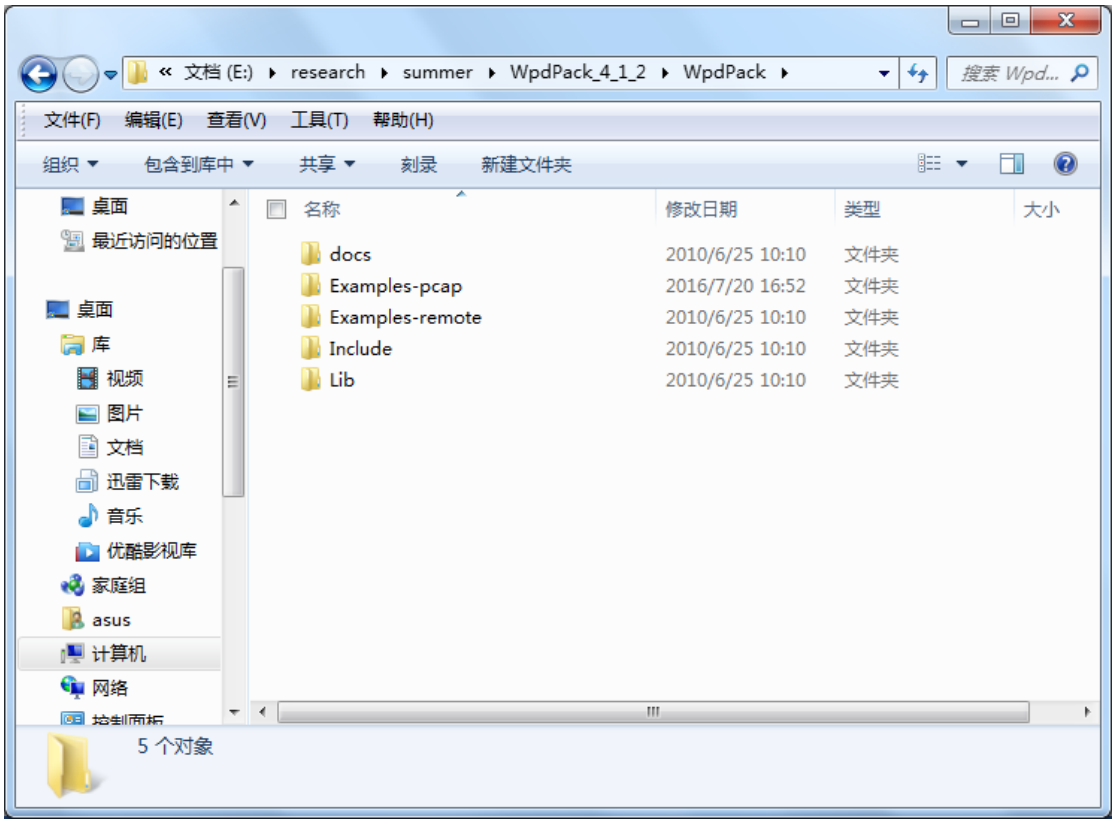
WinPcap 配置指南

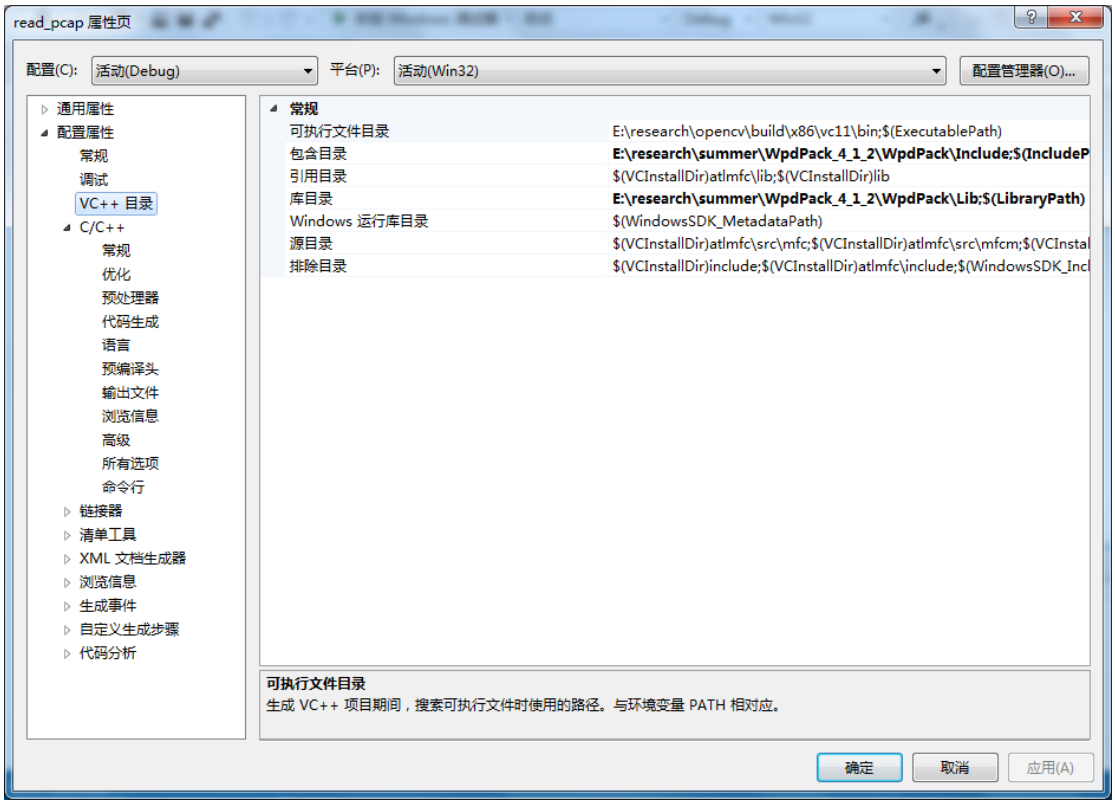
本篇配置环境为 vs2012 和 WpdPack_4_1_2 以及 WinPcap4_1_3

在解压 WpdPack 的时候，可以得到以下内容：

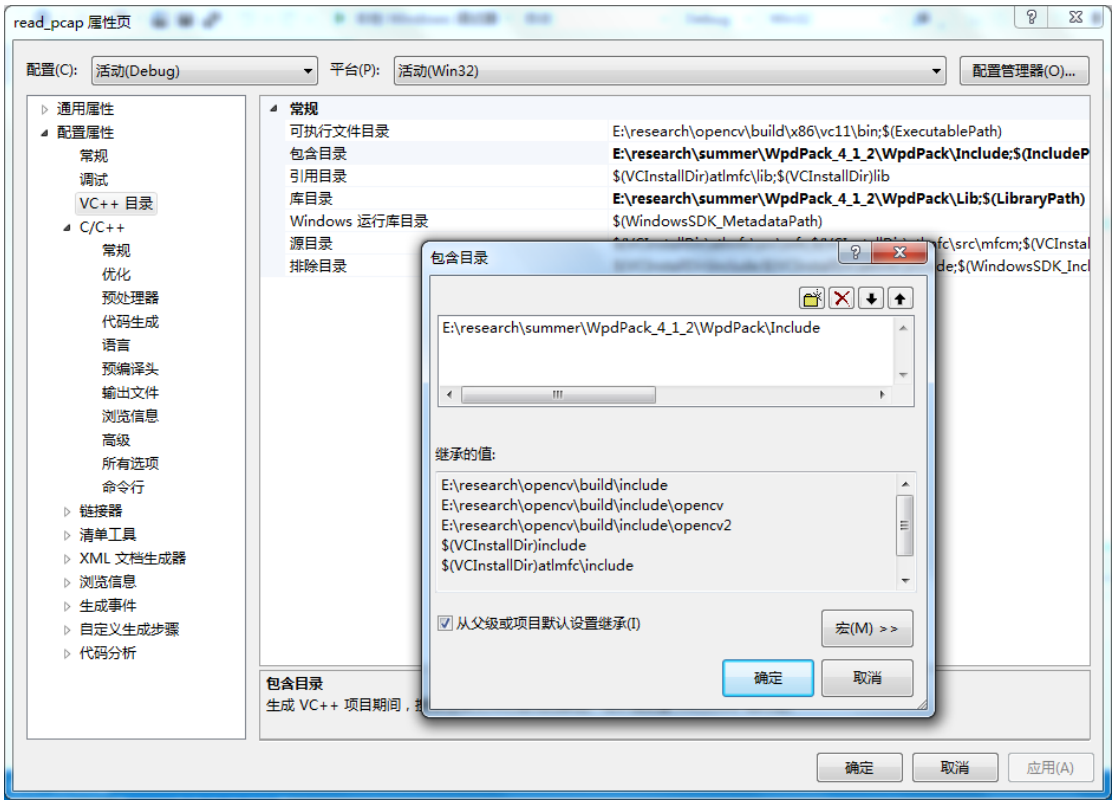


然后，我们进行环境的配置。

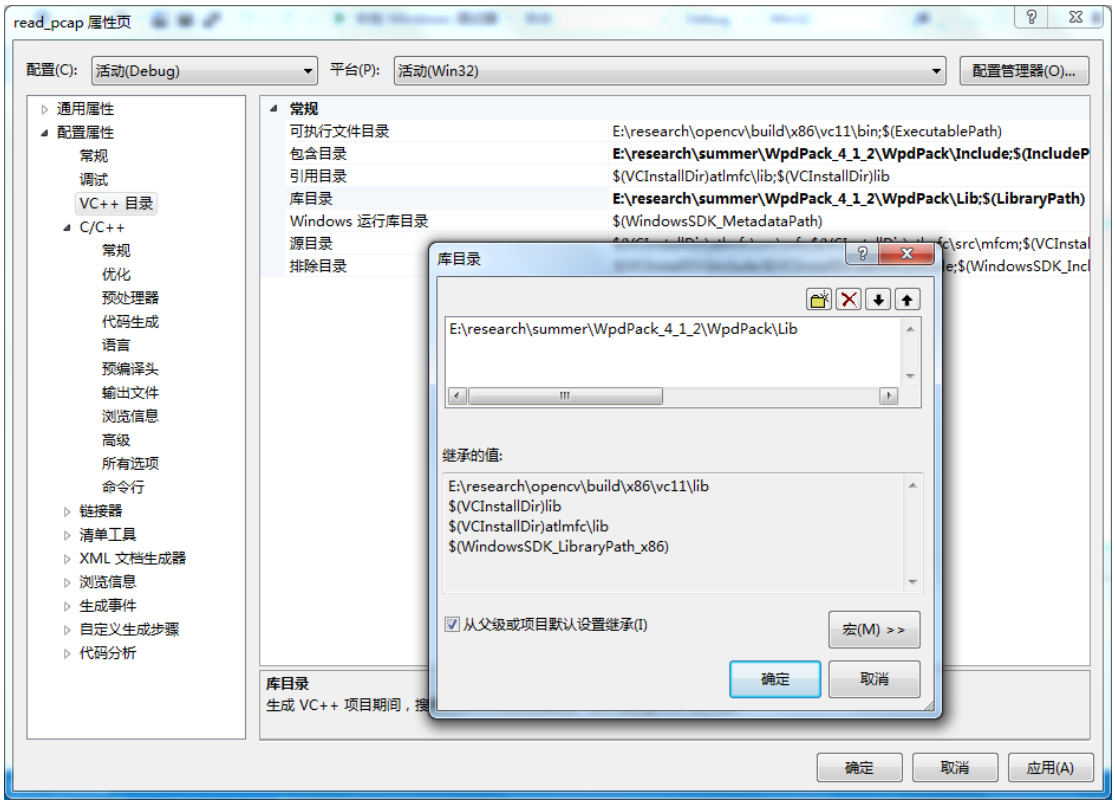
首先，新建 vs 工程，然后，进去 VC++目录：



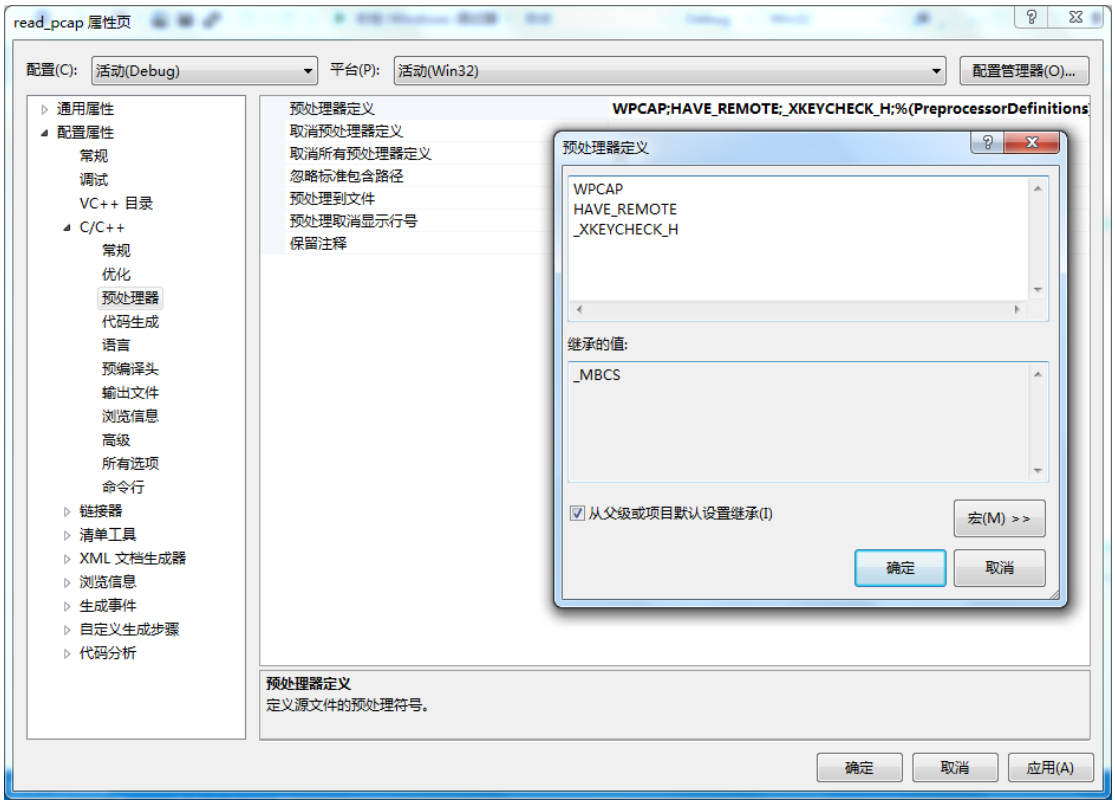
然后，对库目录和包含目录进行配置，首先是包含目录：



将 Include 的路径加进去。
库目录同样的道理：

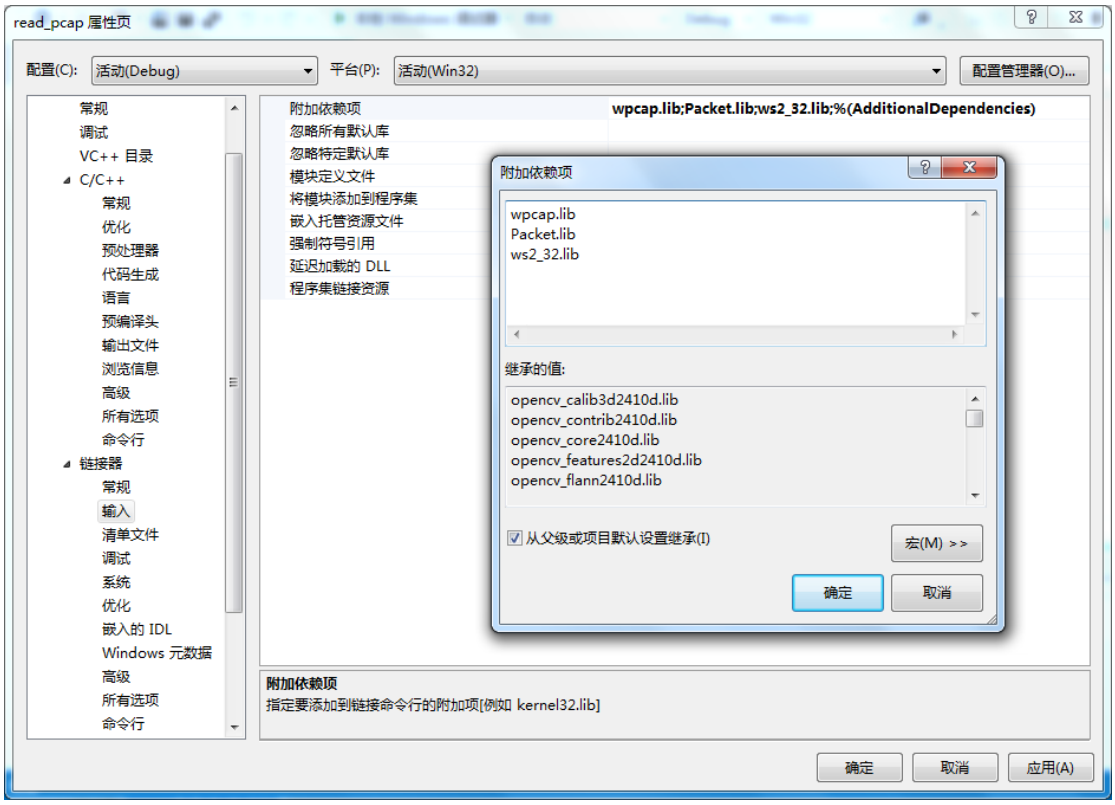


我们加入 lib 的路径，然后，在 C/C++，预处理器中，我们加入以下内容：



WPCAP
HAVE_REMOTE
_XKEYCHECK_H

然后，在链接器输入的时候，加入以下链接库：



wpcap.lib
Packet.lib
ws2_32.lib

在进行了上面的环境部署后，我们进行测试，用以下测试代码进行网络数据的抓包：

```
#include<stdlib.h>
#include "pcap.h"
#include<iostream>
/* 4 字节的 IP 地址 */
typedef struct ip_address{
    u_char byte1;
    u_char byte2;
    u_char byte3;
    u_char byte4;
}ip_address;

/* IPv4 首部 */
typedef struct ip_header{
    u_char  ver_ihl;        // 版本 (4 bits) + 首部长度 (4 bits)
    u_char  tos;            // 服务类型(Type of service)
    u_short tlen;           // 总长(Total length)
    u_short identification; // 标识(Identification)
    u_short flags_fo;       // 标志位(Flags) (3 bits) + 段偏移量(Fragment offset) (13 bits)
    u_char  ttl;            // 存活时间(Time to live)
    u_char  proto;          // 协议(Protocol)
    u_short crc;            // 首部校验和(Header checksum)
    ip_address  saddr;      // 源地址(Source address)
    ip_address  daddr;      // 目的地址(Destination address)
    u_int   op_pad;         // 选项与填充(Option + Padding)
}ip_header;

/* UDP 首部*/
typedef struct udp_header{
    u_short sport;          // 源端口(Source port)
    u_short dport;          // 目的端口(Destination port)
    u_short len;            // UDP 数据包长度(Datagram length)
    u_short crc;            // 校验和(Checksum)
}udp_header;

void packet_handler(u_char *param, const struct pcap_pkthdr *header, const u_char *pkt_data);
int main(int argc, char **argv)
{
pcap_if_t *alldevs;
```

```
pcap_if_t *d;
int inum;
int i=0;
pcap_t *adhandle;
char errbuf[PCAP_ERRBUF_SIZE];
pcap_dumper_t *dumpfile;
char filename[]="E:\\traffic.pcap";



    if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) == -1)
    {
        fprintf(stderr,"Error in pcap_findalldevs: %s\n", errbuf);
        exit(1);
    }


    for(d=alldevs; d; d=d->next)
    {
        printf("%d. %s", ++i, d->name);
        if (d->description)
            printf(" (%s)\n", d->description);
        else
            printf(" (No description available)\n");
    }
    if(i==0)
    {
        printf("\nNo interfaces found! Make sure WinPcap is installed.\n");
        return -1;
    }

    printf("Enter the interface number (1-%d):",i);
    scanf("%d", &inum);

    if(inum < 1 || inum > i)
    {
        printf("\nInterface number out of range.\n");

        pcap_freealldevs(alldevs);
        return -1;
    }


    for(d=alldevs, i=0; i< inum-1 ;d=d->next, i++);
```

```
    if ( (adhandle= pcap_open(d->name,          // name of the device
                              65536,            // portion of the packet to capture
                                                // 65536 guarantees that the whole packet will be captured on all the link layers
                              PCAP_OPENFLAG_PROMISCUOUS,    // promiscuous mode
                              1000,             // read timeout
                              NULL,             // authentication on the remote machine
                              errbuf            // error buffer
                              ) ) == NULL)
    {
        fprintf(stderr,"\nUnable to open the adapter. %s is not supported by WinPcap\n", d->name);

        pcap_freealldevs(alldevs);
        return -1;
    }

    dumpfile = pcap_dump_open(adhandle, filename);
    if(dumpfile==NULL)
    {
        fprintf(stderr,"\nError opening output file\n");
        return -1;
    }

    printf("\nlistening on %s... Press Ctrl+C to stop...\n", d->description);


    pcap_freealldevs(alldevs);


    pcap_loop(adhandle, 200, packet_handler, (unsigned char *)dumpfile);
    system("pause");
    return 0;
}

void packet_handler(u_char *dumpfile, const struct pcap_pkthdr *header, const u_char *pkt_data)
{
    struct tm *ltime;
    char timestr[16];
```

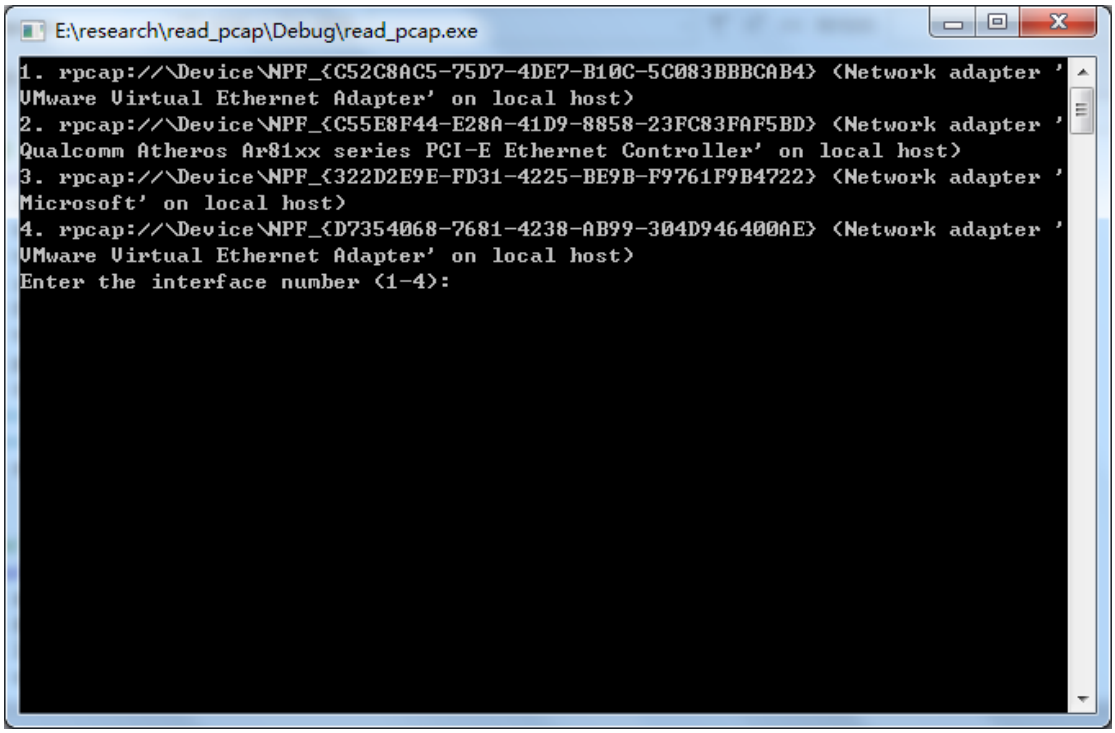
```
    time_t local_tv_sec;
    ip_header *ih;
    udp_header *uh;
    u_int ip_len;
    u_short sport,dport;
    /* 将时间戳转换成可识别的格式 */
    local_tv_sec = header->ts.tv_sec;
    ltime = localtime(&local_tv_sec);
    strftime(timestr, sizeof timestr, "%H:%M:%S", ltime);
    pcap_dump(dumpfile, header, pkt_data);
    printf("time: %s, ms : %.6d len: %d", timestr, header->ts.tv_usec, header->len);
        /* 获得 IP 数据包头部的位置 */
    ih = (ip_header *) (pkt_data +
        14); //以太网头部长度

    /* 获得 UDP 首部的位置 */
    ip_len = (ih->ver_ihl & 0xf) * 4;
    uh = (udp_header *) ((u_char*)ih + ip_len);

    /* 将网络字节序列转换成主机字节序列 */
    sport = ntohs( uh->sport );
    dport = ntohs( uh->dport );

    /* 打印 IP 地址和 UDP 端口 */
    printf(" IP: %d.%d.%d.%d -> %d.%d.%d.%d %d->%d\n",
        ih->saddr.byte1,
        ih->saddr.byte2,
        ih->saddr.byte3,
        ih->saddr.byte4,
        ih->daddr.byte1,
        ih->daddr.byte2,
        ih->daddr.byte3,
        ih->daddr.byte4,
        sport,
        dport);
}
```

如果出现以下界面，则配置成功：


E:\research\read_pcap\Debug\read_pcap.exe
1. rpcap://\Device\NPF_{C52C8AC5-75D7-4DE7-B10C-5C083BBBCAB4} (Network adapter '
VMware Virtual Ethernet Adapter' on local host)
2. rpcap://\Device\NPF_{C55E8F44-E28A-41D9-8858-23FC83FAF5BD} (Network adapter '
Qualcomm Atheros Ar81xx series PCI-E Ethernet Controller' on local host)
3. rpcap://\Device\NPF_{322D2E9E-FD31-4225-BE9B-F9761F9B4722} (Network adapter '
Microsoft' on local host)
4. rpcap://\Device\NPF_{D7354068-7681-4238-AB99-304D946400AE} (Network adapter '
VMware Virtual Ethernet Adapter' on local host)
Enter the interface number (1-4):