Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный университет имени П.М. Машерова»
Кафедра информатики и информационных технологий

В.В. Новый, Т.Г. Алейникова

СЕТЕВЫЕ ТЕХНОЛОГИИ И СЕРВИСЫ

Методические рекомендации

Витебск
ВГУ имени П.М. Машерова
2017

УДК 004.41(075.8)
ББК 32.971.35я73
Н76

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 1 от 19.10.2017 г.

Авторы: старший преподаватель кафедры прикладного и системного программирования ВГУ имени П.М. Машерова **В.В. Новый;** доцент кафедры информатики и информационных технологий ВГУ имени П.М. Машерова, кандидат физико-математических наук **Т.Г. Алейникова**

Р е ц е н з е н т :
доцент кафедры математики и информационных технологий
УО «ВГТУ», кандидат физико-математических наук
Т.В. Никонова

Новый, В.В.

Н76 Сетевые технологии и сервисы : методические рекомендации / В.В. Новый, Т.Г. Алейникова. – Витебск : ВГУ имени П.М. Машерова, 2017. – 50 с.

Учебное издание раскрывает основы использования современных сетевых технологий как на программном уровне, так и на уровне прикладных сервисов.

Методические рекомендации адресованы студентам очной и заочной форм получения образования специальности «Программное обеспечение информационных технологий», а также могут использоваться при изучении дисциплин «Компьютерные системы и сети», «Облачные сервисы».

УДК 004.41(075.8)
ББК 32.971.35я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одной из особенностей профессии программиста в настоящее время является высокая степень включенности в систему глобальных коммуникаций. Открытость, прозрачность, сотрудничество на всех этапах – стандарты современной разработки ПО. Для реализации таких стандартов широкое применение находят так называемые «облачные» сервисы. Миграция к «облачным» технологиям – устойчивая мировая тенденция. В Концепции информатизации системы образования Республики Беларусь на период до 2020 года [5] формирование образовательной среды на базе «облачных» технологий отнесено к числу первоочередных задач.

Предлагаемое учебное издание раскрывает основы функционирования и программирования сервисов компьютерных сетей. В практической части рассматриваются прикладные протоколы, лежащие в основе распространенных сетевых технологий, а также разработка собственных сетевых приложений.

На базе современных web-сервисов студенты учатся использовать методы и средства достижения образовательных целей в мобильной информационной среде, эффективно интегрировать средства информатизации в свою будущую профессиональную деятельность.

Методические рекомендации представлены по целому ряду наиболее популярных в настоящее время сервисов: системам коллективного поиска и хранения информации, средствам визуализации информации, инструментальным средствам облачного программирования. Это издание может служить руководством при изучении студентами IT-специальностей таких дисциплин, как «Компьютерные системы и сети», «Облачные сервисы». Предлагаемые задания для самостоятельной работы могут служить ориентиром для организации самообучения. Для студентов заочной формы обучения предлагается достаточно подробный теоретический материал, необходимый для выполнения практических заданий.

1. НАСТРОЙКА КОМПЬЮТЕРА ДЛЯ РАБОТЫ В СЕТИ

Для работы в компьютерной сети, кроме наличия проводного или беспроводного сетевого адаптера (сетевой карты), а также физического подключения, должна быть установлена поддержка стека протоколов TCP/IP и задана адресная информация: IP-адрес, маска подсети, адреса DNS-серверов и шлюза по умолчанию.

1.1. Настройка сети в Windows

Настройка видимости компьютера в сети. Современные версии Windows поддерживают 3 режима работы сети: «домашняя сеть» (home network), «рабочая сеть» (work network) и «публичная сеть» (public network). «Домашняя сеть» предусматривает включение функции Network Discovery (Сетевое обнаружение) и возможности присоединения к Home Group (Домашней группе, которая предоставляет простой способ доступа к разделяемым ресурсам в пределах до-

машней сети по заданному паролю). «Рабочая сеть» предусматривает включение Network Discovery и отключение Home Group. «Публичная сеть» задает наиболее безопасные настройки: Network Discovery и Home Group – отключены.

Настройка TCP/IP. В Windows предлагается 3 способа настройки адресной информации:

- Ручная настройка. IP-адрес задается вручную (т.н. статический IP-адрес);
- Динамическое назначение адреса. IP-адрес назначается сервером DHCP. Этот вариант используется по умолчанию;
- Назначение альтернативного адреса (автонастройка). Если выбран вариант использования DHCP, но DHCP-сервер не доступен, то сетевому адаптеру автоматически назначается IP-адрес из приватного диапазона 169.254.0.1 – 196.254.255.254 с маской /16 или выбранный пользователем альтернативный адрес. Этот вариант применяется только для IPv4.

Выполним настройку IP-адреса. В Центре управления сетями и общим доступом (Network and Sharing Center) сделайте клик на Local Area Connection. Откроется окно статуса соединения. Кликните на кнопку Свойства для открытия окна Свойств.

Выберите версию TCP/IP, которую Вы используете, и кликните Свойства. Откроется окно настройки TCP/IP.

Если в сети используется динамическое назначение IP-адресов – оставьте поля незаполненными (Obtain an IP address automatically).

Если в сети используется статическая IP адресация, выберите Use the following IP address и задайте IP-адрес (IP address), маску сети (subnet mask) и шлюз по умолчанию (default gateway). Вы также можете задать IP адреса DNS-серверов при наличии этой информации.

Дважды кликните Ok и закройте окно статуса соединения.

Для лэптопов, которые переносятся из одной сети в другую и одна из сетей использует статические IP адреса, можно задать на первой вкладке настройки IP-адреса Obtain an IP address automatically, затем на вкладке Alternate Configuration задать настройки статического IP адреса. Лэптоп попытается получить IP адрес от DHCP-сервера. Если это не получится, он применит указанный статический IP-адрес.

ЗАДАНИЕ. НАСТРОЙКА ДИНАМИЧЕСКОГО НАЗНАЧЕНИЯ IP-АДРЕСА

Загрузите виртуальную машину с Windows и войдите с правами администратора в систему. Настройте Windows на получение IP-адреса от DHCP-сервера. Проверьте работоспособность выполненных Вами настроек.

Для быстрого получения информации о сетевых настройках в Windows может быть использована утилита командной строки ipconfig. Кроме просмотра сетевых настроек она также позволяет освобождать и обновлять IP-адреса, полученные по протоколу DHCP. Команда ipconfig /release – освобождает текущий IP-адрес, команда ipconfig /renew – заставляет запросить IP-адрес у DHCP-сервера.

ЗАДАНИЕ. ИССЛЕДОВАНИЕ ПРОТОКОЛА DHCP

Распакуйте сетевой сниффер Wireshark в образе виртуальной машины Windows, запустите захват трафика на проводном сетевом интерфейсе локальной сети. При помощи утилиты ipconfig сбросьте текущий IP-адрес и запросите выделение нового IP-адреса. В протоколе захвата трафика в Wireshark отследите запросы DHCP-клиента и ответы DHCP-сервера. Определите очередность и направление передачи команд протокола DHCP. Определите какая адресная информация принимается от DHCP-сервера и в каком из сообщений она впервые передается.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. НАСТРОЙКА СТАТИЧЕСКОГО IP-АДРЕСА

Перенастройте Windows в виртуальной машине на использование статического IP-адреса. Отключите использование IPv6. Задайте для сетевого интерфейса IP-адрес, сетевую маску, шлюз по умолчанию и адреса DNS-серверов. Проверьте работоспособность настроенной Вами конфигурации.

1.2. Настройка сети в Linux

В современных дистрибутивах Linux используется два основных способа настройки адресной информации для работы в компьютерной сети – использование сервиса NetworkManager или Wicd (чаще всего для лэптопов и настольных компьютеров) и настройка через файлы конфигурации.

NetworkManager позволяет задать как получение IP-адреса по протоколу DHCP, так и его статическую настройку. Для доступа к NetworkManager, как правило, используются средства окружения рабочего стола (апплеты). Например, в Gnome доступ к настройке NetworkManager осуществляется через иконку NetworkManager в верхней панели. Вы можете выбрать нужное сетевое соединение – проводное или беспроводное, после чего нажать кнопку Настроить (Configure). После этого следует выбрать способ настройки – Automatic (DHCP) или Ручное задание (Manual).

При получении адресной информации по протоколу DHCP NetworkManager автоматически вписывает её в необходимые файлы. Например, при получении адресов DNS-серверов, они вписываются в файл /etc/resolv.conf.

ЗАДАНИЕ. НАСТРОЙКА ДИНАМИЧЕСКОГО НАЗНАЧЕНИЯ IP-АДРЕСА ПРИ ПОМОЩИ NETWORKMANAGER

Загрузите виртуальную машину с Debian GNU/Linux. Ознакомьтесь с интерфейсом апплета настройки NetworkManager и настройте проводное соединение на получение IP-адреса от DHCP-сервера. Протестируйте выполненные Вами настройки. Просмотрите и запомните содержимое файла resolv.conf.

При выборе ручного задания адресной информации (Manual) Вы должны указать IP-адрес (Address) для выбранного интерфейса, маску подсети (Netmask), адрес шлюза по умолчанию (Gateway), DNS-серверов (DNS servers) и список поиска для доменов (Search domains). Последняя опция позволяет при работе с сетевыми командами использовать короткие относительные DNS-

имена. Например, при задании в списке поиска для доменов ad.vsu Вы можете обращаться к somehost.ad.vsu по имени somehost.

Стоит также отметить, что изменения настроек, сделанные для существующего соединения, будут применены после завершения редактирования и клика на имени соединения или его переименования.

ЗАДАНИЕ. НАСТРОЙКА СТАТИЧЕСКОГО IP-АДРЕСА ПРИ ПОМОЩИ NETWORKMANAGER

Перенастройте NetworkManager на использование статического IP-адреса для проводной сети. Протестируйте выполненные Вами настройки.

Для настройки IP-адреса вручную желательно вначале остановить сервис NetworkManager'а (или его альтернативу - Wicd), выполнив:

```
sudo /etc/init.d/network-manager stop
```

или

```
sudo systemctl stop NetworkManager.service
```

(`sudo service NetworkManager stop` для некоторых дистрибутивов).

Затем для настройки информации о получении IP-адреса следует отредактировать файл `/etc/network/interfaces`, отвечающий за настройку сетевых интерфейсов. Синтаксис файла interfaces достаточно прост, например:

```
auto eth0
iface eth0 inet static
address 192.168.1.1
netmask 255.255.255.0
gateway 192.168.1.111
dns-nameservers 192.168.1.100 8.8.8.8
```

- Интерфейс, который должен активироваться при загрузке ОС записывается скомандой `auto`. Традиционно, ядро присваивает проводным адаптерам Ethernet имена вида `eth0`, `eth1` и т.д. В ряде дистрибутивов сейчас используются т.н. предсказуемые имена сетевых интерфейсов, которые зависят от физического расположения адаптера вида `enp1s0`, `eno1` и т.д. Имя интерфейса в Вашей системе можно уточнить, например, из вывода команд `ip link` или `ifconfig -a`;
- Настройки интерфейса задаются командой вида `iface имя параметры`. Здесь параметры, как правило, задаются в виде `inet static` – для статического задания IPv4 адреса или `inet dhcp` – для получения IPv4 адреса по протоколу DHCP.
- При настройке статического IP-адреса в последующих строках идут такие настройки как `address` (IP-адрес), `netmask` (маска подсети), `gateway` (шлюз по умолчанию) с нужными значениями, записанными после пробельного символа.

Для применения настроек после их изменения следует перезапустить сетевую подсистему:

```
/etc/init.d/networking restart
```

Следует заметить, что большинство файлов настройки в Linux доступны

для редактирования только суперпользователю – root, поэтому Вы можете использовать для запуска от его имени текстового редактора команды gksu (для графического режима) или sudo (для консольного режима). Например, после нажатия сочетания клавиш Alt+F2 в оконной среде набрать:

```
gksu gedit
```

или

```
gksu leafpad
```

или для консоли:

```
sudo nano /etc/network/interfaces
```

ЗАДАНИЕ. НАСТРОЙКА СТАТИЧЕСКОГО IP-АДРЕСА ПРИ ПОМОЩИ РЕДАКТИРОВАНИЯ ФАЙЛА КОНФИГУРАЦИИ

Используя файлы конфигураций настройте Linux на использование статического IP-адреса для интерфейса проводной локальной сети. Перезапустите сетевую подсистему и протестируйте работоспособность выполненных Вами настроек.

Для использования DHCP в указанном выше файле следует заменить строку вида `iface eth3 inet static` на строку `iface eth3 inet dhcp` и удалить ненужные параметры.

Чтобы запросить после настройки у сервера DHCP IP-адрес можно выполнить команду:

```
dhclient eth3
```

где `eth3` – это имя настроенного Вами интерфейса.

Также следует помнить, что при перенастройке сети на статический адрес, Вы должны будете остановить работу dhclient, например, запуском с ключом `-r` или `-x`.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. НАСТРОЙКА ДИНАМИЧЕСКОГО НАЗНАЧЕНИЯ IP-АДРЕСА ПРИ ПОМОЩИ РЕДАКТИРОВАНИЯ ФАЙЛА КОНФИГУРАЦИИ

Используя файлы конфигурации перенастройте Linux на использование динамического IP-адреса. Перезапустите сетевую подсистему и протестируйте работоспособность выполненных Вами настроек.

2. ПРОТОКОЛЫ ПРИКЛАДНОГО УРОВНЯ

2.1. Протоколы прикладного уровня. Электронная почта

Протокол SMTP

Основным протоколом работы с электронной почтой является SMTP (Simple Mail Transfer Protocol, простой протокол передачи почты) [16]. Протокол SMTP поддерживает передачу сообщений электронной почты между произвольными узлами сети Internet. Полное описание протокола SMTP можно найти в RFC 2821. В настоящее время активно используется ESMTP (Extended SMTP) – расширение протокола SMTP, добавляющее новые возможности, в том числе, аутентификацию пользователя.

Описание команд протокола SMTP

Протокол SMTP работает поверх транспортного протокола TCP (обычно используется порт 25 TCP).

HELO домен – начало сеанса SMTP. Ответом на эту команду является код 250 с именем отвечающего домена.

MAIL FROM:<имя@домен> - начать новую транзакцию передачи сообщения. Код подтверждения – "250 OK".

RCPT TO: <имя@домен> - задать адрес получателя сообщения. Данная команда может повторяться несколько раз для отправки одного сообщения нескольким пользователям. Код ошибки – 550.

DATA – начало передачи сообщения. После получения кода подтверждения 354 отправитель должен передать заголовок сообщения и тело сообщения. Заголовок сообщения должен отделяться от тела сообщения пустой строкой (если заголовок отсутствует, перед телом сообщения должна быть передана хотя бы одна пустая строка). Для обозначения конца сообщения служит строка, содержащая одну точку "**.**". В заголовке сообщения обычно указываются адреса получателя и отправителя сообщения, дата и время отправки сообщения, тема и идентификатор сообщения, информация о способе представления данных, сведения о промежуточных узлах и т.д. Почтовые программы и сервера обычно самостоятельно заполняют поля заголовка сообщения. Некоторые поля заголовка (From, Subject) впоследствии отображаются почтовыми клиентами.

QUIT – завершить сеанс SMTP.

Пример протокола работы с SMTP-сервером

```
220 shttp.srv Simple Mail Transfer Service Ready
HELO shttp.srv
250 shttp.srv
MAIL FROM:<user1@shttp.srv>
250 Ok
RCPT TO:<user2@shttp.srv>
250 Ok
DATA
354 send the mail data, end with .
From: "User 1" <user1@shttp.srv>
To: "User 2" <user2@shttp.srv>
Date: 08 Oct 2017 11:20:00
Subject: My first letter
Message-ID: <123456789.user1@shttp.srv>

This is test.
Bye!
.
250 Ok
QUIT
221 Bye
```

Протокол POP3

POP3 – это простейший протокол для работы пользователя с содержимым своего почтового ящика. Его назначение - забрать почту из почтового ящика сервера на рабочую станцию клиента и удалить ее из почтового ящика на сервере. Всю дальнейшую обработку почтовое сообщение проходит на компьютере клиента.

POP3-сервер не отвечает за отправку почты, он работает только как универсальный почтовый ящик для группы пользователей. Когда пользователю необходимо отправить сообщение, он должен установить соединение с каким-либо SMTP-сервером и отправить туда свое сообщение по SMTP. Этот SMTP-сервер может быть тем же хостом, где работает POP3-сервер, а может располагаться совсем в другом месте (в другом домене или, вообще говоря, где угодно в Internet). Альтернативой протоколу POP3 является протокол IMAP4.

Полное описание протокола POP3 находится в RFC 1939.

Описание команд POP3-сервера

Протокол POP3 работает поверх транспортного протокола TCP (обычно используется порт 110). Соединение с POP3-сервером устанавливается по инициативе клиента и прекращается сервером после получения от клиента команды завершения соединения.

Ответы POP3-сервера на команды состоят из статус-индикатора (положительный ответ "+OK" и отрицательный ответ "-ERR"), ключевого слова и дополнительной информации. Если ответ POP3-сервера состоит из нескольких строк, то окончание ответа обозначается строкой, содержащей только лишь один символ – точку "**.**".

USER имя – ввод имени пользователя.

PASS пароль – ввод пароля, используется после положительного ответа на команду USER.

STAT – используется для просмотра состояния текущего почтового ящика. Возвращает количество сообщений в почтовом ящике и их общий объем.

LIST – просмотр информации об объеме каждого сообщения в почтовом ящике. Возвращает для каждого сообщения его номер (постоянный в течение всего сеанса) и размер в байтах.

RETR номер – используется для передачи клиенту запрашиваемого сообщения.

DELE номер – удалить сообщение из почтового ящика. Сообщение только помечается как удаленное и удаляется при завершении сеанса работы.

RSET – если какие-то сообщения были помечены как удаленные, сервер снимает пометку об удалении и возвращает положительный отклик.

NOOP – сервер не выполняет никаких операции с почтовым ящиком и посылает положительный отклик.

QUIT – закрыть POP3-сессию.

Пример протокола работы с POP3-сервером

```
+OK POP3 server ready
USER User1
+OK
PASS password
+OK 1 476
LIST
+OK 1 476
1 476
.
RETR 1
+OK 476    bytes for user1
Received: from user2@shttp.srv (some-host [10.0.0.21]) Mon, 18
Sep 2017 13:09:27 GMT
Date: Mon, 18 Sep 2017 16:09:27 +0300
From: User2 <user2@shttp.srv>
To: User1 <user1@shttp.srv>
Subject: Hello
Message-Id: <20170918160927.c16c0d4114a965f725618591@shttp.srv>
X-Mailer: Sylpheed 3.6.0 (GTK+ 2.24.30; i686-pc-mingw32)
Mime-Version: 1.0
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit

Hello.
How are you?

--
User2 <user2@shttp.srv>
.
DELE 1
+OK
QUIT
+OK
```

Подготовка программного обеспечения

Для работы с электронной почтой и выполнения последующих заданий (FTP, HTTP) необходимо следующее программное обеспечение:

- SMTP и POP3-сервер;
- FTP-сервер (FileZilla server, small HTTP server, …);
- Почтовый клиент (Mozilla Thunderbird, …);
- Сетевой терминал (PuTTY, …);
- Web-сервер или доступ в Internet.

В качестве примера рассмотрим настройку Small HTTP Server:

1. Загрузите Small HTTP Server (http://smallsrv.com/) и установите, например, на диск C:\ в каталог shttps. В окне инсталлятора задайте пароль для

учетной записи «admin» сервера.

2. После запуска сервера, для выполнения настройки, откройте системное меню приложения и выберите пункт Server | Настройки…
3. В главном меню появившегося окна «Configuration» выберите имя протокола, для которого требуется активировать сервер (SMTP, POP3 или FTP) и отключите чекбокс «Запретить <ИМЯ_ПРОТОКОЛА> сервер.. (Необходим перезапуск)». Полезной может быть также установка опции «Подробный отчет для POP/SMTP/FTP. Иначе фиксируются только основные события» на экране «Общие».

4. После включения нужного сервера следует настроить учетные записи пользователей, которые будут использоваться для тестирования: в окне «Configuration» выберите пункт «Другое» | «Пользователи», укажите имя пользователя, его пароль и существующий каталог на диске, который будет использоваться для хранения файлов пользователя FTP-сервером и хранилище почтовых сообщений. Подтвердите выполненные настройки нажатием кнопки «Ok».

Для тестирования работы SMTP-сервера и POP3-сервера, а также проверки корректности отправки сообщений электронной почты можно использовать один из почтовых клиентов (Microsoft Outlook, Mozilla Thunderbird, Opera Mail, Claws Mail, …).

Рассмотрим настройку почтового клиента на примере Mozilla Thunderbird:

1. Скопируйте к себе на диск или во временный каталог (например, C:\TEMP) и запустите ThunderbirdPortable.paf.exe. Во время распаковки программы оставляйте предлагаемые параметры без изменения.

2. После запуска программа автоматически перейдет в режим создания новой учетной записи электронной почты.

3. Используя данные зарегистрированной учетной записи, заполните нужной информацией поля формы.

4. Если программа была установлена ранее, то для создания учетной записи электронной почты необходимо выбрать пункт меню **Инструменты | Параметры учетной записи** и нажать кнопку "Действия для учетной записи | Добавить учетную запись почты".

5. Настройте параметры сервера исходящей почты: **Инструменты | Параметры учетной записи | Сервер исходящей почты | Добавить.**

ЗАДАНИЕ. ИЗУЧЕНИЕ ВОЗМОЖНОСТЕЙ ЭЛЕКТРОННОЙ ПОЧТЫ

1. Установите на машине почтовый клиент Mozilla Thunderbird и создайте свой почтовый ящик (информацию об учетной записи получите у преподавателя).

2. Изучите интерфейс программы. С помощью почтового клиента выполните следующие действия:

a) добавьте адреса своих знакомых в адресную книгу;

b) создайте шаблон нового письма;

c) создайте письмо и пошлите его по одному из адресов;

d) создайте письмо и разошлите его по всем адресам из адресной книги (используя список рассылки);

e) создайте письмо, прикрепите к нему произвольный файл и перешлите письмо на свой адрес. Получите письмо и попробуйте открыть прикрепленный файл;

f) сохраните на диске полученное письмо, вложенный файл.

3. Выполните следующие действия с POP3-сервером с помощью терминальной программы (telnet, PuTTY):

a) просмотрите содержимое писем в почтовом ящике;

b) изучите содержимое заголовков писем. Как записываются и что обозначают поля заголовков? (Полное описание формата писем дано в RFC 2822).

c) очистите почтовый ящик.

Указание. Если почтовый ящик случайно окажется пустым, то пошлите себе не менее трех писем с помощью любимой почтовой программы.

4. Выполните следующие действия с SMTP-сервером с помощью терминальной программы:

a) отправьте письмо одному получателю;

b) отправьте письмо нескольким адресатам.

Указание. При отправке писем обязательно следуйте стандартам оформления писем. В заголовке писем обязательно заполняйте поля "From:", "To:", "Subject:" и "Date:".

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. ИЗУЧЕНИЕ ВОЗМОЖНОСТЕЙ ЭЛЕКТРОННОЙ ПОЧТЫ

1. При помощи терминальной программы отправьте сообщение, содержащее прикрепленный файл одному получателю. Проконтролируйте при помощи почтового клиента корректность передачи файла.

2. При помощи терминальной программы отправьте сообщение, содержащее в теле письма только одну точку. Проконтролируйте при помощи почтового клиента корректность выполнения задания.

2.2. Протоколы прикладного уровня. Протокол FTP

FTP (File Transfer Protocol – протокол передачи файлов) – это основа одного из первых сервисов Internet, предназначенного для предоставления доступа к разделяемым файлам. Появление FTP относится к 1971 году. С тех пор FTP многократно модифицировался и изменялся.

Подобно протоколам SMTP и POP3 протокол FTP передает информацию, в том числе имя пользователя и пароль, в открытом виде без шифрования.

Полное описание протокола FTP находится в RFC 959.

FTP поддерживает одновременно два соединения – канал передачи команд (и статусов их обработки) и канал передачи данных.

Канал передачи команд используется для передачи команд от клиента и кратких результатов их обработки со стороны сервера. После того, как устанавливается управляющее соединение (обычно на порт 21 TCP), клиент может отправлять на сервер команды. FTP-команды определяют параметры соединения передачи данных: роли участников соединения (активный или пассивный),

порт соединения, тип передачи, тип передаваемых данных, структуру данных и управляющие директивы, обозначающие действия, которые пользователь хочет совершить, например, сохранить, считать, удалить файл и др.

Канал передачи данных может использоваться как в одном, так и в другом направлениях, кроме того, он может многократно открываться и закрываться по командам управляющих модулей в процессе работы. После того, как согласованы все параметры канала передачи данных, один из участников соединения, который является пассивным, переходит в режим ожидания открытия соединения на заданный для передачи данных порт. После этого активный модуль открывает соединение и определяет направление перемещения (прием или передача) данных. Окончание передачи определяется закрытием канала передачи данных. Сессия FTP считается закрытой только после закрытия управляющего соединения.

Описание команд FTP-сервера

USER пользователь – ввести идентификатор пользователя для работы с файловой системой.

PASS пароль – команда должна следовать непосредственно за командой USER и содержать пароль пользователя.

PASV – перевести FTP-сервер в пассивный режим. В ответ на эту команду сервер возвращает шесть десятичных чисел, определяющих его адрес и выделяемый порт, по которому будет осуществляться передача данных. Номер порта вычисляется по формуле:

(пятое число) * 256 + (шестое число).

Перед передачей данных клиент должен установить ещё одно соединение с этим портом FTP-сервера.

CWD путь – указать путь каталога удаленной файловой системы, в котором желает работать пользователь (change work directory).

PWD – вернуть строку с текущим каталогом удаленной файловой системы (print work directory).

LIST [путь] – по каналу передачи данных вернуть содержимое текущего [указанного] каталога удаленной файловой системы. Перед вводом этой команды необходимо установить еще одно соединение с FTP-сервером, указав порт, полученный по команде PASV.

RETR [путь]файл – команда указывает FTP-серверу передать указанный файл по каналу передачи данных. Перед вводом этой команды необходимо установить еще одно соединение с FTP-сервером, указав порт, полученный по команде PASV. После передачи файла сервер прекращает соединение.

STOR [путь]файл – сохранить данные в виде файла на FTP-сервер. Перед вводом этой команды необходимо установить еще одно соединение с FTP-сервером, указав порт, полученный по команде PASV. Все данные, передаваемые по этому соединению, сохраняются на FTP-сервере в файле с указанным именем. Для окончания передачи данных необходимо закрыть соединение.

RNFR [путь]СтароеИмя – переименовать указанный файл (rename from).

RNTO [путь]НовоеИмя – команда должна следовать непосредственно за командой RNFR и задавать новое имя файла (rename to).

DELE [путь]файл – удалить указанный файл (delete).
MKD [путь] – создать каталог (make directory).
RMD [путь] – удалить пустой каталог (remove directory).
QUIT – закончить FTP-сессию.

Каждой команде в диалоге соответствует ответ, состоящий из кода ответа и сообщения. Коды ответов состоят из трех цифр, каждая из которых имеет определенное назначение:

1xx - указывают на начало выполнения операции;
2xx – указывают на успешное выполнение команды;
3xx – указывают на успешное достижение промежуточной точки;
4xx – сигнализируют о временной ошибке;
5xx - свидетельствуют о постоянной ошибке.

Пример работы с FTP-сервером

```
220 FTP server ready
USER stud00
331 Password reguest
PASS stud
230 Logged in successfully
PASV
227 Entering Passive Mode (127,0,0,1,4,100)
CWD pub
250 Ok
PWD
257 “/pub” Ok
устанавливаем соединение с портом 1124 = 4*256+100
LIST
150 Ok
226 Ok
и там видим содержимое каталога /pub
опять устанавливаем соединение с портом 1124
RETR ftp.txt
150 Ok
226 Ok
и там видим содержимое файла /pub/ftp.txt
QUIT
```

ЗАДАНИЕ. ИЗУЧЕНИЕ ПРОТОКОЛА FTP

1. С помощью терминальной программы выполните следующие действия:

a) просмотрите содержимое корневого каталога на FTP-сервере;

b) просмотрите содержимое произвольного текстового файла, хранящегося на FTP-сервере;

c) создайте на FTP-сервере подкаталог;

d) в созданном подкаталоге создайте текстовый файл, просмотрите его;

e) переименуйте созданный файл;

f) удалите созданный подкаталог.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ.
ИЗУЧЕНИЕ ПРОТОКОЛА FTP

1. С помощью терминальной программы реализуйте передачу файла между двумя FTP-серверами напрямую (т.н. FXP), не используя передачу на промежуточный узел.

2.3. Протоколы прикладного уровня. Протокол HTTP

Начиная с 1990 года наиболее активно развивается часть Internet, называемая **WWW** – **World Wide Web** (**всемирная паутина**). WWW предоставляет доступ к связанным межу собой документам – страницам и основана на модели клиент-сервер. В качестве одного из основных протоколов информационной системой WWW используется протокол HTTP.

HTTP – это протокол прикладного уровня, который, как правило, работает поверх транспортного протокола TCP. При работе по TCP сервер HTTP обычно использует **порт 80**. Протокол построен на схеме «запрос-ответ». Чаще всего HTTP-соединение открывается клиентом перед каждым запросом и обычно закрывается сервером после отправки ответа.

Полное описание протокола HTTP версии 1.1 содержится в **RFC 2616**, а версии HTTP/2, принятой в мае 2015 – в **RFC 7540**.

Структура HTTP-сообщения

После установления TCP-канала между клиентом и сервером, агент пользователя отправляет на сервер сообщение-запрос. Это сообщение имеет следующий вид:

```
Начальная строка (start line)
[Заголовки (Headers)]
Пустая строка (Empty Line)
[Тело сообщения (Message Body)]
```

Каждая строка заголовочной части должна заканчиваться **символами конца строки <CR><LF>** (символ возврата каретки (ASCII 13) и символ перевода строки (ASCII 10), в программе - \x0D\x0A или \r\n). Заголовок запроса заканчивается **пустой строкой** (т.е. последовательностью символов <CR><LF><CR><LF>).

Начальная строка запроса содержит запрашиваемую команду, включающую метод запроса, URI (Uniform Resource Identifier, универсальный идентификатор ресурса), версию протокола HTTP и имеет следующую структуру:

```
<метод запроса><URI ресурса><версия HTTP>
```

Протокол HTTP поддерживает в запросах следующие **основные методы**:

1. Метод "**GET**" используется для получения ресурса, расположенного по заданномуURL. Как правило, запрашиваемый ресурс представляет собой текстовый (HTML, TXT) или графический файл. Если адрес ресурса ассоциирован с исполняемым файлом (CGI-скриптом), то этот файл будет запущен, и клиенту будут переданы результаты стандартного потока вывода.
2. Метод "**POST**" используется для передачи клиентом на сервер данных, которые должны быть обработаны ресурсом, указанным в URL. Данный метод

чаще всего используется для работы с CGI-скриптами. Метод “POST” передает параметры ресурсу в теле сообщения, поэтому при его использовании не требуется соблюдать никаких ограничений на длину передаваемой строки параметров.

3. Метод "**HEAD**" аналогичен методу “GET”, за исключением того, что клиенту возвращается только заголовок ответа (усеченный “GET”). Этот метод, в основном, используется для тестирования гиперссылок и проверки доступа к ресурсам.

4. Метод "**DELETE**" используется чтобы запросить удаление определенного ресурса.

6. Метод "**PUT**" используется, когда клиент желает сохранить на сервере передаваемую информацию в месте, заданном URL.

7. Метод "**OPTIONS**" используется для определения возможностей Web-сервера или проверки его работоспособности, а также параметров для конкретного ресурса. Чаще всего в виде: "`OPTIONS * HTTP/1.1`".

Кроме указанных методов существуют такие методы как: PATCH, TRACE, LINK, UNLINK, CONNECT.

Методы GET и HEAD должны поддерживаться любым сервером, остальные методы – не обязательно. В случае если сервер не может распознать метод, он возвращает код 501 (Not Implemented). Если метод распознан, но запрещен для запрошенного ресурса – возвращается код 405 (Method Not Allowed).

Наиболее распространенным вариантом задания URI в запросе является передача абсолютного пути к ресурсу в `<URI ресурса>` и сетевого расположения ресурса в поле "`Host:`" заголовка. При этом абсолютный путь не может быть пустым. Если его нет в запрашиваемом адресе, то он задается как "`/`" (без кавычек, разумеется).

Версия протокола задается как "HTTP/0.9", "HTTP/1.0" или как "HTTP/1.1" в зависимости от требований.

HTTP-запрос может состоять всего лишь из одной-двух строк, в которых должны быть указаны имя запрашиваемого ресурса и поддерживаемая клиентом версия протокола HTTP, например,

```
GET /catalog/index.htm HTTP/1.0
Host: www.host.net
```

После получения запроса, HTTP-сервер обрабатывает его и отправляет **результат** обработки клиенту. Начальная строка сообщения-ответа выглядит следующим образом:

```
<версия HTTP><код возврата><текстовое описание>
```

Код возврата – это код результата попытки HTTP-сервера понять и выполнить запрос. Код возврата должен состоять из трех цифр. Первая цифра кода возврата определяет класс кода. Две следующие цифры никак не категоризируются. Существует пять классов кодов:

`1xx` – **информационные** (запрос принят, продолжение процесса)

100 Continue

101 Switching Protocols

`2xx` – **коды успешного завершения** (запрос был успешно принят, распознан и выполнен)

200 OK
201 Created
202 Accepted
203 Non-Authoritative Information
204 No Content
205 Reset Content
206 Partial Content

3xx – **перенаправление** (дальнейшее действие должно быть перенаправлено для завершения запроса)

300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
304 Not Modified
305 Use proxy
307 Temporary Redirect

4xx – **ошибка клиента** (неверный синтаксис запроса или недостаточно клиентских данных для выполнения запроса)

400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Time-out
409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Request Entity Too Large
414 Request-URI Too Large
415 Unsupported Media Type
416 Requested range not satisfiable
417 Expectation Failed

5xx – **ошибка сервера** (невозможно полностью выполнить запрос)

500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Time-out
505 HTTP Version not supported

Заголовки могут использоваться как в запросе, так и в ответе сервера и представляют собой наборы из текстовых строк в формате:

```
ИмяПоля: ЗначениеПоля
```

Примерами полей заголовка могут быть "**Date:**" (содержит время создания сообщения), "**Pragma:**" (используется для указания дополнительных параметров участникам соединения). Например:

```
Date: Sat, 30 Sep 2017 01:23:45 GMT
Pragma: no-cache
```

Заголовок сообщения-запроса может включать, например, следующие поля:

Authorization: - содержит информацию аутентификации пользователя,

From: - может содержать Internet-адрес пользователя,

If-Modified-Since: - используется при работе методом "GET". Если запрашиваемый ресурс не изменялся с момента, указанного в этом поле, то данный ресурс не возвращается, а по запросу возвращается только заголовок сообщения ответа и соответствующий код возврата,

Referer: - содержит адрес ресурса, с которого был выполнен запрос,

User-Agent: - содержит информацию о программном обеспечении (браузере) клиента,

Accept – задает форматы данных, приемлемые в ответном сообщений,

Accept-Charset: - указывает, какая кодировка приемлема для ответа,

Accept-Encoding: - указывает, какими алгоритмами кодирования (сжатия) ответа следует ограничиться,

Accept-Language: - указывает, какие языки ответа предпочтительны,

Host: - содержит адрес узла Интернета и номер порта для ресурса (если номер порта не указан, то подразумевается порт 80),

Connection – желаемые опции для текущего соединения (использовать или нет долговременные соединения);

Expect: - указывает на поведение, ожидаемое клиентом от сервера

и другие.

Пример части заголовка запроса:

```
Host: example.test : 8080
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
Accept-Encoding: gzip, deflate
Accept-Language: ru, en
Connection: keep-alive
If-Modified-Since: Sat, 30 Sep 2017 01:23:45 GMT
Referer: http://test.domain/query/index.html
User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:48.0)
Gecko/20100101 Firefox/48.0
DNT: 1
```

Заголовок сообщения-ответа содержит информацию о структуре и формате тела сообщения, или, если тело сообщения отсутствует, информацию о запрашиваемом ресурсе. Например:

Allow: - содержит список методов, поддерживаемых ресурсом, и носит исключительно информационный характер,

Content-Encoding: - содержит идентификатор типа дополнительной кодировки ресурса, как правило, способ сжатия,

Content-Language: - содержит язык ресурса,

Content-Length: - определяет длину тела сообщения,

Content-Type: - определяет тип ресурса и таблицу кодировки данных,

Expires: - содержит дату окончания срока действия ресурса,

Last-Modified: - содержит дату и время последнего изменения ресурса и другие.

Заголовок также позволяет передавать дополнительную информацию обработки запроса, которую нельзя поместить в строку статуса и может содержать, например, такие поля:

Location: - содержит полный URL ресурса, который отвечает на отправленный запрос, имеет смысл при перенаправлении запроса на другой сервер,

Server: - содержит описание программного обеспечения WWW-сервера, отвечающего на запрос,

WWW-Authenticate: - содержит параметры схемы (метода) аутентификации и, по крайней мере, одно название области аутентификации.

ЗАДАНИЕ. ИЗУЧЕНИЕ ПРОТОКОЛА HTTP

1. С помощью терминальной программы сформируйте запросы к HTTP-серверам: program.vsu, www.belta.by, www.google.com, msdn.microsoft.com, php.net. Объясните ответы серверов.
2. С помощью терминальной программы получите одну из страниц любого из приведенных выше сайтов целиком (HTML-код страницы, подключенные скрипты и таблицы стилей, рисунки).

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. ИЗУЧЕНИЕ ПРОТОКОЛА HTTP

1. При помощи сетевого терминала продемонстрируйте работу не менее четырех различных методов протокола HTTP.
2. Составьте запросы к различным сайтам, которые вернут ответы с кодами статуса из всех 5 классов.
3. Составьте не менее 5 запросов, которые вернут различные коды статуса из класса 4xx.

3. GOOGLE APPS КАК СРЕДСТВО ВЗАИМОДЕЙСТВИЯ В ОБЛАЧНОЙ СРЕДЕ

GoogleApps [10] – это облачное решение, которое обеспечивает постоянный доступ к информационным ресурсам в любом месте и в любое время. Сервисы Google делают возможной коллективную работу над любым контентом, а процесс обучения открытым и доступным.

Рассмотрим наиболее популярные web-приложения Google.

Gmail – бесплатный почтовый сервис с большим объёмом места для хранения сообщений и удобным web-интерфейсом. Регистрация почтового ящика сопровождается созданием персонального аккаунта и предоставляет доступ для всех служб Google.

Google Диск – это файловый хостинг, созданный и поддерживаемый компанией Google. Его функции включают хранение файлов в Интернете, общий доступ к ним и совместное редактирование.

Google Календарь – сервис для планирования встреч, событий и дел. Он позволяет задавать время встречи, создавать повторяющиеся мероприятия, устанавливать напоминания, а также приглашать других участников (им высылается сообщение по электронной почте). Возможно совместное использование календаря группой пользователей. Кроме того, сервис интегрирован с Gmail.

GoogleHangouts – это программное обеспечение для мгновенного обмена сообщениями (видео- и голосовая связь) между двумя и более (до 100) участниками и для видеоконференций.

Google Документы – это бесплатный пакет офисных web-приложений, интегрированный с Google диском, включающий в себя текстовый, табличный процессор и сервис для создания презентаций, а также интернет-сервис облачного хранения файлов с функциями файлообмена. Этот набор офисных приложений позволяет:

– создавать, загружать имеющиеся, редактировать текстовые документы, электронные таблицы, презентации, формы, рисунки;

– вести совместную работу (чтение или редактирование) в пределах группы (класса) или всего учебного заведения в режиме реального времени;

– обмениваться сообщениями во время редактирования документов;

– публиковать для пользователей всего мира окончательные версии документов.

Google Группы – web-сервис, который позволяет объединять пользователей для общения и обмена информацией. С помощью этого инструмента можно:

– участвовать в дискуссии на конкретную тему;

– создавать группы поддержки, в которых можно проконсультироваться со специалистами;

– организовывать встречи, конференции и другие мероприятия для участников группы;

– находить людей со схожими интересами и увлечениями;

– читать сообщения группы по электронной почте или через web-интерфейс.

ЗАДАНИЕ 1. GOOGLE ГРУППЫ

1. **Создание личного аккаунта Google.**

1.1. Откройте браузер и в адресной строке введите gmail.com.

1.2. Если у вас имеется аккаунт на gmail, то войдите в него. Если нет – создайте новую учетную запись, т.е. зарегистрируйте на свое имя почтовый ящик и войдите в аккаунт.

2. **Создание Google Группы.**

2.1. Объединитесь внебольшие группы (из 3-4 участников), определите координатора (администратора группы) и придумайте название группы.

2.2. Создайте Google Группу. Для этого:

Координатору группы:

1. В основном перечне сервисов Google выберите ***Группы*** (может находиться в списке, открывающемся по ссылке ***Еще***).

2. Откройте ***Справку*** по созданию Google Групп в отдельной вкладке браузера, ознакомьтесь подробнее с разделом “Работа с Группами Google в роли администратора”.

3. Нажмите кнопку ***Создать Группу*** в верхней части страницы. Введите название вашей группы, придумайте адрес электронной почты, в описании перечислите кратко общие интересы, выберите тип - список рассылки. Нажмите кнопку***Создать***.

4. Перейдите по ссылке “Приглашение участников группы”. Введите адреса электронной почты членов вашей группы, разделяя их запятыми. Напишите приветственное сообщение для участников группы. Нажмите кнопку ***Отправить приглашения***.

5. Перейдите по ссылке ***Участники*** и убедитесь, что все члены группы приняли приглашение. Перейдите в режим ***Управление***, отметьте всех приглашенных участников, в меню ***Действия*** выберите *Настроить разрешения на отправку сообщений,* далее пункт *Переопределение - разрешение отправки сообщений*.

Участникам группы:

1.Откройте ***Справку*** по созданию Google Групп в отдельной вкладке браузера, ознакомьтесь подробнее с разделом “*Работа с Группами Google в роли участника*”.

2.Дождитесь от координатора приглашения в Группу (сообщение придет на вашу электронную почту), ответьте на него согласием.

3.В основном перечне сервисов Google выберите ***Группы*** (может находиться в списке, открывающемся по ссылке ***Еще***). Найдите в списке группу, к которой вы присоединились. Прочтите сообщения.

4.Создайте собственное сообщение, например, о том, какие облачные сервисы могут заинтересовать участников группы, используя кнопку ***Новая тема***.

ЗАДАНИЕ 2. GOOGLE ДОКУМЕНТЫ

1. **Работа в группах по созданию презентаций.**

1.1. Объединитесь внебольшие группы (3-4 участника), определите координатора.

1.2. Создайте Google Презентацию группы на тему *«Облачные сервисы»*.

Координатору группы:

1.Создайте Google Презентацию на своем диске и предоставьте участникам группы доступ с правом редактирования.

2.Организуйте обсуждение структуры презентации и распределите между исполнителями работу над разделами.

3.Создайте и наполните информацией первый и второй слайды презентации *(На титульном слайде разместите заголовок и список авторов, на втором слайде создайте содержание с гиперссылками для перехода к соответствующим слайдам, предусмотрите возможность возврата к содержанию).*

4.Выполните общие настройки презентации (тема, анимация, показ).

5.Координируйте работу участников с целью согласования содержания и оформления.

6.Откройте доступ к презентации преподавателю с правом оставлять комментарии.

Участникам группы:

1. Создайте слайды раздела, который вам поручен в группе. Придерживайтесь единого стиля оформления.

2. На завершающем раздел слайде необходимо создать кнопку *Содержание* и настроить гиперссылку для перехода на слайд 2.

3. Выполните редактирование слайдов в соответствии с замечаниями координатора и преподавателя.

4. Познакомьтесь с другими презентациями коллег и выскажите свое мнение, приняв участие в интерактивном опросе, организованном с помощью сервиса Google Форм.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. GOOGLE ПРЕЗЕНТАЦИЯ

1. **Создание презентации курсового (дипломного) проекта.**

1.1. Скачайте с сайта библиотеки “Инструкцию об организации, проведении и требованиях к содержанию, оформлению и защите рефератов, курсовых проектов (работ), дипломных проектов (работ), магистерских диссертаций”[1]. Изучите рекомендации по представлению результатов исследования.

1.2. Познакомьтесь с советами о том, как сделать хорошую презентацию ([2], [3]).

1.3. Разработайте структуру презентации, сопровождающей ваш доклад на защите курсового (дипломного) проекта.

1.4. Подготовьте иллюстративный материал для презентации. Соблюдайте авторское право (*в инструментах поиска картинок в сети интернет используйте соответствующие лицензионные настройки*).

1.5. СоздайтеGoogle Презентацию на тему курсового (дипломного) проекта.

1.6. Откройте доступ к презентации преподавателю с правом оставлять комментарии.

1.7. Выполните редактирование презентации в соответствии с замечаниями преподавателя, ответьте на комментарии.

4. ИНСТРУМЕНТЫ ПОИСКА ИНФОРМАЦИИ И ХРАНЕНИЯ ЗАКЛАДОК

Крупнейшие компьютерные корпорации разрабатывают всевозможные средства интеллектуализации поиска информации. Единой оптимальной схемы поиска информации в Интернете не существует. В зависимости от специфики нужной информации можно использовать соответствующие поисковые инструменты и службы. От того, как грамотно будут подобраны поисковые службы, зависит качество результатов поиска.

Работа поисковой системы заключается в том, чтобы по запросу пользователя найти документы, содержащие или указанные ключевые слова, или слова, как-либо связанные с ключевыми словами. При этом поисковая система генерирует страницу результатов поиска, котораяможет содержать, например, web-страницы, изображения, аудиофайлы. Примеры поисковых систем: Google.com, Bing.com, Yahoo.com, Rambler.ru, Yandex.ru.

Для комбинированного поиска в нескольких поисковых системах удобнее обратиться к системам мета-поиска – поисковым машинам, которые посылают запрос на огромное количество разных поисковых систем, затем обрабатывают полученные результаты, удаляют повторяющиеся адреса ресурсов и представляют более широкий спектр в сети Интернет. Одна из популярных систем мета-поиска (объединенный поисковый сервер) – Search.com, которая включает в себя почти два десятка поисковых систем.

4.1. Специализированные поисковые системы

Чтобы найти информацию по какой-то узкой теме, можно воспользоваться специализированными поисковыми системами.

Например, если нужны ответы на действительно сложные вопросы в области математики, физики, медицины, статистики, истории, лингвистики и других областей науки, то можно воспользоваться вычислительно-поисковой системой WolframAlpha[7], способной предложить пользователю чуть ли не энциклопедические ответы на самые необычные вопросы. Интеллектуальная поисковая система Нигма [8] является одной из наиболее популярных среди подобных русскоязычных систем. Помимо обычных инструментов поиска она предоставляет возможность решать различные задачи по математике и химии.

Академия Google позволяет без труда выполнять обширный поиск научной литературы, а также среди огромного количества научных трудов найти исследование, наиболее точно соответствующее запросу. Используя единую форму запроса, можно выбрать дисциплины, виды источников, включая прошедшие рецензирование статьи, диссертации, книги, рефераты и отчеты, опубликованные издательствами научной литературы, профессиональными ассоциациями, высшими учебными заведениями и другими научными организациями. С помощью Академии Google можно осуществлять поиск:

- статей, рефератов и библиографических ссылок;
- полного текста документа в библиотеке или сети;
- информации об основных работах в любой области исследований.

Русскоязычный сегмент Академии Google доступен по адресу http://www.scholar.ru/.

Система пользовательского поиска Google (СПП) — сервис, представляющий собой систему персонального (пользовательского) интернет-поиска. Она позволяет пользователю создавать специализированные поисковики, учитывающие их личные предпочтения и тематические интересы, задавать контекст поиска. В простейшем случае осуществляет поиск по набору указанных пользователем сайтов.

4.2. Коллекции закладок

Для хранения персональных гиперссылок сегодня появились специализированные интернет-сервисы, получившие название сервисов социальных закладок, которые предоставляют пользователю возможности:

- создавать и хранить свои коллекции закладок (ссылок на web-ресурсы) в Интернете;
- структурировать информацию по определенным тематикам с помощью категорий или меток;

– при необходимости делиться коллекциями закладок с другими пользователями.

Рисунок 1. Коллекция закладок, созданная с помощью Symbaloo [6].

Примеры популярных сервисов социальных закладок: delicious.com, bobrdobr.ru, symbaloo.com.

ЗАДАНИЕ. СПЕЦИАЛИЗИРОВАННЫЙ ПОИСК В GOOGLE.

1. **Создание СПП на базе Google CSE.**

1.1. Познакомьтесь с разделом “Пользовательский поиск” Справки Google.

1.2. С помощью обычногоweb-поиска Google найдите сайт или блог, в который встроен пользовательский поиск. Воспользуйтесь им для поиска по одному или нескольким ключевым словам. Оцените качество и релевантность результата.

1.3. Объединитесь в мини-группу для создания СПП.

1.4. Выберите тему для поиска информации.

1.5. Найдите наиболее релевантные источники информации по выбранной тематике (порталы, сайты, отдельные web-страницы), которые будут включены в СПП.

1.6. Создайте СПП вашей группы. Для этого:

Координатору группы:

1. В сервисах Google выбрать ***Пользовательский поиск*** или использовать ссылку www.google.com/cse/;
2. Выберите функцию ***Новая поисковая система***;
3. Определите ***язык*** и ***название*** поисковой системы и нажмите кнопку ***Создать***;
4. Выберите функцию ***Изменение поисковой системы***;
5. На вкладке ***Панель администратора*** добавьте пользователей – участников вашей группы;
6. На вкладке ***Основные сведения*** скопируйте ***Общедоступный URL*** вашей поисковой системы и отправьте его преподавателю;
7. Включите опцию ***Поиск изображений***;

Участникам группы:

1. Откройте ССП вашей группы, используя ссылку, полученную от координатора;
2. Выберите функцию ***Изменение поисковой системы*** и дополните ССП своими источниками информации;
3. Протестируйте СПП вашей группы. Осуществите тематический поиск по нескольким ключевым словам. Оцените качество и релевантность результатов;
4. Встройте СПП в ваш блог или сайт (в случае, если они имеются).

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. ПОИСК НАУЧНОЙ ЛИТЕРАТУРЫ В АКАДЕМИИ GOOGLE.

1. Откройте приложение Google *Академия (https://scholar.google.ru/)***.**
2. Найдите последние результаты исследований по определенной теме, например, для курсовой работы или дипломного проекта.
3. Измените временной период поиска, ограничьте его 2-3 годами.
4. Просмотрите полнотекстовые источники (в перечне указаны справа ссылки и формат файлов).
5. Сохраните в *Моей библиотеке* источники, представляющие для Вас наибольший интерес. Обратите внимание на возможность копирования данных источника по ссылке *[Цитировать]*.
6. Изучите возможности *Расширенного поиска*. Доступ к нему открывается в выпадающем списке в правой верхней части окна.
7. В новом текстовом документе[1] укажите свою фамилию, тему поиска, перечень библиографических ссылок на выбранные вами источники.
8. Откройте преподавателю доступ к документу с правом комментирования. Отредактируйте документ в соответствии с комментариями преподавателя.

5. МЕНТАЛЬНЫЕ КАРТЫ

Ментальные карты (или интеллект-карты) – один из удобных инструментов для отображения процесса мышления и структурирования информации в визуальной форме с помощью схем, построенных по определенным правилам.

Ментальные карты используют разнообразные способы визуализации информации, чтобы активировать восприятие: разная толщина линий, разные цвета ветвей, точно выбранные ключевые слова, использование образов и символов.

Ментальная карта полезна в образовательной деятельности при подготовке к экзаменам, в исследовательских проектах, составлении планов, для фиксации мыслей при мозговом штурме, при составлении конспекта и т.д.

Для создания ментальных карт можно использовать сервисы: MindMaps, Mind42, Xmind, DropMind, MindMeister, Mindomo, Bubbl.us, SpiderScribe.

[1] Вместо текстового файла можно использовать вебмикс сервиса коллективного хранения закладок, например Symbaloo.

Преимущества сервиса MindMeister [9]:

– возможность совместной работы над проектом в режиме реального времени;

– многофункциональный интерфейс, который прост в использовании;

– доступ к картам в любое время и в любом месте;

– высокие стандарты безопасности и возможность резервного хранения данных;

– мобильный доступ к приложению с возможностью редактирования и синхронизации;

– возможность работать в режиме offline с сохранением данных локально с их последующей синхронизацией;

– возможность использовать сценарии и шаблоны, которые делятся по группам.

ЗАДАНИЕ. МЕНТАЛЬНЫЕ КАРТЫ

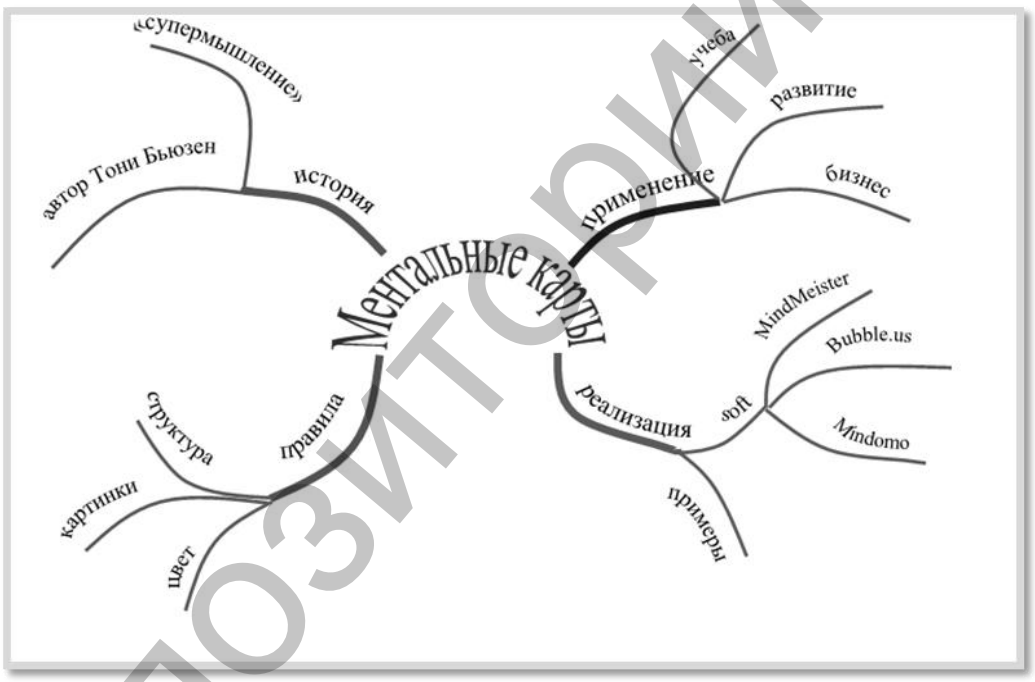

Рисунок 2. Пример ментальной карты.

1. **Работа с сервисом MindeMeister.**

1.1. Откройте сайт программы http://www.mindmeister.com и ознакомьтесь с основными возможностями интеллектуальных карт.

1.2. Если у вас имеется аккаунт Google, то вы можете ***подключиться посредством Google***, щелкнув по соответствующей кнопке на главной странице http://www.mindmeister.com/. Иначе, пройдите процедуру регистрации на сайте.

1.3. Выберите базовый пакет, чтобы начать работу бесплатно.

1.4. Для удобства работы (по желанию) в настройках вашей страницы измените язык интерфейса на русский.

1.5. Ознакомьтесь с каталогом карт, которые имеются в открытом доступе.

1.6. В категории ***Мои карты*** откройте карту, созданную по умолчанию ***MyFirstMindMap***.

1.7. Ознакомьтесь с основными элементами карты (идеи, под-идеи, картинки, символы, связи, ссылки и др.), изучите панели инструментов, возможности экспорта файла. Эту карту в процессе изучения вы можете изменять.

2. **Создание ментальной карты по образцу.**

2.1. Изучите рисунок 2 с ментальной картой, которую необходимо создать, определите ее основные элементы.

2.2. Создайте компьютерный вариант ментальной карты. Для этого:

a) создайте **новую карту**, используя пустой шаблон. Выберите тему (инструмент ***Тема*** в нижней панели редактора);

b) оформите иконку основной идеи: измените текст, цвет символов и фон, вставьте рисунок из имеющихся стандартных рисунков или воспользуйтесь подбором Google (в категории Изображения– кнопка ***WunderBild***);

c) добавьте под-идею **ИСТОРИЯ**, для этоговыделите центральную иконку, а затем примените инструмент ***Добавить идею*** (из верхней панели редактора). Переместите и отформатируйте элемент **ИСТОРИЯ**;

d) добавьте к элементу **ИСТОРИЯ** две под-идеи:

Автор: Тони Бьюзен, фото автора вставьте с помощью кнопки WunderBild

«Супермышление», подберите рисунок из имеющихся стандартных картинок.

a) добавьте к элементу **ИСТОРИЯ** стиль оформления границы;

b) элемент **ПРАВИЛА** создайте аналогично предыдущему. В качестве иллюстраций под-идей используйте рисунки из библиотеки значков;

c) создайте связь между элементами **ИСТОРИЯ** и **ПРАВИЛА** (инструмент ***Добавить связь между идеями*** из верхней панели редактора);

d) дополните карту ветвями **ПРИМЕНЕНИЕ** и **РЕАЛИЗАЦИЯ**. Оформите их в соответствии с образцом;

e) сохраните полученную карту в формате pdf и предоставьте доступ к файлу преподавателю.

Примерный результат вашей работы должен выглядеть как на рисунке 3.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. СОТРУДНИЧЕСТВО ПРИ РАЗРАБОТКЕ МЕНТАЛЬНОЙ КАРТЫ

1. Определите соавтора для разработки ментальной карты. Выберите себе тему для карты.

2. Создайте Google Документ для подготовки информации. Создайте многоуровневый список идей для ментальной карты. Откройте доступ к документу преподавателю для комментирования.

3. Установите дополнение (меню ***Дополнения… Установить дополнения… MindMeister...*)**.

4. Из многоуровнего списка создайте ментальную карту (меню ***Дополнения…MindMeister...Insert as Mind Map)***. Откройте доступ преподавателю для комментирования.

5. Разработайте ментальную карту для своей темы в одном из сервисов. Сделайте ее общедоступной. Поделитесь ссылкой с преподавателем**.**

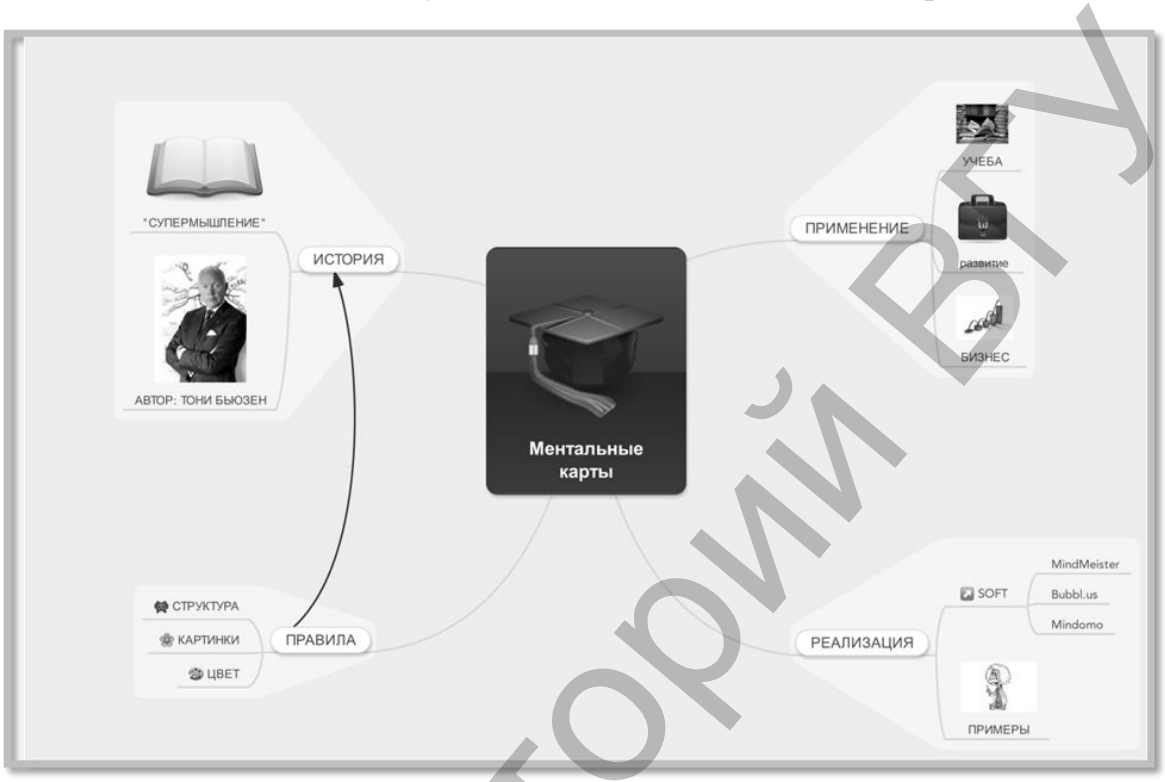

Рисунок 3. Ментальная карта.

6. ПРОГРАММИРОВАНИЕ ПРИКЛАДНЫХ ПРОТОКОЛОВ

Наиболее распространенным интерфейсом сетевого программирования в настоящее время является интерфейс сокетов. Его реализация в ОС UNIX/Linux часто упоминается как сокеты Беркли (по названию университета, в котором был разработан интерфейс). В Windows библиотека сокетов носит название Windows Sockets [12] или кратко WinSock.

Сокет представляет собой дескриптор, используя который можно получать и отправлять данные. Создавая сокет, следует указать какое семейство адресов будет использоваться: **TCP/IP** или **IPX/SPX**. Для **TCP/IP** поддерживаются два основных типа сокетов:

- `SOCK_STREAM` (потоковый или connection-based socket);
- `SOCK_DGRAM` (дейтаграммный или connectionless socket).

Первый тип сокетов использует в качестве транспортного протокола **TCP**, второй – **UDP**.

Для объявления экземпляра сокета используется тип `SOCKET`, а для создания сокета используется функция `socket`:

```
SOCKET socket(int af, int type, int protocol);
```

где `af` – спецификация семейства протоколов (для **TCP/IP** должен иметь

значение AF_INET или AF_INET6);
type – спецификация типа для создаваемого сокета, – какой транспортный протокол следует использовать (SOCK_STREAM или SOCK_DGRAM);
protocol – протокол, используемый для передачи данных через сокет (например, IPPROTO_TCP, IPPROTO_UDP или 0).

Если ошибок при создании сокета не возникло, функция возвращает дескриптор сокета, иначе значение INVALID_SOCKET (более подробные сведения в WinSock можно получить, используя WSAGetLastError(), коды ошибок приведены в справке к **Winsock**).

Пример использования:

```
SOCKET s = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (s == INVALID_SOCKET)
{
      …
      error = WSAGetLastError();
      …
}
```

После создания сокета, дальнейшие действия клиентского и серверного приложения различаются.

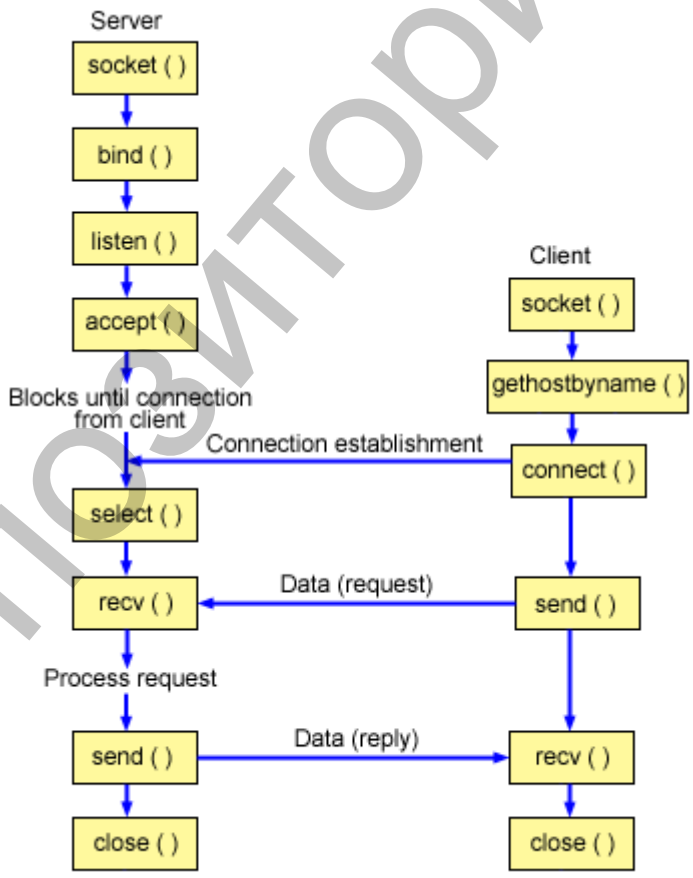

Рисунок 4. Взаимодействие TCP-клиента иTCP-сервера.

Клиентская часть приложения. После создания сокета его можно использовать для обмена данными с другими узлами. Для этого нужно устано-

вить соединение с другим узлом, задав его IP-адрес и порт, который прослушивает приложение-сервер:

```
int connect(SOCKET s, const struct sockaddr* name, int namelen);
```

s – неподключенный сокет;

name – задает сокет, к которому выполняется подключение. Структура sockaddr зависит от используемого семейства протоколов. Для IPv4 используется sockaddr_in;

namelen – размер параметра name (структуры) в байтах.

Функция возвращает нуль, если соединение создано или, в противном случае, SOCKET_ERROR.

Используемая в качестве параметра name структура для IPv4 описывается так:

```
struct sockaddr_in {
   short sin_family;
   u_short sin_port;
   struct in_addrsin_addr;
   char sin_zero[8];
};
```

где sin_family – задает семейство адресов (AF_INET);

sin_port – порт, к которому выполняется подключение;

sin_addr – IP-адрес в сетевом порядке байт, к которому выполняется подключение. Для его хранения используется структура in_addr;

sin_zero – заполнитель, выравнивающий размер структуры с размером sockaddr.

Напомним формат структуры in_addr:

```
struct in_addr {
     union {
          struct { u_char s_b1,s_b2,s_b3,s_b4; } S_un_b;
          struct { u_short s_w1,s_w2; } S_un_w;
          u_long S_addr;
     } S_un;
};
```

здесь S_un_b – адрес в виде четырех u_char;

S_un_w – адрес в виде двух u_short;

S_addr – адрес в виде u_long.

Пример использования:

```
// объявляем переменную для хранения адреса
sockaddr_in s_addr;
// очищаемее
ZeroMemory(&s_addr, sizeof(s_addr));
// задаем семейство адресов Internet
s_addr.sin_family = AF_INET;
/* задаем адрес узла, к которому выполняем подключение. Не забываем преобразовать адрес из строкового десятичного формата в
```

```
бинарный. */
s_addr.sin_addr.S_un.S_addr = inet_addr("192.168.1.1");
/* задаем порт приложения-сервера. Не забываем преобразовывать
номер порта в сетевой порядок байт. */
s_addr.sin_port = htons(2018);
…
// открываем соединение
intcon_status;
con_status = connect(s, (sockaddr *)&s_addr, sizeof(s_addr));
if (con_status == SOCKET_ERROR)
{
…
error = WSAGetLastError();
…
}
```

После установки соединения можно начинать обмен данными (см. ниже).

Замечание: **UDP**-сокеты не требуют установки соединения, поэтому для таких сокетов обычно не используется `connect`. Однако при его использовании **UDP**-сокет может обмениваться данными не только с помощью `sendto` и `recvfrom`, но и с помощью более простых `send` и `recv`.

Серверная часть приложения. На стороне сервера, прежде чем использовать сокет, нужно связать его с локальным адресом и портом, который будет прослушиваться в ожидании входящих подключений. Если у узла сервера есть несколько IP-адресов, то можно указать один из них или привязать сокет сразу ко всем IP-адресам (для этого нужно указать в качестве IP-адреса константу `INADDR_ANY`, равную нулю). Для связывания сокета с локальным адресом и портом используется вызов:

```
int bind(SOCKET s, const struct sockaddr* name, int namelen);
```

`s` – сокет, к которому выполняется привязка;

`name` – задает адрес, к которому привязывается сокет;

`namelen` – задает длину параметра `name` в байтах;

возвращает `0`, если привязка завершилась успешно, или `SOCKET_ERROR` в противном случае.

Замечание. Клиент тоже должен связывать свой сокет с локальным адресом, но за него это выполняет connect, связывая клиента со случайно выбранным портом из диапазона `1024-5000`. Сервер же должен использовать строго определенный протоколом порт, поэтому выполняет связывание «вручную».

Выполнив связывание, TCP-сервер переходит в режим ожидания подключений клиентов с помощью:

```
int listen(SOCKET s, int backlog);
```

`s` – сокет сервера;

`backlog` – максимальный размер очереди ожидающих подключений. Ограничивает количество одновременных соединений. При его превышении очередному клиенту будет отказано в подключении к серверу;

Вызов возвращает нуль в случае успешного выполнения или `SOCKET_ERROR` в противном случае.

Замечание.UDP-сервера не используют функцию `listen` так как не устанавливают соединение и после связывания сразу же могут читать входящие данные.

После начала прослушивания серверного сокета приложение может начать извлекать запросы на соединение из очереди ожидающих подключение. Это выполняется вызовом `accept`, который автоматически создает новый сокет для входящего подключения, выполняет связывание и возвращает его дескриптор, а в структуру `sockaddr` заносит сведения о подключившемся клиенте (IP-адрес и порт):

```
SOCKET accept(SOCKET s, struct sockaddr* addr, int* addrlen);
```

здесь `s` – серверный «прослушиваемый» сокет;

`addr` – адрес извлеченного из очереди ожидающих подключений сокета;

`addrlen` – длина структуры `sockaddr`, извлеченной из очереди;

функция возвращает дескриптор сокета для обмена данными с подключившимся клиентом или `INVALID_SOCKET`, если произошла ошибка. Если очередь входящих подключений пуста, то функция не возвращает управление, пока с сервером не будет установлено соединение.

Замечание. Для параллельной работы с несколькими клиентами после получения сокета следует создать новый поток или процесс, передать ему полученный от `accept` сокет и вернуться к обработке входящих соединении. Иначе, пока не будет завершена обработка одного клиента, сервер не сможет обслужить другого.

Передача и прием данных. Для передачи данных по установленному соединению используется `send`:

```
int send(SOCKET s, const char* buf, intlen, int flags);
```

где `s` – подключенный сокет;

`buf` – указывает на буфер с данными для передачи;

`len` – длина передаваемых данных;

`flags` – флаги, задающие способ, которым выполняется вызов;

функция возвращает количество отправленных байтов или `SOCKET_ERROR` в случае ошибки.

Например:

```
if (send(s, (char *)&buff, 512,0) == SOCKET_ERROR)
{
…
error = WSAGetLastError();
…
}
```

Длина пакета данных ограничена протоколом. Функция не возвращает значение до тех пор, пока данные не будут отправлены.

Замечание. Функция не гарантирует, что данные были доставлены клиенту. Она возвращает число посланных клиенту байт, а не число им полученных.

Для приема данных от узла, с которым установлено соединение, используется функция `recv`:

```
int recv(SOCKET s, char* buf, int len, int flags);
```

здесь `s` – дескриптор подключенного сокета;

`buf` – буфер для входящих данных;

len – длина буфера в байтах;
flags – флаги, задающие способ, которым выполняется вызов;
функция возвращает число полученных байт, если соединение было закрыто – нуль, SOCKET_ERROR в случае ошибки. Если данных нет, то функция не возвращает управление, пока не придет пакет (или пока не истечет тайм-аут).
Замечание. recv получает данные порциями, которыми они передавались функцией send (если был предоставлен буфер достаточного размера).

Пример использования:

```
int recv_len = 0;
recv_len = recv(s, (char *)&buff, MAX_PACKET_SIZE, 0);
if (recv_len == SOCKET_ERROR)
{
…
error = WSAGetLastError();
…
}
```

Дейтаграммный сокет (**UDP**) может использовать функции send и recv, если передними был выполнен вызов **connect** или использовать собственные вызовы: sendto и recvfrom.

```
int sendto(SOCKET s, const char* buf, intlen, int flags, const
struct sockaddr* to, int tolen);
```

где s – дескриптор сокета;
buf – буфер с данными;
len – длина данных в буфере отправки в байтах;
flags – флаги, задающие способ, которым выполняется вызов;
to – указывает на структуру sockaddr, содержащую адрес сокета получателя;
tolen – размер адреса в to;
функция возвращает число отправленных байт или SOCKET_ERROR, если при отправке возникла ошибка.

```
int recvfrom(SOCKET s, char* buf, int len, int flags, struct
sockaddr* from, int* fromlen);
```

где s – дескриптор сокета;
buf – буфер для входящих данных;
len – размер буфера в байтах;
flags – флаги, задающие способ, которым выполняется вызов;
from – указатель на адрес сокета с которого пришли данные;
fromlen – длина адреса;
возвращает число полученных байт или SOCKET_ERROR при возникновении ошибки.
В отличие от рассмотренных выше send и recv функциям sendto и recvfrom требуется указывать адрес узла, передающего или принимающего данные.
Замечание. Рассмотренные выше функции могут управляться с помощью флагов flags. Они могут принимать значение MSG_PEEK или MSG_OOB. Флаг MSG_PEEK заставляет функции получения данных просматривать данные вместо

их получения. Флаг MSG_OOB предназначен для передачи/приема срочных данных.

Закрытие соединения. После завершения передачи данных соединение должно быть закрыто. Для этого используются функции shutdown и closesocket:

```
int shutdown(SOCKET s, int how);
```

s – закрываемый сокет;

how – способ закрытия.

```
int closesocket(SOCKET s);
```

s – закрываемый сокет.

Оба вызова возвращают нуль в случае успеха или SOCKET_ERROR, если произошла ошибка.

Замечание. Существуют более сложные способы закрытия соединения (выборочное закрытие соединения на одной из сторон, оставляя другую сторону в рабочем состоянии). Для этого используется функция shutdown и флаг how (SD_RECEIVE закрывает канал «сервер-клиент», SD_SEND закрывает канал «клиент-сервер», SD_BOTH – оба).

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. РАЗРАБОТКА СЕТЕВЫХ ПРИЛОЖЕНИЙ

Согласно Вашему варианту разработайте клиентское и серверное сетевые приложения на базе интерфейса Windows Sockets (или сокетов в Linux/UNIX) для указанного сетевого протокола или сервиса:

Вариант 1. Протокол Echo (RFC 862).
Вариант 2. Протокол Active Users (RFC 866).
Вариант 3. Протокол DayTime (RFC 867).
Вариант 4. Протокол Time (RFC 868).
Вариант 5. Протокол QotD (Quote of the Day) (RFC 865).
Вариант 6. Протокол PWDGEN (RFC 972).
Вариант 7. Протокол CharGen (RFC 864).
Вариант 8. Протокол MessageSend (RFC 1312).
Вариант 9. Протокол Systat (wikipedia; RFC 866).
Вариант 10. RLogin (RFC 1282).
Вариант 11. Протокол WhoIs (RFC 3912).
Вариант 12. Протокол Telnet (RFC 854).
Вариант 13. Протокол Finger (RFC 742).
Вариант 14. Сервис rwho (см. https://pentestlab.blog/tag/rwho/).

7. ПРОГРАММИРОВАНИЕ КЛИЕНТСКИХ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕК ЯВУ

В современных приложениях гораздо чаще используются библиотечные классы, инкапсулирующие низкоуровневые сетевые механизмы, чем прямая работа на низком уровне с интерфейсом сокетов. Подобные библиотеки классов доступны для C++, C#, Java и для других языков высокого уровня. Они содержат как специализированные классы, предназначенные, например, для работы с протоколом HTTP или протоколами электронной почты, так и классы

общего назначения, которые будут использоваться в дальнейших лабораторных работах.

Язык Java

Создание клиентских приложений с помощью класса Socket

Для разработки клиентских сетевых приложений на языке Java часто используются объекты класса `Socket` из пакета `java.net` [13].

Схема использования класса java.net.Socket:

1. Создание соединения с сервером. При создании объекта класса `Socket` следует указать IP-адрес компьютера, с которым нужно соединиться и номер порта. Вместо IP-адреса можно указать имя компьютера, например, "ftp.example.com".

```
Socket ClientSocket = null;
ClientSocket = new Socket("ftp.example.com", 21);
```

Для задания IP-адреса в Java часто используется объект класса `InetAddress`. У этого класса нет доступных пользователю конструкторов, вместо этого можно использовать ряд статических методов:

`getLocalHost()` – возвращает объект, содержащий IP-адрес и имя компьютера, на котором запущена программа;

`getByName(String host)` – возвращает объект, содержащий IP-адрес указанного компьютера *`host`*;

`getAllByName(String host)` – возвращает массив объектов `InetAddress` в котором содержатся все IP-адреса заданного компьютера `host`;

`getByAddress(byte[] addr)` – возвращает объект, содержащий имя компьютера, по заданному в виде байтового массива IP-адресу;

`getByAddress(String host, byte[] addr)` – возвращает объект класса *`InetAddress`* не проверяя существование компьютера с таким именем и IP-адресом.

При использовании этих методов следует предусмотреть обработку исключительной ситуации `UnknownHostException`.

Для проверки доступности заданного при помощи `InetAddress` компьютера можно использовать метод `isReachable(int timeout)`, возвращающий значение `true` или `false` в зависимости от того, доступен компьютер или нет. Здесь `timeout` задает время в миллисекундах ожидания ответа от проверяемого компьютера.

Для объекта, возвращенного указанными методами можно использовать несколько нестатических методов:

`getAddress()` – возвращает массив из 4 байтов, задающих адрес компьютера;

`getHostName()` – возвращает имя компьютера, соответствующее указанному адресу.

```
InetAddress iAddress = InetAddress.getByName("example.com");
Socket ClientSocket = new Socket(iAddress, 8080);
```

Нужно заметить, что после подключения к другому компьютеру информацию о IP-адресе и номерах портов можно получить из объекта класса `Socket`:

`getInetAddress()` – возвращает `InetAddress`, задающий компьютер с которым соединен объект класса `Socket`;

`getLocalPort()` – задает порт, к которому подключен объект класса `Socket` на удаленной стороне;

`getPort()` – задает порт, который используется на локальном компьютере для подключения к удаленному.

2. Передача/приём данных. После открытия соединения, клиент может с помощью метода `getInputStream()` получить поток ввода и с помощью метода `getOutputStream()` получить поток вывода, связанные с сокетом. После этого можно начинать чтение или запись информации в сокет блоками байтов или использовать один из буферизованных классов.

```
BufferedReader br = null;
PrintWriter pw = null;
br = new BufferedReader ( new InputStreamReader (Client-
Socket.getInputStream()));
pw = new PrintWriter(new BufferedWriter(new OutputStream-
Writer( ClientSocket.getOutputStream())), true);
pw.println("echo!");
String response = br.readLine();
```

3. Закрытие соединения. После выполнения всех необходимых действии сокет должен быть закрыт с помощью метода `close()`.

Язык C#

Создание клиентских приложений при помощи класса TcpClient

Для разработки клиентских сетевых приложений на базе протокола TCP на платформе .NET могут быть использованы классы `TcpClient`, `UdpClient` и `Socket` из пространства имен `System.Net.Sockets` [15]. Класс `TcpClient` инкапсулирует клиентскую сторону TCP-соединения и позволяет управлять основными свойствами соединения. Класс `Socket` является низкоуровневым классом, обеспечивающим максимальный уровень управления в сетевом программировании для .NET.

Схема использования класса System.Net.Sockets.TcpClient

1. Создание соединения с сервером. При создании объекта класса `TcpClient`следует указать имя компьютера-сервера и порт, на который необходимо выполнить соединение, например:

```
TcpClient tcpClient = new TcpClient("mail.vsu", 25);
```

Альтернативой заданию символьного имени сервера служит использование объекта класса `IPEndPoint`, задающего удалённую сторону сетевого соединения.

2. Передача/приём данных. После открытия соединения, клиент может с помощью метода `GetStream()` получить определенный в пространстве имён `System.Net.Sockets` объект `NetworkStream`, который представляет собой поток данных из сети.

```
NetworkStream ns = tcpClient.GetStream();
StreamReader sr = new StreamReader(ns);
string data = sr.ReadToEnd();
```

3. Закрытие соединения. После выполнения всех необходимых действии поток и сетевое соединение должны быть закрыты при помощи метода `Close()`.

```
ns.Close();
tcpClient.Close();
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. СОЗДАНИЕ КЛИЕНТСКИХ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕК ЯВУ

Согласно Вашему варианту разработайте клиентское сетевое приложение.

Вариант 1. Разработайте приложение на основе протокола POP3 для автоматического уведомления пользователя о пришедших новых сообщениях на его почтовый адрес. Программа должна отображать темы и адрес отправителя для новых сообщений.

Вариант 2. Разработайте приложение, отображающее список новостей (обновлений сайта) по указанному пользователем адресу RSS-ленты сайта. Приложение должно позволять указывать адрес RSS-ленты и получать её обновления по требованию пользователя.

Полезная информация: http://validator.w3.org/feed/docs/rss2.html - стандарт RSS 2.0, http://www.opennet.ru/docs/RUS/rss_naklon/ - русскоязычное описание работы RSS.

Вариант 3. Разработайте приложение, создающее off-line галерею изображений с указанной пользователем web-страницы. Приложение должно загружать все изображения (форматы: jpg, png, gif) с указанного адреса и формировать из них web-страницу для локального просмотра.

Вариант 4. Разработайте приложение, отображающее текущую погоду и прогноз на ближайшее время (так называемый метео-информер).

Полезная информация: http://www.gismeteo.by/informers/, http://pogoda.yandex.by/vitebsk/informer/, http://www.meteo-tv.ru/about/informer/.

Вариант 5. Разработайте приложение для организации рассылки новостей по заранее заданному списку адресов электронной почты из текстового файла. Чтение конфигурации и сообщения выполнять из текстового файла настройки.

Вариант 6. Разработайте приложение, отображающее официальный курс различных валют на текущий день. Текущий курс должен быть получен путём разбора результатов выполнения HTTP-запроса к соответствующему серверу, например, http://www.nbrb.by/statistics/rates/ratesDaily.asp (смотрите пояснение по использованию на странице http://www.nbrb.by/statistics/Rates/XML/) или http://myfin.by/scripts/informer/informer-vitebsk.php.

Вариант 7. Разработайте приложение, для управления сообщениями в почтовом ящике пользователя. Приложение должно подключаться к указанному ящику по протоколу POP3, показывать список тем сообщений в почтовом ящике и позволять удалять отмеченные пользователем сообщения.

Вариант 8. Разработайте приложение – простейший браузер, которое получает и сохраняет на диск HTML-страницу, расположенную по указанному пользователем URL. При сохранении страницы HTTP-заголовок ответа не сохраняй-

те. Приложение должно корректно обрабатывать и отображать пользователю основные статусные коды ответа HTTP-сервера.

Вариант 9. Разработайте приложение для чтения электронной почты с указанного пользователем сервера на основе протокола POP3. Пользователь должен иметь возможность указать свой почтовый адрес и пароль, просмотреть список сообщений на сервере и прочитать выбранное им сообщение (или сообщения).

Вариант 10. Разработайте приложение для отправки письма электронной почты на указанный пользователем адрес. Приложение должно позволять пользователю указать SMTP-сервер для отправки, а также контролировать корректное заполнение основных заголовков письма.

Вариант 11. Разработайте приложение для скачивания указанного пользователем файла по протоколу HTTP в 3-потока. Приложение должно предоставлять пользователю возможность выбора каталога для сохранения файла, а также корректно обрабатывать статусные коды ответа протокола HTTP и выводить соответствующие сообщения пользователю (отсутствие файла по указанному адресу, успешное завершение закачки части файла и т.д.).

Вариант 12. Разработайте приложение, уведомляющее пользователя об изменении указанных им web-страниц. Приложение должно позволять пользователю указать URLнекоторой (некоторых) web-страниц и информировать пользователя при их изменениях.

Вариант 13. Разработайте приложение для автоматизированной проверки обновления указанного пользователем каталога на FTP-сервере. Приложение должно позволять пользователю указывать отслеживаемый адрес FTP-сервера и каталог на этом сервере и информировать пользователя о любых изменениях количества файлов в этом каталоге (добавление или удаление).

Вариант 14. Разработайте приложение для автоматической очистки почтового ящика пользователя от спама. Приложение должно позволять пользователю указать адрес почтового сервера и информацию для аутентификации, а также перед удалением сообщений показать их общее количество и запросить подтверждение.

8. ПРОГРАММИРОВАНИЕ СЕРВЕРНОЙ СТОРОНЫ СЕТЕВЫХ ПРИЛОЖЕНИЙ (ЯВУ)

Язык Java. Создание серверных приложений на основе ServerSocket

Для создания серверных приложений в Java в качестве одного из основных классов используется `ServerSocket` из пакета `java.net`[14].

После открытия серверного сокета на определенном порту он переводится в режим прослушивания соединений для ожидания подключений клиентов. После подключения клиента для него открывается отдельный сокет и сервер продолжает прослушивание серверного сокета в ожидании новых подключений. Полученный новый сокет используется для обмена данными с подключившимся клиентом. Приложение, основанное на серверных сокетах, позволяет обслуживать сразу множество клиентов.

Схема использования объекта класса java.net.ServerSocket

1. Создание экземпляра серверного сокета, прослушивающего нужный порт.

Для создания серверного сокета следует использовать конструктор класса `ServerSocket`. При этом конструктор получает номер порта, который сервер должен прослушивать:

```
ServerSocket srvsckt = new ServerSocket(2018);
```

В качестве номера порта рекомендуется задавать не занятое значение из диапазона от 1025 до 65535.

2. Ожидание подключения клиента и получение клиентского сокета.

Для того чтобы получить клиентский сокет следует использовать метод `accept()` объекта серверного сокета:

```
Socketclient = srvsckt.accept();
```

3. Дальнейший обмен информацией полностью аналогичен работе с клиентским сокетом, рассмотренной выше.

Язык C#. Создание серверных приложений на основе TcpListener

На платформе Microsoft .NETFramework для создания серверных приложений может использоваться класс `TcpListener` из пространства имён `System.Net.Sockets` [17]. Класс `TcpListener`, как и рассматривавшиеся выше классы и компоненты, предназначен для прослушивания входящих подключений от TCP-клиентов. После подключения клиента для взаимодействия с ним возвращается `Socket` или `TcpClient`, а `TcpListener` продолжает прослушивание серверного сокета в ожидании новых подключений.

Свойства и методы класса достаточно подробно документированы в MSDN (в том числе и на русском языке): http://msdn.microsoft.com/ru-ru/library/System.Net.Sockets.TcpListener(v=vs.110).aspx

Схема использования объекта класса `TcpListener`

1. Создание экземпляра `TcpListener` и запуск прослушивания нужного порта на нужном интерфейсе.

Для создания экземпляра следует использовать один из 3 конструкторов: по конечной точке (`IPEndPoint`), по IP адресу и порту или только по порту (устаревший):

```
TcpListener server = new TcpListener(IPAddress.Any, 2018);
```

или, если необходим определенный интерфейс для прослушивания:

```
IPAddressip = IPAddress.Parse("127.0.0.1");
TcpListener server = new TcpListener(ip, 2018);
```

В качестве номера порта рекомендуется задавать не занятое значение из диапазона от 1025 до 65535.

После создания экземпляра класса его следует перевести в режим прослушивания входящих соединений методом `Start()`:

```
server.Start();
```

2. Ожидание подключения клиента и получение клиентского сокета.

Для проверки есть ли в очереди ожидающие клиентские подключения используется метод `Pending()`.

Для того чтобы получить клиентский сокет следует использовать метод `AcceptSocket()` или метод `AcceptTcpClient()` для получения экземпляра `TcpClient`. При этом для того чтобы сохранить отзывчивость интерфейса поль-

зователя код ожидания и обработки клиентских подключений следует вынести в отдельный поток или использовать `Application.DoEvents();`

```
...
Application.DoEvents();
if(server.Pending()==true)
{
TcpClient client = server.AcceptTcpClient();
    ...
}
...
```

3. Дальнейший обмен информации полностью аналогичен работе с объектом клиентского класса и `NetworkStream`, рассмотренной в предыдущих лабораторных работах.

4. По окончании работы приложения остановить прослушивание сокета методом `Stop()`.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. ПРОГРАММИРОВАНИЕ СЕРВЕРНОЙ СТОРОНЫ СЕТЕВЫХ ПРИЛОЖЕНИЙ

Согласно варианту, разработайте серверное сетевое приложение с использованием библиотек одного из языков высокого уровня.

Вариант 1. Разработайте приложение – простейший HTTP-сервер: приложение должно обрабатывать GET-запросы от клиента (браузера пользователя) и возвращать ответ с нужным кодом (200, 404 и т.д.) и содержимым (файлы из заранее заданного каталога). При работе сервера входящие запросы и заголовки ответов на них должны протоколироваться в файл.

Использовать для разработки приложения специализированные компоненты или классы для реализации функции протокола HTTP (такие как, например, HttpListener в C#) запрещено.

Вариант 2. Разработайте приложение – простейший почтовый POP3-сервер: приложение должно корректно обрабатывать команды: USER, PASS, RETR, DELE, LIST, STAT, TOP, QUIT. Приложение должно корректно информировать клиента о статусе выполнения команды (+OK, -ERR). Сообщения пользователей должны храниться в заранее заданном каталоге и соответствовать принятой структуре сообщения электронной почты.

Пояснения, примеры и список ссылок на стандарты, например, https://ru.wikipedia.org/wiki/POP3

Вариант 3. Разработайте сетевое приложение – простейший почтовый SMTP-сервер. Приложение должно реализовать поддержку команд MAIL FROM, RCPT TO, QUIT и DATA. На все получаемые команды сервер должен возвращать соответствующие коды ответа. По возможности сервер должен реализовать ограничение доступа к серверу по местоположению. Приложение может реализовать функции SMTP-транслятора: вместо отправки сообщений почтовому серверу получателя передавать сообщение следующему почтовому серверу.

Вариант 4. Разработайте сетевое приложение – сервер для игры «Крестики-нолики». Сервер должен предусматривать подключение 2 клиентов. Если подключен только 1 клиент, то он должен ожидать подключения второго пользователя. После подключения клиентов сервер должен контролировать очередность ходов клиентов и правильность каждого хода (запрещать ставить «крестик» или «нолик» в занятую клетку и т.д.). Для тестирования приложения можно использовать сетевой терминал PuTTY.

Вариант 5. Разработайте сетевое приложение – простейший чат-сервер. Приложение должно прослушивать определенный порт, принимать от клиентов подключения и реализовать для каждого подключенного клиента набор команд, включающий: задание имени пользователя, получение списка пользователей на сервере, отправку сообщения всем пользователям, отправку сообщения определенному пользователю (т.н. приватное сообщение), установку темы чата, получение темы чата.

Пример команд и их формат можно получить в документации по IRC, например, по ссылкам с https://ru.wikipedia.org/wiki/IRC.

Вариант 6. Разработайте сетевое приложение – простейший сервер сетевых викторин. Суть проекта: сервер содержит базу вопросов и ответов. По команде пользователя запускается очередной тур викторины. Сервер выбирает случайным образом вопрос из базы и задаёт его всем подключенным игрокам. Первый, кто даёт верный ответ считается победителем и ему начисляются призовые очки. Приложение должно прослушивать определенный порт, принимать подключения от клиентов и реализовать для каждого подключенного клиента набор команд, включающий: задание имени пользователя, запуск/перезапуск викторины (если вопросов было задано не менее некоторого порогового числа, например, 10), предъявление вопроса пользователям, приём ответа на вопрос от пользователя, вывод рейтинга игроков в зависимости от количества правильных ответов.

Вариант 7. Разработайте сетевое приложение – простейший сервер – информационного бота. Приложение должно реализовать следующие команды: вывод текущей погоды, вывод текущего времени и даты, вывод текущего курса валют, вывод текущей телепрограммы для указанного пользователем телеканала. В ответ на запрос с соответствующей командой должен отправляться ответ с требуемой информацией. Для простоты выводимая информация может храниться в файле на сервере и считываться по мере надобности, а не собираться в Интернете на профильных сайтах.

Вариант 8. Разработайте сетевое приложение – простейший сервер для текстовой ролевой компьютерной игры (многопользовательские версии таких приложений называются MUD). Приложение должно прослушивать определенный порт, принимать подключение от пользователя и реализовать возможность одновременного присутствия на некоторой виртуальной карте нескольких персонажей. Для пользователей должно быть доступно не менее 10 команд (например, пройти вперед, повернуть влево, повернуть вправо, сказать что-либо и др.).

Более подробное описание, примеры сообщений и ссылки на существующие разработки можно найти, например, на http://ru.wikipedia.org/wiki/MUD

Вариант 9. Разработайте сетевое приложение – простейший сервер для текстовой пошаговой игры жанра «файтинг». Сервер должен предусматривать подключение 2 клиентов. Если подключен только 1 клиент, то он должен ожидать подключения второго пользователя. После подключения второго пользователя, сервер предоставляет каждому из игроков возможность ввода команды вида: «хук левой», «блок левой», «джеб правой», «уклон вправо» и т.д. После получения команды от обоих игроков, сервер сравнивает их действия и решает исход очередного раунда схватки. В зависимости от результата игрокам начисляются очки (уменьшаются заранее заданные). Игра идёт до получения одним из игроков требуемого количества очков, после чего объявляется победитель и сервер переходит в режим приёма подключений от следующих игроков.

Вариант 10. Разработайте сетевое приложение – простейший сервер Telnet. Приложение должно прослушивать стандартный для Telnet порт, принимать входящие подключения, в рамках подключения принимать команды и исполнять их на локальном компьютере. Вывод исполняемых команд должен отправляться подключенному клиенту. Предусмотрите простейшую аутентификацию пользователя (например, имя пользователя и пароль в открытом виде).

Краткое описание на русском языке: https://ru.wikipedia.org/wiki/Telnet

9. ПРОГРАММИРОВАНИЕ В ОБЛАЧНОМ СЕРВИСЕ

Представители специализированныхонлайн сервисов для программирования, такие как Ideone, Cloud9 и Codebox и др., построены на «облачных» мощностях. Благодаря высокой продуктивности компьютерного программного обеспечения они вызывают все больший интерес разработчиков. Эти сервисы позволяют в режиме онлайн создавать тексты программ на разных языках программирования и запускать эти программы на исполнение с возможностью анализа полученных результатов.

Типичным примеромподобных облачных технологий является web-сервис Ideone [11]. Преимуществами сервиса Ideone являются следующие возможности:

– создание, редактирование и тестирование небольших программ без использования «тяжеловесных» интегрированных сред;

– подсветка синтаксиса, возможность сохранения программ;

– онлайн компиляция и выполнение кода на более чем 60 языках программирования прямо в браузере;

– возможность делиться фрагментами кода с другими программистами, используя предоставляемую ссылку;

– возможность разработки небольших программ на мобильных устройствах на любых языках, посредством одного легковесного приложение.

Вместе с тем, несмотря на название, сервис не является полноценной IDE (Интегрированной средой разработки), его функционал существенно меньше. Имеется ряд ограничений на использование памяти, времени компиляции и выполнения.

Интерфейс программы предельно прост и понятен (рисунок 5). Большую часть окна занимает рабочая область для создания и редактирования кода. Под ней расположена область входных данных и инструменты управления.

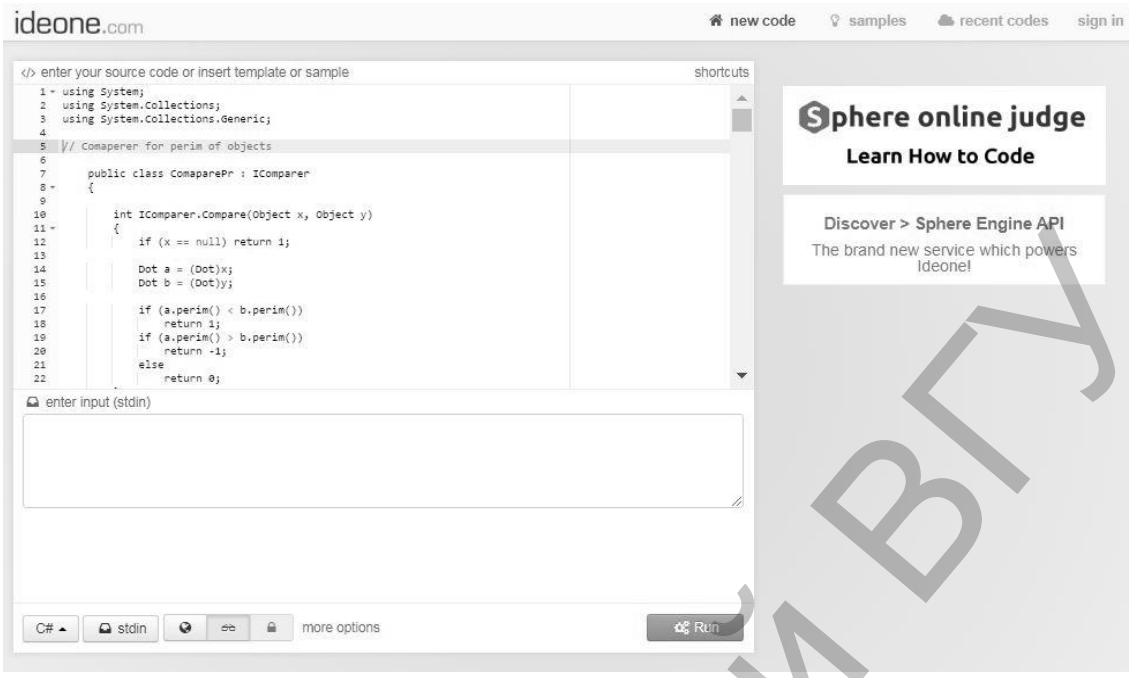

Рисунок 5.Интерфейс облачного сервиса программирования Ideone.

ЗАДАНИЕ. СЕРВИС IDEONE

1. Изучите возможности и ограничения сервисаIdeone. Для этого воспользуйтесь ресурсом https://ideone.com/faq.
2. Зарегистрируйтесь на сайтеhttps://ideone.com/. Уточните настройки своего аккаунта.
3. Подготовьте пробный пример для тестирования сервиса (небольшой фрагмент кода). Можно использовать предлагаемые сервисом примеры.
4. Выберите язык программирования (в нижней части окна) и вставьте код в окно редактирования. Можно включить подсветку синтаксиса (в дополнительных опциях или по умолчанию в настройках аккаунта).
5. Скомпилируйте, отладьте и выполните программу.
6. Изучите инструменты сетевого взаимодействия (возможности поделиться своим кодом и открыть чужой). Обратите внимание на то, как настроить статус доступности ваших программ.
7. Перешлите ссылку на свой пример коллеге.
8. Откройте чужой пример по ссылке, которой поделились с вами.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ. СОВМЕСТНАЯ РАЗРАБОТКА ПРОГРАММЫ.

1. Выберите соавтора для разработки программы. Уточните номер варианта задания для вашего мини-коллектива у преподавателя.
2. Договоритесь с соавтором о распределении работы по написанию и отладке кода программы. Используйте разные аккаунты для онлайн сервиса программирования.
3. Обменяйтесь фрагментами кода с соавтором. Ваша общая программа должна быть отлажена и безошибочно работать у каждого из участников.
4. Поделитесь ссылкой на готовую программу с преподавателем.

Варианты заданий:

Создать иерархию классов. В тестирующей программе продемонстрировать наследование и полиморфизм.

№	**Классы**	**Методы**
1.	Сотрудник (поля: имя, p – минимальная зарплата) Менеджер (поле объем продаж в тоннах) Инженер (поле количество разработанных проектов – n)	Доход: k *p, где k – повышающий коэффициент Доход менеджера изменить в зависимости от объема продаж (если >H, то увеличить на 1% от H) Доход инженера увеличить на 4.8*n.
2.	Квартира (поля: номер, стоимость 1м^2, площадь) Квартира в центре (поля: номер этажа, этажность дома) Квартира в пригороде (поле расстояние от центра r)	Стоимость Увеличить стоимость с учетом надбавки за расположение в центре на 1%, и уменьшить на 1000$ для квартир на 1 и последнем этажах. Изменить стоимость в зависимости от расстояния: если < 10км увеличить на 3%, в противном случае уменьшить на 0,01*r.
3.	Агрополе (поля: название, вес r посеянных семян на единицу площади, S – площадь в га) Фермерское (поле: количество внесенных удобрений m) Приусадебное (поле: сроки посева – ранний, средний, поздний)	Урожай: k*r, где k- коэффициент, зависящий от культуры Найти урожай с учетом увеличения на 0,001*mна единицу площади. Изменить урожай с учетом сроков (+10% для раннего, – 5% для позднего) и площади посевов (<0,01 увеличить на 50% от S)
4.	Стол (поля: название,площадь S в см) Письменный (поле – используемый материал, стоимость отделки) Обеденный (поле – форма)	Стоимость: $S^2/3+30$ Увеличить стоимость на стоимость отделки. Изменить стоимость в зависимости от формы: увеличить для прямоугольной формы на 10%, овальной – на 20% при S<0,5 м^2 и S>2 м^2.
5.	Автобус (поля: количество пассажиров, стоимость билета)	Выручка

№	**Классы**	**Методы**
	Экспресс (поле – средняя скорость v, марка автобуса)	Изменить выручку с учетом увеличения на 0.05*v к цене билета.
	Пригородный (поле – расстояние r)	Изменить выручку с учетом уменьшения на 0.01*r к цене билета.
6.	Транспортное средство (поля: название, расстояние, цена за 1 км)	Стоимость проезда
	Самолет (высота – h км, скорость – v км/ч)	Увеличить стоимость проезда на 100*h*v
	Корабль (количество палуб k, номер палубы n)	Увеличить стоимость проезда на палубах №3-4 на k^2%
7.	Пособие (поля: повышающий коэффициент k, минимальное пособие r)	Размер пособия: k*r
	Инвалид (поле – номер группы)	Увеличить пособие для инвалидов 1-й группы на 30%, 2-й – на 20%.
	Многодетные семьи (поле – количество детей)	Увеличить пособие от 3 до 5 детей на 10%, > 5 – на 20%.
8.	Автомобиль (поля: марка, расход горючего на 100км N, расстояние R в км)	Объем горючего: N*R
	Грузовой (поле – грузоподъемность p в т)	Увеличить объем горючего на величину $M = \sqrt{p} * R$
	Легковой (поле – объем двигателя V в л)	Увеличить объем горючего для объема > 3 на величину M= 0,005*V*R
9.	Постройка (поля: название, высота здания V м)	Высота фундамента: 0,03*V
	Офис (поле – количество этажей N)	Изменить высоту фундамента: для офиса с N>10 на+ 0,005*N
	Завод (поле – вес оборудования G)	для завода на +0,000002*G
10.	Аудитория (поля: номер, площадь S $м^2$)	Количество мест: [S/1,2]
	Лекционная (поле количество ярусов K)	Увеличить количество мест в лекционной аудитории на 2*K
	Компьютерная (поле количество компьютеров P)	Заменить количество мест в компьютерной аудитории на P-1

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Инструкция об организации, проведении и требованиях к содержанию, оформлению и защите рефератов, курсовых проектов (работ), дипломных проектов (работ), магистерских диссертаций: утв. приказом ректора ВГУ им. П.М. Машерова от 18.03.2016. – Витебск, 2016. – 49с. – Режим доступа: http://lib.vsu.by/files_site/news/instruction.pdf Дата доступа: 01.09.2017.
2. Каптерев, А. Смерть через PowerPoint и как ее избежать. Режим доступа: https://www.slideshare.net/thecroaker/death-by-powerpoint-rus. – Дата доступа: 01.09.2017.
3. Каптерев, А. Мастерство презентации. – М.: Манн, Иванов, Фарбер, Эксмо. 2014. – 336 с. – Режим доступа: http://avidreaders.ru/read-book/masterstvo-prezentacii-kak-sozdavat-prezentacii-kotorye.html. – Дата доступа: 01.09.2017.
4. Абламейко, С.В. «Облачные» технологии в образовании / С.В. Абламейко, Ю.И. Воротницкий, Н.И. Листопад // Электроника: ежемесячный журнал для специалистов. – Минск: БГУ, 2013. – № 9. – С. 30–34.
5. Концепция информатизации системы образования Республики Беларусь на период до 2020 г. // Официальный интернет-портал Министерства образования Республики Беларусь / Режим доступа:http://edu.gov.by/doc-437693. – Дата доступа: 01.09.2017.
6. Сервис визуальных закладок Symbaloo. – Режим доступа: http://symbaloo.com. – Дата доступа: 01.09.2017.
7. Сервис вычислительно-поисковой системы. – Режим доступа: https://www.wolframalpha.com. – Дата доступа: 01.09.2017.
8. Сервис интеллектуально-поисковой системы. – Режим доступа: http://www.nigma.ru. – Дата доступа: 01.09.2017.
9. Сервис ментальных карт MindMeister. – Режим доступа: http://mindmeister.com. – Дата доступа: 01.09.2017.
10. Сервисы Google. – Режим доступа: http://www.google.by/intl/ru/about/products/. – Дата доступа: 01.09.2017
11. Сервис онлайн программирования Ideone. – Режим доступа: https://ideone.com/ . – Дата доступа: 01.09.2017.
12. Windows Sockets 2. – Режим доступа: https://msdn.microsoft.com/en-us/library/windows/desktop/ms740673(v=vs.85).aspx. – Дата доступа: 01.09.2017.
13. Class Socket. – Режим доступа: http://docs.oracle.com/javase/8/docs/api/java/net/Socket.html. – Дата доступа: 01.09.2017.

14. Class ServerSocket. – Режим доступа: http://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html. – Дата доступа: 01.09.2017.
15. Кровчик, Э. .Net сетевое программирование для профессионалов / Э. Кровчик, В. Кумар, Н. Лагари, А. Мунгале, К. Нагель, Т. Паркер, Ш. Шивакумар. – М.: Лори, 2014. – 420с.
16. Фейт, С. TCP/IP. Архитектура, протоколы, реализация / С. Фейт. – М.: Лори, 2015. – 450с.
17. Пространство имен System.Net.Sockets. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/system.net.sockets(v=vs.110).aspx. – Дата доступа: 01.09.2017.

Учебное издание

НОВЫЙ Вадим Владимирович
АЛЕЙНИКОВА Татьяна Григорьевна

СЕТЕВЫЕ ТЕХНОЛОГИИ И СЕРВИСЫ

Методические рекомендации

Технический редактор *Г.В. Разбоева*
Компьютерный дизайн *Т.Е. Сафранкова*

Подписано в печать 02.11.2017. Формат 60x84 $^{1}/_{16}$. Бумага офсетная.
Усл. печ. л. 2,90. Уч.-изд. л. 2,46. Тираж 45 экз. Заказ 179.

Издатель и полиграфическое исполнение – учреждение образования
«Витебский государственный университет имени П.М. Машерова».

Свидетельство о государственной регистрации в качестве издателя,
изготовителя, распространителя печатных изданий
№ 1/255 от 31.03.2014 г.

Отпечатано на ризографе учреждения образования
«Витебский государственный университет имени П.М. Машерова».
210038, г. Витебск, Московский проспект, 33.