网络技术与应用课程实验报告

实验名称：编程获取 IP 地址与 MAC 地址的对应关系

学号：2211489 姓名：冯佳明 专业：物联网工程

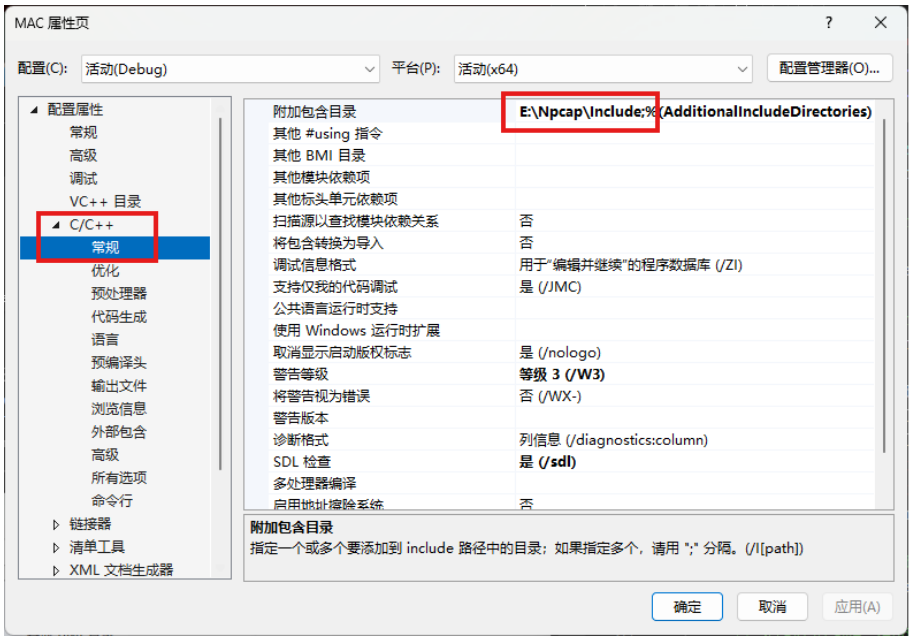
一、实验要求：通过编程获取 IP 地址与 MAC 地址的对应关系

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。

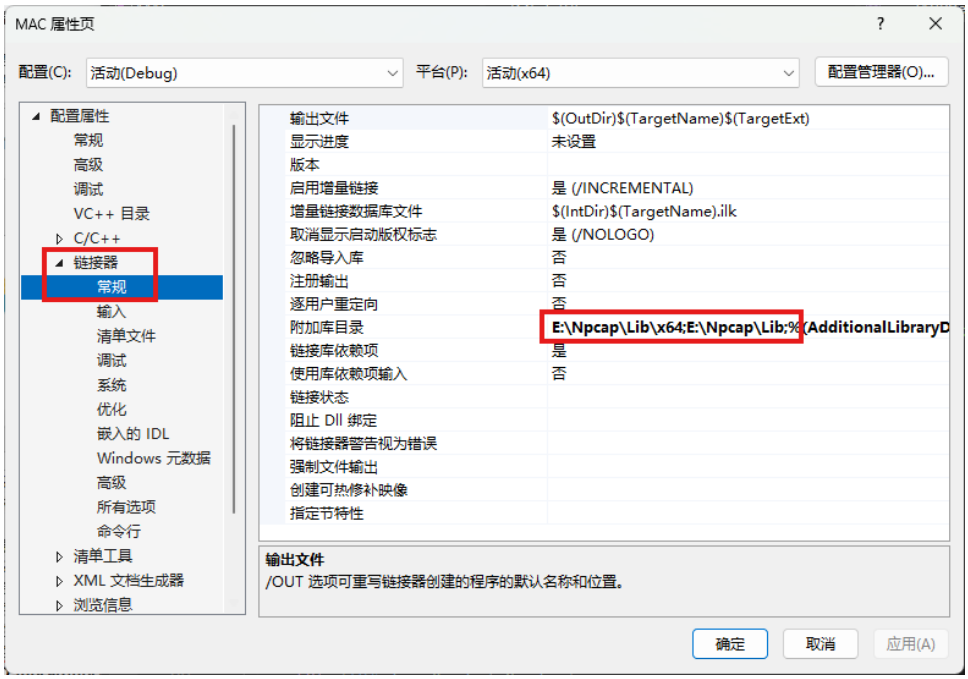
二、实验步骤

1. 准备工作：环境配置

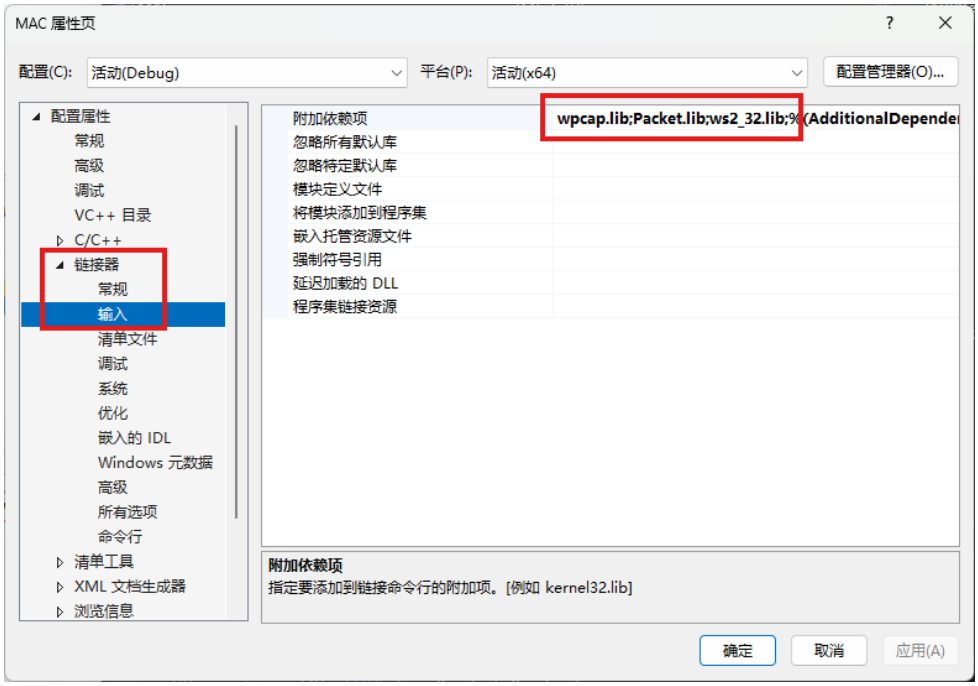
（1） 添加包含文件目录：项目 – 属性 – 配置属性 – C/C++ – 常规 – 附加包含目录



（2） 添加库文件目录：项目 – 属性 – 配置属性 – 连接器 – 常规 – 附加库目录



（3） 添加链接时使用的库文件



2. 帧结构定义：依据以太网中 ARP 报文格式，对 ARP 帧结构进行定义

硬件类型：以太网接口类型值为 1

协议类型：IP 协议类型为 0800（16）

硬件地址长度：MAC 地址长度为 6B

协议地址长度：IP 地址长度为 4B

操作：ARP 请求为 1，ARP 应答为 2

源 MAC 地址：发送方的 MAC 地址

源 IP 地址：发送方的 IP 地址

目的 MAC 地址：ARP 请求中该字段没有意义，ARP 响应中为接收方的 MAC 地址

目的 IP 地址：ARP 请求中为请求解析的 IP 地址，ARP 响应中为接收方的 IP 地址


```
//定义帧首部
typedef struct FrameHeader_t {
    BYTE DesMAC[6];     //目的地址
    BYTE SrcMAC[6];     //源地址
    WORD FrameType;     //帧类型
}FrameHeader_t;

//定义IP首部
typedef struct IPHeader_t {
    BYTE Ver_HLen;      //IP版本和头部长度
    BYTE TOS;           //服务类型
    WORD TotalLen;      //总长度
    WORD ID;            //标识
    WORD Flag_Segment;  //片偏移
    BYTE TTL;           //生存时间
    BYTE Protocol;      //协议
    WORD Checksum;      //首部校验和
    ULONG SrcIP;        //源IP
    ULONG DstIP;        //目的IP
}IPHeader_t;

//定义包含帧首部和IP首部的数据包
typedef struct Data_t {
    FrameHeader_t FrameHeader;
    IPHeader_t IPHeader;
}Data_t;
```

```
//定义ARP帧
typedef struct ARPFrame_t {
    FrameHeader_t FrameHeader;
    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
    BYTE SendHa[6];     //源MAC地址
    DWORD SendIP;       //源IP地址
    BYTE RecvHa[6];     //目的MAC地址
    DWORD RecvIP;       //目的IP地址
}ARPFrame_t;

#pragma pack()    //恢复缺省对齐方式
```

3. 获取本机网络接口（同 IP 数据包捕获与分析实验，不再赘述）

```
int main()
{
    int i = 0;
    int num = 0; //接口序号
    pcap_if_t* alldevs; //指向设备链表首部的指针
    pcap_if_t* d;
    pcap_addr_t* a; //表示接口地址指针
    char errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区

    //获取当前网卡列表
    if (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 //出错信息保存缓冲区
    ) == -1)
    {
        cout << "获取本机网卡列表时出错:" << errbuf << endl;
        return 1;
    }

    //打印网卡的列表
    for (d = alldevs; d != NULL; d = d->next)
    {
        num++;
        //打印网络接口设备的名字和描述信息
        cout << num << ". " << d->name << "->" << d->description << ";" << endl;
    }
```

4. 选择打开的设备，并打印设备的详细信息（同 IP 数据包捕获与分析实验，不再赘述）

```
    //选择网卡
    cout << "请选择设备（1-" << num << "):" << endl;
    int dev_select_num = 0;
    cin >> dev_select_num;

    while (dev_select_num < 1 || dev_select_num > num)
    {
        cout << "字符非法，请重新输入（1-" << num << "):" << endl;
        cin >> dev_select_num;
        if (dev_select_num >= 1 && dev_select_num <= num)
        {
            break;
        }
    }

    //转到选择的设备
    for (d = alldevs, i = 0; i < dev_select_num - 1; i++)
    {
        d = d->next;
    }

    //打印选择的设备的详细信息
    cout << "您选择的设备信息为：" << d->name << ";" << endl;
    cout << "描述信息：" << d->description << endl;

    for (a = d->addresses; a != NULL; a = a->next)
    {
        if (a->addr->sa_family == AF_INET)
        {
            cout << "IP地址：" << inet_ntoa(((struct sockaddr_in*)a->addr)->sin_addr) << endl;
            HostIP = inet_ntoa(((struct sockaddr_in*)a->addr)->sin_addr);

            cout << "子网掩码：" << inet_ntoa(((struct sockaddr_in*)a->netmask)->sin_addr) << endl;
            cout << "广播地址：" << inet_ntoa(((struct sockaddr_in*)a->broadaddr)->sin_addr) << endl;
            HostBroadaddr = inet_ntoa(((struct sockaddr_in*)a->broadaddr)->sin_addr);
            cout << endl;
            break;
        }
    }
```

5. 打开所选择的设备（同 IP 数据包捕获与分析实验，不再赘述）

```
232     //打开网络接口
233     pcap_t* handle = pcap_open_live(d->name, BUFSIZ, 1, 1000, errbuf);
234     if (pcap_open == NULL)
235     {
236         cout << "打开设备" << dev_select_num << "的网络接口失败：" << errbuf << endl;
237         return 1;
238     }
```

6. 初始化 ARP 数据包：设置相关参数

```
240     //定义发送的ARP包并初始化
241     ARPFrame_t ARPFrame;
242
243     ARPFrame.FrameHeader.FrameType = htons(0x8086); //帧类型为ARP
244     ARPFrame.HardwareType = htons(0x0001); //硬件类型为以太网
245     ARPFrame.ProtocolType = htons(0x0800); //协议类型为IP
246     ARPFrame.HLen = 6; //硬件地址长度为6
247     ARPFrame.PLen = 4; //协议地址长度为4
248     ARPFrame.Operation = htons(0x0001); //操作为ARP请求
```

7. 获取本机 MAC 地址：构造 ARP 数据包，将目的 IP 地址设置为选择的接口的的 IP 地址，发送 ARP 数据包，并循环捕获返回的 ARP 响应包；响应包中的源 IP 地址如果与源 IP 地址一致，则获取响应包中的源 MAC 地址并输出

```
62  void get_host_mac(pcap_t* handle, ARPFrame_t ARPFrame)
63  {
64      struct pcap_pkthdr* packet_header;
65      const u_char* packet_data;
66      Data_t* packet;
67
68      for (int i = 0; i < 6; i++)
69      {
70          ARPFrame.FrameHeader.DesMAC[i] = 0xff; //目的Mac地址设置为广播地址
71          ARPFrame.FrameHeader.SrcMAC[i] = HostMAC[i]; //源MAC地址设置为本机网卡MAC地
72          ARPFrame.SendHa[i] = i;
73          ARPFrame.RecvHa[i] = 0x00; //目的MAC地址设置为0
74      }
75      ARPFrame.SendIP = inet_addr("10.10.10.10"); //源IP地址随便设置一个
76      ARPFrame.RecvIP = inet_addr(HostIP.c_str()); //目的IP地址为本机地址
77
78      //发送数据包
79      if (pcap_sendpacket(handle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) != 0);
80
81      //抓取自己发出的包，从而获得自己的MAC地址
82      int res = 0;
83      while ((res = pcap_next_ex(handle, &packet_header, &packet_data)) >= 0)
84      {
85          if (res == 0) continue; //接收数据包超时
86
87          packet = (Data_t*)packet_data;
88          if (inet_ntoa(*(in_addr*)&packet->IPHeader.SrcIP) == HostIP)
89          {
90              for (int i = 0; i < 6; i++)
91              {
92                  HostMAC[i] = packet->FrameHeader.SrcMAC[i];
93              }
94              cout << "本机接口IP地址对应的MAC地址：\n" << HostIP;
95              printf(" --> %02X-%02X-%02X-%02X-%02X-%02X\n\n",
96                  (unsigned int)HostMAC[0],
97                  (unsigned int)HostMAC[1],
98                  (unsigned int)HostMAC[2],
99                  (unsigned int)HostMAC[3],
100                 (unsigned int)HostMAC[4],
101                 (unsigned int)HostMAC[5]);
102             break;
103         }
104     }
105
106 }
```

8. 获取输入的 IP 地址所对应的 MAC 地址：处理逻辑与获取本机 MAC 地址类似，将 ARP 请求发送给目标 IP，收到目标的 ARP 响应后，提取其中的源 MAC 地址并输出

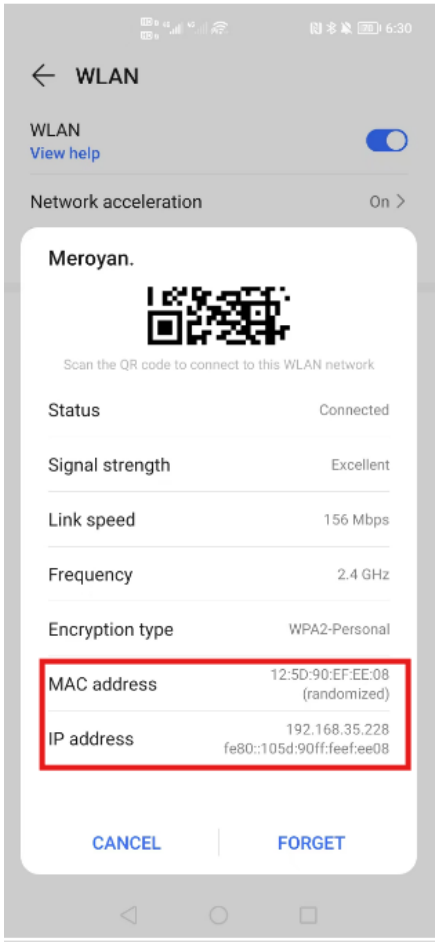
```
void GetMAC(pcap_t* adhandle, ARPFrame_t ARPFrame)
{
    struct pcap_pkthdr* packet_header;
    const u_char* packet_data;
    Data_t* packet;

    cout << "请输入IP地址：";
    string IP;
    cin >> IP;

    //ARP数据包
    for (int i = 0; i < 6; i++)
    {
        ARPFrame.FrameHeader.DesMAC[i] = 0xff; //目的Mac地址设置为广播地址
        ARPFrame.FrameHeader.SrcMAC[i] = HostMAC[i]; //源MAC地址设置为本机网卡MAC地
        ARPFrame.SendHa[i] = HostMAC[i]; //源MAC地址设置为本机网卡MAC地址
        ARPFrame.RecvHa[i] = 0x00; //目的MAC地址设置为0
    }
    ARPFrame.SendIP = inet_addr(HostIP.c_str()); //本机网卡IP地址
    ARPFrame.RecvIP = inet_addr(IP.c_str()); //目的IP地址

    //发送数据包
    pcap_sendpacket(adhandle, (u_char*)&ARPFrame, sizeof(ARPFrame_t));
```

```
    //监听目的IP为本机IP，原始IP为输入IP的数据包
    int res = 0;
    while ((res = pcap_next_ex(adhandle, &packet_header, &packet_data)) >= 0)
    {
        if (res == 0) continue; //接收数据包超时

        ARPFrame_t* ARP_Packet = (ARPFrame_t*)packet_data;
        string desIP = inet_ntoa(*(in_addr*)&ARP_Packet->RecvIP);
        string srcIP = inet_ntoa(*(in_addr*)&ARP_Packet->SendIP);
        if ((desIP == HostIP) && (srcIP == IP))
        {
            BYTE MACs[6];
            for (int i = 0; i < 6; i++)
            {
                MACs[i] = ARP_Packet->FrameHeader.SrcMAC[i];
            }

            cout << IP;
            printf(" --> %02X-%02X-%02X-%02X-%02X-%02X\n\n",
                (unsigned int)MACs[0],
                (unsigned int)MACs[1],
                (unsigned int)MACs[2],
                (unsigned int)MACs[3],
                (unsigned int)MACs[4],
                (unsigned int)MACs[5]);

            break;
        }
    }
}
```

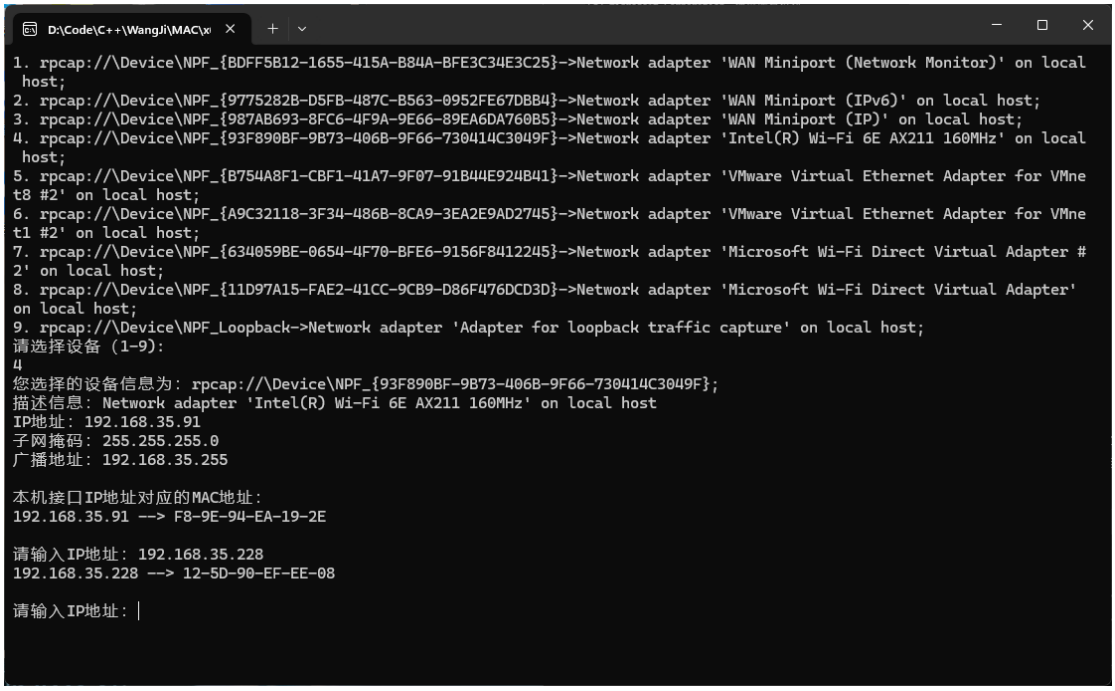
9. 运行结果展示

我将舍友的手机与我自己的电脑一同连到了我手机的热点之下，运行程序，选择 wifi 接口 4，可以看到返回了该接口的物理地址

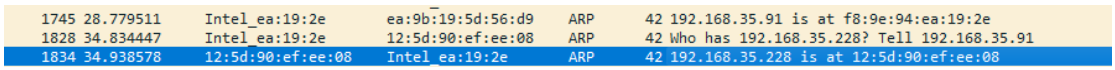
随后输入手机的 ip 地址 192.168.35.228，查询手机对应的 mac 地址



运行结果如下图所示，返回结果与手机显示的 MAC 地址一致，证明程序功能正确



在 wireshark 中，设置过滤为 arp，可以看到捕获的数据包如下



三行数据依次为：

主机响应 ARP，返回本机的 MAC 地址；

主机发送 ARP 请求，询问手机的 MAC 地址；

手机响应 ARP，返回手机的 MAC 地址