|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 5 | | |
| по дисциплине «Операционные системы и компьютерные сети» | | |
| **Анализ функционирования и диагностика IP-сетей** | | |
|  | | |
|  | Бригада 4 | Лойченко ДАНИЛА |
| ПМ-12 | Ушатов сергей |
|  |
|  |
|  |
|  |
| Преподаватели | Кобылянский валерий григорьевич |
|  | Сивак Мария алексеевна |
| Новосибирск, 2023 | | |

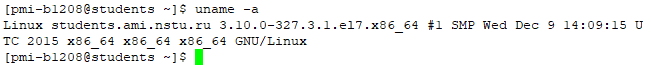
1. **Цель работы**

Приобретение практических навыков работы с сетевыми командами операционных систем Windows и Linux, предназначенными для анализа и диагностики сетей TCP/IP, а также со средствами интерактивной диагностики сетей.

1. **Ход выполнения**

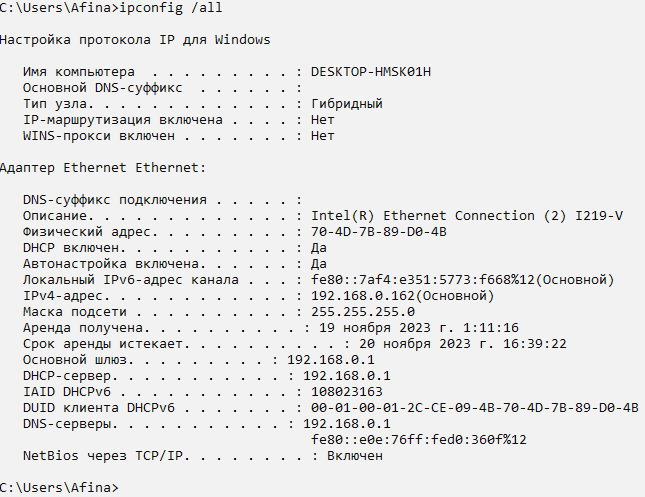
**Этап 1**

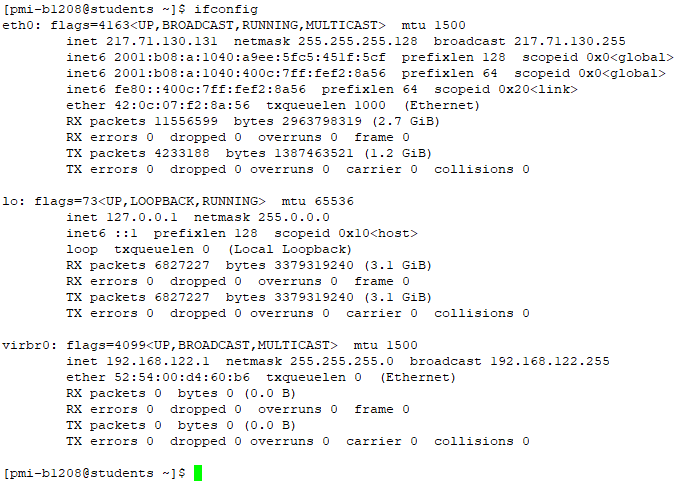
1. Подключиться с помощью клиента Putty к серверу **fpm2.ami.nstu.ru** и с помощью команды **uname** получить полную информацию об установленной операционной системе и аппаратной платформе, полученный результат включить в отчет.



|  |  |
| --- | --- |
| **Имя ядра** | Linux |
| **Имя хоста** | students.ami.nstu.ru |
| **Релиз ядра** | 3.10.0-327.3.1.el7.x86\_64 |
| **Версия ядра** | #1 SMP Wed Dec 9 14:09:15 UTC 2015 |
| **Имя аппаратного имени машины** | x86\_64 |
| **Архитектура процессора** | x86\_64 |
| **Аппаратная платформа** | x86\_64 |
| **Имя ОС** | GNU/Linux |

1. Получить статистику по сетевым интерфейсам РК и сервера **fpm2.ami.nstu.ru**, пояснить результаты.





**eth0** – сетевой интерфейс сервера.

**MAC-адрес**: 42:0c:07:f2:8a:56

**IPv4-адрес**: 217.71.130.131

**IPv6-адрес:** 2001:b08:a:1040:a9ee:5fc5:451f:5cf

**IPv6-адрес:** 2001:b08:a:1040:400c:7ff:fef2:8a56

**IPv6-адрес:** fe80::400c:7ff:fef2:8a56

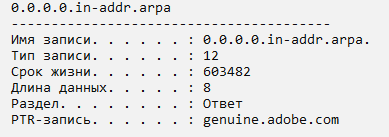
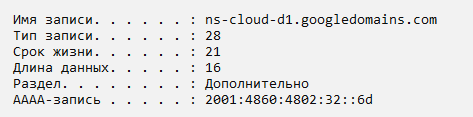
**Макс. размер кадра:** 1500 байт

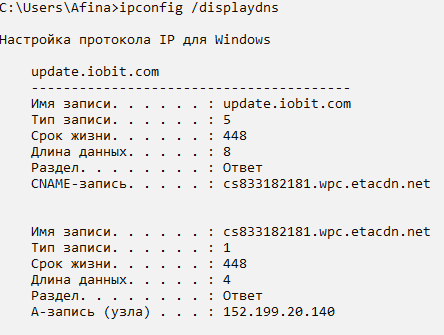
**lo** - это интерфейс внутреннего стека TCP/IP (петля или заглушка), предназначенный для отладки сетевого ПО и имеющий IP-адрес 127.0.0.1.

**virbr0** - интерфейс, являющийся виртуальным мостом, который связывает созданные пользователем виртуальные машины с основной машиной, имеет адрес 192.168.122.1.

**RX** и **TX** в каждом из сетевых интерфейсов показывают, сколько было успешно принятых и отправленных пакетов. **RX errors** и **TX errors** – сколько было ошибок при обмене пакетами.

1. Просмотреть содержимое DNS-кэша, пояснить характеристики записей, очистить кэш.

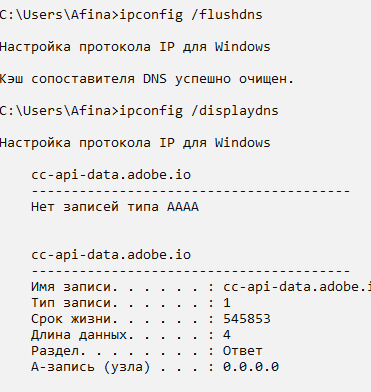
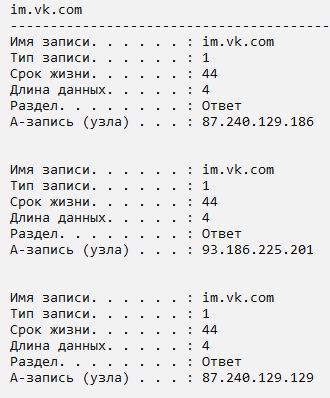
Приведена лишь малая часть DNS-кэша



Характеристики записей:

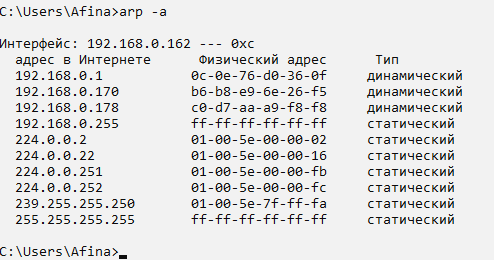
* **Имя записи** – доменное имя, к которому относится запись
* **Тип записи** – формат и назначение записи
* **Срок жизни** – допустимое время хранения записи на DNS сервере
* **Длина данных** – длина записи в байтах
* **Раздел** – тип записи в кэше DNS
* **A-запись (узла)** – связывает имя хоста с адресом протокола IPv4
* **AAAA запись** -связывает имя хоста с адресом протокола IPv6.
* **CNAME-запись -** тип записи DNS, которая привязывает псевдоним к действительному (каноническому) доменному имени (служит для перенаправления).
* **Запись PTR** - обратная DNS-запись или запись указателя связывает IP-адрес хоста с его каноническим именем.

Очистили **DNS-**кэш и вывели его текущее состояние:

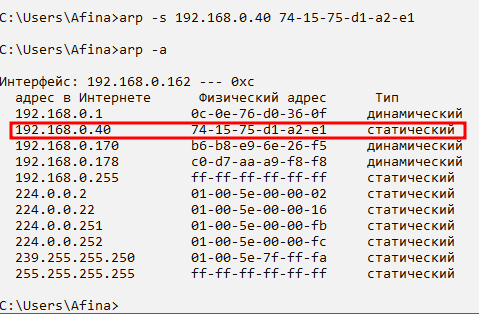
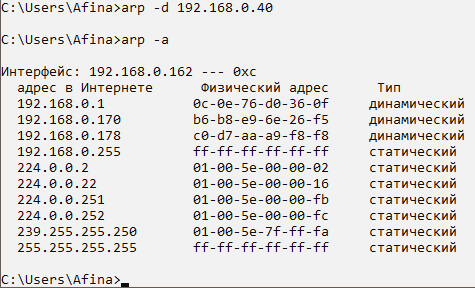
Как можем заметить его наполнение поменялось

1. Просмотреть содержимое ARP-таблицы, пояснить характеристики записей, выполнить добавление и удаление статических записей.

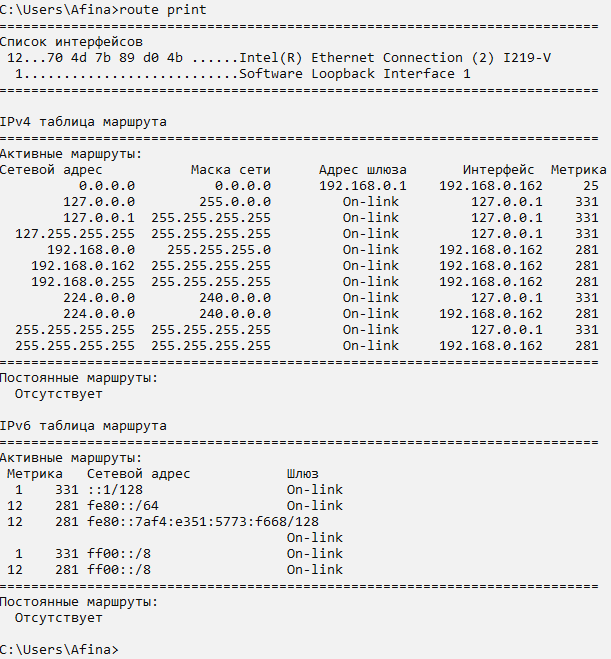


* **Сетевой адрес** – IP-адрес устройства в сети
* **Физический адрес** – уникальный MAC-адрес устройства
* **Тип записи** – информация о том, является ли адрес периодически обновляемым или временным (до перезагрузки устройства)

Добавления записи Удаление записи

1. Просмотреть содержимое таблицы маршрутизации, пояснить характеристики записей.



**Список интерфейсов** – информация об интерфейсах, имеющихся у устройства

**IPv4 таблица маршрута** - содержит информацию о том, каким образом следует доставить пакеты данных к сетевым адресам IPv4

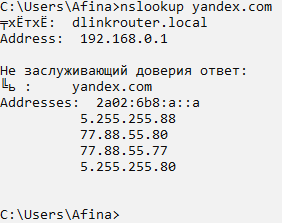
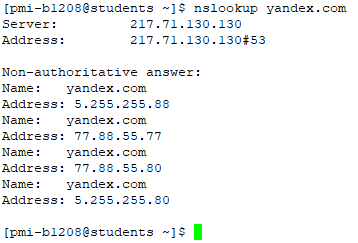
**IPv6 таблица маршрута** - содержит информацию о том, каким образом следует доставить пакеты данных к сетевым адресам IPv6

**Характеристики записей:**

* **Адрес по умолчанию** – 0.0.0.0 с маской 0.0.0.0, далее идут специфические адреса
* **Сетевой адрес** – IP адрес сети, для которой задан данный маршрут
* **Маска сети** - маска, которая определяет диапазон IP-адресов внутри сети
* **Адрес шлюза** – IP адрес узла, через который пакеты будут направляться для достижения адресов
* **Интерфейс** - сетевой интерфейс, через который проходят пакеты для данного маршрута
* **Метрика** - числовое значение, которое определяет приоритет маршрута; чем меньше значение, тем предпочтительнее маршрут

1. В командном режиме на РК и на сервере определить IP-адреса поисковых систем в соответствии с вариантом из таблицы 5.1, пояснить результаты.

|  |  |
| --- | --- |
| Вариант | Название |
| 4 | yandex.com |

В начале нам выводится адрес главного роутера локальной сети.

Для РК это 192.168.0.1, для сервера 217.71.130.130. Это шлюз по умолчанию, адрес на который мы отправим наш запрос.

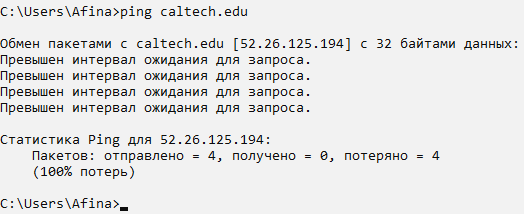
Сообщение "Не заслуживающий доверия ответ:" говорит о том, что выполняющий запрос DNS-сервер, не является владельцем зоны yandex.com т.е. записи для узла yandex.com в его базе отсутствуют, и для разрешения имени использовался рекурсивный запрос к другому DNS-серверу.

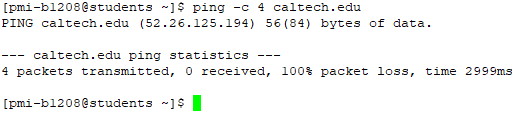
IP-адреса, связанные с доменом в обоих случаях – **5.255.255.88, 77.88.55.80, 77.88.55.77, 5.255.255.80**

1. В командном режиме на РК и на сервере определить IP-адрес узлов сети в соответствии с номером варианта, указанного в таблице 5.2, выполнить его пингование и трассировку.

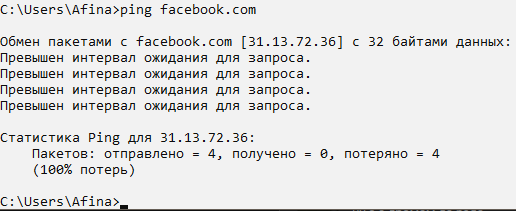
**Пингование**

|  |  |
| --- | --- |
| Вариант | Доменное имя |
| 4 | caltech.edu, facebook.com |





Для **caltech.edu** - было отправлено 4 пакета, но принято было 0

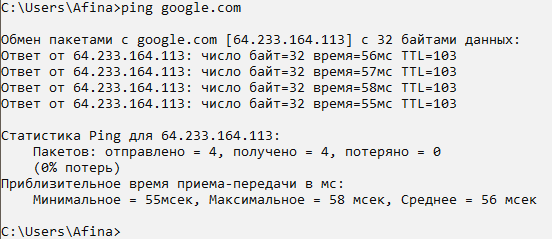




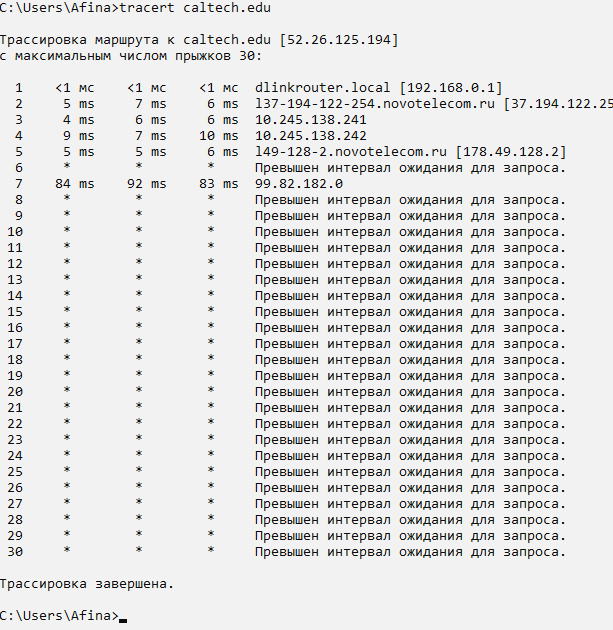
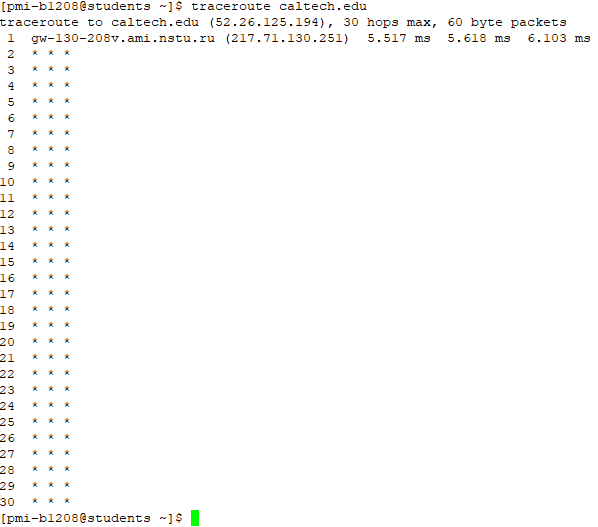
На РК для **facebook.com** было отправлено 4 пакета, но принято было 0

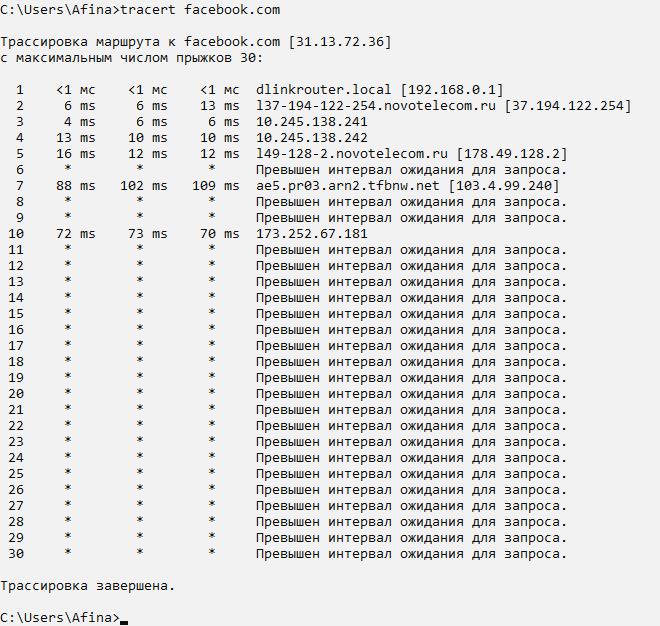
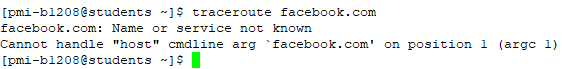
На сервере **facebook.com** является неизвестным доменом.

По некоторым причинам нам не удалось пропинговать **caltech.edu** и **facebook.com,** но чтобы показать работоспособность команды ping пропингуем google.com

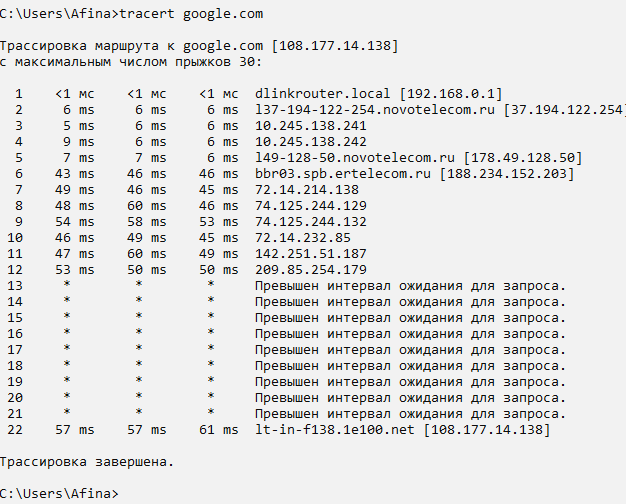
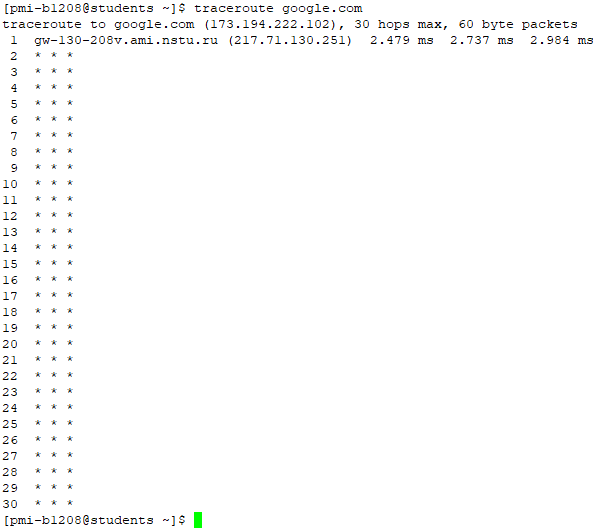


**Трассировка**

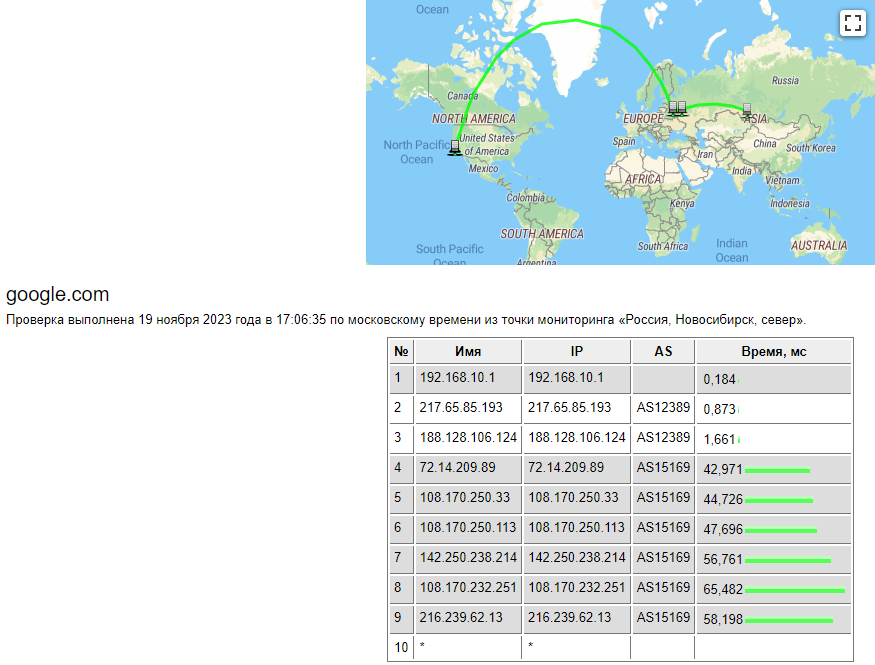


По некоторым причинам нам не удалось выполнить трассировку **caltech.edu** и **facebook.com,** но чтобы показать работоспособность команды tracert/traceroute выполним трассировку google.com

Исходя из скриншотов можно сделать вывод, что доступ к google.com недоступен на сервере, а на РК доступ есть и на трассировке видно путь.

8. С помощью интерактивных сетевых сервисов (например, **ping-admin.ru**) выполнить трассировку, определить местонахождение и владельца узла сети в соответствии с номером варианта, указанного в таблице 5.2. Результат трассировки в виде скриншота географической карты представить в отчете и выполнить его анализ. Начальный пункт трассировки – г. Новосибирск.



Место нахождение: США, Сан-Хосе

**Этап 2**

Реализовать Windows- или Linux-приложение, которое будет выполнять основные функции одной из утилит мониторинга сети в соответствии с таблицей 5.3. Результат работы функций должен быт

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Аналог | ОС | Функция |
| 4 | ping | Windows | запрос маски сети |

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#include <conio.h>

#include <cstdio>

#include <locale>

#include <stdio.h>

#include <winsock2.h>

#include <iphlpapi.h>

#include <icmpapi.h>

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

#pragma comment(lib, "ws2\_32.lib")

void main()

{

setlocale(LC\_ALL, "Russian");

char sendAdr[128], hostIP[128];

char\* sendData = "package";

WSADATA wsaData;

WORD sockVer = MAKEWORD(2, 2);

WSAStartup(sockVer, &wsaData);

printf("Введите адрес: ");

gets\_s(sendAdr);

struct hostent\* host = gethostbyname(sendAdr); //Получть IP адресс из введенного домена/IP адресса

if (host != NULL) //Если адресс найден

{

unsigned long minTime = LONG\_MAX, maxTime = 0, time = 0, pack\_rec = 0, timeout = 1000;

strcpy\_s(hostIP, inet\_ntoa(\*((in\_addr\*)host->h\_addr\_list[0]))); //Запись IP полученного из host в hostIP

unsigned long ipaddr = inet\_addr(hostIP); //Функция возвращает число, сод. подходящее двоичное представления заданного адреса

HANDLE hIcmpFile = IcmpCreateFile(); //Функция IcmpCreateFile открывает дескриптор, на котором можно выдавать эхо - запросы ICMP IPv4.

unsigned long massageSize = sizeof(ICMP\_ECHO\_REPLY) + sizeof(sendData); //Вычисление размера сообщение при помощи len(тех. данных + строки сообщения)

LPVOID massageBuffer = malloc(massageSize); //Задание буфера для эхо-запросов

if (strcmp(sendAdr, hostIP))

printf("Обмен пакетами с %s [%s] с %d байтами данных:\n", sendAdr, hostIP, massageSize);

else

printf("Обмен пакетами с %s с %d байтами данных:\n", sendAdr, massageSize);

for (int i = 0; i < 4; i++)

{

if (int echoReq = IcmpSendEcho(hIcmpFile, ipaddr, sendData, sizeof(sendData), NULL, massageBuffer, massageSize, timeout)) //Функция отправляет эхо-запрос ICMP IPv4 и возвращает любые ответы echo response. Вызов возвращается, когда истекает время ожидания или заполняется буфер ответа.

{

PICMP\_ECHO\_REPLY pEchoReply = (PICMP\_ECHO\_REPLY)massageBuffer;

printf(" Ответ от %s: ", hostIP);

printf("Число байт = %d ", massageSize);

printf("Время = %ld мсек ", pEchoReply->RoundTripTime);

printf("TTL = %d\n", (int)pEchoReply->Options.Ttl);

if (minTime > pEchoReply->RoundTripTime)

minTime = pEchoReply->RoundTripTime;

if (maxTime < pEchoReply->RoundTripTime)

maxTime = pEchoReply->RoundTripTime;

time += pEchoReply->RoundTripTime;

pack\_rec += 1;

}

else

printf(" Превышен интервал ожидания\n");

}

printf("Статистика Ping для %s:\n", hostIP);

printf(" Пакетов: отправлено = 4, получено = %d, потеряно = %d\n", pack\_rec, 4 - pack\_rec);

printf(" (%.0f%% потерь)\n",(100.0 - pack\_rec/4.0\*100.0));

if (pack\_rec != 0) {

printf("Приблизительное время приема-передачи в мс:\n");

printf(" Минимальное = %ld мсек, Максимальное = %ld мсек, Среднее = %ld мсек\n", minTime, maxTime, time / pack\_rec);

}

//PIP\_ADAPTER\_INFO pAdapterInfo = NULL;

//ULONG ulOutBufLen = sizeof(IP\_ADAPTER\_INFO);

//pAdapterInfo = (IP\_ADAPTER\_INFO\*)malloc(sizeof(IP\_ADAPTER\_INFO));

//GetAdaptersInfo(pAdapterInfo, &ulOutBufLen);

//printf("Маска локальной подсети: %s\n", pAdapterInfo->IpAddressList.IpMask.String);

}

else

printf("Данного адреса не существует!\n");

system("pause");

}

Результаты работы программы:

