《操作系统实验4指导书》

[前言 1](#_Toc86595791)

[步骤1 鲲鹏云ECS的构建及内核编译 2](#_Toc86595792)

[1.2 实验介绍 2](#_Toc86595793)

[1.2.1 任务描述 2](#_Toc86595794)

[1.3 实验目的 2](#_Toc86595795)

[1.4 构建云实验环境 2](#_Toc86595796)

[1.4.1 创建VPC 2](#_Toc86595797)

[1.4.2 购买ECS 5](#_Toc86595798)

[1.4.3 通过ssh登录系统 6](#_Toc86595799)

[1.5 openEuler内核编译与安装 8](#_Toc86595800)

[1.6 云环境资源清理 11](#_Toc86595801)

[1.6.1 ECS关机 11](#_Toc86595802)

[1.6.2 删除ECS 11](#_Toc86595803)

[步骤2 安装BPF所需环境与依赖项 14](#_Toc86595804)

[2.1任务描述 14](#_Toc86595805)

[2.2.1安装依赖项 14](#_Toc86595806)

[2.2.2安装其他环境 14](#_Toc86595807)

[2.2.3编译示例代码 15](#_Toc86595808)

[步骤3看懂Makefile文件 22](#_Toc86595809)

[3.1任务描述 22](#_Toc86595810)

[3.2具体要求 22](#_Toc86595811)

[步骤4执行PBF示例程序 23](#_Toc86595812)

[4.1任务描述 23](#_Toc86595813)

[4.2具体过程 23](#_Toc86595814)

[4.2.1查看文件，读懂其功能 23](#_Toc86595815)

[4.2.2执行可执行文件 23](#_Toc86595816)

[4.2.3查看生成的elf格式的可执行文件 23](#_Toc86595817)

[4.2.4执行llvm-objdump命令 24](#_Toc86595818)

[步骤5编写自己的BPF程序 25](#_Toc86595819)

[5.1任务描述 25](#_Toc86595820)

[5.2具体要求 25](#_Toc86595821)

前言

鲲鹏云是“华为云鲲鹏云”的简称，本实验手册是基于鲲鹏云弹性云服务器（ECS）的openEuler操作系统内核编程实验手册，包括了内核编译、内存管理、进程管理、中断异常管理、内核时间管理、设备管理、文件管理以及网络管理等内核相关实验。

本实验手册不仅包含实验步骤，还在每章简要介绍了实验相关的原理和背景知识，以便读者能更好地理解操作系统内核的原理并进行实验。

一、实验网络环境介绍

SG

ECS

VPC

EIP

PC

openEuler操作系统实验的云环境

如上图所示，本实验的云环境主要是一台基于鲲鹏架构的ECS，附加部件有弹性公网IP（EIP）、虚拟私有云（VPC）和安全组（SG）。学生端的PC机使用ssh终端登录ECS。

二、实验设备介绍

关键配置如下表所示：

关键配置

|  |  |
| --- | --- |
| 云资源 | 规格 |
| ECS鲲鹏计算 | 4vCPUs | 8GB | 40GB |
| EIP带宽 | 按流量计费 |

软件方面，本实验需要一台终端电脑与弹性云服务器(ECS)链接以输入操作命令或/和传输文件。对于Windows 10 / macOS / Linux，我们可以用命令行工具ssh和scp完成这个过程。

如果在有些Windows系统下不能运行ssh工具，也可以使用Putty和WinSCP工具软件。其中Putty工具的推荐下载地址：

<https://hcia.obs.cn-north-4.myhuaweicloud.com/v1.5/putty.exe>

WinSCP的推荐下载地址：

<https://winscp.net/eng/index.php>

# 步骤1 鲲鹏云ECS的构建及内核编译

## 实验介绍

本实验通过构建鲲鹏云ECS、编译安装openEuler操作系统新内核以及简单的内核模块编程任务操作带领大家了解操作系统以及内核编程。

### 任务描述

构建鲲鹏云ECS

编译安装openEuler操作系统新内核

简单的内核模块编程实验，在内核模块中打印“Hello, world!”

## 实验目的

学习掌握如何安装构建ECS

学习掌握如何编译操作系统内核

了解内核模块编程。

## 构建云实验环境

### 创建VPC

在浏览器地址栏输入华为云控制台网址[console.huaweicloud.com](https://console.huaweicloud.com/)并按回车键，这时页面将跳转至登录页。



按要求输入账号密码，进行登录。

注意：在此之前您需要在华为云主页注册华为云账号。

登录成功后会自动进入控制台页面，这时将区域选在 “华北-北京四”。



将鼠标悬停于左侧导航栏图标处展开服务列表，然后在服务列表中点击“虚拟私有云VPC”项。



点击“虚拟私有云”控制台页面右上角的“创建虚拟私有云”按钮。



在创建虚拟私有云的页面中按照下表内容配置虚拟私有云参数。

|  |  |
| --- | --- |
| 参数 | 配置 |
| 区域 | 华北-北京四 |
| 名称 | vpc-test |
| 网段 | 192.168.1.0/24 |
| 企业项目 | default |
| 默认子网可用区 | 可用区1 |
| 默认子网名称 | subnet-test |
| 子网网段 | 如192.168.1.0/24 |

配置完成后，点击“立即创建”，创建完成后会自动回到VPC控制台。

点击VPC控制台左侧导航栏的“访问控制”🡪“安全组”，进入安全组控制台。



点击右上角的“创建安全组”。



在弹出的对话框中按“通用Web服务器”配置安全组参数，然后点击“确定”。



### 购买ECS

将鼠标悬停于左侧导航栏图标处展开服务列表。然后在服务列表中点击“弹性云服务器ECS”项。



点击弹性云服务器ECS控制台页面右上角的“购买弹性云服务器ECS”按钮进入购买页面。



按照下表内容配置弹性云服务器ECS的参数。

|  |  |
| --- | --- |
| 参数 | 配置 |
| 计费模式 | 按需计费 |
| 区域 | 华北-北京四 |
| 可用区 | 可用区1 |
| CPU架构 | 鲲鹏计算 |
| 规格 | 鲲鹏通用计算增强型 | kc1.xlarge.2 | 4vCPUs | 8GB |
| 镜像 | 公共镜像 | openEuler | openEuler 20.03 64bit with ARM(40GB) |
| 系统盘 | 通用型SSD | 40GB |

注意：这里“区域”的配置是和VPC的区域配置保持一致的。

配置完成后点击“下一步：网络配置”，进入网络配置，按下表配置网络参数。

|  |  |
| --- | --- |
| 参数 | 配置 |
| 网络 | vpc-test | subnet-test | 自动分配IP地址 |
| 安全组 | sg-test |
| 弹性公网IP | 现在购买 |
| 线路 | 全动态BGP |
| 公网带宽 | 按流量计费 |
| 带宽大小 | 5Mbit/s |

配置完成后，点击“下一步：高级配置”，按下表配置ECS高级配置参数。

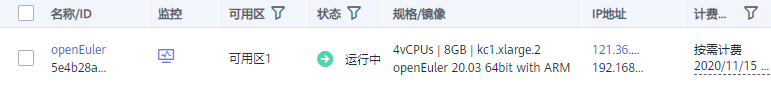
|  |  |
| --- | --- |
| 参数 | 配置 |
| 云服务器名称 | openEuler（输入符合规则名称） |
| 登录凭证 | 密码 |
| 密码 | 请输入8位以上包含大小写字母、数字和特殊字符的密码，如openEuler@123 |
| 确认密码 | 请再次输入密码 |
| 云备份 | 暂不购买 |
| 云服务器组 | 不配置 |
| 高级选项 | 不勾选 |

配置完成后点击右下角“下一步：确认配置”。勾选同意协议，然后点击：立即购买。

在提交任务成功后，点击“返回云服务器列表”，返回ECS控制台。

### 通过ssh登录系统

在ECS控制台查看ECS弹性公网IP地址。



在客户端机器操作系统里的Console控制台或Terminal终端里运行ssh命令：

$ ssh root@*121.36.45.64*

（注意：此处的IP地址*121.36.45.64*即是刚刚购买的弹性公网IP地址。）

在客户端（本地PC）第一次登录时会有安全性验证的提示：

The authenticity of host '119.8.238.181 (119.8.238.181)' can't be established.

ECDSA key fingerprint is SHA256:RVxC1cSuMmqLtWdMw4n6f/VPsfWLkT/zDMT2q4qWxc0.

Are you sure you want to continue connecting (yes/no)? yes

在这里输入yes并按回车键继续：

Warning: Permanently added '119.8.238.181' (ECDSA) to the list of known hosts.

Authorized users only. All activities may be monitored and reported.

root@119.8.238.181's password:

输入密码(注意这里不会有任何回显)并回车，登录后的界面如下所示：

Welcome to Huawei Cloud Service

Last login: Mon May 18 15:35:37 2020

Welcome to Huawei Cloud Service

Last login: Mon May 18 15:35:37 2020

Welcome to 4.19.90-2003.4.0.0036.oe1.aarch64

System information as of time: Sun Nov 15 14:41:58 CST 2020

System load: 0.15

Processes: 131

Memory used: 5.0%

Swap used: 0.0%

Usage On: 9%

IP address: 192.168.1.5

Users online: 1

[root@openeuler ~]#

修改主机名

ECS创建时被命名为“openEuler”,所以系统默认hostname为“openeuler”，为了和本实验手册另外两个版本保持行文上的一致，我们可以将主机名改为“openEuler”或“localhost”（一般在虚拟机中，主机名被默认为localhost，而ECS也是虚拟机。本文的上下文环境中可能同时用到这三种名称，请鉴别）：

[root@openeuler ~]# vi /etc/hostname

[root@openeuler ~]# cat /etc/hostname

openEuler

[root@openeuler ~]# reboot

修改完成后重启系统并重新登录。

## openEuler内核编译与安装

安装工具，构建开发环境：

[root@openEuler ~]# **yum group install -y "Development Tools"**

[root@openEuler ~]# **yum install -y bc**

[root@openEuler ~]# **yum install -y openssl-devel**

备份boot目录以防后续步骤更新内核失败

[root@openEuler ~]# **tar czvf boot.origin.tgz /boot/**

保存当前内核版本信息

[root@openEuler ~]# **uname –r > uname\_r.log**

获取内核源代码并解压

[root@openEuler ~]# **wget https://gitee.com/openeuler/kernel/repository/archive/kernel-4.19.zip**

[root@openEuler ~]# **unzip kernel-4.19.zip**

编译内核

[root@openEuler ~]# **cd kernel**

[root@openEuler kernel]# **make openeuler\_defconfig**

在这里，我们按源代码文件kernel/arch/arm64/configs/openeuler\_defconfig的配置配置内核。

[root@openEuler kernel]# **make help | grep Image**

\* Image.gz - Compressed kernel image (arch/arm64/boot/Image.gz)

Image - Uncompressed kernel image (arch/arm64/boot/Image)

这一步查看了可编译的Image。

[root@openEuler kernel]# **make -j4 Image modules dtbs**

这一步是编译内核的Image、modules和dtbs。

安装内核

[root@openEuler kernel]# **make modules\_install**

**……**

INSTALL sound/soundcore.ko

DEPMOD 4.19.154

[root@openEuler kernel]# **make install**

/bin/sh ./arch/arm64/boot/install.sh 4.19.154 \

arch/arm64/boot/Image System.map "/boot"

dracut-install: Failed to find module 'xen-blkfront'

dracut: FAILED: /usr/lib/dracut/dracut-install -D /var/tmp/dracut.tlIdPu/initramfs --kerneldir /lib/modules/4.19.154/ -m virtio\_gpu xen-blkfront xen-netfront virtio\_blk virtio\_scsi virtio\_net virtio\_pci virtio\_ring virtio

注意：在最后一步“make install”时出现的错误在这里可以忽略。

以VNC登录ECS



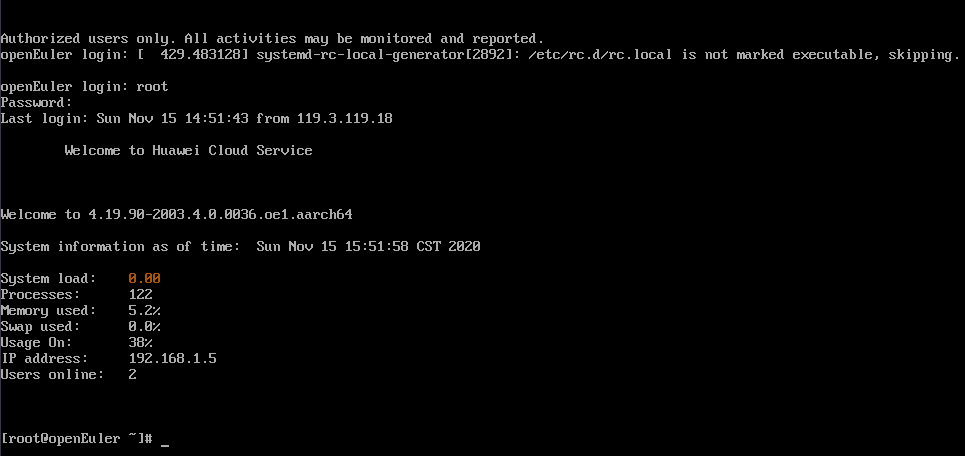
在控制台“弹性云服务器ECS”的页面中点击刚刚创建的虚拟机“openEuler”的名字超链接，在新打开的页面中点击“远程登录”按钮：



然后以控制台提供的VNC方式登录：



与以ssh登录一样，以root身份登录：



大部分的时间，我们仅将此作为一个监视器使用。

重启系统

在ssh终端重启操作系统：

[root@openEuler kernel]# **reboot**

登录并验证

在VNC窗口中选择以新编译出来的内核启动系统：



在这里新编译出来的内核版本为4.19.154。您的子版本号可能与此不一样。

登录系统并查看版本

请以VNC和ssh终端登录系统，并在其中之一查看内核版本：

[root@openEuler ~]# **uname -r**

4.19.154

可以看出内核版本已更新。

## 云环境资源清理

### ECS关机

当所实验完成后，应该对ECS进行关机以节约经费（ECS关机后仍有少量扣费）。

回到ECS控制台，勾选openEuler虚拟机，进行关机。



在弹出的对话框中点击“是”按钮：



点击“是”按钮即可进行关机。

### 删除ECS

可以等到所有的内核实验完成后再删除ECS，否则每次都得重新编译内核。这里给出删除ECS的方法。

待关机完成后点击“更多”🡪“删除”



在弹出的对话框中勾选“释放云服务器绑定的弹性公网IP地址”和“删除云服务器挂载的数据盘”，然后点击“是”，删除ECS。



您可以在控制台点击“更多 | 资源 | 我的资源”菜单项，检查资源是否全部删除



注意：(1) 虚拟私有云VPC和安全组可以不删除，以留下次使用。(2) 若在除“华北-北京四”之外区域购买了ECS和EIP，请切换到那个区域查看。

# 步骤2 安装BPF所需环境与依赖项

## 2.1任务描述

在华为云openeuler操作系统上，完成eBPF需要的依赖项与环境的安装。

### 2.2.1安装依赖项

[root@openEuler ~]yum install libncurses\*

[root@openEuler ~]yum install flex

[root@openEuler ~]yum install bison

[root@openEuler ~]yum install libelf\*

[root@openEuler ~]yum install binutils-dev\*

[root@openEuler ~]yum install libssl\*

[root@openEuler ~]yum install elfutils-libelf-devel

### 2.2.2安装其他环境

执行下列命令

[root@openEuler ~]`yum install llvm`

[root@openEuler ~]`yum install clang`

在开始编译工作之前，确保实验环境的版本满足以下条件：

clang>=version 3.4.0\*\*

llvm >=version 3.7.1\*\*

LLVM是当前唯一提供BPF后端的编译套件。gcc当前不支持BPF。LLVM提 供了一个BPF后端，这样就可以使用像clang这样的工具将C编译成BPF对象文件，然后可以将其加载到内核中。BPF与Linux内核紧密相连，允许在不牺牲本机内核性能的情况下进行完全的编程。典型的工作流程是，BPF程序是用C语言编写的，由LLVM编译成对象/ ELF文件，由用户空间BPF ELF加载器(如iproute2或其他)解析，并通过BPF系统调用推入内核。内核验证BPF指令并JITs指令，为程序返回一个新的文件描述符，然后可以把文件描述符附加到一个子系统(例如网络)。如果支持，子系统可以进一步将BPF程序卸载(offload)到硬件(例如NIC)。

- Clang是一个C++编写、基于LLVM、发布于LLVM BSD许可证下的C/C++/Objective-C/Objective-C++编译器。它与GNU C语言规范几乎完全兼容（当然，也有部分不兼容的内容，包括编译命令选项也会有点差异），并在此基础上增加了额外的语法特性，比如C函数重载（通过\_\_attribute\_\_((overloadable))来修饰函数），其目标（之一）就是超越GCC。Clang将支持其普通lambda表达式、返回类型的简化处理以及更好的处理constexpr关键字。

### 2.2.3编译示例代码

使用make命令编译samples/bpf/目录下的所有bpf示例代码：

cd kernel

make samples/bpf/

或者make M=samples/bpf

Make-C../..//root/kernel/samples/bpf/ BPF\_SAMPLES\_PATH=/root/kernel/samples/bpf

make[1]: Entering directory '/root/kernel'

……

HOSTLD /root/kernel/samples/bpf/task\_fd\_query

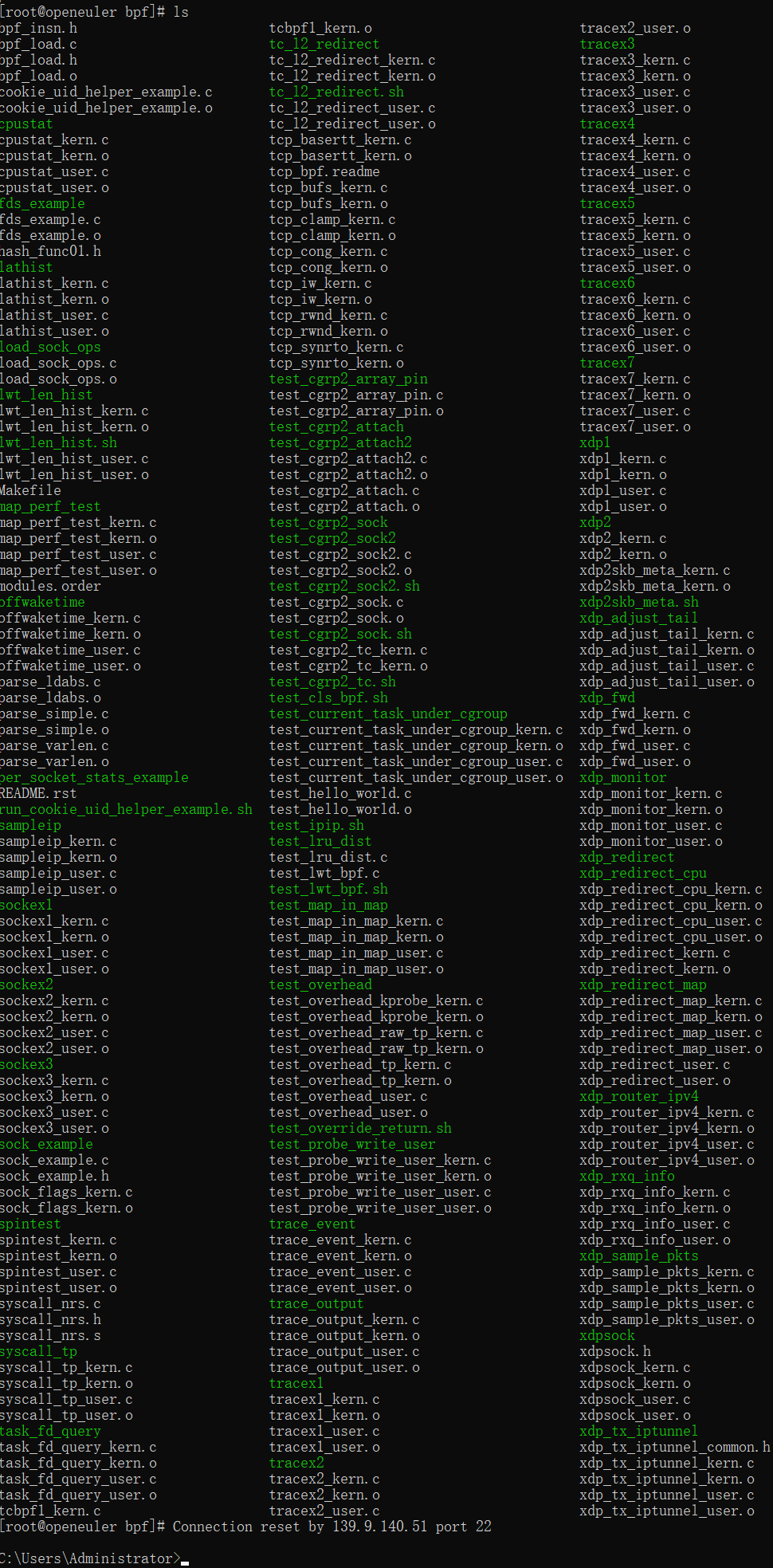
cat: /.build.command: No such file or directory

HOSTLD /root/kernel/samples/bpf/xdp\_sample\_pkts

cat: /.build.command: No such file or directory

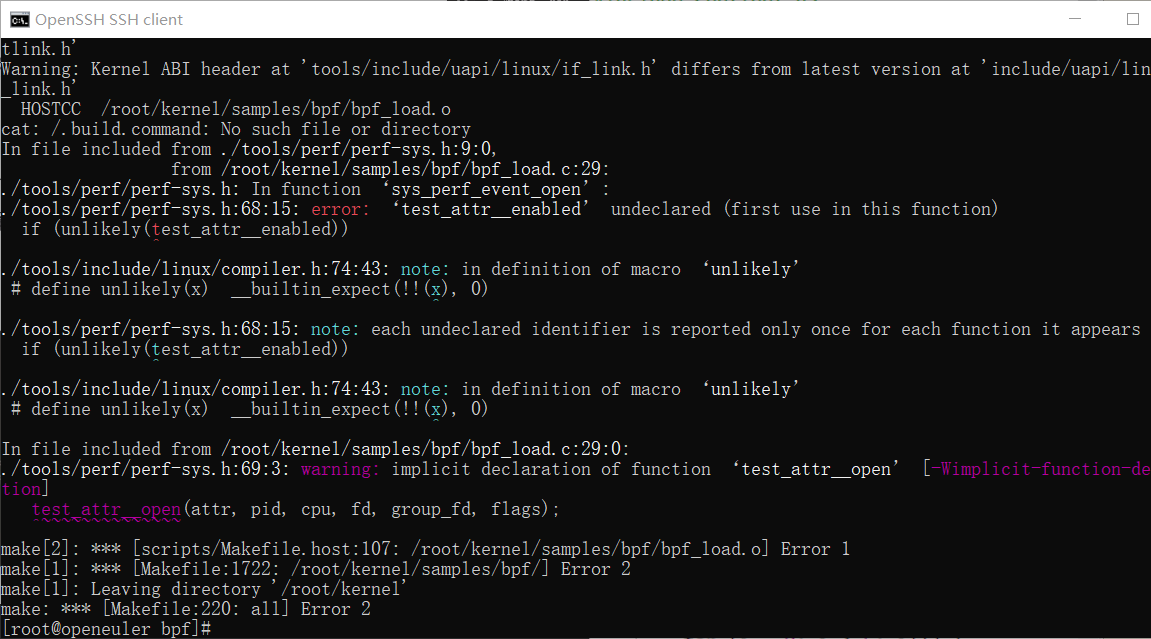
make[1]: Leaving directory '/root/kernel'

生成如下所示的文件，有.o后缀的目标文件以及绿色高亮的可执行文件（kernel/samples/bpf/）：



注：

（1）执行make命令时，可能会有以下错误信息：



根据错误提示，发生错误的文件为./tools/perf/perf-sys.h，可按下面的代码修改perf-sys.h，再重新运行make命令。

修改方法可参考：https://www.runoob.com/linux/linux-vim.html

/\* SPDX-License-Identifier: GPL-2.0 \*/

#ifndef \_PERF\_SYS\_H

#define \_PERF\_SYS\_H

#include <unistd.h>

#include <sys/types.h>

#include <sys/syscall.h>

#include <linux/types.h>

#include <linux/compiler.h>

#include <linux/perf\_event.h>

#include <asm/barrier.h>

#ifdef \_\_powerpc\_\_

#define CPUINFO\_PROC {"cpu"}

#endif

#ifdef \_\_s390\_\_

#define CPUINFO\_PROC {"vendor\_id"}

#endif

#ifdef \_\_sh\_\_

#define CPUINFO\_PROC {"cpu type"}

#endif

#ifdef \_\_hppa\_\_

#define CPUINFO\_PROC {"cpu"}

#endif

#ifdef \_\_sparc\_\_

#define CPUINFO\_PROC {"cpu"}

#endif

#ifdef \_\_alpha\_\_

#define CPUINFO\_PROC {"cpu model"}

#endif

#ifdef \_\_arm\_\_

#define CPUINFO\_PROC {"model name", "Processor"}

#endif

#ifdef \_\_mips\_\_

#define CPUINFO\_PROC {"cpu model"}

#endif

#ifdef \_\_arc\_\_

#define CPUINFO\_PROC {"Processor"}

#endif

#ifdef \_\_metag\_\_

#define CPUINFO\_PROC {"CPU"}

#endif

#ifdef \_\_xtensa\_\_

#define CPUINFO\_PROC {"core ID"}

#endif

#ifndef CPUINFO\_PROC

#define CPUINFO\_PROC { "model name", }

#endif

#ifndef HAVE\_ATTR\_TEST

#define HAVE\_ATTR\_TEST 0

#endif

static inline int

sys\_perf\_event\_open(struct perf\_event\_attr \*attr,

pid\_t pid, int cpu, int group\_fd,

unsigned long flags)

{

int fd;

fd = syscall(\_\_NR\_perf\_event\_open, attr, pid, cpu,

group\_fd, flags);

#if HAVE\_ATTR\_TEST

if (unlikely(test\_attr\_\_enabled))

test\_attr\_\_open(attr, pid, cpu, fd, group\_fd, flags);

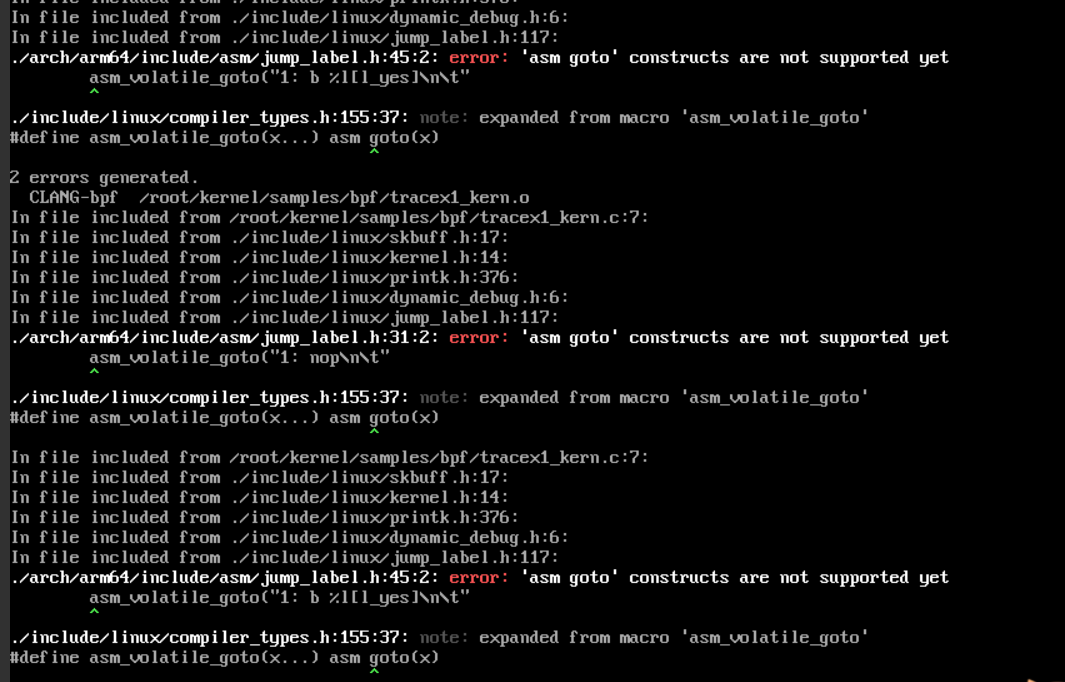
#endif

return fd;

}

#endif /\* \_PERF\_SYS\_H \*/

（2）若有如下报错信息：



可在出错的文件中（./arch/arm64/include/asm/jump\_label.h）按如下方式修改相关代码：

static \_\_always\_inline bool arch\_static\_branch(struct static\_key \*key, bool branch)

{

#if 0

asm\_volatile\_goto("1: nop\n\t"

".pushsection \_\_jump\_table, \"aw\"\n\t"

".align 3\n\t"

".quad 1b, %l[l\_yes], %c0\n\t"

".popsection\n\t"

: : "i"(&((char \*)key)[branch]) : : l\_yes);

return false;

l\_yes:

#endif

return true;

}

static \_\_always\_inline bool arch\_static\_branch\_jump(struct static\_key \*key, bool branch)

{

#if 0

asm\_volatile\_goto("1: b %l[l\_yes]\n\t"

".pushsection \_\_jump\_table, \"aw\"\n\t"

".align 3\n\t"

".quad 1b, %l[l\_yes], %c0\n\t"

".popsection\n\t"

: : "i"(&((char \*)key)[branch]) : : l\_yes);

return false;

l\_yes:

#endif

return true;

}

# 步骤3看懂Makefile文件

## 3.1任务描述

仔细阅读分析给定的Makefile，并写出具体规则。

## 3.2具体要求

使用cgroup实现限制CPU核数。编译一个内核文件的构建规则都写在Makefile文件里面，要学会如何Make命令，就必须学会如何编写Makefile文件。请分析samples/bpf/Makefile文件。（注：若对make不熟悉，可以查看教程http://www.ruanyifeng.com/blog/2015/02/make.html）

对Makefile文件语句功能的解释需要写在实验报告上

# 步骤4执行PBF示例程序

## 4.1任务描述

运行指定的BPF代码，观测结果

## 4.2具体过程

XDP全称为eXpress Data Path，是Linux内核网络栈的最底层。它只存在于RX路径上，允许在网络设备驱动内部网络堆栈中数据来源最早的地方进行数据包处理，在特定模式下可以在操作系统分配内存（skb）之前就已经完成处理。

### 4.2.1查看文件，读懂其功能

[root@openeuler bpf]# cat xdp1\_kern.c

该文件主要功能是从数据报文中解析并判断协议类型，统计数据报文中不同协议的数量分布情况。

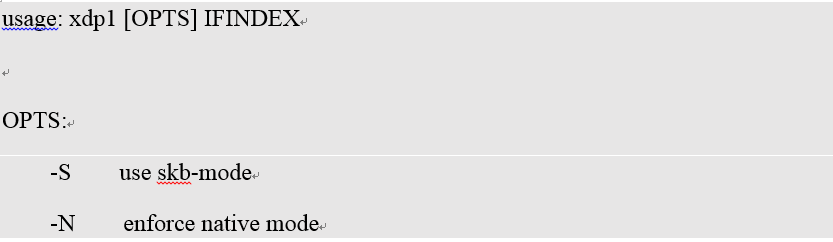
[root@openeuler bpf]# cat xdp1\_user.c

该文件的主要功能是载入 eBPF 程序到内核中，并从内核共享的 map 中获得统计信息并展示。

### 4.2.2执行可执行文件

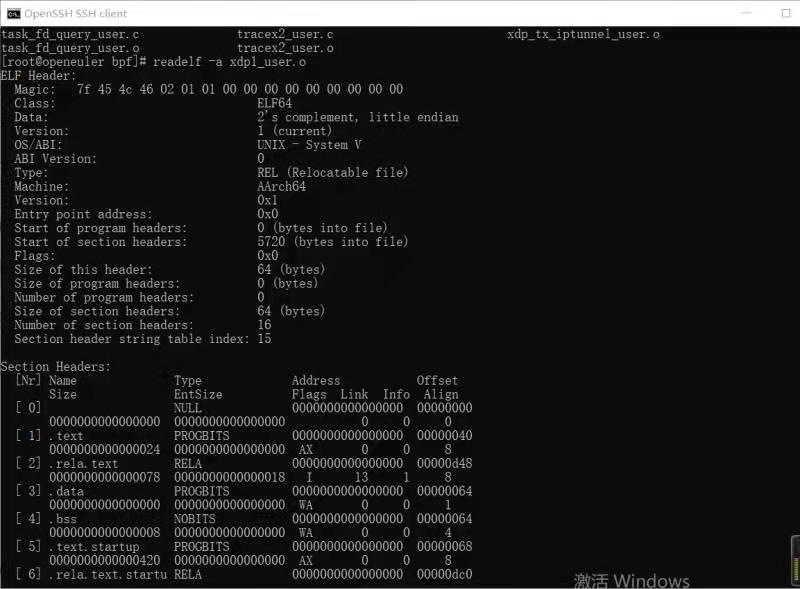
**在 Linux 源码仓库根目录执行 make samples/bpf/，编译完成后会在 samples/bpf 目录下会得到用户态可执行文件 xdp1 和 eBPF 字节码文件 xdp1\_kern.o，直接在当前目录直接执行即可。**

[root@openeuler bpf]# ./xdp1



### 4.2.3查看生成的elf格式的可执行文件

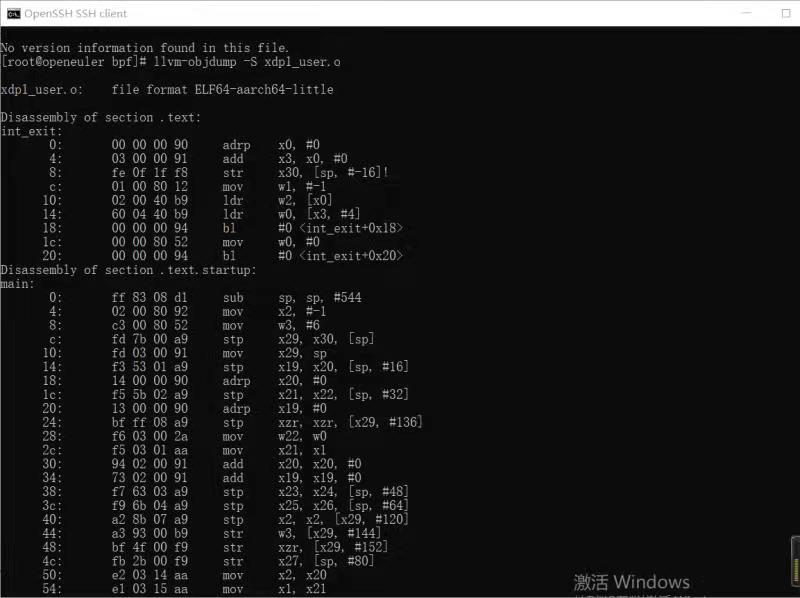
[root@openeuler bpf]# readelf -a xdp1\_user.o



### 4.2.4执行llvm-objdump命令

可以通过llvm-objdump这个工具来分析下这个可执行文件的反汇编指令信息。llvm-objdump命令会在命令行输出目标文件中的内容和最终链接视图

[root@openeuler bpf]#llvm-objdump -S xdp1\_user.o



完成以上功能并将截图写入实验报告！（cat命令执行结果无需截图）

# 步骤5编写自己的BPF程序

## 5.1任务描述

编写一个自定义简单BPF源文件，并编译执行输出结果

## 5.2具体要求

**第1步** 在samples/bpf目录下新建两个文件，hello\_kern.c和hello\_user.c，分别是内核空间程序和用户空间程序。

Hello\_kern.c示例如下：

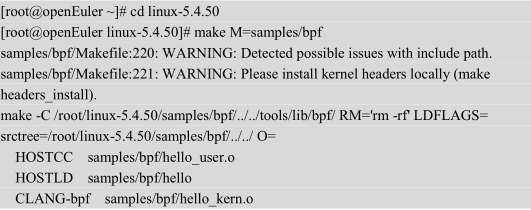
Hello\_user.c示例如下：



**第2步** 编写Makefile文件

（需要自己编写）

,**第3步**编译运行自己的BPF应用程序

结果输出参考如下：



按照以上步骤实现自己BPF的执行过程，

将过程及结果截图写入实验报告。