МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

Утверждаю
Проректор по учебной работе
О.Г. Локтионова
«25» 09 2023 г.

БЕСПРОВОДНЫЕ ТЕХНОЛОГИИ ПЕРЕДАЧИ ДАННЫХ

Методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр)

Курск 2023

УДК 621.(076.1)

Составители: А.А.Кузьмин

Рецензент:
Кандидат технических наук, доцент *Т.Н. Конаныхина*

Беспроводные технологии передачи данных: методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр) / Юго-Зап. гос. ун-т; сост.: А.А.Кузьмин. - Курск, 2023. - 23с.

Содержат методические рекомендации к проведению лабораторных работ по дисциплине «Беспроводные технологии передачи данных». Методические указания по структуре, содержанию и стилю изложения материала соответствуют методическим и научным требования, предъявляемым к учебным и методическим пособиям.

Предназначены для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр)

Текст печатается в авторской редакции

Подписано в печать 25.09.23 Формат 60x84 1/16
Усо.печ.л. 1,3. Уч.-изд.л. 1,2 .Тираж 30 экз. Заказ: 1086 .Бесплатно.
Юго-Западный государственный университет.
305040. г. Курск, ул. 50 лет Октября, 94.

Лабораторная работа №1

Изучение команд операционной системы для контроля функционирования сети

2.1. Цель работы: изучить команды операционной системы, позволяющие оценить наличие сетевого подключения, его свойства, качество связи, пропускную способность сети, список ее вспомогательных узлов и прочие параметры

2.2 Краткие теоретические сведения

Большинство рассматриваемых сетевых утилит для полноценной работы требуют наличия административных привилегий. Для операционных систем семейства Windows 2000/XP достаточно того, чтобы пользователь работал под учетной записью члена группы администраторов. Интерпретатор командной строки cmd.exe можно запустить с использованием меню Пуск - Выполнить - cmd.exe. В среде операционных систем Widows Vista/Windows 7 интерпретатор cmd.exe должен быть запущен для выполнения с использованием пункта контекстного меню "Запустить от имени администратора". Командные файлы, в которых используются сетевые утилиты, также должны выполняться в контексте учетной записи с привилегиями администратора.

В списке представлены сетевые утилиты командной строки для получения информации о сетевых настройках, выполнения операций по конфигурированию и диагностике сети.

В описании команд используется

< текст > - текст в угловых скобках. Обязательный параметр

[текст] - текст в квадратных скобках. Необязательный параметр.

(текст) - текст в круглых скобках. Необходимо выбрать один из параметров.

Вертикальная черта | - разделитель для взаимоисключающих параметров. Нужно выбрать один из них.

Многоточие ... - возможно повторение параметров.

Краткое описание и примеры использования сетевых утилит командной строки Windows:

ARP

IPCONFIG

GETMAC

NBTSTAT

NETSH

NETSTAT

NET

NSLOOKUP

PATHPING

PING

ROUTE

TELNET

TRACERT

2.3. Задание на лабораторную работу

Используя команды операционной системы, такие как ping, tracert, ipconfig, net, команды подсистемы netbios, провести трассировку и изучение параметров сети с использованием узлов, заданных преподавателем.

2.4 Содержание отчета

1. Титульный лист
2. Задание на лабораторную работу
3. Листинг программы
4. Скриншот работы приложения

Лабораторная работа №2

Программирование службы DNS с использованием языка Java

2.1. Цель работы

Изучить классы и их методы, которые позволяют реализовать доменное разрешение имен, в частности преобразование имен в адреса и обратное преобразование

2.2. Краткие теоретические сведения

Если посмотреть большинство более-менее популярных статей об устройстве Интернета, то про DNS там чаще всего будет сказано что-то типа "DNS-сервер обеспечивает трансляцию имен сайтов в IP адреса". В принципе, это действительно является его основной задачей, и для большинства пользователей (и даже компьютерщиков) этих знаний вполне достаточно. Но если вдруг вам придется отлаживать сеть, которой провайдер выделил какой-то блок "честных" адресов, или поднимать в локальной сети свой DNS-сервер, то очень быстро всплывут всякие страшные слова: "зона", "трансфер", "форвардер", "in-addr.arpa" и другие... Поэтому в этой заметке мы попробуем чуть более подробно поговорить о работе DNS.

Очень приблизительно можно сказать, что у каждого компьютера в Интернете есть два основных идентификатора: доменное имя (например, www.listsoft.ru) и IP-адрес (например, 127.0.0.1). Приблизительность заключается в том, что, во-первых, IP-адресов у компьютера может быть несколько (мало того, что у каждого интерфейса свой адрес, так еще и несколько адресов могут "висеть" на одном интерфейсе); во-вторых, имен тоже может быть несколько, причем они могут связываться как с одним, так и с несколькими IP-адресами; в-третьих, у компьютера может и не быть доменного имени... Словом, ясно, что картина уже начинает запутываться.

Основной задачей DNS-сервера является трансляция доменных имен в IP-адреса и обратноКак было сказано выше, основной задачей DNS-сервера является трансляция доменных имен в IP адреса и обратно. На заре становления Интернета (когда он еще был ARPANET'ом) это решалось ведением длинных списков, включающих все компьютеры сети, причем копия такого списка должна была присутствовать на каждом компьютере. Понятно, что с ростом сети эта технология

перестала удовлетворять кого бы то ни было — ведь эти файлы надо было еще и синхронизировать, не говоря уж об их размере... Некоторые "пережитки" этого метода можно обнаружить и сейчас: существует файл HOSTS (и в UNIX, и в Windows), в котором можно прописывать адреса серверов, с которыми вы регулярно работаете (кстати, именно его использование лежит в основе многих "ускорителей Интернета" — такие программы просто записывают адреса серверов, к которым вы обращаетесь, в файл HOSTS и при следующем обращении берут данные из него, не тратя время на запрос к DNS-серверу).

На смену "однофайловой" схеме пришел DNS — иерархическая структура имен. Существует "корень дерева" с именем "." (точка). Так как корень един для всех доменов, то точка в конце имени обычно не ставится (но она используется в описаниях DNS — тут надо быть очень внимательным!). Ниже корня лежат домены первого уровня. Их немного — com, net, edu, org, mil, int, biz, info, gov (есть еще несколько) и домены государств, например, ru. Еще ниже находятся домены второго уровня, например, listsoft.ru. Еще ниже — третьего и т.д.

DNS-сервер работает как хороший компьютерщик: он всегда либо знает ответ, либо знает у кого спросить...Иерархичность DNS-серверов — штука довольно интересная, если проследить прохождение запроса. При установке (точнее, при настройке) клиенту указывается как минимум один DNS-сервер (как правило, их два) — его адрес выдается провайдером. Клиент посылает запрос этому серверу. Сервер, получив запрос, либо отвечает (если ответ ему известен), либо пересылает запрос на "вышестоящий" сервер (если он известен) или на корневой (каждому DNS-серверу известны адреса корневых DNS-серверов). Так выглядит "восходящая иерархия". Затем запрос начинает спускаться вниз — корневой сервер пересылает запрос серверу первого уровня, тот — серверу второго уровня и т.д. Таким образом, каждый DNS-сервер работает как хороший компьютерщик: он всегда либо знает ответ, либо знает, у кого спросить...

Помимо "вертикальных связей", у серверов есть еще и "горизонтальные" отношения — "первичный — вторичный". Действительно, если предположить, что сервер, обслуживающий какой-то домен и работающий "без страховки" вдруг перестанет быть

доступным, то все машины, расположенные в этом домене, окажутся недоступны! Именно поэтому при регистрации домена второго уровня выдвигается требование указать минимум два сервера DNS, которые будут этот домен обслуживать.

Рекурсивные сервера удобно использовать в локальных сетяхDNS-сервера бывают рекурсивные и нерекурсивные. Первые всегда возвращают клиенту ответ — они самостоятельно отслеживают отсылки к другим DNS-серверам и опрашивают их. Нерекурсивные сервера возвращают клиенту эти отсылки, так что клиент должен самостоятельно опрашивать указанный сервер. Рекурсивные сервера удобно использовать на низких уровнях, в частности, в локальных сетях. Дело в том, что они кэшируют все промежуточные ответы, и при последующих запросах ответы будут возвращаться намного быстрее. Нерекурсивные сервера обычно стоят на верхних ступенях иерархии — поскольку они получают очень много запросов, то для кэширования ответов никаких ресурсов не хватит.

Использование "пересыльщиков" ускоряет разрешение именПолезным свойством DNS является умение использовать "пересыльщиков" (forwarders). "Честный" DNS-сервер самостоятельно опрашивает другие сервера и находит нужный ответ, но если ваша сеть подключена к Интернету по медленной (например, dial-up) линии, то этот процесс может занимать довольно много времени. Вместо этого можно перенаправлять все запросы, скажем, на сервер провайдера, а затем принимать его ответ. Использование "пересыльщиков" может оказаться интересным и для больших компаний с несколькими сетями: в каждой сети можно поставить относительно слабый DNS-сервер, указав в качестве "пересыльщика" более мощную машину, подключенную по быстрой линии. При этом все ответы будут кэшироваться на этом мощном сервере, что ускорит разрешение имен для целой сети.

Для каждого домена администратор ведет базу данных DNS. Эта база данных представляет собой набор простых текстовых файлов, расположенных на основном (первичном) сервере DNS (вторичные сервера периодически копируют к себе эти файлы). В файлах конфигурации сервера указывается, в каком именно файле содержатся

описания каких зон, и является ли сервер первичным или вторичным для этой зоны.

Элементы базы DNS часто называют RR (сокращение от Resource Record). Базовый формат записи выглядит так:

[имя] [время] [класс] тип данные

Имя может быть относительным или абсолютным (FQDN — Fully Qualified Domain Name). Если имя относительное (не заканчивается точкой — помните про корневой домен?), то к нему автоматически добавляется имя текущего домена. Например, если в домене listsoft.ru я опишу имя «www», то полное имя будет интерпретироваться как "www.listsoft.ru." Если же это имя указать как "www.listsoft.ru" (без последней точки), то оно будет считаться относительным и будет интерпретировано как "www.listsoft.ru.listsoft.ru."

Время задает интервал времени в секундах, в течение которого данные могут сохраняться в кэше сервера.

Класс определяет класс сети. Практически всегда это будет IN, обозначающее INternet.

Тип может быть одним из следующих:

SOA — определяет DNS зону

NS — сервер имен для зоны

A — преобразование имени в IP-адрес

PTR — преобразование IP-адреса в имя

MX — почтовая станция

CNAME — имена машины

HINFO — описание "железа" компьютера

TXT — комментарии или какая-то другая информация

Есть также некоторые другие типы, но они намного менее распространены.

В записях можно использовать символы # и ; для комментариев, @ для обозначения текущего домена, () — скобки — для написания данных на нескольких строках. Кроме того, можно использовать метасимвол * в имени. Порядок записей не имеет значения за одним исключением: запись SOA должна идти первой. Дальнейшие записи считаются относящимися к той же зоне, пока не встретится новая запись

SOA. Как правило, после записи зоны указывают записи DNS-серверов, а остальные записи располагают по алфавиту, но это не обязательно.

SOA — описание зоны Теперь попробуем рассмотреть записи. Первой описываем зону:

mycompany.ru. IN SOA ns.mycompany.ru. admin.mycompany.ru. (1001 ; serial

21600 ; Refresh — 6 часов

1800 ; Retry — 30 мин

1209600 ; Expire — 2 недели

432000) ; Minimum — 5 дней

Сначала идет имя домена: mycompany.ru. (обратите внимание на точку в конце имени). Вместо имени можно было (и чаще всего так и делают) поставить знак @.

ns.mycompany.ru. — основной сервер имен

admin.mycompany.ru. — почтовый адрес администратора в формате имя(точка)машина

затем в круглых скобках идут поля, необходимые для правильного "восприятия" вашей зоны другими серверами. Первое число — serial — является "версией" файла зоны. При внесении изменений это число надо увеличить — если вторичный сервер увидит, что его версия зоны меньше, чем у первичного сервера, то он перечитает данные. Типичной ошибкой является обновление зоны без обновления этого числа. Очень удобно в качестве serial использовать текущую дату, например, 2003040401 — 4 апреля 2003 года, первое обновление.

Refresh говорит вторичным серверам, как часто они должны проверять значение serial.

Retry говорит о том, как часто вторичный сервер должен пытаться прочитать данные, если первичный сервер не отвечает.

Expire говорит вторичным серверам, в течение какого времени они должны обслуживать домен, если первичный сервер не отвечает. По истечении этого времени вторичные сервера будут считать свои данные устаревшими

2.3. Задание на лабораторную работу

Создайте программу, реализующие следующие функции:

1. Пользователь вводит IP адрес, а программа выдает ему DNS имя, которое относится к этому адресу

2. Пользователь вводит DNS имя сети, а программа выдает ему значение IP адреса, соответствующее этому имени

3. Пользователь вводит DNS имя требуемого узла. Программа производит поиск и печать всего списка IP адресов, принадлежащих заданому узлу.

2.4. Содержание отчета

1. Титульный лист
2. Задание на лабораторную работу
3. Листинг программы
4. Скриншот работы приложения

Лабораторная работа №3

Многопоточность в сетевых приложениях

3.1. Цель работы

Изучить практические аспекты создания программ, реализующих свое функционирование с использованием нескольких потоков выполнения

3.2. Краткие теоретические сведения

Наиболее очевидная область применения многопоточности – это программирование интерфейсов. Многопоточность незаменима тогда, когда необходимо, чтобы графический интерфейс продолжал отзываться на действия пользователя во время выполнения некоторой обработки информации. Например, поток, отвечающий за интерфейс, может ждать завершения другого потока, загружающего файл из интернета, и в это время выводить некоторую анимацию или обновлять прогресс-бар. Кроме того он может остановить поток загружающий файл, если была нажата кнопка «отмена».

Еще одна популярная и, пожалуй, одна из самых хардкорных областей применения многопоточности – игры. В играх различные потоки могут отвечать за работу с сетью, анимацию, расчет физики и т.п.

Процесс — это совокупность кода и данных, разделяющих общее виртуальное адресное пространство. Чаще всего одна программа состоит из одного процесса, но бывают и исключения (например, браузер Chrome создает отдельный процесс для каждой вкладки, что дает ему некоторые преимущества, вроде независимости вкладок друг от друга). Процессы изолированы друг от друга, поэтому прямой доступ к памяти чужого процесса невозможен (взаимодействие между процессами осуществляется с помощью специальных средств).

Для каждого процесса ОС создает так называемое «виртуальное адресное пространство», к которому процесс имеет прямой доступ. Это пространство принадлежит процессу, содержит только его данные и находится в полном его распоряжении. Операционная система же отвечает за то, как виртуальное пространство процесса проецируется на физическую память.

Схема этого взаимодействия представлена на картинке. Операционная система оперирует так называемыми страницами памяти, которые представляют собой просто область определенного фиксированного размера. Если процессу становится недостаточно памяти, система выделяет ему дополнительные страницы из физической памяти. Страницы виртуальной памяти могут проецироваться на физическую память в произвольном порядке.

При запуске программы операционная система создает процесс, загружая в его адресное пространство код и данные программы, а затем запускает главный поток созданного процесса.

Один поток – это одна единица исполнения кода. Каждый поток последовательно выполняет инструкции процесса, которому он принадлежит, параллельно с другими потоками этого процесса.

Следует отдельно обговорить фразу «параллельно с другими потоками». Известно, что на одно ядро процессора, в каждый момент времени, приходится одна единица исполнения. То есть одноядерный процессор может обрабатывать команды только последовательно, по одной за раз (в упрощенном случае). Однако запуск нескольких параллельных потоков возможен и в системах с одноядерными процессорами. В этом случае система будет периодически переключаться между потоками, поочередно давая выполняться то одному, то другому потоку. Такая схема называется псевдо-параллелизмом. Система запоминает состояние (контекст) каждого потока, перед тем как переключиться на другой поток, и восстанавливает его по возвращению к выполнению потока. В контекст потока входят такие параметры, как стек, набор значений регистров процессора, адрес исполняемой команды и прочее. Проще говоря, при псевдопараллельном выполнении потоков процессор мечется между выполнением нескольких потоков, выполняя по очереди часть каждого из них.

После запуска побочного потока его инструкции начинают выполняться вперемешку с инструкциями главного потока. Кол-во выполняемых инструкций за каждый подход не определено.

То, что инструкции параллельных потоков выполняются вперемешку, в некоторых случаях может привести к конфликтам доступа к данным.

3.3. Задание на лабораторную работу

Получите у преподавателя математическую функцию. Создайте программу, которая запрашивает у пользователя пределы интегрирования a и b. Далее программа запрашивает требуемое количество потоков, после чего запускает параллельный поиск интеграла заданной функции на требуемом интервале. Интервал приращения dx принять равным 0.00000001. Количество потоков не должно превышать 20.

3.4. Содержание отчета

1. Титульный лист
2. Задание на лабораторную работу
3. Листинг программы
4. Скриншот работы приложения

Лабораторная работа №4

Передача информации с использованием протокола tcp

1.1. Цель работы: изучить основы передачи данных по протоколу tcp/ip, позволяющит решать задачи сетевого обмена

1.2. Краткие теоретические сведения

Сокеты (сетевые разъёмы) – это логическое понятие, соответствующее разъёмам, к которым подключены сетевые компьютеры и через которые осуществляется двунаправленная поточная передача данных между компьютерами. Сокет определяется номером порта и IP-адресом. При этом IP-адрес используется для идентификации компьютера, номер порта – для идентификации процесса, работающего на компьютере. Когда одно приложение знает сокет другого, создается сокетное протоколо-ориентированное соединение по протоколу TCP/IP. Клиент пытается соединиться с сервером, инициализируя сокетное соединение. Сервер прослушивает сообщение и ждет, пока клиент не свяжется с ним. Первое сообщение, посылаемое клиентом на сервер, содержит сокет клиента. Сервер, в свою очередь, создает сокет, который будет использоваться для связи с клиентом, и посылает его клиенту с первым сообщением. После этого устанавливается коммуникационное соединение.

Сокетное соединение с сервером создается клиентом с помощью объекта класса Socket. При этом указывается IP-адрес сервера и номер порта. Если указано символьное имя домена, то Java преобразует его с помощью DNS-сервера к IP-адресу. Например, если сервер установлен на этом же компьютере, соединение с сервером можно установить из приложения клиента с помощью инструкции:

```
Socket socket = new Socket("ИМЯ_СЕРВЕРА", 8030);
```

Сервер ожидает сообщения клиента и должен быть заранее запущен с указанием определенного порта. Объект класса ServerSocket создается с указанием конструктору номера порта и ожидает сообщения клиента с помощью метода accept()класса ServerSocket, который возвращает сокет клиента:

```
ServerSocket server = new ServerSocket(8030);

Socket socket = server.accept();
```

Таким образом, для установки необходимо установить IP-адрес и номер порта сервера, IP-адрес и номер порта клиента. Обычно порт клиента и сервера устанавливаются одинаковыми. Клиент и сервер после установления сокетного соединения могут получать данные из потока ввода и записывать данные в поток вывода с помощью методов getInputStream() и getOutputStream() или к PrintStream для того, чтобы программа могла трактовать поток как выходные файлы.

В следующем примере для отправки клиенту строки "привет!" сервер вызывает метод getOutputStream() класса Socket. Клиент получает данные от сервера с помощью метода getInputStream(). Для разъединения клиента и сервера после завершения работы сокет закрывается с помощью метода close() класса Socket. В данном примере сервер отправляет клиенту строку "привет!", после чего разрывает связь.

/* пример # 5 : передача клиенту строки : MyServerSocket.java */

```
package chapt15;
import java.io.*;
import java.net.*;
public class MyServerSocket {
public static void main(String[] args) {
Socket s = null;
try { // отправка строки клиенту
//здание объекта и назначение номера порта
ServerSocket server = new ServerSocket(8030);
s = server.accept();//ожидание соединения
PrintStream ps =
new PrintStream(s.getOutputStream());
// помещение строки "привет!" в буфер
ps.println("привет!");
// отправка содержимого буфера клиенту и его очищение
ps.flush();
p.close();
} catch (IOException e) {
System.err.println("Ошибка: " + e);
 finally {
if (s != null)
s.close(); // разрыв соединения

}
```

4.3. Задание для лабораторной работы.

Напишите программу Java, которая позволяет организовать символьный обмен даными между двумя различными компьютерами в сети.
Запишите текст класса и основного метода, чтобы создать объект класса

4.4 Содержание отчета
1. Титульный лист
2. Задание на лабораторную работу
3. Листинг программы
4. Скриншот работы приложения

Лабораторная работа №5

Передача информации с использованием протокола tcp

5.1. Цель работы: изучить основы передачи данных по протоколу udp/ip, позволяющие решать задачи сетевого обмена

5.2. Краткие теоретические сведения

UDP (User Datagram Protocol) не устанавливает виртуального соединения и не гарантирует доставку данных. Отправитель просто посылает пакеты по указанному адресу; если отосланная информация была повреждена или вообще не дошла, отправитель об этом даже не узнает. Однако достоинством UDP является высокая скорость передачи данных. Данный протокол часто используется при трансляции аудио- и видеосигналов, где потеря небольшого количества данных не может привести к серьезным искажениям всей информации.

По протоколу UDP данные передаются пакетами. Пакетом в этом случае UDP является объект класса DatagramPacket. Этот класс содержит в себе передаваемые данные, представленные в виде массива байт. Конструкторы класса:

DatagramPacket(byte[] buf, int length)

DatagramPacket(byte[] buf, int length,

InetAddress address, int port)

DatagramPacket(byte[] buf, int offset, int length)

DatagramPacket(byte[] buf, int offset, int length,

InetAddress address, int port)

DatagramPacket(byte[] buf, int offset, int length,

SocketAddress address)

DatagramPacket(byte[] buf, int length,

SocketAddress address)

Первый конструктор используется в тех случаях когда датаграмма только принимает в себя пришедшие данные, так как созданный с его

помощью объект не имеет информации об адресе и порте получателя. Остальные конструкторы используются для отправки датаграм.

Класс DatagramSocket может выступать в роли клиента и сервера, то есть он способен получать и отправлять пакеты. Отправить пакет можно с помощью метода send(DatagramPacket pac), для получения пакета используется метод receive(DatagramPacket pac).

```
/* пример # 9 : отправка файла по UDP протоколу : Sender.java */
package chapt15;
import java.io.*;
import java.net.*;
public class Sender {
public static void main(String[] args) {
try {
byte[] data = new byte[1000];
DatagramSocket s = new DatagramSocket();
InetAddress addr =
InetAddress.getLocalHost();
/*файл с именем toxic.mp3 должен лежать в корне проекта*/
FileInputStream fr =
new FileInputStream(
new File("toxic.mp3"));
DatagramPacket pac;
while (fr.read(data) != -1) {
//создание пакета данных
pac = new DatagramPacket(data, data.length, addr, 8033);
s.send(pac);//отправление пакета
}
fr.close();
System.out.println("Файл отправлен");
} catch (UnknownHostException e) {
// неверный адрес получателя
e.printStackTrace();
}catch (SocketException e) {
// возникли ошибки при передаче данных
e.printStackTrace();
} catch (FileNotFoundException e) {
// не найден отправляемый файл
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
```

5.3. Задание для лабораторной работы.

Напишите программу Java, которая позволяет организовать символьный обмен даными между двумя различными компьютерами в сети.
Запишите текст класса и основного метода, чтобы создать объект класса

5.4 Содержание отчета
1. Титульный лист
2. Задание на лабораторную работу
3. Листинг программы
4. Скриншот работы приложения

Лабораторная работа №6

Клиент-серверные взаимодействия и протокол HTML

5.1. Цель работы: изучить основы передачи данных по протоколу HTML, позволяющие решать задачи сетевого обмена

Документ, написанный на языке HTML, представляет собой текст, в который вписаны **теги** (markup tags) или теги разметки. Теги помогают программе просмотра разобраться, как должен быть расположен текст на экране, в каком месте будут находиться рисунки, хранящиеся в совсем других файлах, и т.д. С помощью тегов формируются связи с другими web-документами и ресурсами Интернета. Текст с тегами называется **исходным кодом** (Source). Просмотреть готовый файл, написанный на HTML, можно в программах просмотра (MicrosoftInternet Explorer). На экране просмотра теги не отображаются.

Теги – определенные последовательности символов, заключенные между знаками

< (меньше) и > (больше). Символ < обозначает начало тега, символ > обозначает конец тега. Все, что заключено между тегами, является HTML-документом.

Любой HTML-документ состоит их двух частей. Первая часть – заголовок, который находится между тегами < HEAD > и < / HEAD >. В нем содержится информация о документе, которая не выводится на экран. Название странички располагается между тегами < TITLE > и < / TITLE > и появляется в верхней части окна программы просмотра. Например, < TITLE > моя страничка < / TITLE >

Прописные или строчные буквы, используются в написание тегов, значения не имеет. Компьютер одинаково отреагирует на записи <title > и < TITLE >.

Вторая часть – тело, которое выводится на экран программой просмотра – текст, картинки, видеофрагменты и т.д. Оно заключается между тегами < BODY > и < / BODY>. Форматирующие теги бывают парными и непарными, открывающими и закрывающими. Область действия парного тега начинается с того места, где стоит открывающий тег, а кончается там, где встречается закрывающий. Признак закрывающего тега – символ /. Почти все форматирующие теги всегда следуют парами, и для открывающего тега должен существовать закрывающий.

Непарных тегов мало, например, для перехода на новую строку используется тег
 (Line Break).

Задание. Наберите в текстовом редакторе Блокнот.

```
<html >
 < head>
<title >Моя страничка </title >
</head >
< body> Привет! Это моя личная страничка!
</body >
</html>
```

Сохраните созданный файл в вашей папке. Для этого выполните **Файл\Сохранить как**, затем откройте свою папку. В поле **Имя файла**наберите 1.html, а в поле **Тип файла** из раскрывающегося списка выберите **Все файлы.** Нажмите кнопку **Сохранить.**

Закройте Блокнот. Откройте свою папку. Найдите документ в формате html, он имеет такую же пиктограмму, как и документы из Интернета. Например,

Откройте документ и посмотрите результат.

Чтобы разбить текст на параграфы поставьте тег перед началом параграфа. Когда программа обнаружит этот тег, она сама вставит перед началом параграфа пустую строку.

 текст

Вместе с тегом параграфа можно задать параметры выравнивания (align):

right – по правому краю.

left – по левому краю.

center – по центру.

Тег обеспечивает выравнивание теста параграфа по правому краю.

Тег обеспечивает выравнивание теста параграфа по левому краю.

Тег обеспечивает выравнивание теста параграфа по центру.

Еще один способ выравнивания текста

<right > текст </ right >

< left> текст </ left >

< center > текст </ center >

Задание. Создайте в вашей папке файл с именем ind5.htm следующего содержания.

<HTML>
<HEAD>
<TITLE> Термины </TITLE>
</HEAD>
<BODY>
<dl>
<dt> Web server
<dd>Web-сервер. Сервер, хранящий и пересылающий HTML-документы и другие информационные ресурсы Интернет с использованием протокола HTTP. Его называют так же HTTP-сервером. </P>
<dt> HOME PAGE
<dd> «Домашняя страница». Головная, начальная страница, локальный архив. Первая страница какого-либо Web-сервера или логически связанной группы HTML документов. </P>
</dl>
</BODY>
</HTML>

Откройте и посмотрите получившуюся страницу.

6.3. Задание на лабораторную работу. Создайте свой собственный HTML-документ, в котором должны присутствовать:

Название странички.

Заголовки нескольких уровней, выровненные по центру, левому или правому полю.

Параграфы с текстом.

Фрагменты выделенного текста (с помощью полужирного шрифта, курсива, размера шрифта, горизонтальной линии).

Списки.

Сохраните этот файл в вашей папке под именем index.htm и просмотрите его с помощью программы просмотра.

6.4 Содержание отчета

1. Титульный лист
2. Задание на лабораторную работу

3. Листинг программы
4. Скриншот работы приложения

Контрольные вопросы.
Что такое тег?
Что такое исходный код?
Какой символ обозначает начало тега? Какой символ обозначает конец тега?
Что такое парные теги? Приведите пример.
Из каких частей состоит HTML документ?
Как выделяется заголовок документа?
Как выделяется тело документа?
Какие теги позволяет создать параграф (новый абзац)?
Какие теги позволяют создавать маркированные и нумерованные списки?