

网络安全实验报告

题 目 基于 libnet 的程序设计

专 业 视听觉信息处理 .

学 号 7203610121 .

学 生 刘天瑞 .

指 导 教 师 王彦 .

# 一、实验目的

掌握 libnet 数据包的构造原理。

**二、实验内容**

1. 掌握 libnet 数据包的构造原理
2. 编程实现基于 libnet 的数据包构造，结合前面实验给出验证过程。能够对源码进行解释。

# 三、实验过程

**基于 libnet 的数据包构造实验基本信息：**

实验环境：Ubuntu20.04 x64

编程语言：C 语言

## 1. 需求分析

需要使用 libnet 构造并发送一个数据包，并验证这个数据包被成功发送了。验证这一过程需要用到实验二中的捕包程序 pcap，将生成的数据包从虚拟机 B 发送到虚拟机 A，虚拟机 A 中的捕包程序会自动将其捕获，通过检查各项信息，证明捕获的数据包就是从虚拟机 B 此程序 createPac 中发送的数据包。

追加：在虚拟机 A 中编写接收来自相应端口 udp 数据包的程序 recvUDP，验证该数据包确实可以被正确接收。

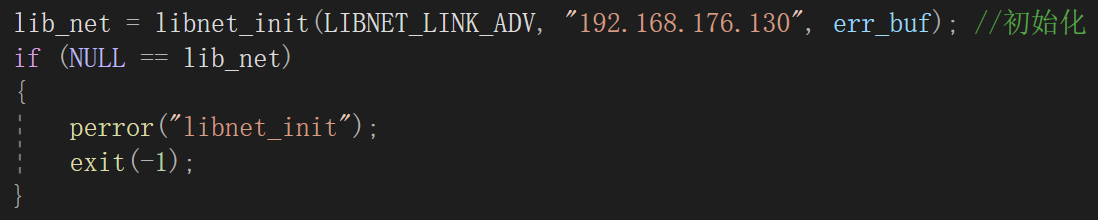
## 2. 程序结构

利用libnet函数库开发应用程序的基本步骤：

1. 数据包内存初始化

调用libnet\_init()函数，选择构造数据包的类型和发送该数据包的网络接口，获得libnet句柄。

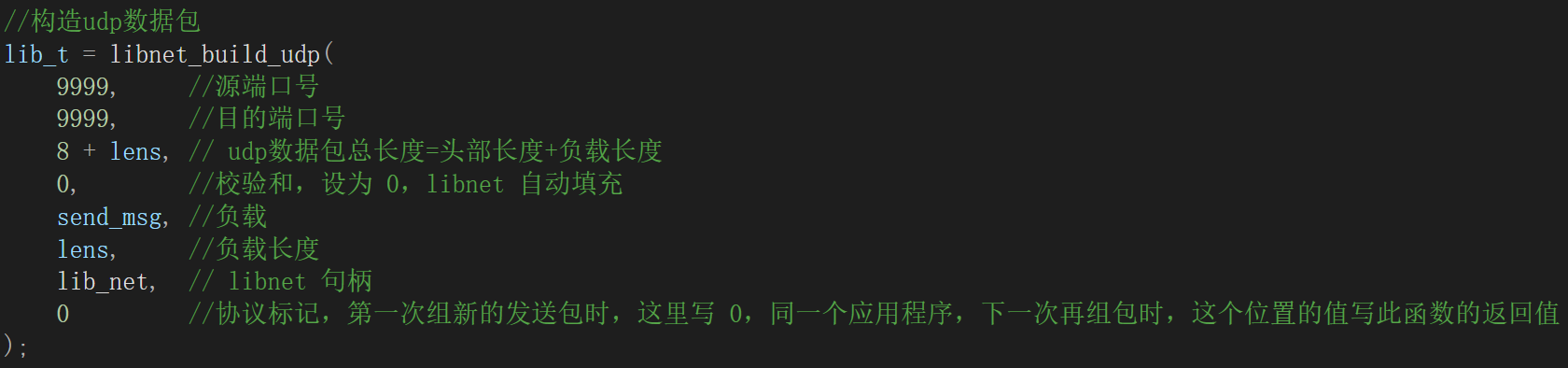
这里选择IP地址为克隆虚拟机“192.168.176.130”的网络接口发送，具体部分运行代码如下图所示：



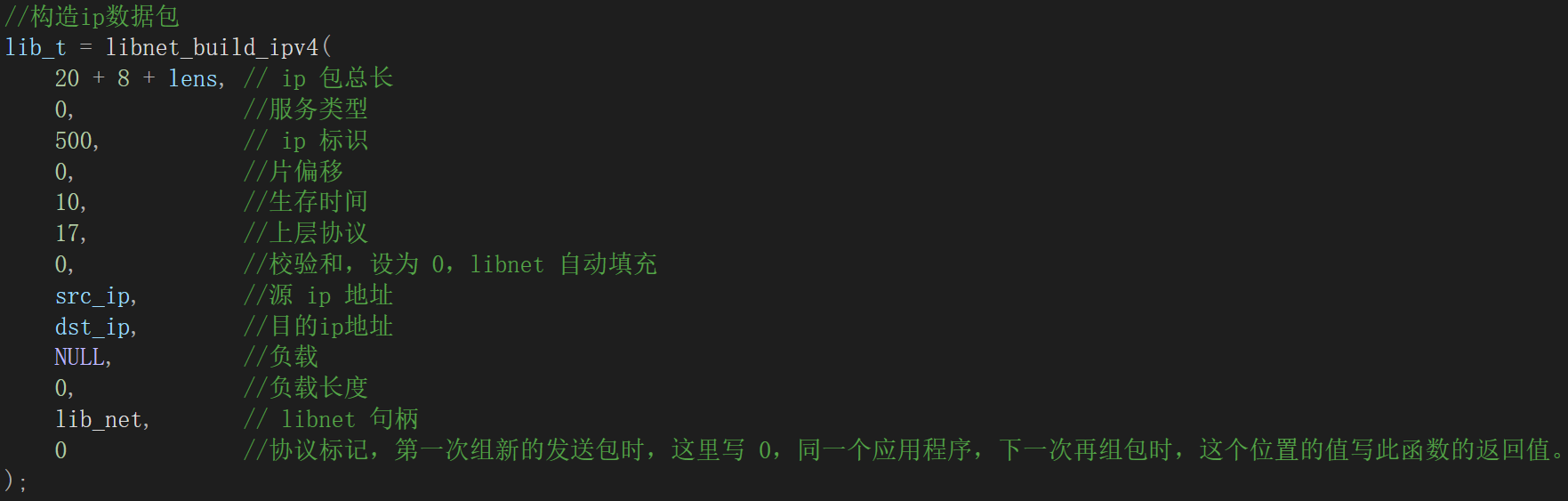
1. 构造数据包

分别调用函数构造各层数据包，从UDP数据报，到IP分组，最后再到以太网帧进行构建：分别先构建上一层的PDU，作为SDU传递给下一层，逐层封装；注意这里顺序不能改变。

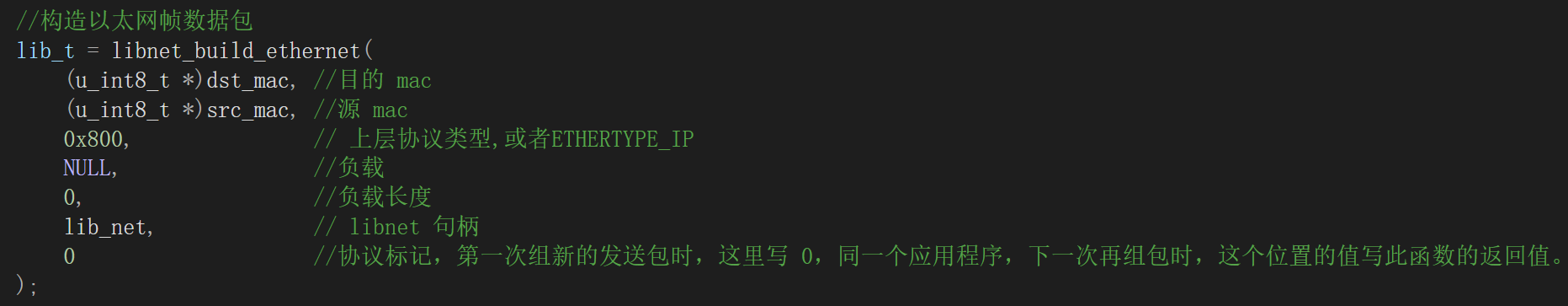
UDP数据报的部分运行代码如下图所示：



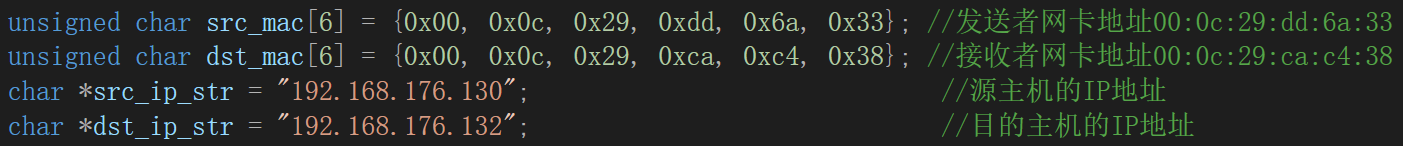
IP分组的部分运行代码如下图所示：

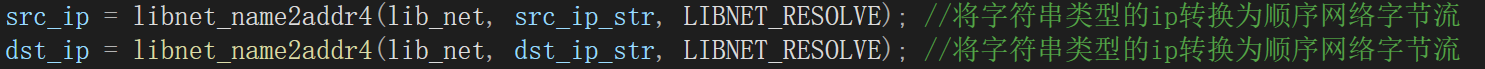


以太网帧的部分运行代码如下图所示：



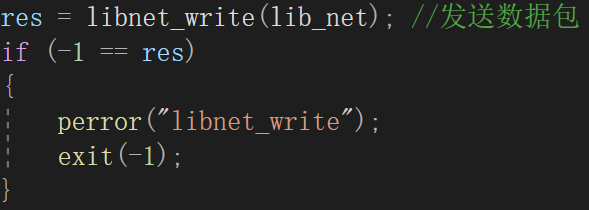
其中源端口和目的端口均选择为9999号，其中的MAC物理地址和IP地址均为实验一所做的两个虚拟机的MAC物理地址和IP地址（详情部分请见网络安全实验一）。





1. 发送数据

调用libnet\_write()函数来发送数据包，如下图所示：



1. 释放资源

调用libnet\_destroy()函数来释放并销毁资源，如下图所示：

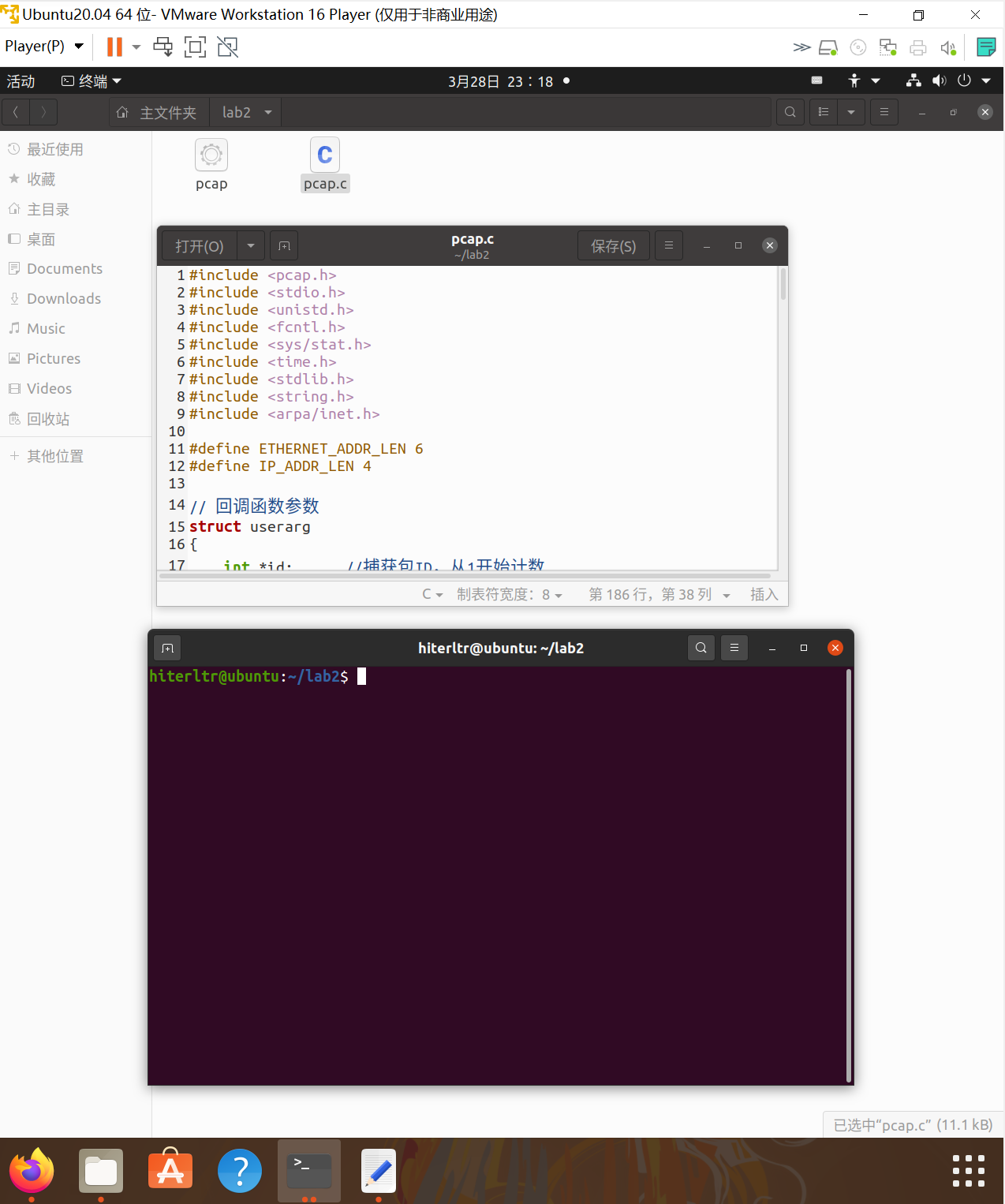


## 3. 进一步验证

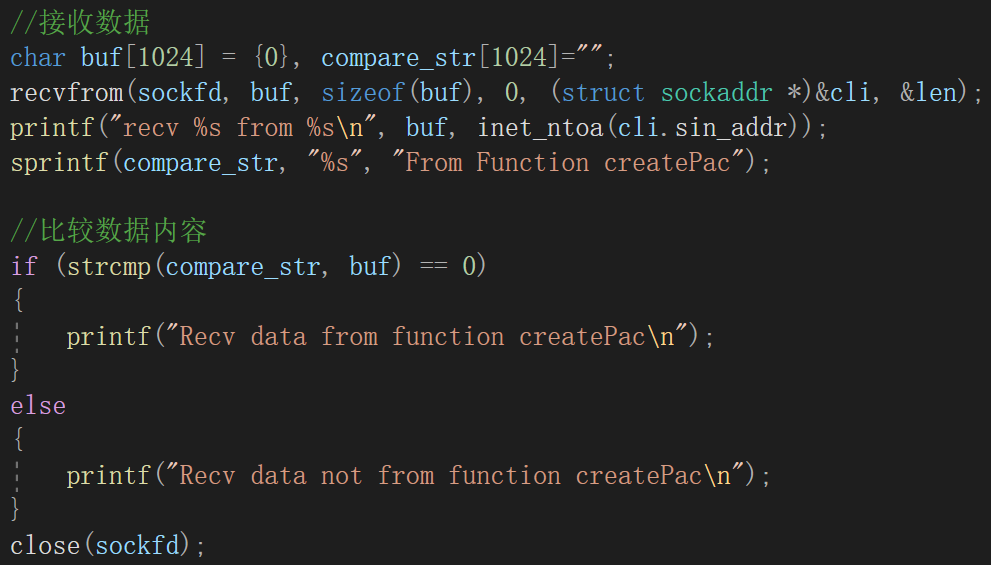
在验证是否发送成功阶段可以将udp类型数据包负载的数据内容设置为“From Function createPac”，从而方便验证udp数据包是否被发送成功同时也验证其是否来自createPac()函数。如下图所示：



此时便可以使用实验二中已经编写好的捕包程序：pcap.c，同时使用过滤规则“udp”（或者其他更为具体的过滤规则，例如”host 192.168.176.130”等等，但是由于我设置的网络端口非混合模式接收udp数据包的数量本身较少，所以这里只需过滤udp即可）捕获由另一个虚拟机构造并发送成功的数据包，然后查看其MAC物理地址以及IP地址是否对应正确。如下图所示：



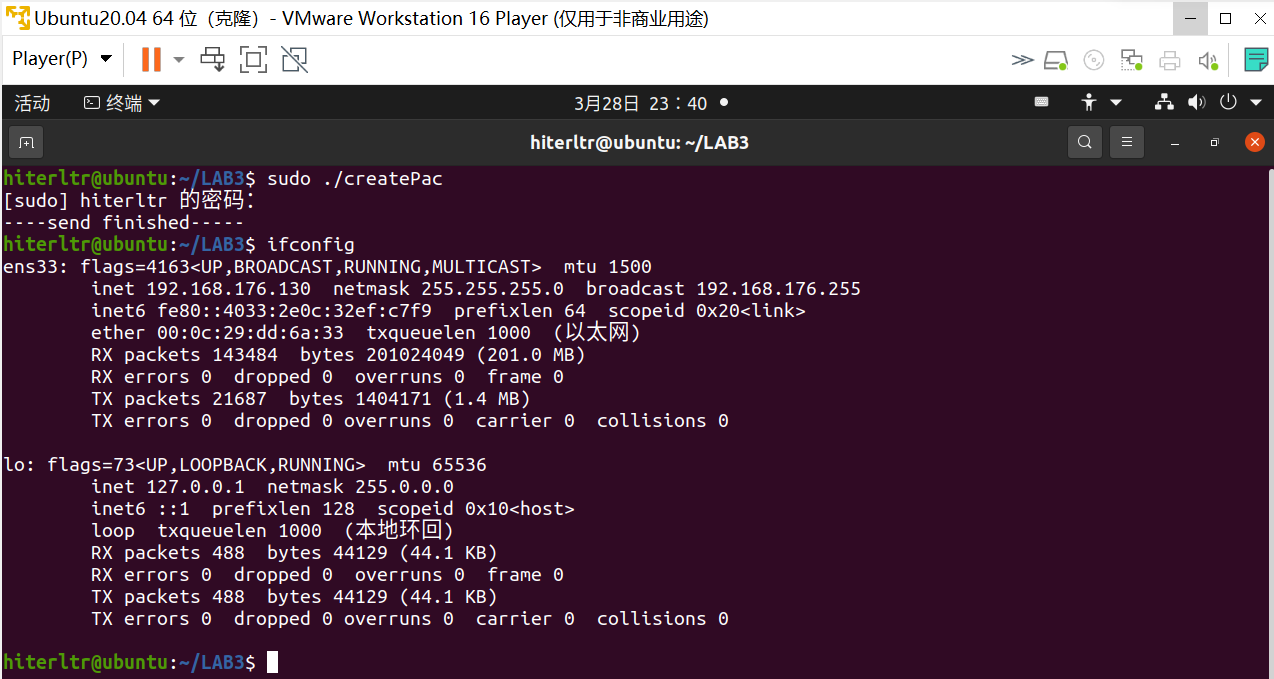
最后一部分便是接着编写UDP数据包的接收程序：recvUDP.c，此程序需运行在接收端的虚拟机上，能够查看是否接受到udp数据包，并且对比判断输出其数据内容是否为“From Function createPac”。如下图所示：



# 四、实验结果

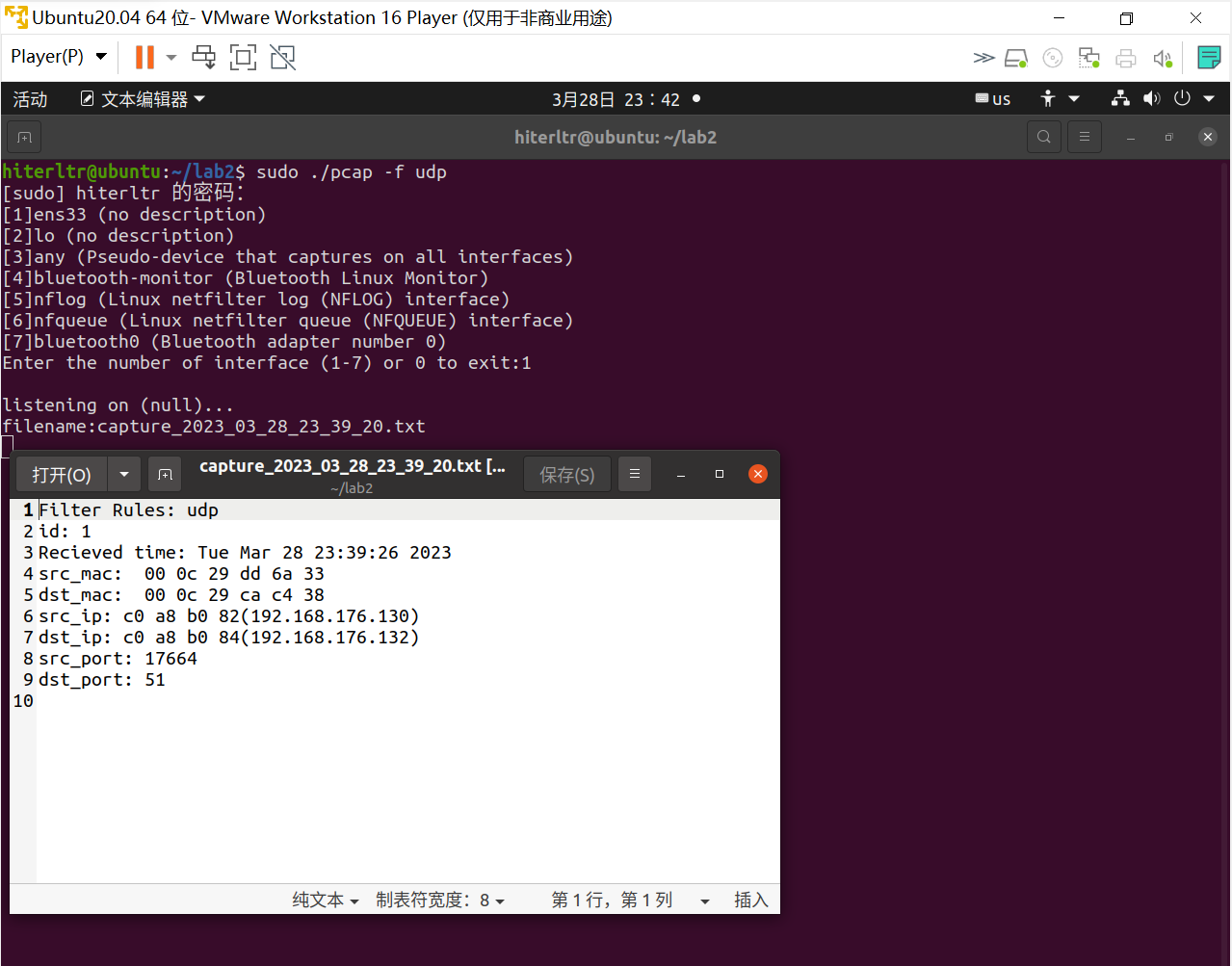
原虚拟机（IP地址为192.168.172.130）作为消息接收方，其分别调用程序recvUDP.c和pcap.c来接收构造好的udp数据包，而克隆虚拟机（IP地址为192.168.172.132）则作为消息发送方，其调用程序createPac来发送使用libnet构造的udp数据包（以下所展示的程序运行截图是原虚拟机（IP地址为192.168.172.130）上的程序代码，在克隆虚拟机（IP地址为192.168.172.132）上调用该程序代码时需要进行调换源和目的的MAC物理地址和IP地址等等操作，较为繁琐且没必要，故这里不再进行展示克隆虚拟机（IP地址为192.168.172.132）上的程序运行截图了）。

1.在克隆虚拟机（IP地址为192.168.172.132）上使用运行createPac.c程序代码来发送构造好的udp数据包（逻辑上是先在原虚拟机（IP地址为192.168.172.130）上分别调用recvUDP.c和pcap.c程序代码来阻塞并且等待克隆虚拟机（IP地址为192.168.172.132）发送udp数据包）。如下图所示：

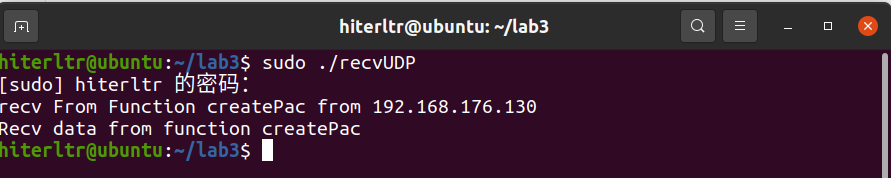


2.在原虚拟机（IP地址为192.168.172.130）上分别运行使用recvUDP.c和pcap.c程序代码来接收构造好的udp数据包。如下图所示：

1）pcap.c：



2）recvUDP.c：



# 五、心得体会 （出现问题分析）

做完本次实验后，我认为使用libnet来构造数据包时，其中的函数调用方法是较为简单的，但是在编写程序代码后进行验证时程序却总是出错。刚开始就出现了recvUDP.c程序运行起来显示能够被正确地接收，但是反而之前编写的pcap.c程序运行起来在日志中捕获不到数据包的奇怪情况。再后来我发现了自己是误把fprintf()函数认为成是行缓存，而使得程序持续进行捕获，并没有考虑到缓存导致其不能够进行实时地输出，于是最后由于udp数据包数量本身就过少，才导致pcap.c程序运行没有输出结果，还误认为没有捕获到。解决办法是将fprintf()函数的缓存大小手动设置为0，之后程序便能够实时地输出捕获到的数据包。除此之外，在编写recvUDP.c程序时，在创建网络通信对象部分我刚开始我忘记将sin\_addr.s\_addrMAC物理地址设置为默认的任意物理地址：INADDR\_ANY导致程序一直出错，这使得我更加注意赋值细节的把握。总体而言通过这次实验我掌握了 libnet 数据包的构造原理，可谓收获颇丰。