实验 3：通过编程获取 IP 地址与 MAC 地址的对应关系

学号：2110049　　姓名：张刘明

Github 仓库地址：

https://github.com/newstarming/InternetTechnology.git
代码目录：coderun/work3.cpp
可执行文件路径：coderun/work3.exe

实验要求：

通过编程获取 IP 地址与 MAC 地址的对应关系实验，要求如下：

（1）在 IP 数据报捕获与分析编程实验的基础上，学习 NPcap 的数据包发送方法。

（2）通过 NPcap 编程，获取 IP 地址与 MAC 地址的映射关系。

（3）程序要具有输入 IP 地址，显示输入 IP 地址与获取的 MAC 地址对应关系界面。界面可以是命令行界面，也可以是图形界面，但应以简单明了的方式在屏幕上显示。

（4）编写的程序应结构清晰，具有较好的可读性。

前期准备（10），实验过程（40），　程序及规范性（25），　实验报告（25），　总分（100）。

实验过程：

代码实现流程：
1. 获取本机设备列表；

```
pcap_findalldevs_ex(PCAP_SRC_IF_STRING,    //获取本机的接口设备
        NULL,                 //无需认证
        &alldevs,             //指向设备列表首部
        errbuf                //出错信息保存缓存区
```

```
    )
```

2. 打开指定的网络接口；

```
adhandle = pcap_open(d->name,      //设备名
        65536,      //要捕获的数据包的部分
        PCAP_OPENFLAG_PROMISCUOUS,      //混杂模式
        1000,           //超时时间
        NULL,       //远程机器验证
        errbuf      //错误缓冲池
    );//打开网络接口设备
```

使用 pcap_open 函数来打开一个网络接口设备；
d->name 是要打开的设备名，根据输入的端口序号从 pcap_findalldevs 获取到的设备列表中选择。
65536 是要捕获的数据包的最大长度，这里设置为 65536 字节，即 64KB。
PCAP_OPENFLAG_PROMISCUOUS 表示以混杂模式打开设备，这种模式下可以捕获该接口上的所有数据包，而不仅仅是发往该接口的数据包。
1000 是超时时间，表示等待数据包到达的最长时间（以毫秒为单位）。
NULL 表示不进行远程机器验证。
errbuf 是一个错误缓冲区，用于存储错误信息，如果发生错误，将会把错误信息写入这个缓冲区。

3. 设置 APR 帧的内容；

APR 帧结构定义：

```
typedef struct ARPFrame_t {//IP 首部
    FrameHeader_t FrameHeader;//帧首部
    WORD HardwareType;//硬件类型
    WORD ProtocolType;//协议类型
    BYTE HLen;//硬件地址长度
    BYTE PLen;//协议地址长度
    WORD Operation;//操作类型
    BYTE SendHa[6];//发送方 MAC 地址
    DWORD SendIP;//发送方 IP 地址
    BYTE RecvHa[6];//接收方 MAC 地址
    DWORD RecvIP;//接收方 IP 地址
}ARPFrame_t;
```

APR 帧内容的初始化：

```
//将 APRFrame.FrameHeader.DesMAC 设置为广播地址
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.FrameHeader.DesMAC[i] = 0xff;
    }
```

```
    //将 APRFrame.FrameHeader.SrcMAC 设置为本机网卡的 MAC 地址
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.FrameHeader.SrcMAC[i] = 0x66;
    }
    ARPFrame.FrameHeader.FrameType = htons(0x0806);// 帧类型为 ARP
    ARPFrame.HardwareType = htons(0x0001);//硬件类型为以太网
    ARPFrame.ProtocolType = htons(0x0800);//协议类型为 IP
    ARPFrame.HLen = 6;//硬件地址长度为 6
    ARPFrame.PLen = 4;//协议地址长为 4
    ARPFrame.Operation = htons(0x0001);//操作为 ARP 请求
    //将 ARPFrame.SendHa 设置为本机网卡的 MAC 地址
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.SendHa[i] = 0x66;
    }
    //将 ARPFrame.SendIP 设置为本机网卡上绑定的 IP 地址
    ARPFrame.SendIP = inet_addr("112.112.112.112");
    //将 ARPFrame.RecvHa 设置为 0
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.RecvHa[i] = 0x00;//表示目的地址未知
    }
    //将 ARPFrame.RecvIP 设置为请求的 IP 地址
    ARPFrame.RecvIP = inet_addr(ip);
```

4. 发送 APR 帧;

```
pcap_sendpacket(adhandle, (u_char*)&ARPFrame, sizeof(ARPFrame_t))
```

pcap_sendpacket 函数用于发送封装好的数据包。参数含义如下:

adhandle: 表示打开的网络适配器的句柄。这个句柄在代码中通过 pcap_open 打开，代表了网络适配器的连接，允许对网络进行操作。

(u_char*)&ARPFrame: 表示待发送数据包的指针，指向 APRFrame 结构体，被强制转换为 u_char 类型。这里的 ARPFrame 是代码中自定义的结构体，包含了网络数据包的内容，确保 pcap_sendpacket 函数可以正确地发送这个数据包。

sizeof(ARPFrame_t): 表示要发送的数据包的大小，以字节为单位，返回结构体所占的字节数。

5. 捕获数据包并提取 MAC 地址;

```
while (1)
    {
```

```
        pcap_pkthdr* pkt_header1;   // 数据包头
        const u_char* pkt_data1;
        int rtnNew = pcap_next_ex(adhandle, &pkt_header1,
&pkt_data1);  // 捕获数据包
        // 如果捕获成功
        if (rtnNew == 1)
        {
            IPPacket1 = (ARPFrame_t*)pkt_data1; // 将捕获的数据包转换为
IPPacket 结构体
            if ((ntohs(IPPacket1->FrameHeader.FrameType) == 0x0806) &&
(ntohs(IPPacket1->Operation) == 0x0002)&&(IPPacket1->SendIP ==
inet_addr(ip1)))//如果帧类型为 ARP 并且操作为 ARP 应答
            {
                printf("Mac 地址: \n");
                printf("%02x:%02x:%02x:%02x:%02x:%02x\n",
                    IPPacket1->FrameHeader.SrcMAC[0],
                    IPPacket1->FrameHeader.SrcMAC[1],
                    IPPacket1->FrameHeader.SrcMAC[2],
                    IPPacket1->FrameHeader.SrcMAC[3],
                    IPPacket1->FrameHeader.SrcMAC[4],
                    IPPacket1->FrameHeader.SrcMAC[5]
                );
                break;
            }
        }
    }
```

6. 操作实例及解释

```
25:rpcap://\Device\NPF_{74E3646E-B10B-4BB9-9110-F7F7D101587E}
Network adapter 'Hyper-V Virtual Ethernet Adapter #2' on local host
Address Family Name:AF_INET     IP_Address:     172.29.64.1
Address Family Name:AF_INET6
------------------------------------------------------------------------------------------------------------
26:rpcap://\Device\NPF_{481523DD-B21C-47BB-8A57-1BAFA79915E5}
Network adapter 'Hyper-V Virtual Ethernet Adapter' on local host
Address Family Name:AF_INET     IP_Address:     172.20.224.1
Address Family Name:AF_INET6
------------------------------------------------------------------------------------------------------------
27:rpcap://\Device\NPF_Loopback
Network adapter 'Adapter for loopback traffic capture' on local host
------------------------------------------------------------------------------------------------------------
请输入要打开的网络接口号（1~27）：
14
监听：Network adapter 'VMware Virtual Ethernet Adapter for VMnet8' on local host
Mac地址：
00:50:56:c0:00:08
请输入IP：
192.168.106.137
发送成功
Mac地址：
00:0c:29:54:90:78
```

选择的网络接口号是本机的 VMnet8，其基本信息通过命令行使用指令 ipconfig /all 查看如下：

```
以太网适配器 VMware Network Adapter VMnet8:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : VMware Virtual Ethernet Adapter for VMnet8
   物理地址. . . . . . . . . . . . . : 00-50-56-C0-00-08
   DHCP 已启用 . . . . . . . . . . . : 否
   自动配置已启用. . . . . . . . . . : 是
   本地链接 IPv6 地址. . . . . . . . : fe80::e7e5:c92f:5b7b:c5c6%27(首选)
   IPv4 地址 . . . . . . . . . . . . : 192.168.106.1(首选)
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . :
   DHCPv6 IAID . . . . . . . . . . . : 771772502
   DHCPv6 客户端 DUID  . . . . . . . : 00-01-00-01-2B-E4-60-B2-DC-71-96-2B-61-CD
   TCPIP 上的 NetBIOS  . . . . . . . : 已启用
```

对比发现，获取的 mac 地址正确；

同样在虚拟机中使用指令 ifconfig 查看 ip 为 192.168.106.137 的网卡的 mac 地址如下：

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.106.137  netmask 255.255.255.0  broadcast 192.168.106.255
        inet6 fe80::563e:f90c:e209:f120  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:54:90:78  txqueuelen 1000  (Ethernet)
        RX packets 268876  bytes 310284231 (310.2 MB)
        RX errors 1639  dropped 54  overruns 0  frame 0
        TX packets 153773  bytes 31595735 (31.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000
```

对比发现，获取虚拟机目的 ip 的 mac 地址成功；