需要特别说明的是“存储”一栏中，第一IDE控制器主通道是正常的存储硬盘，分配了40GB容量，而第二IDE控制器主通道则是一个光驱，其实oepnEuler操作系统的安装文件。当第一次安装操作系统时，必须设置启动顺序为：光驱-硬盘。这样才能保证光驱的操作系统安装文件先执行，从而顺利安装操作系统。

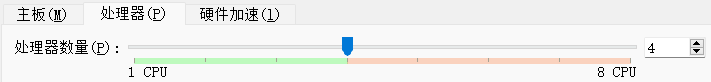
**步骤2：设置虚拟电脑并安装。**

在安装操作系统之前需要我们先进行一些基础设置。

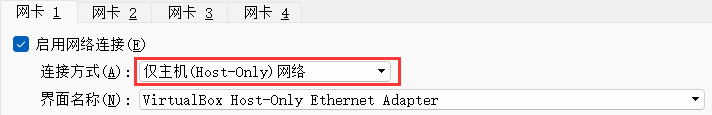
首先就是我们前面已经提到的启动顺序问题，第一次安装操作系统，我们要将光驱放在第一位，硬盘放在第二位，**如下图所示：**

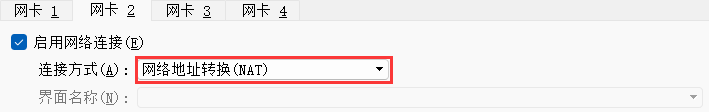


对CPU数进行设置，这里我们设置4个处理器，**如下图所示：**



接下来是网络设置，这里我们需要设置两个网卡，网卡1连接方式下拉选择为＂仅主机（Host-Only）网络＂，主要用于本地电脑与虚拟电脑相连；网卡2设置，勾选＂启用网络连接＂，并且＂连接方式选择＂下拉选择为＂网络地址转换（NAT）＂，主要用于虚拟电脑与互联网相连，**如下图所示：**

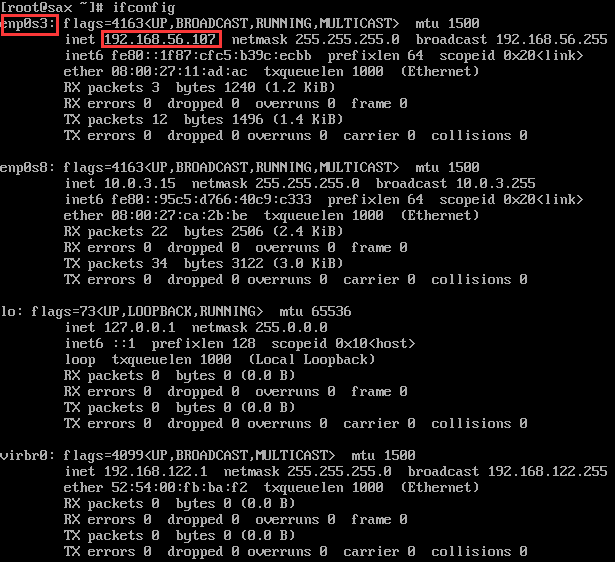




这样我们便设置完成，点击“启动”即可进行操作系统的安装。安装的具体步骤就不再赘述，由于openEuler是图形化安装界面，安装也十分简单。几个需要特别注意的地方有：1.设置的两张网卡都要确认打开；2.在“安装位置”设置中最好选择为自定义分区——标准分区；3. ＂软件选择＂一栏中基本环境选择为“虚拟化主机”，附加选项选择＂Linux的远程管理＂、＂开发工具＂、＂安全性工具＂、＂系统工具＂。

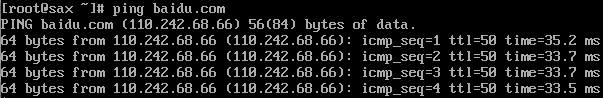
**步骤3：网络确认**

通过ifconfig来查看两张网卡是否都正常启动，**如下图所示：**



不难看出两张网卡都能正常启动，其中网卡“enp0s3”的IP地址十分重要，因为它是我们用来和本地电脑相联的网卡，之后使用MobaXtrem与虚拟机远程连接时将会用到。

执行命令ping baidu.com确认是否能上网，**如下图所示：**



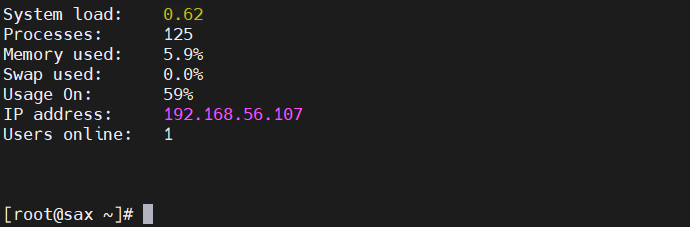
**步骤4：进行远程连接**

为了操作方便，可以使用SSH工具从本地电脑通过配置enp0s3网卡的IP地址来连接虚拟机，并使用root用户来登录。**如下图所示，**是使用MobaXtrem对Session连接进行配置，先配置用户名和密码再配置IP地址：



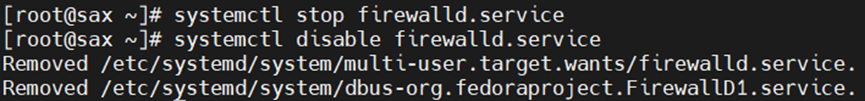


配置好后即可进行远程登录，**如下图所示：**



**步骤5：环境准备**

执行以下二个命令将防火墙关闭，**如下图所示：**



设置字符集，**如下图所示：**



关闭swap交换内存，**如下图所示：**



调整系统参数值。执行指令打开performance.sh文件，用#注释sysctl -w vm.min\_free\_kbytes=112640 &> /dev/null，**如下图所示：**





准备yum环境。首先备份原有的yum配置文件，**如下图所示：**

****

下载可用源的repo文件，**如下图所示：**



至此，x86 的虚拟机安装已经全部完成，下面我们就会**以该虚拟机为基础**进行内核编程实验和系统编程环境实验。

2.2 aarch64架构虚拟机的安装

**步骤1：安装QEMU**

首先需要安装qemu虚拟机。因为大部分电脑的架构都是amd64的，因为架构不同无法运行arrch64的操作系统，所以我们需要使用qemu来模拟arm64的架构，以此来安装aarch64架构的OpenEuler。qemu的安装十分简单，在此不再赘述。

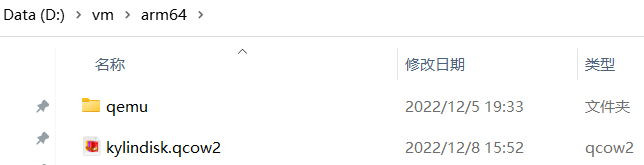
此外我们还需下载“UEFI”，其是对BIOS的替代方案。

**步骤2：生成虚拟硬盘**

qemu安装完成后，打开cmd，在qemu的安装路径下输入指令，生成一个40G的虚拟硬盘，**如下图所示：**

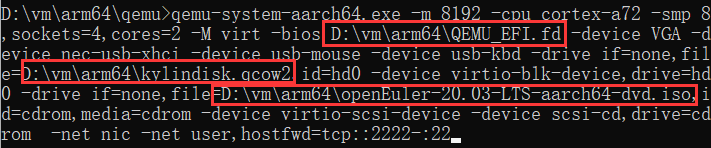


可以看到，在指定文件夹下确实生成了一个qcow2文件，其之后将作为虚拟机的硬盘进行数据的存储，**如下图所示：**



**步骤3：启动虚拟机并安装**

同样打开cmd，在qemu的安装路径下输入指令，即可启动虚拟机进行操作系统的安装，**如下图所示：**



值得注意的是，上图三个红框分别指定了虚拟机的**bios地址**、**硬盘地址**以及**操作系统安装的光驱地址**。运行上面这条指令就可以运行虚拟机并进行安装了。具体的安装过程不再赘述，与x86架构下的安装基本无异。

三、内核编程实验

3.1 实验介绍

本实验在鲲鹏云ECS上编译、安装openEuler操作系统内核并编写一个简单的内核模块以验证安装是否成功。

3.2 实验目的

* 熟悉openEuler内核的编译与安装；
* 了解内核模块编程的过程。

3.3 内核的编译与安装

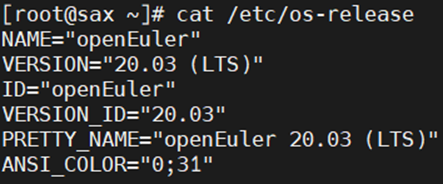
**步骤1：查看当前的系统信息**

使用uname指令可以查看系统的CPU架构以及内核版本号，**如下图所示：**

可以看到，我们的虚拟机其CPU架构为x86\_64，当前所用内核版本为4.19.XX-XXXX.X.X.XXXX.XXX.x86\_64。

之后使用cat指令查看/etc/os-release文件，即查看当前操作系统相关信息，其可以查看操作系统的名字、版本号、ID等信息，**如下图所示：**

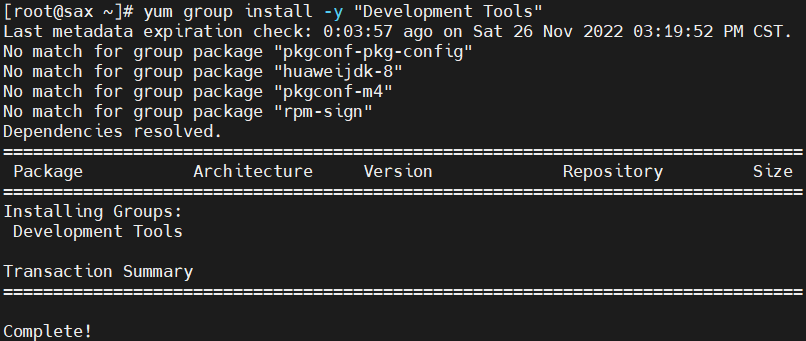


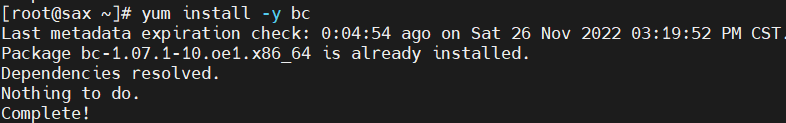
再使用指令，查看CPU数，因为在虚拟机设置时我们分配了4个CPU，所以会显示“4”，**如下图所示：**



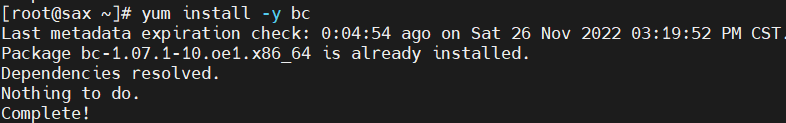
**步骤2：准备内核编译环境**

使用yum指令安装必要的文件，为内核编译做准备，具体的指令及结果**如下图所示：**

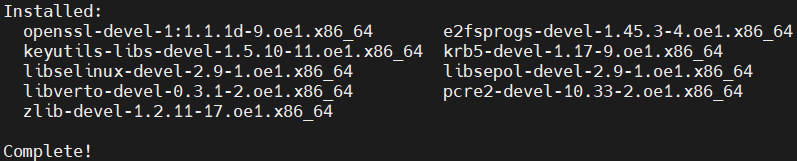












**步骤3：获取内核源码并解压**

我们先使用wget在gitee上下载我们准备使用的内核源码，**如下图所示：**



可以看到我们使用的是操作系统版本号仍然是20.03，但我们自己编译的内核任何系统自带有一定差距，后面便能看得到。之后我们再使用unzip指令，将下载的zip文件解压，**如下图所示：**



使用ls指令查看我们是否成功下载及解压成功，**如下图所示：**

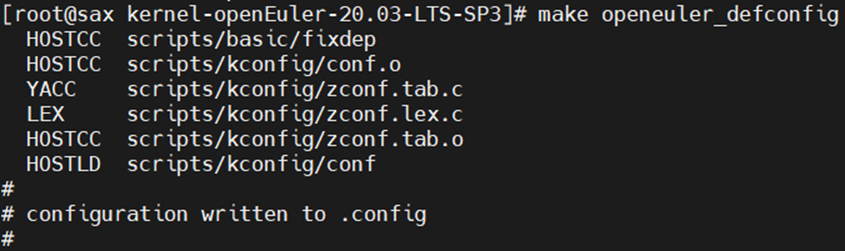


当前目录下多了kernel-openEuler-20.03-LTS-SP3文件夹，这便是我们解压后的内核源码，之后的工作都将在该文件下进行，所以我们使用cd指令进入该文件夹，**如下图所示：**



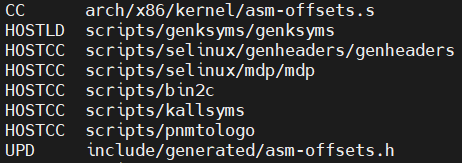
**步骤4：内核编译及安装**

首先要对内核进行相关配置，使用make openeuler\_defconfig指令即可，**如下图所示：**

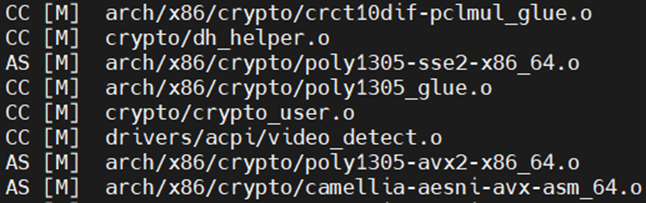


配置完成后，即可正式开始内核的编译，使用make -j16 bzImage进行内核编译，然后再使用make -j16 modules进行内核模块的编译。这里需要说明的是，“-j16”表示使用16个线程进行编译，但实际上我们的虚拟机只配置了4个CPU，这里使用16是没有太大意义的，具体**如下图所示：**



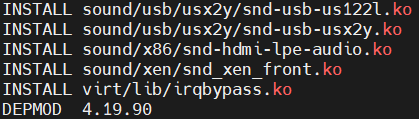






运行完以上两条make指令即可完成内核编译，内核编译完成后则需要进行安装。先使用make modules\_install安装模块，再使用make install安装内核，**如下图所示：**



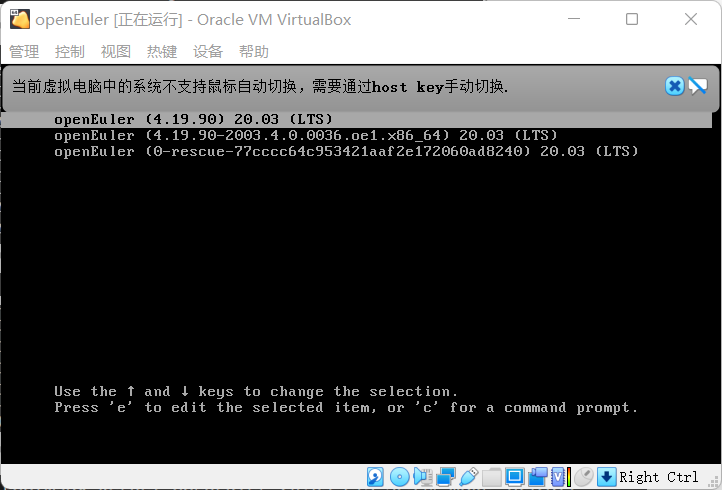




至此一个可用的新的内核即安装完成，下面可以正式进行相关的操作了。

**步骤5：重启系统**

使用reboot指令即可重启系统，可以发现虚拟机重新启动了，之后在操作系统选择界面我们可以看到一个新的“openEuler（4.19.90）20.03（LTS）”选项供我们选择，这说明我们的内核编译安装成功，**如下图所示：**



选中并按回车键进入该内核，使用MobaXtrem重新连接虚拟机，使用uname -r查看当前内核版本，**如下图所示：**

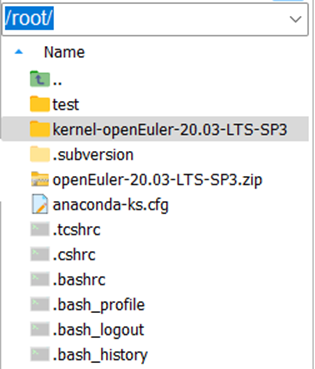


可以看到这里我们查看内核版本变成了4.19.90，而之前的系统内核版本是4.19.XX-XXXX.X.X.XXXX.XXX.x86\_64。所以即使是同版本的内核，系统自带和我们自己编译其版本号还是有所不同。

3.4 字符打印内核模块

**步骤1：编写C程序进行内核验证**

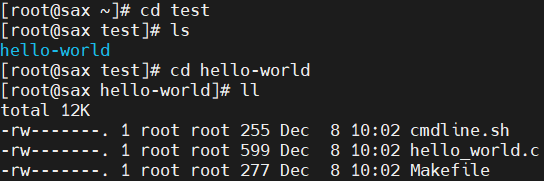
我们编写一个仅打印“Hello, world!”内核模块，包括.c源文件和Makefile文件。由于实验的软件包中已经给了相关的源码，我们只需将其复制到虚拟机中即可。MobaXtrem可以十分方便的进行文件复制，其在左边可以显示当前远程连接的电脑的文件系统。我们可以使用直接拖拽的方式将源码复制到虚拟机中，**如下图所示：**



我们再使用ls指令确定是否复制成功，**如下图所示：**

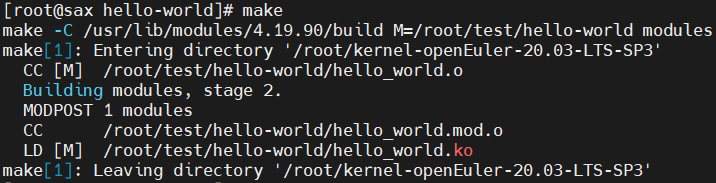


当前目录下多了文件夹test，说明复制成功。使用cd指令进入test文件夹再进入hello-world文件夹准备进行实验，**如下图所示：**



**步骤2：编译源文件**

由于Makefile文件已经写好且和源文件位于同一文件夹下，我们只需要make指令即可进行源文件的编译，**如下图所示：**



**步骤3：加载内核模块**

我们使用insmod指令加载编译完成的内核模块，并传送一定的参数值，此时的hello\_world.ko与内核同等，**如下图所示：**



执行上述指令后，我们可以在虚拟机界面上看到我们传送的参数，**如下图所示：**



我们再使用lsmod | grep hello指令查看内核加载结果，**如下图所示：**



**步骤4：卸载内核模块**

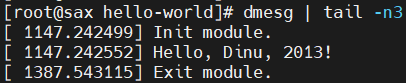
我们使用rmmod指令即可完成内核模块的卸载工作，**如下图所示：**



执行上述指令后，虚拟机界面也会给出反馈，显示“Exit module”，即退出模块，说明内核模块成功卸载，**如下图所示：**

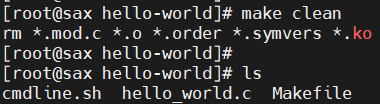


使用dmesg | tail -n3指令查看最后显示的三条指令信息，其结果与虚拟机显示器的结果是一样的，**如下图所示：**



**步骤5：清除源文件的编译结果**

使用make clean指令可以清除上次的make命令所产生的object文件,**如下图所示：**



可以看到执行完make clean指令后，hello-world文件夹下只保存了最初的三个文件，编译的结果被完全清除了。

四、系统编程环境实验

4.1 实验介绍

本实验通过运行shell命令查看系统信息以达到了解openEuler操作系统的目的。另外运行简单的C程序以了解基于ARMv8-64的开发环境。

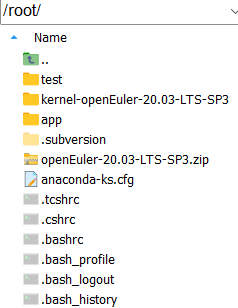
4.2 实验目的

* 了解openEuler操作系统的基本信息；
* 了解基于Kunpeng架构的openEuler操作系统开发环境。

4.3 x86架构下实验

**步骤1：编写C程序进行内验证**

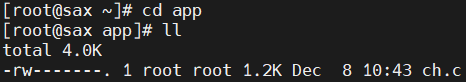
下面我们将以C程序为例，比较鲲鹏架构（aarch64架构）和x86\_64架构的不同点，由于实验的软件包中已经给了相关的源码，我们只需将其复制到虚拟机中即可。和前面一样通过拖拽方式直接复制即可，**如下图所示：**



我们再使用ls指令确定是否复制成功，**如下图所示：**

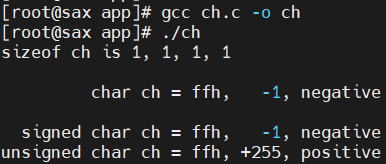


当前目录下多了文件夹app，说明复制成功。使用cd指令进入app文件夹准备进行实验，**如下图所示：**



**步骤2：编译并执行**

由于平台自带gcc，所以我们可以直接用指令运行该c代码，先使用gcc指令进行编译，再直接运行即可，**如下图所示：**



可见，由于计算机中的整数用补码表示的原因，-1在内存中的数值是0xff。在x86平台上char的默认数据类型与signed char相同。

4.4 aarch64架构下实验

**步骤1：编写C程序进行内验证**

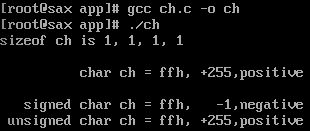
由于aarch64架构的虚拟机使用的是qemu虚拟机，而该虚拟机对于网卡的配置十分麻烦，所以这里我们便没有配置aarch64的网卡，所以其不能上网也不能连接主机，我们只能创建ch.c文件并手动编写程序。**如下图所示：**





**步骤2：编译并执行**

和在x86平台一样直接编译执行即可，**如下图所示：**



可见，在aarch64上char的默认数据类型与unsigned char相同。

其实，C语言标准并没有规定char应该是unsigned char还是signed char：

“The C standards do say that "char" may either be a "signed char" or "unsigned char" and it is up to the compilers implementation or the platform which is followed.”所以，在C语言中，char类型是不是相当于unsigned char和signed char泛型，即仅仅代表一个8bit位数的集合，至于这个集合的具体含义，是unsigned char还是signed char，那是和C编译器与CPU指令集都有关的，可以看出：信息就是上下文。

五、总结

5.1 实验过程中遇到的问题及解决方案

**问题一：aarch64架构虚拟机搭建**

**问题描述：**我先尝试使用Oracle VM VirtualBox进行aarch64架构虚拟机的搭建，本以为只需要换一个启动光驱即可，但在设定好之后发现启动光驱后直接报错，无法继续进行安装。

**解决办法：**使用qemu虚拟机进行aarch64安装，我参考了CSDN上的一篇文章（<https://blog.csdn.net/weixin_47151919/article/details/125391572>），成功安装了aarch64架构的openEuler。

**问题二：内核编译内存不足**

**问题描述：**最初我在x86平台下进行内核编译实验时，总是在执行指令“make -j16 modules”这一步时出错，出错的原因是“out of memory”，即存储容量不足。

**解决办法：**我尝试了各种解决办法，一开始怀疑是内存不足导致内存泄露于是加大了虚拟机的内存进行重试，发现还是会报错。后来又尝试了加CPU数量、加大硬盘容量等方法。后来发现就是硬盘容量一开始给的太少了（一开始只给了10GB）导致磁盘满了，之后改为40GB硬盘即成功解决问题。

**问题三：qemu虚拟机网卡设置**

**问题描述：**qemu无法上网也无法连接到主机。

**解决办法：**最初尝试了使用OpenVpn搭建网桥的方式，但最后以失败告终。最后只能在x86平台进行内核编译实验，而qemu虚拟机的aarch64架构内核则用于进行系统编程环境实验的对照，也算是没有办法的办法吧。

5.2 分工情况

|  |  |
| --- | --- |
| **成员** | **工作** |
|  |  |
|  |  |
|  |  |
|  |  |

5.3 心得体会

这是我们大学期间第三次接触华为云的相关实验了，第一次华为云实验是软件课程设计(I)课程中的 openGauss 数据库实验，第二次则是高性能计算引论的基于华为鲲鹏计算云服务进行的并行编程实验。而本次实验是openEuler内核编程实验，由小组成员孙傲歆、韦永贤、汪乾傲、邓龙欣共同完成，我们通力合作、相互配合，最终很好地完成了实验。

虽然之前学过操作系统这一门课，但是理论终究还是理论，真正去编译一个操作系统内核是我们从来没有接触过的东西，这次软件课程设计（III）的openEuler内核编程实验对我们来说不仅是一次十分新奇的体验。

本次实验我们并没有使用华为云的服务器（因为一直没发代金券，又舍不得自己买……），我们是自己搭建的openEuler虚拟机来进行实验的。实际上在之前软件课程设计(I)中我们就已经学过了openEuler虚拟机的搭建，而且也比较熟练。而高性能计算引论的实验又让我们更加熟悉了openEuler操作系统的相关操作以及各种linux指令。加上华为老师仔细的讲解和实验手册的指导，本次实验总体来说难度不是很大，更多的是研究不同架构虚拟机的安装问题，以及调整虚拟机的配置。

此次华为云的内核编程实验让我们对操作系统的结构及其运行原理有了更深的理解，此外系统编程环境实验让我们了解了一个相同C语言数据存储类型可能会因CPU架构的不同而不同，这或许也是说C语言的移植性不好的原因吧。总之，此次实验让我们受益匪浅，学到了很多。

在最后，感谢赵庆玲老师在本次课程设计中的指导与所做的努力，祝老师身体健康，科研顺利！