计算机网络

**课程设计报告**

题目： Tracert与Ping程序设计与实现

**姓名： 唐韬**

**学号： 1305010206**

**班级： 13计算机二班**

**指导老师： 姜磊**

**湖南科技大学计算机科学与工程学院**

**2016年 1 月**

1. **实验题目**

Tracert与Ping程序设计与实现

1. **实验内容**

1、了解Tracert程序的实现原理

2、参照附录2，将Tracert程序调试通过

3参考Tracert程序和教材4.4.2节，编写一个Ping程序，能测试本局域网的所有机器是否在线

1. **实验步骤**

1、阅读课程设计指导书，了解了解Tracert程序的实现原理

2、阅读课程设计指导书，参照附录2，将Tracert程序调试通过

3、通过课程设计指导书，参照附录2，重新编写一个Ping程序，能够测试主机名或者IP是否能够接通

4、在3的基础上，将程序改为能够测试一个局域网内所有或者任意一个段内的机器是否在线

1. **代码部分**

// Tarcert.cpp : 定义控制台应用程序的入口点。

//10.1.12.219

#include "stdafx.h"

#include <iostream>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <IPHlpApi.h>

#include <string>

using namespace std;

#pragma comment(lib, "Ws2\_32.lib")

#pragma comment(lib,"iphlpapi.lib")

//IP报头

typedef struct

{

unsigned char hdr\_len:4; //4位头部长度

unsigned char version:4; //4位版本号

unsigned char tos; //8位服务类型

unsigned short total\_len; //16位总长度

unsigned short identifier; //16位标识符

unsigned short frag\_and\_flags; //3位标志加位片偏移

unsigned char ttl; //8位生存时间

unsigned char protocol; //8位上层协议号

unsigned short checksum; //16位校验和

unsigned long sourceIP; //32位源IP地址

unsigned long destIP; //32位目的IP地址

} IP\_HEADER;

//ICMP报头

typedef struct

{

BYTE type; //8位类型字段

BYTE code; //8位代码字段

USHORT cksum; //16位校验和

USHORT id; //16位标识符

USHORT seq; //16位序列号

} ICMP\_HEADER;

//报文解码结构

typedef struct

{

USHORT usSeqNo; //序列号

DWORD dwRoundTripTime; //往返时间

in\_addr dwIPaddr; //返回报文的IP地址

}DECODE\_RESULT;

//计算网际校验和函数

USHORT checksum(USHORT \*pBuf,int iSize)

{

unsigned long cksum=0;

while(iSize>1)

{

cksum+=\*pBuf++;

iSize-=sizeof(USHORT);

}

if(iSize)

{

cksum+=\*(UCHAR \*)pBuf;

}

cksum=(cksum>>16)+(cksum&0xffff);

cksum+=(cksum>>16);

return (USHORT)(~cksum);

}

string GetMAC(string IP) // 取IP地址对应的MAC地址

{

int rs;

unsigned char mac[6];

ULONG MacLen=6;

ULONG DestIP=inet\_addr(IP.c\_str());

rs=SendARP(DestIP,(ULONG)NULL,(PULONG)mac,(PULONG)&MacLen);

string MACs="";

if (rs==0)

{

char buf[32];

sprintf(buf,"%02X-%02X-%02X-%02X-%02X-%02X",

(unsigned int)mac[0],

(unsigned int)mac[1],

(unsigned int)mac[2],

(unsigned int)mac[3],

(unsigned int)mac[4],

(unsigned int)mac[5]);

MACs=buf;

}

return(MACs);

}

//对数据包进行解码

BOOL DecodeIcmpResponse(char \* pBuf,int iPacketSize,DECODE\_RESULT &DecodeResult,BYTE ICMP\_ECHO\_REPLY,BYTE ICMP\_TIMEOUT)

{

//检查数据报大小的合法性

IP\_HEADER\* pIpHdr = (IP\_HEADER\*)pBuf;

int iIpHdrLen = pIpHdr->hdr\_len \* 4;

if (iPacketSize < (int)(iIpHdrLen+sizeof(ICMP\_HEADER)))

return FALSE;

//根据ICMP报文类型提取ID字段和序列号字段

ICMP\_HEADER \*pIcmpHdr=(ICMP\_HEADER \*)(pBuf+iIpHdrLen);

USHORT usID,usSquNo;

if(pIcmpHdr->type==ICMP\_ECHO\_REPLY) //ICMP回显应答报文

{

usID=pIcmpHdr->id; //报文ID

usSquNo=pIcmpHdr->seq; //报文序列号

}

else if(pIcmpHdr->type==ICMP\_TIMEOUT)//ICMP超时差错报文

{

char \* pInnerIpHdr=pBuf+iIpHdrLen+sizeof(ICMP\_HEADER); //载荷中的IP头

int iInnerIPHdrLen=((IP\_HEADER \*)pInnerIpHdr)->hdr\_len\*4; //载荷中的IP头长

ICMP\_HEADER \* pInnerIcmpHdr=(ICMP\_HEADER \*)(pInnerIpHdr+iInnerIPHdrLen);//载荷中的ICMP头

usID=pInnerIcmpHdr->id; //报文ID

usSquNo=pInnerIcmpHdr->seq; //序列号

}

else

{

return false;

}

//检查ID和序列号以确定收到期待数据报

if(usID!=(USHORT)GetCurrentProcessId()||usSquNo!=DecodeResult.usSeqNo)

{

return false;

}

//记录IP地址并计算往返时间

DecodeResult.dwIPaddr.s\_addr=pIpHdr->sourceIP;

DecodeResult.dwRoundTripTime=GetTickCount()-DecodeResult.dwRoundTripTime;

//处理正确收到的ICMP数据报

if (pIcmpHdr->type == ICMP\_ECHO\_REPLY ||pIcmpHdr->type == ICMP\_TIMEOUT)

{

//输出往返时间信息

if(DecodeResult.dwRoundTripTime)

cout<<"\t在线"<<flush;

else

cout<<"\t在线"<<flush;

}

return true;

}

void main()

{

//初始化Windows sockets网络环境

WSADATA wsa;

WSAStartup(MAKEWORD(2,2),&wsa);

char IpAddress[255];

cout<<"请输入一个IP地址前三位(XXX.XXX.XXX.)：";

cin>>IpAddress;

for(int i=1;i<=255;i++){

//得到IP地址

char temp[50];

char IpTemp[255];

itoa(i,temp,10);

strcpy(IpTemp,IpAddress);

strcat(IpTemp,temp);

//cout<<IpTemp<<endl;

u\_long ulDestIP=inet\_addr(IpTemp);

//转换不成功时按域名解析

if(ulDestIP==INADDR\_NONE)

{

hostent \* pHostent=gethostbyname(IpTemp);

if(pHostent)

{

ulDestIP=(\*(in\_addr\*)pHostent->h\_addr).s\_addr;

}

else

{

cout<<"输入的IP地址或域名无效!"<<endl;

WSACleanup();

return;

}

}

//cout<<"Tracing roote to "<<IpAddress<<" with a maximum of 30 hops.\n"<<endl;

//填充目地端socket地址

sockaddr\_in destSockAddr;

ZeroMemory(&destSockAddr,sizeof(sockaddr\_in));

destSockAddr.sin\_family=AF\_INET;

destSockAddr.sin\_addr.s\_addr=ulDestIP;

//创建原始套接字

SOCKET sockRaw=WSASocket(AF\_INET,SOCK\_RAW,IPPROTO\_ICMP,NULL,0,WSA\_FLAG\_OVERLAPPED);

//超时时间

int iTimeout=5;

//接收超时

setsockopt(sockRaw,SOL\_SOCKET,SO\_RCVTIMEO,(char \*)&iTimeout,sizeof(iTimeout));

//发送超时

setsockopt(sockRaw,SOL\_SOCKET,SO\_SNDTIMEO,(char \*)&iTimeout,sizeof(iTimeout));

//构造ICMP回显请求消息，并以TTL递增的顺序发送报文

//ICMP类型字段

const BYTE ICMP\_ECHO\_REQUEST=8; //请求回显

const BYTE ICMP\_ECHO\_REPLY=0; //回显应答

const BYTE ICMP\_TIMEOUT=11; //传输超时

//其他常量定义

const int DEF\_ICMP\_DATA\_SIZE=32; //ICMP报文默认数据字段长度

const int MAX\_ICMP\_PACKET\_SIZE=1024;//ICMP报文最大长度（包括报头）

const DWORD DEF\_ICMP\_TIMEOUT=30; //回显应答超时时间

const int DEF\_MAX\_HOP=30; //最大跳站数

//填充ICMP报文中每次发送时不变的字段

char IcmpSendBuf[sizeof(ICMP\_HEADER)+DEF\_ICMP\_DATA\_SIZE];//发送缓冲区

memset(IcmpSendBuf, 0, sizeof(IcmpSendBuf)); //初始化发送缓冲区

char IcmpRecvBuf[MAX\_ICMP\_PACKET\_SIZE]; //接收缓冲区

memset(IcmpRecvBuf, 0, sizeof(IcmpRecvBuf)); //初始化接收缓冲区

ICMP\_HEADER \* pIcmpHeader=(ICMP\_HEADER\*)IcmpSendBuf;

pIcmpHeader->type=ICMP\_ECHO\_REQUEST; //类型为请求回显

pIcmpHeader->code=0; //代码字段为

pIcmpHeader->id=(USHORT)GetCurrentProcessId(); //ID字段为当前进程号

memset(IcmpSendBuf+sizeof(ICMP\_HEADER),'E',DEF\_ICMP\_DATA\_SIZE);//数据字段

USHORT usSeqNo=0; //ICMP报文序列号

int iTTL=1; //TTL初始值为

BOOL bReachDestHost=FALSE; //循环退出标志

int iMaxHot=DEF\_MAX\_HOP; //循环的最大次数

DECODE\_RESULT DecodeResult; //传递给报文解码函数的结构化参数

while(!bReachDestHost&&iMaxHot--)

{

//设置IP报头的TTL字段

setsockopt(sockRaw,IPPROTO\_IP,IP\_TTL,(char \*)&iTTL,sizeof(iTTL));

cout<<i<<flush; //输出当前序号

//填充ICMP报文中每次发送变化的字段

((ICMP\_HEADER \*)IcmpSendBuf)->cksum=0; //校验和先置为

((ICMP\_HEADER \*)IcmpSendBuf)->seq=htons(usSeqNo++); //填充序列号

((ICMP\_HEADER \*)IcmpSendBuf)->cksum=checksum((USHORT \*)IcmpSendBuf,sizeof(ICMP\_HEADER)+DEF\_ICMP\_DATA\_SIZE);//计算校验和

//记录序列号和当前时间

DecodeResult.usSeqNo=((ICMP\_HEADER\*)IcmpSendBuf)->seq; //当前序号

DecodeResult.dwRoundTripTime =GetTickCount(); //当前时间

//发送TCP回显请求信息

sendto(sockRaw,IcmpSendBuf,sizeof(IcmpSendBuf),0,(sockaddr\*)&destSockAddr,sizeof(destSockAddr));

//接收ICMP差错报文并进行解析处理

sockaddr\_in from; //对端socket地址

int iFromLen=sizeof(from); //地址结构大小

int iReadDataLen; //接收数据长度

while(1)

{

//接收数据

iReadDataLen=recvfrom(sockRaw,IcmpRecvBuf,MAX\_ICMP\_PACKET\_SIZE,0,(sockaddr\*)&from,&iFromLen);

if(iReadDataLen!=SOCKET\_ERROR)//有数据到达

{

//对数据包进行解码

if(DecodeIcmpResponse(IcmpRecvBuf,iReadDataLen,DecodeResult,ICMP\_ECHO\_REPLY,ICMP\_TIMEOUT))

{

//到达目的地，退出循环

if(DecodeResult.dwIPaddr.s\_addr==destSockAddr.sin\_addr.s\_addr)

bReachDestHost=true;

//输出IP地址

string tip = GetMAC(IpTemp);

cout<<'\t'<<inet\_ntoa(DecodeResult.dwIPaddr)<<"\t"+tip<<endl;

break;

}

}

else if(WSAGetLastError()==WSAETIMEDOUT) //接收超时，输出\*号

{

cout<<"\t"<<"离线"<<endl;

bReachDestHost=true;

break;

}

else

{

break;

}

}

iTTL++; //递增TTL值

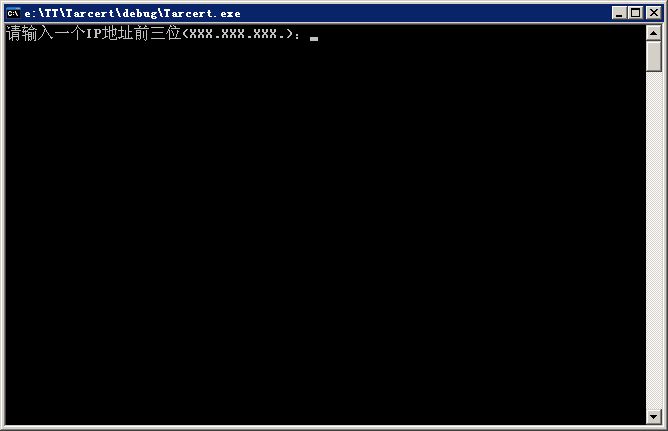
}

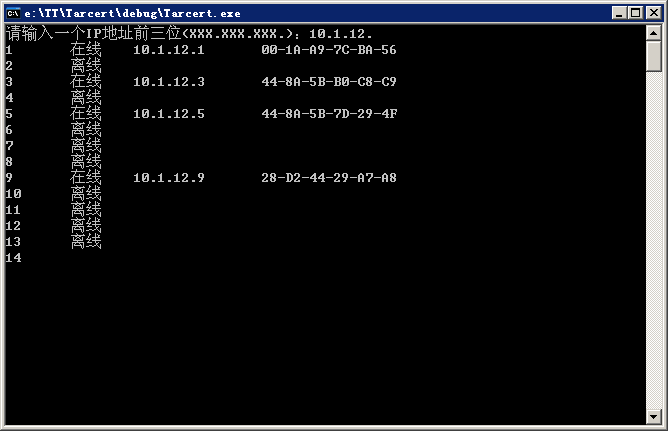
}

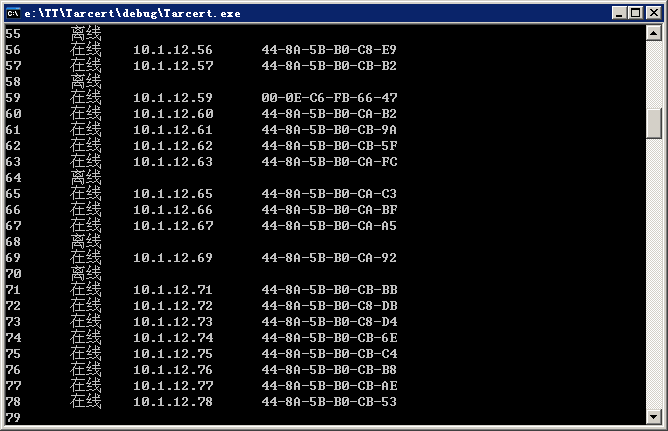
system("pause");

**五**.**实验结果及结果分析**

程序图：







**六.心得体会**

这个实验很有趣啊，通过遍历所有的局域网IP段就可以实现目的，实际上在cmd里面自带了ping功能，只要在我的程序里调用系统命令，比如说system(“ping xxx.xxx.xxx.xxx”);然后把输出结果重定向到文件，最后读出文件对文件内容进行整理，这个实验的功能也就实现了，可是我觉得不能这么做，这样做了之后根本学不到什么，还是得从底层写起来。