МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №6

по дисциплине: Компьютерные сети
тема: «Протоколы DHCP и DNS»

Выполнил: ст. группы ПВ-223
Игнатьев Артур Олегович

Проверили:
Рубцов Константин Анатольевич

Белгород 2025 г.

Цель работы: изучить протоколы DHCP, DNS и составить программы согласно заданию.

Краткие теоретические сведения

Протокол DHCP

DHCP (англ. DynamicHostConfigurationProtocol — протокол динамической конфигурации узла) - это сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP, и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве сетей TCP/IP.

Распределение IP-адресов

Протокол DHCP предоставляет три способа распределения IP-адресов:

1. Ручное распределение. При этом способе сетевой администратор сопоставляет аппаратному адресу (для Ethernet сетей это MAC-адрес) каждого клиентского компьютера определённый IP-адрес. Фактически, данный способ распределения адресов отличается от ручной настройки каждого компьютера лишь тем, что сведения об адресах хранятся централизованно (на сервере DHCP), и потому их проще изменять при необходимости.

2. Автоматическое распределение. При данном способе каждому компьютеру на постоянное использование выделяется произвольный свободный IP-адрес из определённого

администратором диапазона.

3. Динамическое распределение. Этот способ аналогичен автоматическому распределению, за исключением того, что адрес выдаётся компьютеру не на постоянное пользование, а на определённый срок. Это называется арендой адреса. По истечении срока аренды IP-адрес вновь считается свободным, и клиент обязан запросить новый. Кроме того, клиент сам может отказаться от полученного адреса.

Некоторые реализации службы DHCP способны автоматически обновлять записи DNS, соответствующие клиентским компьютерам, при выделении им новых адресов. Это производится при помощи

протокола обновления DNS, описанного в RFC 2136.

Помимо IP-адреса, DHCP также может сообщать клиенту дополнительные параметры, необходимые для нормальной работы в сети. Эти параметры называются опциями DHCP. Список стандартных

опций можно найти в RFC 2132. Некоторыми из наиболее часто используемых опций являются:

- IP-адрес маршрутизатора по умолчанию;
- маска подсети;
- адреса серверов DNS;
- имя домена DNS.

Протокол DHCP является клиент-серверным, то есть в его работе участвуют клиент DHCP и сервер DHCP. Передача данных производится при помощи протокола UDP, при этом сервер принимает сообщения от клиентов на порт 67 и отправляет сообщения клиентам на порт 68.

Все сообщения протокола DHCP разбиваются на поля, каждое из которых содержит определённую информацию. Все поля, кроме последнего (поля опций DHCP), имеют фиксированную длину.

Рассмотрим пример процесса получения IP-адреса клиентом от сервера DHCP. Предположим, клиент ещё не имеет собственного IP- адреса, но ему известен его предыдущий адрес — 192.168.1.100. Процесс состоит из четырёх этапов.

1. Обнаружение DHCP.

Вначале клиент выполняет широковещательный запрос по всей физической сети с целью обнаружить доступные DHCP-серверы. Он отправляет сообщение типа DHCPDISCOVER, при этом в качестве IP-адреса источника указывается 0.0.0.0 (так как компьютер ещё не

имеет собственного IP-адреса), а в качестве адреса назначения - широковещательный адрес 255.255.255.255.

Клиент заполняет несколько полей сообщения начальными значениями:

- В поле xid помещается уникальный идентификатор транзакции, который позволяет отличать данный процесс получения IP-адреса от других, протекающих в то же время.
- В поле chaddr помещается аппаратный адрес (MAC-адрес) клиента.
- В поле опций указывается последний известный клиенту IP-адрес. В данном примере это 192.168.1.100. Это необязательно и может быть проигнорировано сервером.

Сообщение DHCPDISCOVER может быть распространено за пределы локальной физической сети при помощи специально настроенных агентов ретрансляции DHCP, перенаправляющих поступающие от клиентов сообщения DHCP серверам в других подсетях.

2. Предложение DHCP.

Получив сообщение от клиента, сервер определяет требуемую конфигурацию клиента в соответствии с указанными сетевым администратором настройками. Пусть в данном случае DHCP-сервер согласен с запрошенным клиентом адресом 192.168.1.100. Сервер отправляет ему ответ (DHCPOFFER), в котором предлагает конфигурацию. Предлагаемый клиенту IP-адрес указывается в поле yiaddr. Прочие параметры (такие, как адреса маршрутизаторов и DNS-серверов) указываются в виде опций в соответствующем поле.

Далее это сообщение DHCP-сервер отправляет хосту, пославшему DHCPDISCOVER, на его MAC, при определенных обстоятельствах сообщение может распространяться как широковещательная рассылка. Клиент может получить несколько различных предложений DHCP от разных серверов; из них он должен выбрать то, которое его «устраивает».

3. Запрос DHCP.

Выбрав одну из конфигураций, предложенных DHCP-серверами, клиент отправляет запрос DHCP (DHCPREQUEST). Он рассылается широковещательно; при этом к опциям, указанным клиентом в сообщении DHCPDISCOVER, добавляется специальная опция -

идентификатор сервера - указывающая адрес DHCP-сервера, выбранного клиентом (в данном случае - 192.168.1.1).

4. Подтверждение DHCP.

Наконец, сервер подтверждает запрос и направляет это подтверждение (DHCPACK) клиенту. После этого клиент должен настроить свой сетевой интерфейс, используя предоставленные опции.

Протокол DNS

В стеке TCP/IP применяется доменная система имен, которая имеет иерархическую древовидную структуру.

Иерархия доменных имен аналогична иерархии имен файлов, принятой в файловых системах. В отличие от имен файлов запись доменного имени начинается с самой младшей составляющей, и заканчивается самой старшей. Составные части доменного имени отделяются друг от друга точкой.

Разделение имени на части позволяет разделить административную ответственность за назначение уникальных имен между различными людьми или организациями в пределах своего уровня иерархии. Разделение административной ответственности позволяет решить проблему образования уникальных имен без

взаимных консультаций между организациями, отвечающими за имена одного уровня иерархии. Поэтому должна существовать одна организация, отвечающая за назначение имен верхнего уровня иерархии.

Совокупность имен, у которых несколько старших составных частей совпадают, образуют домен (domain) имен.

Если один домен входит в другой домен как его составная часть, то такой домен могут называть поддоменом (subdomain). Обычно поддомен называют по имени той его старшей составляющей, которая

отличает его от других поддоменов. Имя поддомену назначает администратор вышестоящего домена. Хорошей аналогией домена является каталог файловой системы.

По аналогии с файловой системой в доменной системе имен различают краткие имена, относительные имена и полные доменные имена. Краткое имя - это имя конечного узла

сети. Относительное имя - это составное имя, начинающееся с некоторого уровня иерархии, но не самого верхнего. Например, wwwl.zil — это относительное имя. Полное доменное имя (fullyqualifieddomainname, FQDN) включает составляющие всех уровней иерархии, начиная от краткого имени и кончая корневой точкой.

Каждый домен администрируется отдельной организацией, которая обычно разбивает свой домен на поддомены и передает функции администрирования этих поддоменов другим организациям. Чтобы получить доменное имя, необходимо зарегистрироваться в какой-либо организации, которой организация InterNIC делегировала свои полномочия по распределению имен доменов. В России такой организацией является РосНИИРОС, которая отвечает за делегирование имен поддоменов в домене ru.

Доменная система имен реализована в Интернете, но она может работать и как автономная система имен в любой крупной корпоративной сети, которая также использует стек TCP/IP, но не связана с Интернетом.

Соответствие между доменными именами и IP-адресами может устанавливаться как средствами локального хоста, так и средствами централизованной службы. На раннем этапе развития Интернета на

каждом хосте вручную создавался текстовый файл с известным именем hosts.txt. Этот файл состоял из некоторого количества строк, каждая из которых содержала одну пару «IP-адрес — доменное имя».

В настоящее время используется масштабируемая служба для разрешения имен — система доменных имен (DomainNameSystem, DNS). Служба DNS использует в своей работе протокол типа «клиент-

сервер». В нем определены DNS-серверы и DNS-клиенты. DNS-серверы поддерживают распределенную базу отображений, а DNS-клиенты обращаются к серверам с запросами о разрешении доменного имени в IP-адрес.

Служба DNS использует текстовые файлы, которые администратор подготавливает вручную. Однако служба DNS опирается на иерархию доменов, и каждый сервер службы DNS хранит только часть имен сети, а не все имена. При росте количества узлов в сети проблема масштабирования решается созданием новых доменов и поддоменов имен и добавлением в службу DNS новых серверов.

Для каждого домена имен создается свой DNS-сервер. Обычно сервер домена хранит только имена, которые заканчиваются на следующем ниже уровне иерархии по сравнению с именем домена. Именно при такой организации службы DNS нагрузка по разрешению имен распределяется более-менее равномерно между всеми DNS-серверами сети. Каждый DNS-сервер кроме таблицы отображений имен содержит ссылки на DNS-серверы своих поддоменов. Эти ссылки связывают отдельные DNS-серверы в единую службу DNS. Ссылки представляют собой IP-адреса соответствующих серверов. Для обслуживания корневого домена выделено несколько дублирующих друг друга DNS-серверов, IP-адреса которых являются широко известными (их можно узнать в InterNIC).

Процедура разрешения DNS-имени во многом аналогична процедуре поиска файловой системой адреса файла по его символьному имени. Для определения IP-адреса по доменному имени необходимо просмотреть все DNS-серверы, обслуживающие цепочку поддоменов, входящих в имя хоста, начиная с корневого домена.

Существенным отличием является то, что файловая система расположена на одном компьютере, а служба DNS по своей природе является распределенной.

Существует две основные схемы разрешения DNS-имен. В первом варианте работу по поиску IP-адреса координирует DNS-клиент, последовательно обращаясь к DNS-серверам, начиная с корневого. Такая схема взаимодействия называется нерекурсивной, или итеративной. Так как эта схема загружает клиента достаточно сложной работой, то она применяется редко. Во втором варианте DNS-клиент запрашивает локальный DNS-сервер, обслуживающий поддомен, к которому принадлежит имя клиента. Если локальный DNS-сервер знает ответ, то он сразу же возвращает его клиенту. Это может соответствовать случаю, когда запрошенное имя входит в тот же поддомен, что и имя клиента, а также случаю, когда сервер уже узнавал данное соответствие для другого клиента и сохранил его в своем кэше. Если локальный сервер не знает ответ, то он выполняет итеративные запросы к корневому серверу и к нижним по иерархии. Получив ответ, он передает его клиенту. Такая схема называется косвенной, или рекурсивной.

Для ускорения поиска IP-адресов DNS-серверы широко применяют процедуру кэширования проходящих через них ответов. Чтобы служба DNS могла оперативно отрабатывать изменения, происходящие в сети, ответы кэшируются на определенное время — обычно от нескольких часов до нескольких дней.

Используемые функции

Библиотека Winsock:

• gethostbyname – получает IP-адрес по доменному имени (устаревшая, но используемая в коде).

name – указатель на строку с доменным именем,

Возвращает: указатель на структуру hostent с результатами

• gethostbyaddr – получает доменное имя по IP-адресу (устаревшая).

addr – указатель на IP-адрес,

len – длина адреса (4 для IPv4),

type – тип адреса (AF_INET для IPv4),

Возвращает: указатель на структуру hostent

• inet_addr – преобразует строку с IP-адресом в бинарный формат.

cp – строка с IP (например "192.168.1.1"),

Возвращает: IP-адрес в сетевом порядке байт

• inet_ntoa – преобразует бинарный IP в строку.

in – структура in_addr с адресом,

Возвращает: указатель на строку с IP

• WSAStartup – инициализирует Winsock (обязателен перед использованием других функций).

wVersionRequested – запрашиваемая версия (MAKEWORD(2,2)),

lpWSAData – указатель на структуру WSADATA

• WSAGetLastError – получает код последней ошибки Winsock.

Не принимает параметров,

Возвращает: целочисленный код ошибки

Библиотека IPHlpApi:

• GetAdaptersInfo – возвращает информацию о сетевых адаптерах.

AdapterInfo – указатель на буфер с данными адаптеров,

SizePointer – указатель на размер буфера (при недостатке размера возвращает ERROR_BUFFER_OVERFLOW с требуемым размером)

• GetInterfaceInfo – возвращает данные сетевых интерфейсов.

pIfTable – указатель на буфер таблицы интерфейсов,

dwOutBufLen – указатель на размер буфера (при ERROR_INSUFFICIENT_BUFFER требует перевызова)

• IpRenewAddress – обновляет IP-адрес адаптера.

AdapterInfo – указатель на структуру адаптера для обновления

• IpReleaseAddress – освобождает IP-адрес адаптера.

AdapterInfo – указатель на структуру адаптера для освобождения

• GetIpNetTable – возвращает ARP-таблицу (устаревший вариант).

pIpNetTable – указатель на буфер ARP-таблицы,

pdwSize – указатель на размер буфера,

bOrder – флаг сортировки (TRUE/FALSE)

• GetIfTable – возвращает таблицу интерфейсов.

pIfTable – указатель на буфер таблицы,

pdwSize – указатель на размер буфера,

bOrder – флаг сортировки (TRUE/FALSE)

• GetNetworkParams – возвращает сетевые параметры.

pFixedInfo – указатель на структуру параметров,

pOutBufLen – указатель на размер буфера

• GetTcpTable – возвращает таблицу TCP-соединений.

TcpTable – указатель на буфер таблицы,

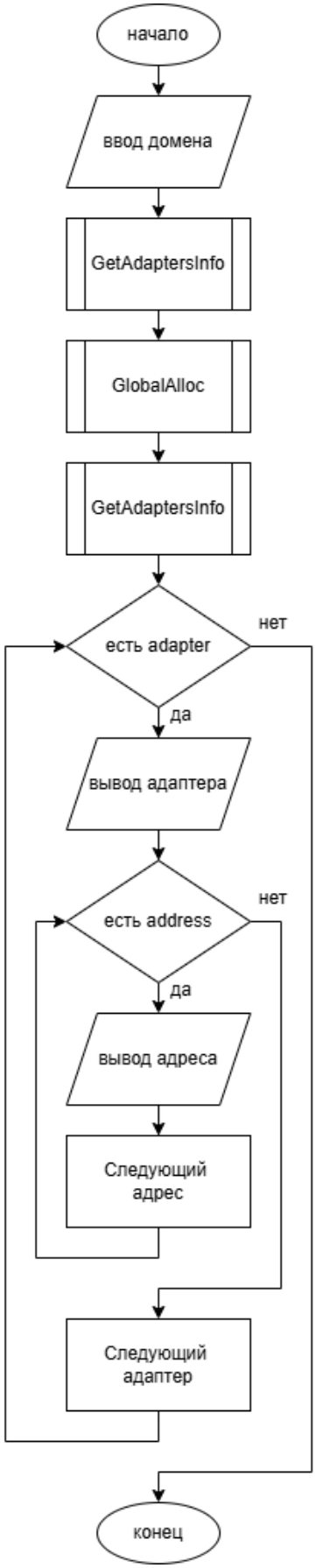
SizePointer – указатель на размер буфера,

Order – флаг сортировки (TRUE/FALSE)

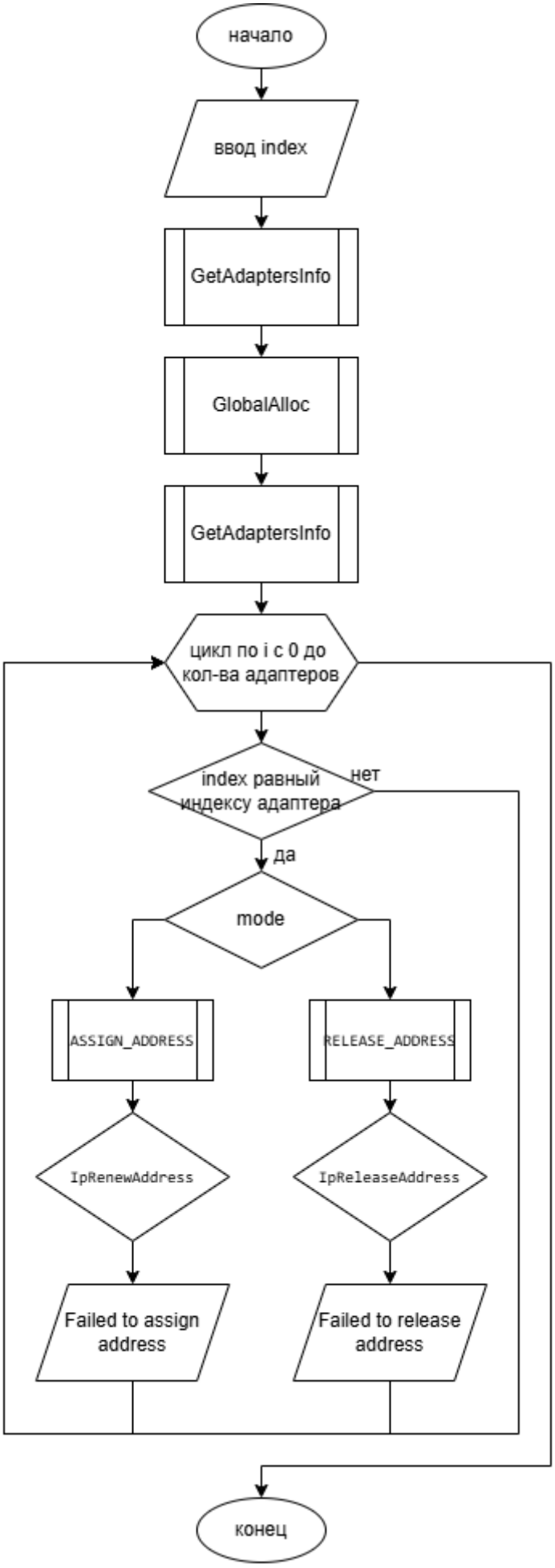
Разработка программы. Блок-схемы программы

DHCP

view_ip():

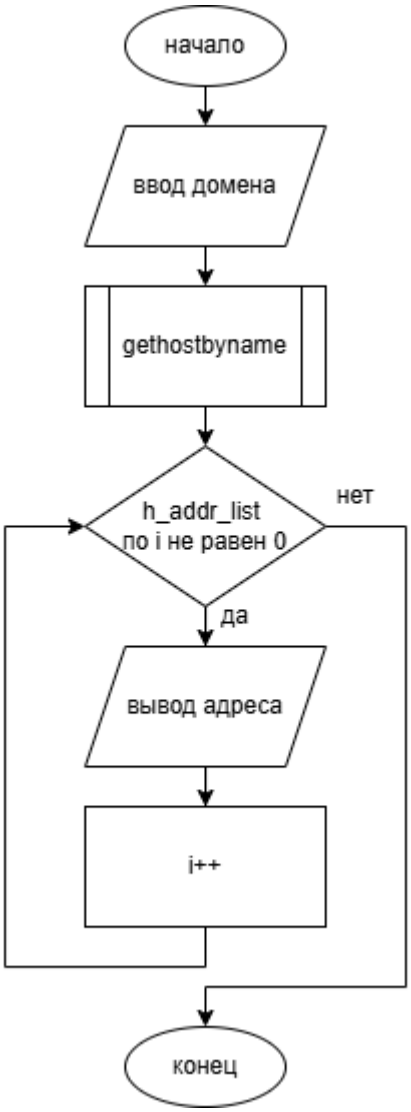


control_ip_address(AddressControllerMode mode):

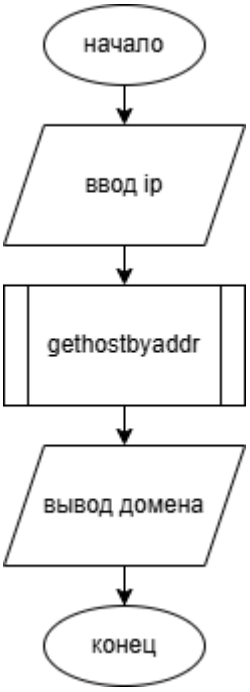


DNS

view_ip_by_domain():



view_domain_by_ip():



Анализ функционирования программы

DHCP

Вывод адаптеров

```
Select action:
  1. Show table
  2. Assign address
  3. Release address
  4. Exit
1
Adapter: {E5788AAB-E5E2-4A4C-BCFA-341C0DCE09E6}
Index: 13
DHCP enabled: yes
DHCP address: 192.168.1.1
----------
Context: 1694607552
IP address: 192.168.1.101
Mask: 255.255.255.0
```

Запрос IP адреса

```
Select action:
  1. Show table
  2. Assign address
  3. Release address
  4. Exit
2
Enter adapter index to assign new address
13
Select action:
  1. Show table
  2. Assign address
  3. Release address
  4. Exit
1
Adapter: {E5788AAB-E5E2-4A4C-BCFA-341C0DCE09E6}
Index: 13
DHCP enabled: yes
DHCP address: 192.168.1.1
----------
Context: 1694607552
IP address: 192.168.1.101
Mask: 255.255.255.0
```

Сброс IP адреса

```
Select action:
  1. Show table
  2. Assign address
  3. Release address
  4. Exit
3
Enter adapter index to release address
13
Select action:
  1. Show table
  2. Assign address
  3. Release address
  4. Exit
1
Adapter: {E5788AAB-E5E2-4A4C-BCFA-341C0DCE09E6}
Index: 13
DHCP enabled: yes
DHCP address: 0.0.0.0
----------
Context: 2541616809
IP address: 169.254.125.151
Mask: 255.255.0.0
```

DNS

```
1. Find IP address by domain name
2. Find domain name by IP address
3. Exit
Choice:
1
Enter domain: ya.ru
        IP address #1: 77.88.44.242
        IP address #2: 5.255.255.242
        IP address #3: 77.88.55.242

1. Find IP address by domain name
2. Find domain name by IP address
3. Exit
Choice:
2
Enter IP address: 77.88.44.242
        Domain name for IP address: ya.ru

1. Find IP address by domain name
2. Find domain name by IP address
3. Exit
Choice:
2
Enter IP address: 5.255.255.242
        Domain name for IP address: ya.ru
```

Код программы:

Файл DHCP.cpp

```
#include <iostream>
#include <WinSock.h>
#include <IPHlpApi.h>
#include <string>

#define STR_BUF_SIZE 1024  // размер буфера строк

using namespace std;

// Перечисление режимов управления IP-адресом
enum AddressControllerMode
{
    ASSIGN_ADDRESS,   // Назначить IP-адрес
    RELEASE_ADDRESS   // Освободить IP-адрес
};

// Вывод сообщения об ошибке с кодом последней ошибки WinSock
void throw_err_with_code()
{
    std::string err_msg = "Error with code: ";
    err_msg += to_string(WSAGetLastError());  // получение кода последней ошибки
    throw std::runtime_error(err_msg);        // выбрасываем исключение с текстом ошибки
}

// Отображение информации о всех сетевых адаптерах
void view_ip()
{
    PIP_ADAPTER_INFO adapter_info, adapter;
    ULONG adapter_info_value;
    int error;

    adapter_info_value = 0;

    // Запрашиваем размер буфера, необходимого для хранения информации об адаптерах
    error = GetAdaptersInfo(NULL, &adapter_info_value);
    if ((error != 0) && (error != ERROR_BUFFER_OVERFLOW))
    {
        cout << "Buffer overflow" << endl;
        return;
    }

    // Выделяем память под буфер для информации об адаптерах
    if ((adapter_info = (PIP_ADAPTER_INFO)GlobalAlloc(GPTR, adapter_info_value)) == NULL)
    {
        cout << "Memory allocation error" << endl;
        return;
    }

    // Получаем информацию об адаптерах
    if (GetAdaptersInfo(adapter_info, &adapter_info_value) != 0)
    {
        cout << "Unknown error" << endl;
        return;
    }

    adapter = adapter_info;
    while (adapter)
    {
        // Выводим основную информацию об адаптере
        cout << "Adapter: " << adapter->AdapterName
             << "\nIndex: " << adapter->Index
             << "\nDHCP enabled: " << (adapter->DhcpEnabled ? "yes" : "no")
```

```
             << "\nDHCP address: " << adapter->DhcpServer.IpAddress.String << endl;

        // Выводим IP-адреса, привязанные к адаптеру
        PIP_ADDR_STRING address = &(adapter->IpAddressList);
        while (address)
        {
            cout << "----------" << endl;
            cout << "Context: " << address->Context << endl;
            cout << "IP address: " << address->IpAddress.String << endl;
            cout << "Mask: " << address->IpMask.String << endl;
            address = address->Next;
        }
        cout << endl << endl;

        adapter = adapter->Next; // переходим к следующему адаптеру
    }
}

// Управление IP-адресом: назначение или освобождение
void control_ip_address(AddressControllerMode mode)
{
    unsigned int index;
    cout << "Enter adapter index "
         << (mode == ASSIGN_ADDRESS ? "to assign new address" : "to release address")
         << endl;
    cin >> index;

    DWORD interface_info_size = 0;
    PIP_INTERFACE_INFO interface_info;

    // Получаем необходимый размер буфера для хранения информации об интерфейсах
    if (GetInterfaceInfo(NULL, &interface_info_size) != ERROR_INSUFFICIENT_BUFFER)
    {
        cerr << "Buffer error" << endl;
        return;
    }

    // Выделяем буфер нужного размера
    if ((interface_info = (PIP_INTERFACE_INFO)GlobalAlloc(GPTR, interface_info_size)) == NULL)
    {
        cerr << "Not enough memory" << endl;
        return;
    }

    // Получаем таблицу интерфейсов
    if (GetInterfaceInfo(interface_info, &interface_info_size) != 0)
    {
        cerr << "Failed to get table" << endl;
        return;
    }

    // Ищем адаптер с нужным индексом и применяем действие
    for (int i = 0; i < interface_info->NumAdapters; ++i)
    {
        if (index == interface_info->Adapter[i].Index)
        {
            switch (mode)
            {
            case ASSIGN_ADDRESS:
                if (IpRenewAddress(&interface_info->Adapter[i]) != 0)
                {
                    cerr << "Failed to assign address" << endl;
                    return;
                }
                break;
            case RELEASE_ADDRESS:
                if (IpReleaseAddress(&interface_info->Adapter[i]) != 0)
```

```
                {
                    cerr << "Failed to release address" << endl;
                    return;
                }
                break;
            }
        }
    }
}

int main(int args, char *argv[])
{
    int selected_value = 1;

    while (selected_value != 4)
    {
        // Меню действий
        cout << "Select action:\n"
             << "  1. Show table\n"
             << "  2. Assign address\n"
             << "  3. Release address\n"
             << "  4. Exit" << endl;
        cin >> selected_value;

        switch (selected_value)
        {
        case 1:
            view_ip(); // показать информацию об IP-адресах
            break;
        case 2:
            control_ip_address(ASSIGN_ADDRESS); // назначить IP-адрес
            break;
        case 3:
            control_ip_address(RELEASE_ADDRESS); // освободить IP-адрес
            break;
        default:
            break;
        }
    }

    return 0;
}
```

Файл DNS.cpp

```
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdio.h>
#include <iostream>
#include <string>

#define STR_BUF_SIZE 1024  // размер буфера для ввода строки

using namespace std;

// Выбрасывает исключение с текстом и кодом последней ошибки Winsock
void throw_error_with_code()
{
    std::string error_msg = "Error with code: ";
    error_msg += to_string(WSAGetLastError());  // получаем код ошибки
    throw std::runtime_error(error_msg);        // выбрасываем исключение с сообщением
}

// Получение IP-адресов по доменному имени
void view_ip_by_domain()
```

```
{
    struct hostent *remote_host;     // структура с информацией о хосте
    struct in_addr addr = {0};       // структура для IP-адреса
    char host_name[STR_BUF_SIZE];    // буфер для доменного имени

    cout << "Enter domain: ";
    cin.ignore();                    // очистка буфера ввода
    cin.getline(host_name, STR_BUF_SIZE);  // считываем строку

    remote_host = gethostbyname(host_name);  // получаем IP-адрес по доменному имени
    if (remote_host == NULL)
    {
        cerr << "Remote host not defined";  // если не найден
        return;
    }

    if (WSAGetLastError())
        throw_error_with_code();    // обработка возможной ошибки

    if (remote_host->h_addrtype == AF_INET)  // проверяем, что адрес IPv4
    {
        int i = 0;
        // Перебираем список адресов
        while (remote_host->h_addr_list[i] != 0)
        {
            addr.s_addr = *(u_long *)remote_host->h_addr_list[i++];  // получаем очередной адрес
            printf("\tIP address #%d: %s\n", i, inet_ntoa(addr));    // печатаем адрес
        }
    }
    cout << endl;
}

// Получение доменного имени по IP-адресу
void view_domain_by_ip()
{
    struct hostent *remote_host;

    cout << "Enter IP address: ";
    char ip_addr[STR_BUF_SIZE];
    cin.ignore();                               // очистка потока ввода
    cin.getline(ip_addr, STR_BUF_SIZE);         // считываем IP-адрес как строку

    DWORD ip = inet_addr(ip_addr);              // преобразуем строку в число
    remote_host = gethostbyaddr((char *)&ip, 4, AF_INET);  // ищем имя по IP
    if (remote_host == NULL)
    {
        cerr << "Could not resolve IP address" << endl;
        return;
    }

    // Выводим найденное доменное имя
    cout << "\tDomain name for IP address: "
         << remote_host->h_name
         << '\n'
         << endl;
}

int main()
{
    while (true)
    {
        WSADATA wsa_data;
        int result;
        struct hostent *remote_host;
        struct in_addr addr = {0};

        // Инициализируем библиотеку Winsock
```

```
        result = WSAStartup(MAKEWORD(2, 2), &wsa_data);
        if (result != 0)
        {
            // Если инициализация не удалась — завершаем программу
            cerr << "WSAStartup failed: " << result << endl;
            return 1;
        }

        // Выводим меню
        cout << "1. Find IP address by domain name\n"
             << "2. Find domain name by IP address\n"
             << "3. Exit\n"
             << "Choice: "
             << endl;

        int selected_value;
        cin >> selected_value;

        switch (selected_value)
        {
        case 1:
            view_ip_by_domain();    // поиск IP по доменному имени
            break;
        case 2:
            view_domain_by_ip();    // поиск домена по IP
            break;
        case 3:
        default:
            return 0;               // выход из программы
        }
    }
    WSACleanup();
}
```