**网络技术与应用课程实验报告**

姓名：孟启轩

学号：2212452

专业：计算机科学与技术

**实验2：数据包捕获与分析**

一、实验内容：

（1）了解NPcap的架构。

（2）学习NPcap的设备列表获取方法、网卡设备打开方法，以及数据包捕获方法。

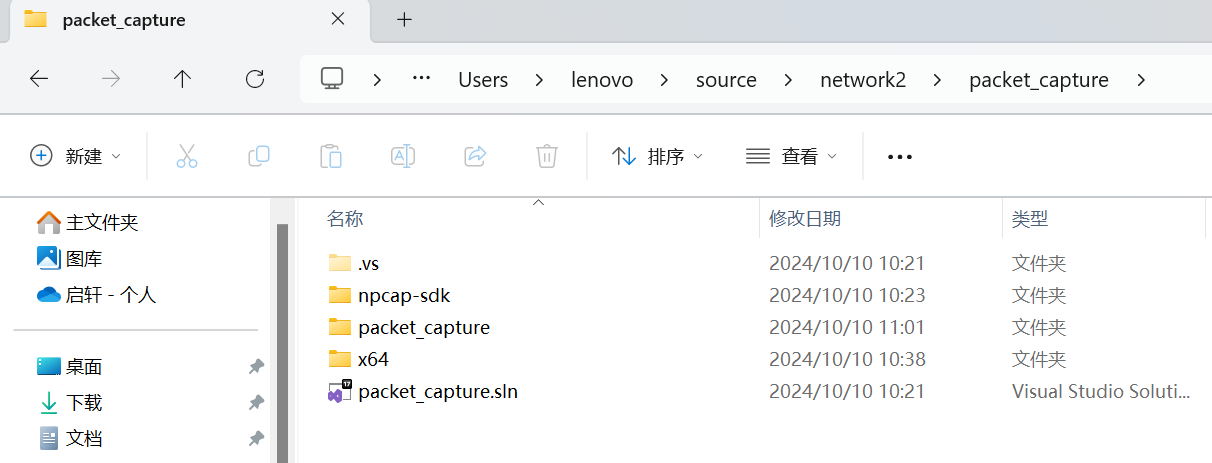
（3）通过NPcap编程，实现本机的数据包捕获，显示捕获数据帧的源MAC地址和目的MAC地址，以及类型/长度字段的值。

（4）捕获的数据包不要求硬盘存储，但应以简单明了的方式在屏幕上显示。必显字段包括源MAC地址、目的MAC地址和类型/长度字段的值。

（5）编写的程序应结构清晰，具有较好的可读性。

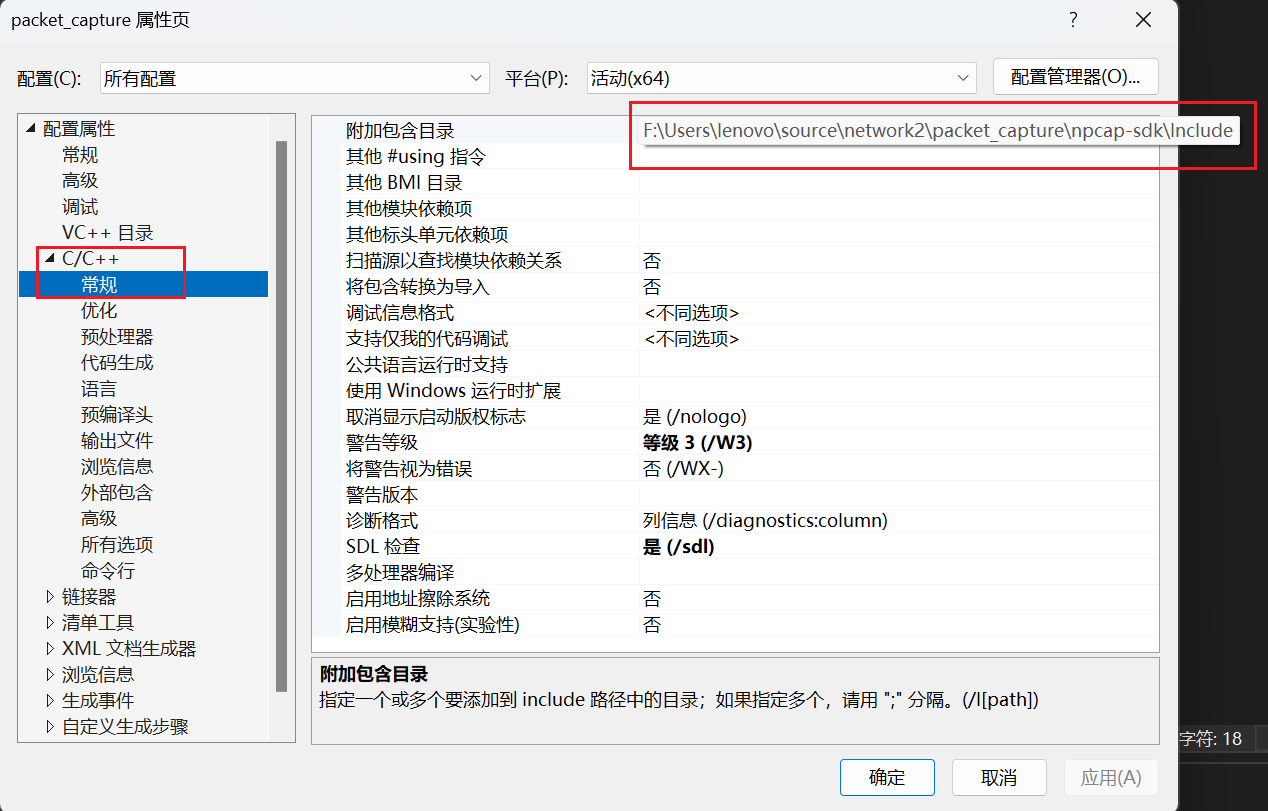
二、实验准备：

使用visual studio 2022进行编程实现，首先，到Npcap官网下载Npcap驱动程序和DLL程序以及sdk依赖，在visual studio中新建项目，并将npcap中的sdk移至项目目录下。

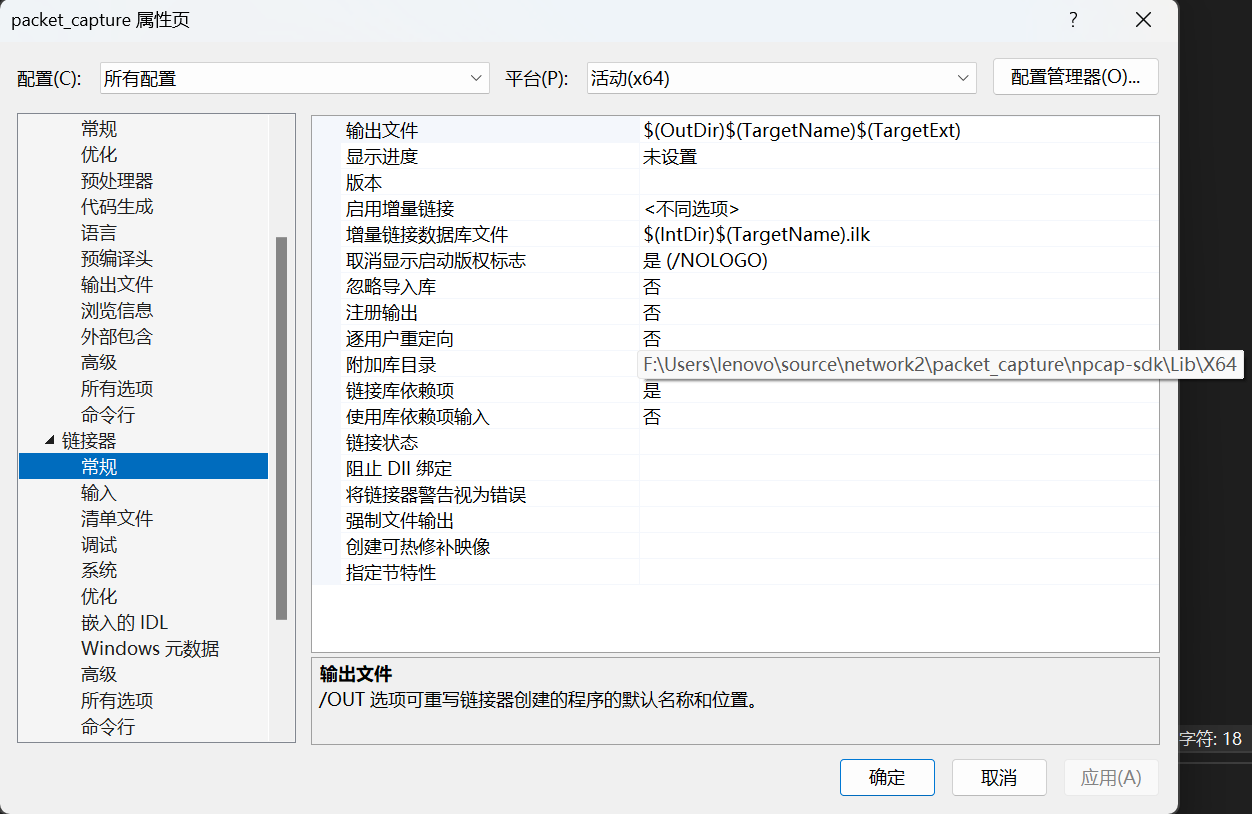


再进入项目文件中对npcap进行相关配置：

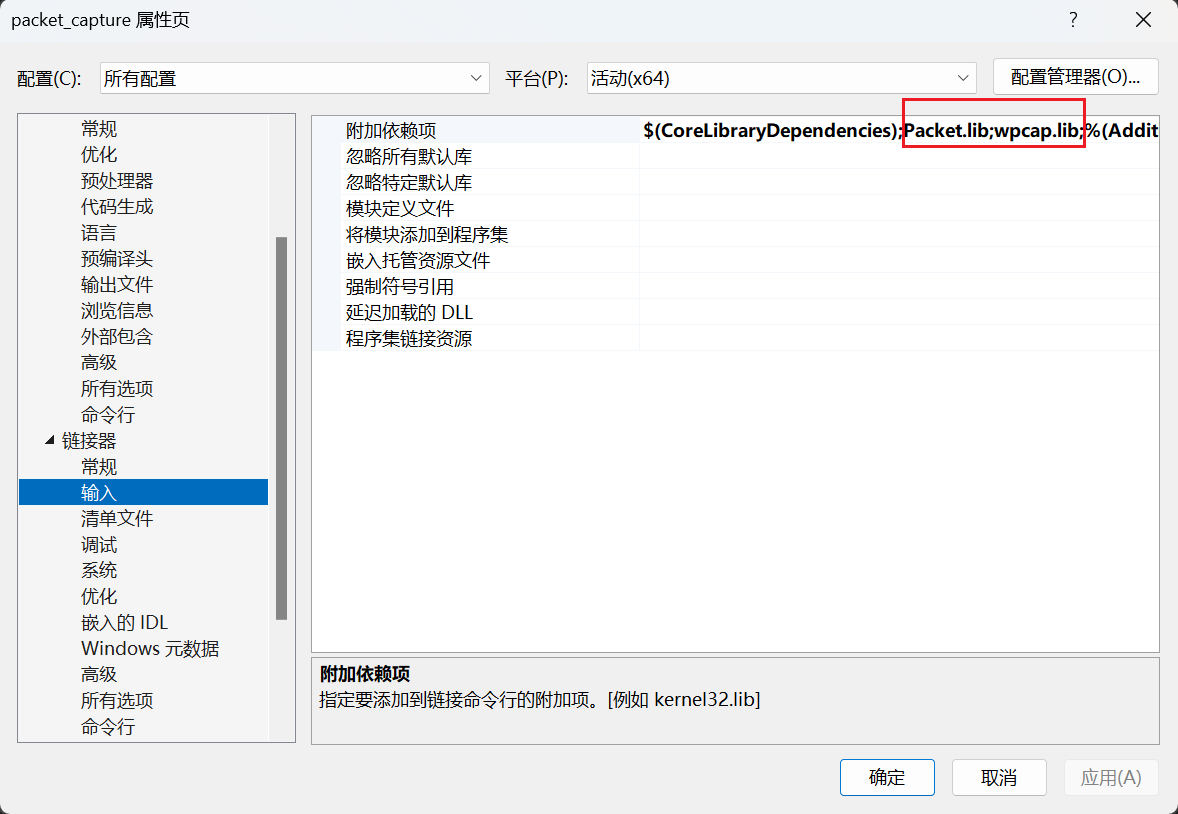
在C/C++属性的常规属性中，将include目录添加至附加包含目录中



在链接器的附加库目录中添加Lib目录（注意这里一定要添加Lib中的X64）



在输入的附加依赖项中添加Packet.lib和wpcap.lib



三、实验过程：

**（1）了解NPcap的架构。**

NPcap是Windows平台上的高性能网络数据包捕获库，它基于WinPcap项目的分支，专门为现代Windows系统设计，提供了增强的功能和改进的性能。NPcap的架构主要包括以下几个关键组件：

1.内核级驱动程序 (Kernel-Level Driver)

NPcap的核心是一个位于内核级别的驱动程序，它直接与Windows网络适配器进行交互。该驱动程序允许捕获通过网络适配器发送和接收的数据包，实现低延迟和高吞吐量的网络数据包捕获。此外，它还支持数据包注入功能，能够将用户自定义的数据包发送到网络上。内核级驱动的设计使得NPcap能够深入操作系统网络栈，提供准确和高效的流量捕获。

2.用户级API (User-Level API)

NPcap提供用户级别的API，方便开发者进行应用程序开发。开发者可以通过API配置捕获参数、启动或停止数据包捕获，以及访问和处理捕获的数据包。NPcap的API兼容WinPcap接口，支持多种编程语言（如C/C++和Python），使得已有的WinPcap应用程序可以较为轻松地迁移到NPcap。

3.数据包过滤与注入 (Packet Filtering and Injection)

NPcap支持使用BPF（Berkeley Packet Filter）语法定义数据包过滤规则，这些规则可以基于IP地址、端口、协议类型等条件进行配置，从而选择需要捕获的数据包类型。NPcap还支持数据包注入功能，允许开发者发送自定义的数据包到网络上，从而实现模拟网络通信、测试和调试等功能。

4.回环捕获 (Loopback Capture)

NPcap支持本地回环接口的捕获，可以捕获本地应用程序之间的通信流量。该功能在调试和分析本地服务、检测本地系统行为时非常有用。这一功能弥补了WinPcap不支持回环捕获的不足，使NPcap成为更全面的网络流量分析工具。

5.辅助库和工具 (Libraries and Utilities)

NPcap提供了一些辅助库和实用工具，帮助开发者方便地与NPcap进行交互。这些工具可以用于检测可用的网络适配器、配置适配器参数、分析和解码捕获的数据包等，提升开发效率。

6.与Wireshark的集成 (Integration with Wireshark)

NPcap常常被用于与Wireshark等网络协议分析工具相结合，以实现全面的网络数据分析解决方案。Wireshark能够与NPcap无缝集成，用户可以使用Wireshark通过NPcap捕获数据包，并对数据包进行深入的协议分析和可视化展示。这种组合使NPcap成为网络安全、故障排查和网络性能优化等领域的重要工具。

7.安全增强和性能改进

NPcap在WinPcap的基础上做了许多改进。它增加了安全性措施，如对驱动程序签名的严格要求、在捕获过程中默认禁用混杂模式等。此外，NPcap采用了更高效的内存管理机制和改进的驱动架构，显著提升了数据包捕获的性能和可靠性。

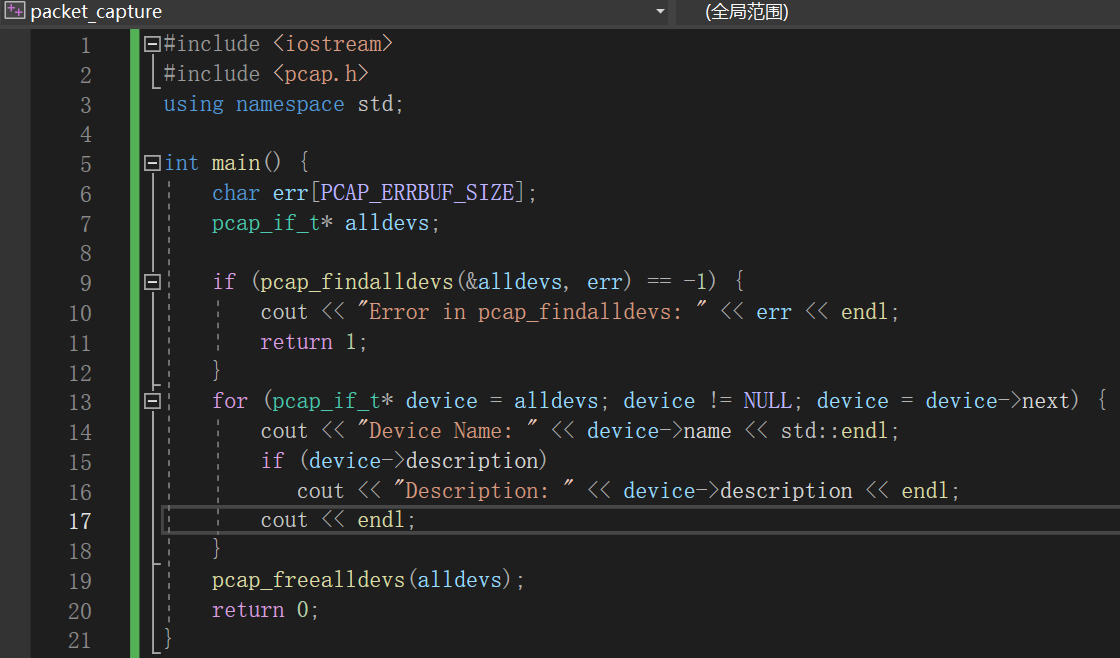
综上所述，NPcap通过内核级驱动程序、灵活的用户级API以及多种实用功能，如数据包过滤、注入和回环捕获等，为网络流量分析、网络调试和安全监测等任务提供了强大的工具支持。它的广泛应用不仅限于开发者，还包括网络安全专家、系统管理员和网络工程师等专业人员。

1. **学习NPcap的设备列表获取方法、网卡设备打开方法，以及数据包捕获方法。**

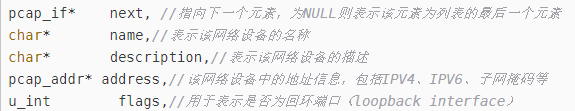
1.网络接口设备（网卡）列表获取方法

NpCap 提供了 pcap\_findalldevs\_ex 和 pcap\_findalldevs 函数来获取计算机上的网络接口设备的列表；此函数会为传入的 pcap\_if\_t 赋值——该类型是一个表示了设备列表的链表头；每一个这样的节点都包含了 name 和 description 域来描述设备。除此之外，pcap\_if\_t 结构体还包含了一个 pcap\_addr 结构体；后者包含了一个地址列表、一个掩码列表、一个广播地址列表和一个目的地址的列表；此外，pcap\_findalldevs\_ex 还能返回远程适配器信息和一个位于所给的本地文件夹的 pcap 文件列表。

编写代码获取设备列表：

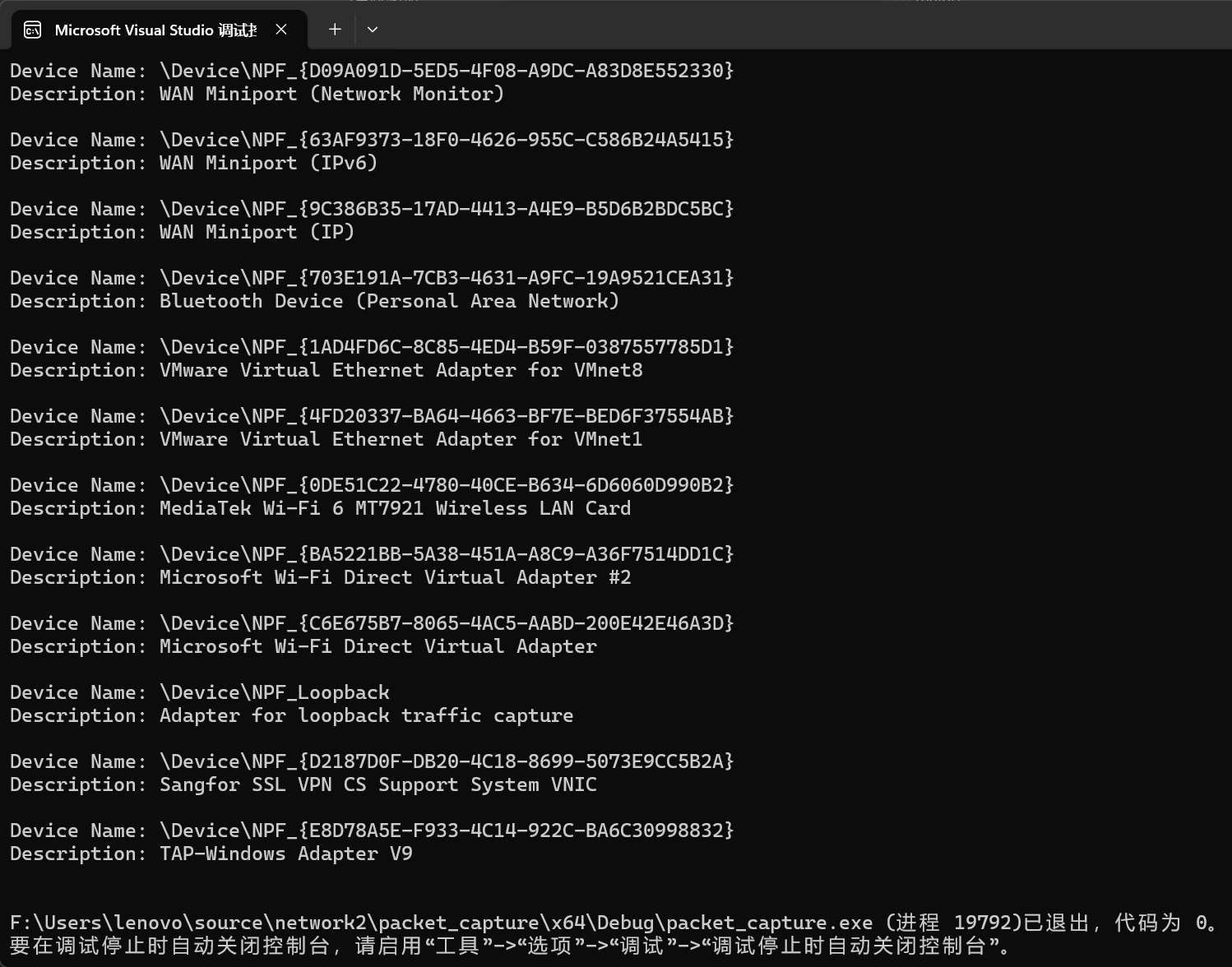


此程序的编写思路：定义一个err数组用来接受错误信息，定义一个指向pcap\_if\_t结构体的指针：



由于pcap\_findalldevs函数传入pcap\_if\_t\*类型的参数，再对alldevs进行取地址操作，若这个函数的返回值为-1，证明获取设备列表有误，结束程序。使用遍历输出alldevs中的数据，当device指向不为空，则证明有对应的pcap\_if\_t，进行输出，若device指向的description不为空，则输出description。

输出的网络接口设备（网卡）列表如下：



2.网卡设备打开方法

`pcap\_open\_live()` 是一个用于获取数据包捕获描述符的函数，开发者可以通过它来查看网络上的数据包。该函数返回一个 pcap\_t 类型的 handle。其主要参数和功能说明如下：

①device：这是一个字符串参数，用于指定要打开的网络设备。在Linux系统（内核版本2.2或更高）上，可以使用特殊的设备名称 `"any"` 或者 `NULL` 来捕获来自所有接口的数据包。

②snaplen：定义要捕获的最大字节数。如果捕获的数据包大小超过 `snaplen`，那么只会捕获数据包的前 `snaplen` 字节，并将其作为数据包数据提供。对于大多数网络，65535的值通常足以捕获数据包中所有可用的数据。

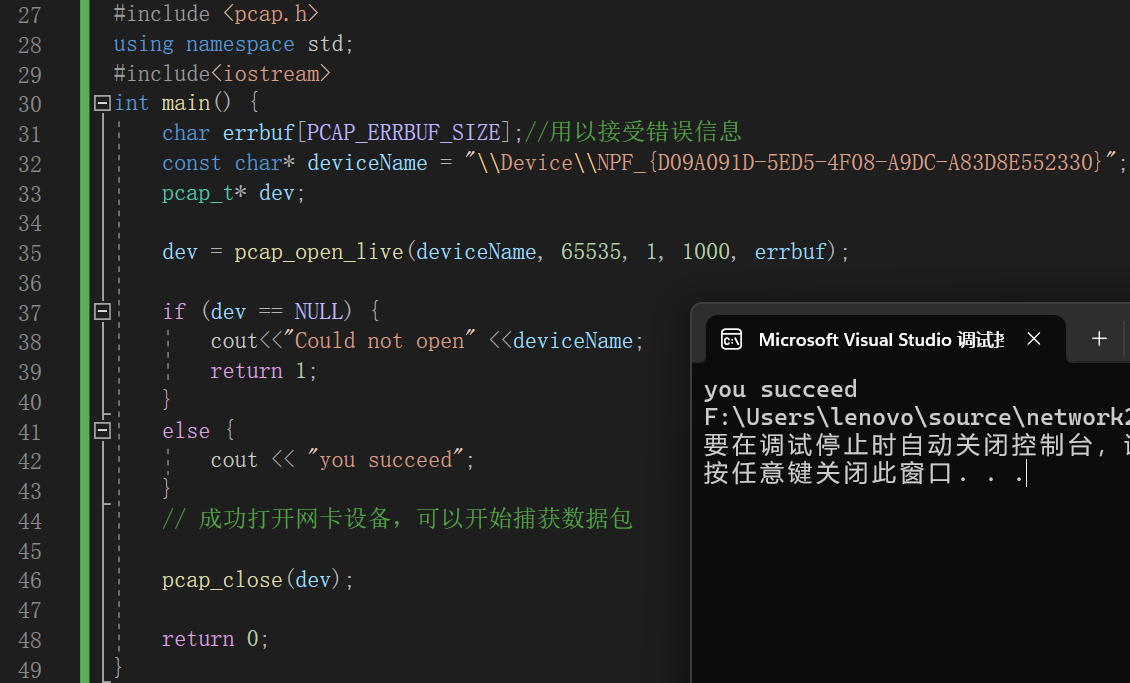
③promisc：用于指定是否将网络接口置于混杂模式。如果设置为 `true`，接口将进入混杂模式，从而捕获经过网络的所有数据包，而不仅仅是发送到该接口的包。请注意，即使设置为 `false`，网络接口可能由于其他原因处于混杂模式。对于设备 `"any"` 或 `NULL`，该参数会被忽略。

④to\_ms：指定读取操作的超时时间（以毫秒为单位）。读取超时允许在捕获到数据包后稍作等待，以便获取更多数据包，从而减少频繁的读取操作。这样可以在单次操作中从操作系统内核读取多个数据包，提高捕获效率。

⑤errbuf：用于存储错误信息字符串的缓冲区

通过这些参数，`pcap\_open\_live()` 提供了一种灵活的方式来设置数据包捕获行为，适用于各种网络流量分析和监控需求。

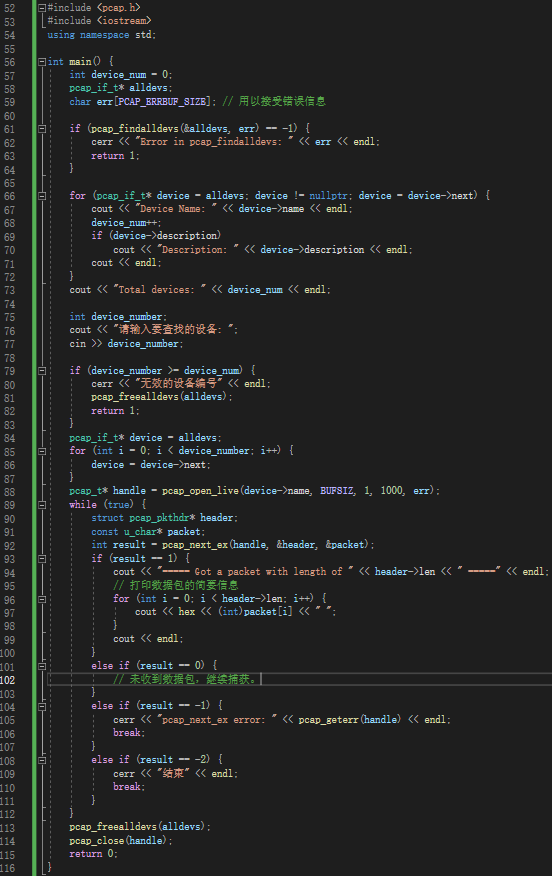
进行编程实现与验证：



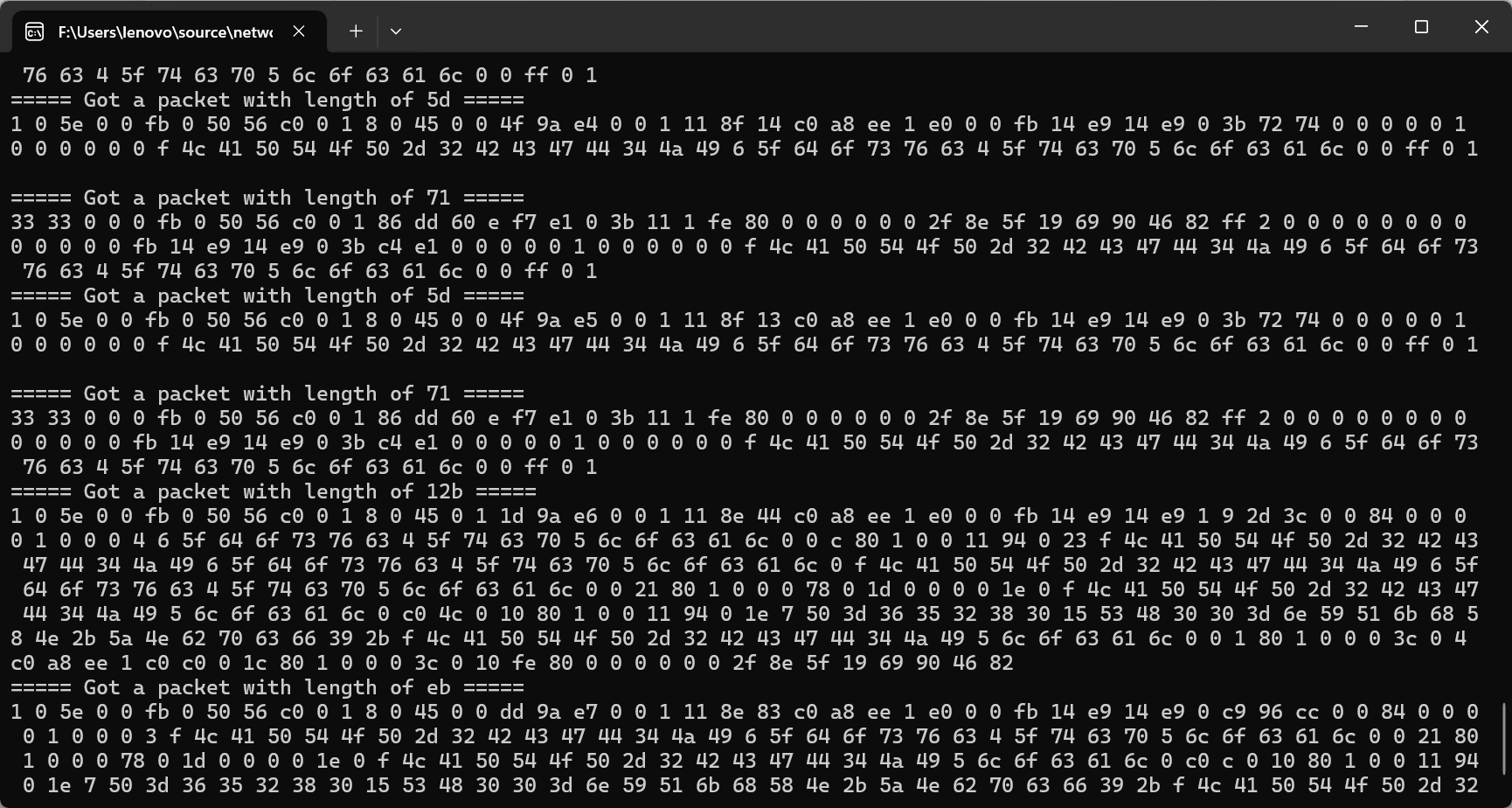
3.数据包捕获方法

在使用pcap\_open\_live打开网卡设备后，输入要查看的网卡编号，手动编写死循环，使用pcap\_next\_ex捕获数据包，捕获成功时对数据包进行打印，捕获失败或捕获结束时结束循环。

完成捕获后释放网卡设备并关闭pcap会话句柄。



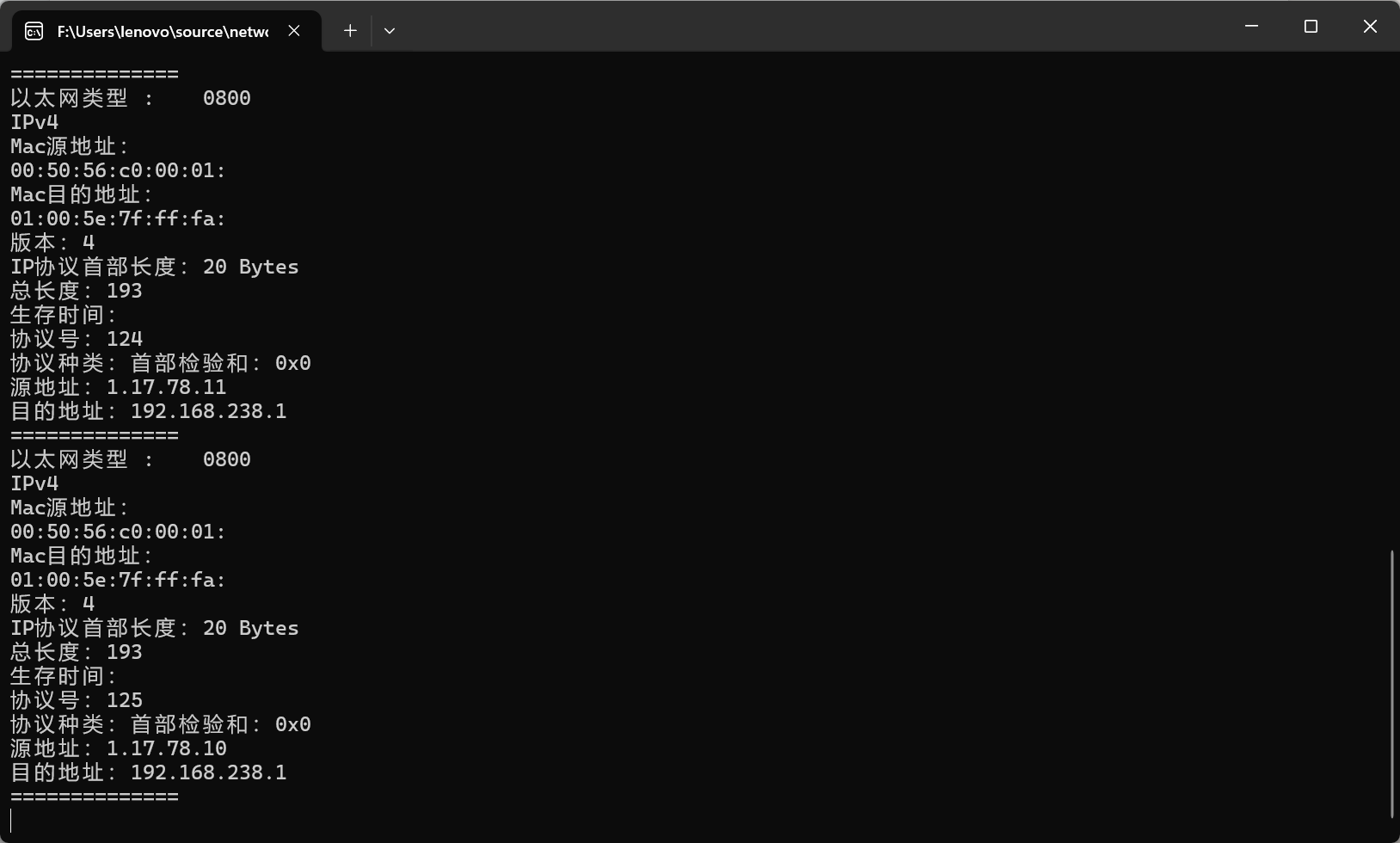
捕获的数据包如下：



1. **通过NPcap编程，实现本机的数据包捕获，显示捕获数据帧的源MAC地址和目的MAC地址，以及类型/长度字段的值。**
2. **捕获的数据报不要求硬盘存储，但应以简单明了的方式在屏幕上显示。必显字段包括源MAC地址、目的MAC地址和类型/长度字段的值。**

编写程序对已抓到的数据包进行分析，输出源MAC地址、目的MAC地址和类型/长度字段的值,总长度,生存周期,协议,头部校验码。代码见 **packet\_capture.cpp**

运行结果：



四、实验感悟：

通过本次NPcap数据包捕获与分析实验，我深入理解了NPcap的架构及其高效的数据包捕获机制，掌握了设备列表获取、网络接口打开和数据包捕获的基本流程，并通过编程实践提取了数据帧的关键信息。实验不仅强化了我对网络协议的理解，还培养了良好的编程习惯和代码规范意识，为后续的实验打下了坚实的基础。