РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

А. В. Королькова, Д. С. Кулябов

Прикладные протоколы Интернет и www

курс лекций

Москва
Российский университета дружбы народов
2012

УДК 004.72+004.057.4
ББК 32.97
К 68

Утверждено
РИС Учёного совета
Российского университета
дружбы народов

Рецензент —
начальник сектора телекоммуникаций УИТО РУДН
кандидат физико-математических наук, доцент *К. П. Ловецкий*

Королькова, А. В.

К 68 Прикладные протоколы Интернет и www [Текст] : курс лекций / А. В. Королькова, Д. С. Кулябов. — М. : РУДН, 2012. — 146 с. : ил.

Учебное пособие предназначено для студентов направлений «Прикладная математика и информатика», «Математика и компьютерные науки», «Фундаментальная информатика и информационные технологии», «Бизнес-информатика».

ISBN 978-5-209-04950-0

Оглавление

Глава 1. Интернет. Введение в понятие протоколов прикладного уровня

1.1. Базовые понятия

Определение 1.1. *Интернет (Internet)* — глобальная телекоммуникационная сеть информационных и вычислительных ресурсов.

Определение 1.2. *World Wide Web (WWW)* — глобальное информационное пространство, основанное на физической инфраструктуре сети Интернет и протоколе передачи данных HTTP (HyperText Transfer Protocol).

WWW-страницы обычно содержат гиперссылки к другим страницам или мультимедиа файлам.

Определение 1.3. *Веб-браузер* — специальная программа, основная функция которой — отображение гипертекста, являющегося основным методом представления информации в веб.

Определение 1.4. *Гипертекст* — принцип организации текстовых массивов, при котором отдельные информационные элементы связаны между собой ассоциативными отношениями (гиперссылками), обеспечивающими быстрый и удобный поиск необходимой информации и/или просмотр взаимозависимых данных.

Определение 1.5. *Язык HTML (Hypertext Markup Language)* — стандартный язык разметки документов в Интернете, при помощи которого создаются все веб-страницы.

1.2. Предпосылки возникновения Интернета

Историю возникновения Интернета можно отсчитывать с конца 60-х г. XX в. К предпосылкам возникновения Интернет можно отнести следующие моменты:
- **1949 г.** успешное испытание в СССР ядерной бомбы;
- **1952 г.** успешное испытание в СССР водородной бомбы;
- **1957 г.** в СССР был выведен на орбиту первый искусственный спутник Земли;
- **1958 г.** руководством США принято решение о создании глобальной системы раннего оповещения о пусках ракет (сеть наземных станций на вероятных маршрутах подлёта ракет) — началась разработка системы NORAD (North American Aerospace Defence Command); две

проблемы NORAD: обработка результатов наблюдения воздушных объектов и зависимость размещения NORAD от маршрутов подлёта ракет (были протянуты компьютерные и другие линии связи, соединившие центр управления NORAD со станциями слежения, рабочими постами и правительственными органами);
- **середина 1960-х гг.** подключение к NORAD службы управления, затем гражданских авиационных служб авиаполётами — по сути глобальная компьютерная сеть Министерства обороны;
- **1962 г.** публикация работы Дж. Ликлайдера (J.C.R. Licklider) «Galactic Network» о потенциальной возможности создания глобальной компьютерной сети для мгновенного доступа к программам и базам данных из любой точки земного шара;
- **1964 г.** в свет выходит книга Л. Клейнрока (Leonard Kleinrock) по теории коммутации пакетов для передачи данных;
- **1965 г.** первая организация межкомпьютерного взаимодействия (TX-2 в Массачусетсе с ЭВМ Q-32 в Калифорнии) по низкоскоростной коммутируемой телефонной линии;
- **вторая половина 1960-х гг.** проблема разработки такой архитектуры глобальной сети, которая не выходила бы из строя даже в случае поражения одного или нескольких узлов — в рамках Министерства обороны США организовано Агентство передовых исследовательских проектов (Advanced Research Projects Agency, ARPA); начало разработки концепции сети ARPANET на базе университетов США;
- **1967 г.** цели ARPANET:
  - организация сетевого взаимодействия исследовательских учреждений;
  - проведение экспериментов в области компьютерных коммуникаций;
  - изучение способов поддержания связи в условиях ядерного нападения;
  - разработка концепции децентрализованного управления военными и гражданскими объектами в период ведения войн;
- **1968–1969 гг.**
  - реализация компанией Bolt Beranek and Newman (BBN) проекта по созданию первого коммутатора пакетов — интерфейсного процессора сообщений (Interface Message Processor, IMP);
  - проработка архитектуры ARPANET (Боб Кан);
  - проектирование и оптимизация топологии сети (группа из Network Analysis Corporation);
  - разработка системы измерения характеристик сети (группа Л. Клейнрока из Калифорнийского университета Лос-Анджелеса, UCLA);
- **1969 г.** подключение в одну компьютерную сеть исследовательских центров — University of California Los Angeles (UCLA), Stanford

Research Institute (SRI), University of California at Santa Barbara (UCSB), University of Utah;
- **1970 г.** первая версия протокола управления сетью (Network Control Protocol, NCP);
- **1972 г.**
  - первое пользовательское приложение — электронная почта (Рэй Томлинсон (Ray Tomlinson) из BBN);
  - проект «Internetting» — разработка концепции открытой сетевой архитектуры (для организации взаимодействия сетей произвольной архитектуры) с требованиями:
    - универсальность — объединяемые сети не должны зависеть от собственного внутреннего устройства и типа аппаратного и программного обеспечения;
    - связь должна быть максимально надёжной даже при заведомо низком качестве линий связи и оборудования;
    - обеспечение передачи больших объёмов информации;
- **1974 г.** разработка универсального протокола передачи данных и объединения сетей Transmission Control Protocol / Internet Protocol (TCP/IP);
- **1976 г.** разработка протокола UUCP для связи компьютеров по модемному соединению и построения на этом принципе сети Dial-Up Network of Unix Systems;
- **1979 г.** проект по созданию научно-исследовательской компьютерной сети — CSNET (Computer Science research NETwork);
- **1980 г.** стандартизация TCP/IP и разработка концепции объединения CSNET и ARPANET;
- **1983 г.** внедрение протокола TCP/IP; разделение сети на MILNET — для военных целей, и ARPANET — для научных исследований;
- **1985–1990 гг.** разработка межрегиональной сети NSFNET;
- **1989–1991 гг.** разработка концепции гипертекстовых документов — World Wide Web (Тим Бернерс-Ли (Tim Berners-Lee));
- **1993 г.** появление службы World Wide Web (WWW), основанной на пользовательском протоколе передачи данных HTTP и на особом формате представления данных — HTML; первый графический интерфейс Mosaic, обеспечивающий отправку запросов и приём сообщений в формате HTML;
- **1994 г.** соглашение об основании WWW-консорциума (World Wide Web consortium) с целью дальнейшего развития приложения веб, стандартизации протоколов и взаимодействия между отдельными сайтами;
- **4 октября 1995 г.** Федеральный сетевой совет (FNC) единодушно одобрил резолюцию, определяющую термин «Интернет».

После этого работа в Интернете перестала быть доступна только профессионалам. Интернет превратился в распределённую по миллионам серверов единую базу данных.

Основные вехи развития современных гипертекстовых технологий:

- **1945 г.** Ванневер Буш (Vannevar Bush) создал первую фотоэлектрическую память и приспособление Memex (memory extension), которое представляет собой справочник, реализованный с помощью гиперссылок в пределах документа;
- **1965 г.** Тед Нельсон (Ted Nelson) ввёл термин «гипертекст» и создал гипертекстовую систему Xanadu с двусторонними гиперсвязями;
- **1980 г.** Тим Бернерс-Ли (T. Berners-Lee), консультант CERN (Европейская организация ядерных исследований), написал программу, которая разрешает создавать и пересматривать гипертекст, реализующий двунаправленные связи между документами из коллекции;
- **1990 г.** для поддержки документации, которая циркулирует в CERN Бернерс-Ли начал работу над графическим интерфейсом пользователя (GUI, Graphical User Interface) для гипертекста. Эта программа была названа «WorldWideWeb». До 1992 г. уже были созданы такие GUI, как Erwise и Viola;
- **1993 г.** М. Андрессен (M. Anderssen) из NCSA (Национальный Центр Суперкомпьютерных приложений США, www.ncsa.uiuc.edu) закончил первую версию программы визуализации гипертекста Mosaic для популярного графического интерфейса Xwindow System под UNIX. Одновременно CERN развивал и улучшал HTML — язык гипертекстовой разметки текстов и HTTP — протокол передачи гипертекста, а также сервер обработки гипертекстовых документов — CERN HTTPD.

1.3. Структура и схема административного устройства Интернета

Сеть Интернет не имеет единого, централизованного управления. Можно говорить лишь об элементах управления сетью Интернет.

- *Политика приемлемого использования (Accepted Use Policy, AUP)* Правила и ограничения, накладываемые на использование сетей, входящих в Интернет, являющиеся неотъемлемой частью политики информационной безопасности. Документы AUP призваны уменьшить вероятность судебных исков от пользователей в адрес разместивших информацию владельцев веб-ресурсов.
- *Интернет-сообщество (Internet Society (ISOC))* Международная профессиональная некоммерческая организация, основной задачей которой является развитие и разработка интернет-технологий, обеспечение доступности ресурсов Интернета в мировом масштабе. ISOC официально владеет правами на все документы RFC (Request for Comments) — технические спецификации и стандарты, применяемые в Интернете.

– *Технические комитеты и комиссии*
  Комитеты поддерживают системы стандартов, на которых базируется вся сеть:
  – *Internet Architecture Board (IAB)* — комиссия по архитектуре сети Интернет, основная задача которой — разработка и оформление стандартов взаимодействия готовых информационных систем;
  – *Internet Engineering Task Force (IETF)* — открытое международное сообщество проектировщиков, учёных, сетевых операторов и провайдеров, которое непосредственно отвечает за разработку протоколов и архитектуры Интернета;
  – *Internet Research Task Force (IRTF)* — исследовательское подразделение по развитию долгосрочных перспективных интернет-технологий;
  – *Internet Assigned Numbers Authority (IANA)* — ведёт реестр всех идентификаторов, связанных с протоколами Интернета, поддерживает хранилище документов, управляет распределением IP-адресов, доменов верхнего уровня;
  – *Internet Computer Emergency Response Team (CERT)* — занимается вопросами безопасности сети Интернет;
  – *Reseaux IP Euopeens (RIPE)* — координация развития сетей в Европе, занимается распределением IP-адресов, развивает технические виды сервиса по маршрутизации и системы доменных имён;
  – *InterNIC* — центр сетевой информации, контролирует ресурсы Интернета (IP-адреса, доменные имена, справочные службы и хранилища документов);
  – *Информационный центр MERIT* — центр специализированной информации по маршрутизации, оптимизации адресного пространства и пр.

В России создано отделение ISOC — РАЙНЕТ. В рамках Ассоциации документальной электросвязи в России интернет-комитет будет выполнять функции по сертификации интернет-операторов. Функции по администрированию домена RU, присвоению IP-адресов, поддержанию хранилища документов выполняет Российский НИИ развития общественных сетей (РосНИИРОС).

1.4. Документация, регламентирующая развитие и эксплуатацию сети Интернет

Разработка стандартов Интернета происходит на открытых сессиях Комитета по технологической поддержке сети Интернет (Internet Engineering Task Force, IETF). Разработанные таким образом протоколы публикуются в виде документов *RFC (Request for Comments, запрос комментариев)*. Документы RFC содержат широкий спектр

интересных и полезных сведений и не ограничиваются формальными спецификациями протоколов обмена данными. Существует три основных типа документов RFC: стандарты (standards, STD), применяющиеся на практике документы (Best Current Practices, BCP), уведомляющие документы (For Your Information, FYI).

Документы RFC, определяющие официальные стандарты протоколов, обозначаются аббревиатурой STD и получают STD-номера в дополнение к RFC-номерам. Создание официального стандарта Интернета — строго последовательный процесс:

- *Заявка на стандарт (Proposed Standard)*
  Спецификация протокола, который является важным и уже получил достаточно широкую поддержку интернет-сообщества, чтобы предлагаться в качестве стандарта, но может и не быть стандартизованным в дальнейшем.
- *Проект стандарта (Draft Standard)*
  Спецификация протокола, для которой существует по меньшей мере две независимых, взаимозаменяемых реализации. Проект стандарта — это окончательная спецификация, используемая в широком тестировании.
  Проект изменяется лишь в том случае, когда этого требуют результаты тестирования.
- *Стандарт Интернета (Internet Standard)*
  Спецификация получает статус стандарта лишь после всеобъемлющего тестирования и лишь в том случае, когда применение определённого этой спецификацией протокола может принести значительную выгоду интернет-сообществу.
  Стандарты делятся на две категории.
  - техническая спецификация (Technical Specification, TS) даёт определение протокола;
  - формулировка применимости (Applicability Statement, AS) описывает случаи, когда протокол следует применять. Применимость стандарта имеет следующие уровни:
    - *обязательный (Required)*
      Стандартный протокол, является обязательной частью любой реализации TCP/IP. Стек протоколов должен включать этот протокол, чтобы соответствовать стандарту.
    - *рекомендованный (Recommended)*
      Стандартный протокол, рекомендованный к включению во все реализации TCP/IP. Его присутствие не является обязательным условием.
    - *факультативный (Elective)*
      Решение по реализации принимает разработчик конкретного пакета приложений.
    - *ограниченного применения и не рекомендован*
      Связаны с документами RFC, существующими обособленно от

процесса стандартизации. Протокол ограниченного применения используется только в особых случаях, например, в ходе экспериментов. Протоколы, не рекомендованные к применению, являются устаревшими либо имеют ограниченную функциональность.

Документы RFC, не принадлежащие процессу стандартизации, бывают трёх типов:

- *экспериментальные (Experimental)* — применение ограничено исследованиями и разработкой;
- *исторические (Historic)* — являются устаревшими, их применение не рекомендуется;
- *уведомляющие (Informational)* — содержат информацию, представляющую интерес для широких слоев сообщества сети Интернет, но не содержат определений протоколов.

Документы FYI (For Your Information) являются уведомляющими RFC и помимо номера RFC имеют номер FYI. Эти документы содержат вводный и подготовительный материал по сети Интернет и сетям TCP/IP в целом.

Документы BCP (Best Current Practices) предназначены для формального документирования различных методов и процедур, например, описывают принципы, которым подчиняются действия самой организации IETF, или содержат рекомендации по функционированию сетей или служб.

1.5. Адресация в Интернете: URI, URL, URN, IRI

Интернет имеет три элемента: форматы данных, протоколы и указатели, связывающие первые два элемента.

URI (Uniform Resource Identifier) — унифицированный идентификатор ресурса (документ, изображение, файл, служба, ящик электронной почты и т.д.)

Форма записи URI:

`<схема>:<идентификатор>`

- схема — схема (протокол) обращения к ресурсу (например, http, ftp, mailto, urn);
- идентификатор — идентификатор ресурса, вид которого зависит от выбранной схемы обращения к ресурсу.

URI-путь представляет собой путь доступа к файлу. URI унаследовали левую косую черту (a/b/c) из традиций UNIX. Синтаксис URI использует двойную левую косую черту для перекрёстного обращения к хостам. Помимо этого, используется синтаксис схем для унификации соглашения о присвоении имён из различных протоколов. Если URI-ссылка начинается **со схемы** и **двоеточия**, то она является *абсолютной*, в противном случае — *относительной*.

Примеры URI:
- `mailto:mbox@domain`
- `ftp://host/file`
- `ftp://loginname:pass@host:port/path`
- `http://domain/path`
- `telnet://domain:port`

URL (Uniform Resource Locator) — унифицированный способ записи адреса ресурса в сети Интернет.

URL является частью концепции URI, чётко определяющий местонахождение ресурса.

Форма записи URL:

`<схема>://<логин>:<пароль>@<хост>:<порт>/<URL-путь>`

- схема — схема (протокол) обращения к ресурсу;
- логин — имя пользователя, используемое для доступа к ресурсу;
- пароль — пароль, ассоциированный с указанным именем пользователя;
- хост — полностью прописанное доменное имя хоста в системе DNS или IP-адрес хоста;
- порт — порт хоста для подключения;
- URL-путь — уточняющая информация о месте нахождения ресурса (зависит от протокола).

Некоторые схемы (протоколы) URL:
- ftp — протокол передачи файлов FTP;
- http — протокол передачи гипертекста HTTP;
- https — специальная реализация протокола HTTP, использующая шифрование (как правило, SSL или TLS);
- mailto — адрес электронной почты;
- news — новости Usenet;
- nntp — новости Usenet через протокол NNTP;
- telnet — ссылка на интерактивную сессию Telnet;
- file — имя локального файла;
- data — непосредственные данные (Data: URL).

URN (Uniform Resource Name) — унифицированный способ задания названия (имени) ресурса.

URN является частью концепции URI и, в отличие от URL, не включает в себя указания на местонахождение и способ обращения к ресурсу. Концепция URN призвана решить проблему изменения местонахождения ресурса в сети.

URN концептуально *обозначает сам ресурс*, а не место, где он находится. Для нахождения ресурсов по URN-имени нужна **система разрешения URN-имён** (URN resolution): при обращении к URN-ресурсу система должна выдать список конкретных мест (серверов), где этот ресурс может находиться.

В 2002 г. была предложена **система динамического обнаружения ресурсов (DDDS, Dynamic Delegation Discovery System)**,

которая разрешает имена URN в URL-ссылки на конкретные местонахождения ресурсов.

Структура URN:

`<URN> ::= urn: <NID> : <NSS>`

- NID — **идентификатор пространства имён (Namespace Identifier)**, представляющий собой синтаксическую интерпретацию NSS (не чувствителен к регистру);
- NSS — строка из определённого **пространства имён (Namespace Specific String)**.

URN книги, идентифицируемой номером ISBN:

`urn:isbn:5170224575`

URN технической спецификации RFC 3406:

`urn:ietf:rfc:3406`

URN конкретного файла MP3, идентифицируемого хэш-кодом по алгоритму SHA1:

`urn:sha1:YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C`

Здесь isbn, ietf, sha1 — пространства имён (NID), а строки за вторым двоеточием — NSS.

Именование адресов с помощью URN не получило распространения в сети, зато подобный способ именования прижился в разнообразных одноранговых сетях (p2p-сетях). Поскольку в них местонахождение ресурса не определено, его нужно идентифицировать каким-то другим параметром. Так как имена не обеспечивают уникальности, то обычно используется значение хэш-функции от содержимого ресурса.

URI и URI-ссылки создаются из ограниченного набора символов ASCII.

При использовании символов не ASCII кода они должны быть перекодированы. Преобразование происходит в два этапа: сначала каждый символ кириллицы кодируется в Юникод (UTF-8), а затем каждый байт этой последовательности записывается в шестнадцатеричном представлении. Перед каждым шестнадцатеричным кодом байта, согласно RFC 2396, ставится знак процента (%)[1].

Поскольку такому преобразованию подвергаются буквы всех алфавитов, кроме базовой латиницы, то URL со словами на подавляющем большинстве языков может утратить способность восприниматься людьми. Эту проблему должны были снять IRI.

IRI (Internationalized Resource Identifier) — интернационализированный идентификатор ресурса.

В IRI может использоваться весь набор символов Unicode. Для каждого IRI существует соответствующая кодировка в формате URI на тот случай, если идентификатор нужно будет использовать в протоколе (например, HTTP), который может работать только с URI.

[1] Отсюда возник английский термин percent-encoding, обозначающий способ кодирования символов в URI.

Глава 2. Представление информации в Интернете. Методология поиска информации. Поисковые машины

2.1. Информационно-поисковые системы

Определение 2.1. *Информационно-поисковая система (Information Retrieval System, IRS)* — система, предназначенная для обеспечения поиска и отображения документов, представленных в базах данных.

Ядром информационно-поисковой системы является *поисковый механизм* — программный модуль, который осуществляет поиск по запросу.

Компоненты поисковых систем:
- веб-сервер;
- поисковый робот (spider/crawler, паук);
- индексатор;
- база данных;
- система выдачи результатов.

Определение 2.2. *Веб-сервер* — сервер поисковой машины, который осуществляет взаимодействие между пользователем и остальными компонентами системы.

Определение 2.3. *Индексатор* — программа-анализатор веб-страниц; анализу подвергаются текст, служебные html-теги, заголовки, особенности стилистики и структурные формы.

Определение 2.4. *База данных* — хранилище для скачанных и обработанных страниц — общая база данных поисковой машины.

Определение 2.5. *Система выдачи результатов* — извлекает результаты поиска из базы данных поисковой системы.

Определение 2.6. *Поисковый робот* — программа, являющаяся составной частью поисковой системы и предназначенная для перебора страниц Интернета с целью занесения информации о них в базу данных поисковика.

При этом:
- поисковый робот анализирует содержимое страницы, сохраняет его в некотором специальном виде на сервере поисковой машины, которой принадлежит, и отправляется по ссылкам на следующие страницы;

– порядок обхода страниц, частота визитов, защита от зацикливания, а также критерии выделения значимой информации определяются поисковыми алгоритмами;
– ограничить индексацию сайта можно с помощью файла robots.txt.

Основные функции информационно-поисковых систем:

– хранение больших объёмов информации;
– быстрый поиск необходимой информации;
– добавление, удаление и изменение хранимой информации;
– вывод информации в удобном для пользователя виде.

Основные этапы развития информационно-поисковых систем:

– **1966 г.** реализация проекта MARC для установления стандартного формата электронных каталогов обеспечила переход к унифицированному обмену электронными данными;
– **1972 г.** получил международное признание стандарт MARC-2, на основе которого были созданы многие национальные стандарты;
– **1970-е гг.** появилась возможность интерактивного поиска на коммерческой основе в тематических базах данных Национальной медицинской библиотеки и Министерства образования США;
– **начало 1990-х гг.** для унификации информационных систем разработан международный стандарт Z39.50 — информационно-поисковый протокол для библиографических систем;
– **1994 г.** университет Джорджии запустил пилотный проект «Галилей» (http://www.usg.edu/galileo/) с использованием Site-Search — пакета программ Огайоского центра, соответствующего стандарту Z39.50. Стандарт Z39.50 также был положен в основу исторически первой службы поиска распределённой информации в Интернете — WAIS (Wide Area Information Service), в настоящее время утратившей свою актуальность.

Основные характеристики поисковых запросов

Определение 2.7. *Релевантность* — степень соответствия найденной в поисковой системе информации запросу пользователя.

Определение 2.8. *Ранжирование* — построение поисковой машиной списка релевантных запросу сайтов, расположенных в соответствии с весом и ценностью запрашиваемой информации.

Определение 2.9. *Степень индексации сайта в поисковых системах* — показатель соотношения числа веб-страниц, проиндексированных конкретной поисковой системой, к действительному числу веб-страниц сайта.

2.2. Методология поиска информации

Определение 2.10. *Терм* — ключевое слово, устойчивое словосочетание.

Основные обозначения:

- i — индекс терма t_i из словаря T $(i = 1, \ldots, M)$;
- $d^{(j)}$ — документ, принадлежащий множеству документов D;
- $w_i^{(j)}$ — вес, ассоциированный с парой $(t_i, d^{(j)})$;
- g_i — инверсная функция: $g_i(d^{(j)}) = w_i^{(j)}$.

  $\forall t_i \notin d^{(j)}\ w_i^{(j)} = \emptyset$.

  Документ $d^{(j)}$ рассматривается как вектор весов: $d^{(j)} = (w_1^{(j)}, \ldots, w_M^{(j)})$.

  Традиционные методы (модели) поиска:
- булева модель;
- векторно-пространственная модель;
- вероятностная модель.

Булева модель:

- документы и запросы представляются в виде множества *термов* — ключевых слов и устойчивых словосочетаний;
- каждый терм представлен как булева переменная:
  - 0 (терм из запроса не присутствует в документе);
  - 1 (терм из запроса присутствует в документе);
- вес: $w_i^{(j)} \in \{0, 1\}$;
- *запрос пользователя* q — логическое выражение, в котором термы связываются логическими операторами конъюнкции (AND, $\wedge$), дизъюнкции (OR, $\vee$) и отрицания (NOT, $\neg$);
- $q \equiv q_{dnf} = \bigwedge_{i=1}^{N} q_{cc}^{(i)}$, где $q_{cc}^{(i)}$ — i-я конъюнктивная компонента формы запроса q_{dnf} (*ДНФ (дизъюнктивная нормальная форма)* — нормальная форма в булевой логике, в которой булева формула имеет вид дизъюнкции нескольких конъюнктивных компонент);
- мера близости документа $d^{(j)}$ и запроса q:

$$sim(d^{(j)}, q) = \begin{cases} 1, & \text{если } \exists q_{cc}^{(i)} : \left(q_{cc}^{(i)} \in d_{dnf}\right) \wedge \\ & \quad \wedge \left(\forall k,\ g_k\left(q_{cc}^{(i)}\right) = g_k(d^{(j)})\right), \\ 0, & \text{иначе} \end{cases}$$

- если $sim(d^{(j)}, q) = 1$, то документ $d^{(j)}$ считается *релевантным* (соответствующим) запросу q.

Архитектура поисковых систем, соответствующих булевой модели, состоит из следующих **таблиц**:

- *текстовой*, содержащей текстовую часть всех документов;
- *указателей текстов*, которая включает указатели на местонахождение документов в текстовой таблице;
- *словарной*, содержащей все уникальные слова, встречающиеся в документах, то есть те слова, по которым может осуществляться поиск;
- *инверсной*, содержащей списки номеров документов и координаты отдельных слов в документах.

Алгоритм поиска для запроса из одного слова

1. Обращение к словарной таблице, по которой определяется, входит ли слово в состав словаря базы данных, и если входит, то определяется ссылка в инверсной таблице на цепочку появлений этого слова в документах.
2. Обращение к инверсной таблице, по которой определяются номера документов, содержащих данное слово, и координаты всех вхождений слова в текстах базы данных.
3. По номеру документа происходит обращение к записи таблицы указателей текстов. Каждая запись этого файла соответствует одному документу в базе данных.
4. По номеру документа происходит прямое обращение к фрагменту текстовой таблицы — документу, после чего следует вывод найденного документа.

Если в запрос входит некоторая комбинация слов, то в результате выполнения поиска по каждому из слов запроса формируется массив записей, которые соответствуют вхождению слова в базу данных. После окончания формирования массивов результатов поиска происходит выявление релевантных документов путём выполнения теоретико-множественных операций над записями этих массивов в соответствии с правилами булевой логики.

Векторно-пространственная модель:

- каждому терму t_i в документе $d^{(j)}$ соответствует некоторый неотрицательный вес $w_i^{(j)}$;
- каждому запросу q_i, представляющему собой множество термов, не соединённых между собой никакими логическими операторами, соответствует вектор весовых значений w_i^q;
- близость документа $d^{(j)}$ к запросу q оценивается как скалярное произведение $\vec{d_j} = \left(w_1^{(j)}, \ldots, w_n^{(j)}\right)$ и $\vec{q} = (w_1^q, \ldots, w_n^q)$;
- правило вычисления веса: $w_i^{(j)} = tf_i^{(j)} \log \frac{N}{n_i}$, $tf_i^{(j)}$ — частота встречаемости терма в документе, n_i — количество документов, в которых используется терм t_i, N — общее количество документов в массиве;

– определение тематической близости двух документов: $sim(d^{(1)}, d^{(2)})$; или документа и запроса: $sim(d^{(j)}, q)$ (т.е. через скалярное произведение соответствующих векторов).

Векторно-пространственная модель обеспечивает:

– обработку запросов без ограничений их длины;
– простоту реализации режима поиска подобных документов (каждый документ может рассматриваться как запрос);
– сохранение результатов поиска с возможностью выполнения уточняющего поиска.

В векторно-пространственной модели *не предусмотрено* использование логических операций в запросах, что существенно ограничивает её применимость.

Вероятностная модель поиска:

– вероятность того, что документ релевантен запросу, основывается на предположении, что термы запроса по-разному распределены среди релевантных и нерелевантных документов;
– релевантность рассматривается как вероятность того, что документ может оказаться интересным пользователю;
– функционирование модели базируется как на экспертных оценках, получаемых в результате обучения модели, так и на последующих оценках вероятности того, что документ является релевантным запросу исходя из состава его термов;
– модель предусматривает определение вероятностей соответствия запросу для документов, сортировку и предоставление документов с ненулевой вероятностью пользователю.

Общий недостаток классических моделей поиска — предположение, что содержание документа определяется множеством слов и устойчивых словосочетаний — термов, которые входят в документ без учёта взаимосвязей и считаются независимыми.

Это ведёт к потере содержательных оттенков, но позволяет реализовать поиск и группирование документов по формальным признакам.

Основные недостатки традиционных моделей:

– *булева модель* — невысокая эффективность поиска, отсутствие контекстных операторов, невозможность ранжирования результатов поиска;
– *векторно-пространственная модель* связана с расчётом массивов высокой размерности и в каноническом виде малопригодна для обработки больших массивов данных;
– *вероятностная модель* характеризуется низкой вычислительной масштабируемостью (т.е. резким снижением эффективности при росте объёмов данных), необходимостью постоянного обучения системы.

2.3. Алгоритмы поиска в пиринговых сетях

Алгоритм поиска ресурсов по ключам:
- используется два вида сущностей: узлы и ресурсы;
- узлам и ресурсам приписываются соответствующие идентификаторы (ID), характеризующиеся ключами (Key);
- сеть может быть представлена двумерной матрицей размерности $M \times N$, где M — число узлов, N — количество ресурсов;
- задача поиска сводится к нахождению ID узла, на котором хранится ключ ресурса.

Метод широкого первичного поиска (Breadth First Search, BFS):
- узел q генерирует запрос, который адресуется всем соседям (ближайшим по некоторым критериям узлам);
- получив запрос, узел p выполняет поиск в своём локальном индексе;
- если узел p принимает запрос (Query) и обрабатывает его, то он генерирует сообщение-отклик (QueryHit), чтобы возвратить результат;
- сообщение QueryHit включает информацию о релевантных документах, которая доставляется по сети запрашивающему узлу;
- узел q, получив QueryHits от более чем одного узла, может загрузить файл с наиболее доступного ресурса.

Недостаток BFS — каждый запрос вызывает чрезмерную нагрузку сети, так как он передаётся по всем связям (в том числе и узлам с высоким временем ожидания).

Положительная черта BFS — гарантирует высокий уровень качества поиска за счёт большого числа переданных сообщений.

Метод BFS широко используется в реальных файлообменных сетях P2P, таких как, например, Gnutella (`http://www.gnutella.com`).

Метод случайного широкого первичного поиска (Random Breadth First Search, RBFS):
- аналогичен BFS;
- узел q пересылает поисковое предписание только части узлов сети, выбранной в случайном порядке.

Преимущество RBFS — не требуется наличия глобальной информации о состоянии контента сети; узел может получать локальные решения так быстро, как это потребуется.

Недостаток RBFS — метод вероятностный, поэтому некоторые большие сегменты сети могут оказаться недостижимыми.

Интеллектуальный поисковый механизм (Intelligent Search Mechanism, ISM):
- состоит из двух компонент – профайла (profile) и способа его ранжирования, так называемого ранга релевантности;

– каждый узел сети строит информационный профайл для каждого из соседних узлов;
– профайл содержит последние ответы каждого из узлов;
– с помощью ранга релевантности осуществляется ранжирование профайлов узлов для выбора тех соседних, которые будут давать наиболее релевантные документы по запросу;
– механизм профайлов используется для того, чтобы сохранять последние запросы, а также количественные характеристики результатов поиска.

Преимущества ISM:

– улучшение скорости и эффективности поиска информации достигается за счёт минимизации затрат на связи, т.е. на число сообщений, передающихся между узлами, и минимизации количества узлов, которые опрашиваются для каждого поискового запроса;
– эффективно работает в сетях, где узлы содержат некоторые специализированные сведения.

Недостаток ISM — поисковые сообщения могут циклично проходить одни и те же узлы сети, не достигая некоторых её частей.

Метод большинства результатов по прошлой эвристике (>RES):

– подобен методу ISM, но использует упрощённую информацию об узлах;
– каждый узел пересылал запрос подмножеству своих узлов, образованному на основании некоторой обобщённой статистики;
– запрос в методе >RES является удовлетворительным, если выдаётся Z или больше результатов (Z – некоторая постоянная);
– узел q пересылает запросы к n узлам, выдавшим наибольшие результаты для последних m запросов

Преимущество >RES — маршрутизирует запросы в большие сегменты сети (которые возможно, также содержат более релевантные ответы).

Недостаток >RES — по сравнению с ISM отсутствует анализ параметров узлов, содержание которых связано с запросом.

Метод случайных блужданий (Random Walkers, Algorithm, RWA):

– каждый узел случайным образом пересылает сообщение с запросом, именуемое посылкой, одному из соседних узлов;
– для сокращения времени получения результатов вместо одной посылки рассматривается k независимых посылок, последовательно запущенных с поискового узла;
– ожидается, что k-посылка после T шагов достигнет тех же результатов, что и одна посылка за kT шагов;
– напоминает метод RBFS, но в RBFS каждый узел пересылает сообщение запроса части соседей;

– в RBFS предполагается экспоненциальное увеличение пересылаемых сообщений, а в методе случайных блужданий – линейное;
– и RBFS, и RWA не используют никаких явных правил для того, чтобы адресовать поисковый запрос к наиболее релевантному содержанию.

Адаптивный вероятностный поиск (Adaptive Probabilistic Search, APS):
– каждый узел развёртывает на своих ресурсах локальный индекс;
– индекс содержит значения условных вероятностей для каждого соседа, который может быть выбран для обработки следующего запроса;
– узел использует в качестве обратной связи результаты предыдущих поисков (в виде условных вероятностей) вместо полностью случайных переходов в RWA.

2.4. Информационно-поисковые языки

Особенности информационно-поисковых языков:
– являются основными компонентами информационно-поисковых систем, с помощью которых, в частности, реализуются интерфейсы между пользователями и системами;
– не существует стандартизированного языка запросов;
– часто языки запросов информационно-поисковых систем приближены к SQL, но обладают особенностями, связанными со следующими моментами:
  – интерпретация операций, задающих порядок расположения слов в тексте (операций контекстной близости);
  – вычисление уровня релевантности найденных документов запросам для представления результатов поиска;
  – применение нестандартных для реляционных СУБД функций, например, таких как нахождение документов по принципу подобия содержания, построение дайджестов из фрагментов документов, включаемых поисковыми системами в списки найденных документов и т.п.

Особенности информационно-поисковых систем:
– применение различных архитектурных решений, охватывающих структуры данных, алгоритмы их обработки, методы организации поиска;
– обеспечение поиска хотя бы по одному слову;
– реализация грамматического поиска как результата применения лингвистического анализа (например, по терму из запроса «человек» находятся не только «человека», «человеку», но и «люди»);
– для группирования термов и операторов:
  – реализация контекстного поиска фразы, заключённой в кавычки;

  - возможность поиска с использованием булевых операторов AND, OR и NOT;
  - возможность указания скобок;
- реализация функции контекстной близости;
- возможность поиска по параметрам документов, которая чаще всего позволяет ограничивать диапазон поиска значениями URL, дат, заголовков и т.п.;
- обеспечение поиска не только по данным в формате HTML, но и в форматах PDF, RTF, DOC (MsWord), PS;
- реализация адаптивных интерфейсов уточнения запросов, чаще всего реализованных путём применения методов кластерного анализа к результатам первичного поиска;
- реализация метода папок поиска (Custom Search Folders) — группирование результатов поиска и представление группы наиболее связанных документов (кластеров) в удобном для пользователей виде.

2.5. Характеристики информационного поиска

Основные характеристики информационного поиска:
- полнота (recall);
- точность (precision);
- пертинентность (соответствие полученных в результате поиска документов информационным потребностям пользователя, а не формальному соответствию документа запросу).

Для вычисления показателей качества поиска принято рассматривать таблицу, которую заполняют по результатам поиска в коллекции документов:

Документы	Выданные	Невыданные
Релевантные	a	c
Нерелевантные	b	d

Показатели информационного поиска рассчитываются следующим образом.

Коэффициент полноты (recall): $r = \dfrac{a}{a+c}$.

Коэффициент точности (precision): $p = \dfrac{a}{a+b}$.

Коэффициент аккуратности (accuracy): $acc = \dfrac{a+d}{a+b+c+d}$.

Ошибка (error): $err = \dfrac{b+c}{a+b+c+d}$.

F-мера (F-measure): $F = 2\left(\frac{1}{p} + \frac{1}{r}\right)^{-1}$.

Средняя точность (average precision): $ArgPrec = \frac{1}{k}\sum_{i=1}^{k} prec\_rel(i)$, где k — количество документов, релевантных некоторому запросу, i — номер релевантного запросу документа, $prec\_rel(i)$ — точность i-го релевантного документа.

Технологические характеристики поисковых систем:
- скорость обработки запросов;
- полнота охвата ресурсов;
- доступность, т.е. вероятность получения ответа от системы;
- нахождение документов, подобных найденным;
- возможность уточнения запросов;
- возможность подключения переводчиков и т.п.

2.6. Концепция Text Mining

Технологии глубинного анализа текстов (Text Mining):
- разработаны на основе статистического и лингвистического анализа, а также методов искусственного интеллекта;
- предназначены для проведения смыслового анализа;
- задачи:
  - выбрать из текстов наиболее ключевую и значимую информацию для пользователей;
  - отнесение документов к некоторым категориям из заданной схемы их систематизации;
- важная компонента технологий Text Mining связана с извлечением из текста характерных элементов или признаков, которые могут использоваться в качестве ключевых слов, метаданных, аннотаций;
- в отличие от традиционных подходов не только находит списки документов, формально релевантных запросам, но и помогает в понимании смысла текстов.

Text Mining — это алгоритмическое выявление прежде неизвестных связей в уже имеющихся данных.

Основные элементы Text Mining:
- классификация (classification, categorization);
- кластеризация (clustering);
- извлечение фактов, понятий (feature extraction);
- реферирование (summarization);
- ответ на запросы (question answering);
- тематическое индексирование (thematic indexing);
- поиск по ключевым словам (keyword searching).

При *классификации* текстов используются статистические корреляции для размещения документов в определённые категории с целью оптимизации числа объектов и их атрибутов при поиске.

В отличие от классификации, при *кластеризации* заранее не фиксируются определённые категории. Результатом кластеризации является автоматическое группирование информации, в результате которого создаются классификационные схемы, обеспечивающие эффективный охват больших объёмов данных.

Text Mining предусматривает также построение семантических сетей, анализ связей, которые определяются появлением дескрипторов (например, ключевых слов) в текстах.

Задачи Text Mining:

- прогнозирование — предсказывание по значениям одних признаков текста значений остальных;
- нахождение исключений, т.е. поиск документов, которые своими характеристиками выделяются из общей массы;
- поиск связанных признаков (ключевых слов, понятий) отдельных документов.

Извлечение понятий (Feature Extraction) из текста — технология, обеспечивающая получение информации в структурированном виде.

В качестве структур могут запрашиваться как относительно простые понятия (ключевые слова, персоны, организации, географические названия), так и более сложные, например, имя персоны, её должность в конкретной организации и т.п.

Технология извлечения понятий основана на применении специальных семантико-лингвистических методов, которые дают возможность получать приемлемую точность и полноту.

Технология извлечения понятий включает в себя **три основных метода**:

- *Entity Extraction* — извлечение слов или словосочетаний, важных для описания содержания текста (например, списки терминов предметной области, персон, организаций, географических названий и др.);
- *Feature Association Extraction* — прослеживание связей между извлечёнными понятиями;
- *Event and Fact Extraction* — извлечение сущностей, распознавание фактов и событий.

Автоматическое реферирование (Automatic Text Summarization) — это составление коротких изложений материалов, аннотаций или дайджестов, т.е. извлечение наиболее важных сведений из одного или нескольких документов и генерация на их основе лаконичных отчётов.

Реализации систем с элементами Text Mining:

- Intelligent Miner for Text от IBM (`http://www.ibm.com`):

  - Language Identification Tool — утилита определения языка документа;
  - Categorisation Tool — утилита автоматической категоризации текста;
  - Clusterisation Tool — утилита кластеризации — документы разбиваются на группы, характеризуемые близостью стиля, формы, характеристиками ключевых слов;
  - Feature Extraction Tool — утилита определения нового — на базе заранее заданного словаря в документе выявляются новые термы (например, названия, сокращения);
  - Annotation Tool — утилита выявления определённого содержания текстов и составления по ним рефератов-аннотаций.
- PolyAnalyst от компании Мегапьютер Интеллидженс (`http://www.megaputer.com`):
  - может применяться для автоматизированного анализа числовых и текстовых баз данных с целью выявления прежде неизвестных, нетривиальных, полезных и доступных пониманию закономерностей
  - имеет возможность построения семантической сети для больших текстов, может осуществлять подготовку резюме текста, поиск по тексту, автоматическую классификацию и кластеризацию текстов.
- Продукты компании Oracle (`http://www.oracle.com`):
  - осуществляют поиск документов по их содержанию;
  - обеспечивают проведение анализа тематики текстов на английском языке (проводится лингвистический и статистический анализ текста, определяются его ключевые темы, строится тематическое, а также общее резюме-реферат).

2.7. Краткий обзор необычных поисковых машин

Нигма (`http://nigma.ru/`) — кластеризующий поисковик, который позволяет уточнить запрос пользователя, группировать и фильтровать результаты поиска по темам (фильтрует результаты других поисковых систем).

Особенности:

- умные поисковые подсказки — Нигма отвечает на вопрос ещё до того, как пользователь введёт его в строку поиска;
- Нигма-математика — сервис, с помощью которого пользователи могут решать различные математические задачи (упрощать выражения, решать уравнения, системы уравнений и т.д.), вводя их прямо в строку поиска в виде обычного текста;
- Нигма-химия — сервис, с помощью которого пользователи могут решать различные задачи по химии;

– Нигма-музыка — сервис, упрощающий поиск музыкальных композиций;
– поиск по торрентам — ведётся только по бесплатным торрент-трекерам, не требует регистрации и не докучает назойливой рекламой;
– инфопоиск — алгоритм индексации интернет-сайтов, который расщепляет содержимое веб-страниц на информационные блоки; затем блоки, содержащие информацию, связанную с запросом пользователя, подмешиваются в результаты поиска с более высоким приоритетом;
– официальные сайты — повышение приоритета первоисточников информации в поисковой выдаче;
– табличный Нигма-поиск — выдача списков объектов на запросы пользователей в виде таблиц;
– борьба со спамом — сервис, дающий возможность пользователям указывать сайты, содержащие поисковый спам;
– расшифровка сокращений — распознает практически любые русско- и англоязычные аббревиатуры;
– поиск по библиотекам — помогает находить полную информацию по любому автору и тексты всех его произведений как русской, так и зарубежной литературы;
– англоподсказка — сервис, который проверяет совместимость и частоту употребления слов в английских выражениях;
– фильтр — на основе запроса пользователя формируется список документов, разделённый на группы (фильтры);
– конвертер — переводит денежные единицы из одной валюты в другую, а также производит вычисления одновременно с разными валютами.

FindSounds.com:
– позволяет искать звуковые файлы разных форматов — wav, mp3, aiff, au;
– находит самые разнообразные звуки — крики животных, скрежет машин, звон, стук, сирены, жужжание насекомых, грохот взрывов и стрельбы, всплеск воды и т.д.;
– поиск осуществляется по разным критериям, например, по размеру, наличию двух или одного канала звучания (стерео/моно), частоте дискретизации и разрядности звучания;
– в результатах поиска ресурс показывает не только ссылки на найденные файлы, но и их основные характеристики, а также график амплитуды звука, по которой можно судить о характере звучания данного семпла.

Gnod.net — подбор музыки, книг и фильмов по вкусу:
– имеет несколько баз данных — по музыкальным исполнителям, фильмам, книгам и людям (четыре сервиса: Gnod Music, Gnod Books, Gnod Movies и Flork);

– по нескольким именам музыкальных исполнителей (или названиям книг, фильмов), которые нравятся пользователю, проводится анализ результатов и предлагается вариант певца или группы, которая должна понравиться пользователю.

Alldll.net — поиск файлов библиотек:
– поисковая база данных по наиболее популярным библиотекам dll;
– файлы рассортированы по алфавиту, присутствует функция поиска.

Medpoisk.ru — поиск медицинской информации:
– предназначен для поиска на медицинских сайтах;
– может использоваться для поиска работы среди медицинских работников;
– содержит каталог медицинских учреждений по регионам.

Taggalaxy.de — поиск картинок и фотографий:
– осуществляет поиск изображений на Flickr.com с предварительным просмотром;
– интерфейс поиска – полностью трёхмерный.

Searchme.com — поисковик с предпросмотром:
– показывает результаты поиска в виде анимированной ленты эскизов, уменьшенных скриншотов веб-страниц, включающих в себя ключевое слово поиска.

Тындекс (tyndex.ru) — специализированная поисковая система, которая осуществляет поиск по прайс-листам, размещённым на сайтах фирм-продавцов.

Научные поисковые системы.
– **Scirus** — ищет веб-страницы с научным содержанием: сайты университетов, библиотек и т.д.
– **Google Scholar** — находит статьи крупных научных издательств, архивы препринтов, публикации на сайтах университетов, научных обществ и других научных организаций; рассчитывает индекс цитирования публикаций и позволяет находить статьи, содержащие ссылки на те, что уже найдены.
– **Science Research Portal** — полнотекстовый поиск в журналах крупных научных издательств, например Elsevier, Highwire, IEEE, Nature, Taylor & Francis и др.; ищет статьи и документы в открытых научных базах данных: Directory of Open Access Journals, Library of Congress Online Catalog, Science.gov и Scientific News.
– **Infotrieve — artical finder** — поиск статей в более чем 35000 журналах по физике, технике, медицине, юриспруденции и др.; возможен поиск только по какой-то определённой области науки; можно читать аннотации.
– **e-Print ArXive** — бесплатный архив электронных публикаций. Темы: физика, математика, нелинейная динамика, компьютерные науки. Содержит поисковую систему по тематическим разделам.

Глава 3. Эмуляция удалённого терминала и удалённый доступ к ресурсам сети. Протоколы Telnet и SSH

3.1. Технология Telnet

Telnet — информационная технология Интернета, включающая в себя **telnet-интерфейс** пользователя, **telnetd-процесс**, **TELNET-протокол**, которая обеспечивает описание и реализацию сетевого терминала для доступа к ресурсам удалённого компьютера.

Telnet даёт возможность устанавливать соединение с удалённым компьютером таким образом, что создаётся впечатление, как будто местный терминал — это терминал удалённой системы.

3.1.1. TELNET-протокол

Назначение TELNET-протокола — общее описание двунаправленного, восьмибитового взаимодействия с целью обеспечения стандартного метода взаимодействия терминального устройства и терминал-ориентированного процесса (возможно взаимодействие «терминал-терминал» (связь) и «процесс-процесс» (распределённые вычисления)).

Основные концепции TELNET-протокола:

- концепция сетевого виртуального терминала (Network Virtual Terminal, NVT);
- принцип договорных опций (согласование параметров взаимодействия);
- симметрия связи «терминал-процесс».

Сетевой виртуальный терминал (Network Virtual Terminal, NVT) — спецификация представления правил функционирования терминального устройства, содержащая стандартное описание наиболее широко используемых возможностей реальных физических терминальных устройств.

NVT позволяет описать и преобразовать в стандартную форму способы отображения и ввода информации.

Команды NVT:

- IP (Interrupt Process, прервать процесс) — механизм прерывания процесса выполнения задачи пользователя;
- AO (Abort Output, прервать процесс выдачи) — механизм прерывания вывода информации на экран;
- AYT (Are You There, «ты ещё здесь?») — проверка связи с удалённой машиной;

– EC (Erase Character, удалить символ) — механизм удаления последнего введённого в буфер символа;
– EL (Erase Line) — механизм удаления последней введённой в буфер строки.

Команда telnet — 2-байтовая последовательность, состоящая из Esc-символа (255) IAC (Interpret as Command) и кода команды (240-255).

Команды, связанные с процедурой согласования параметров сеанса, имеют 3-байтовый формат: третий байт — ссылка на устанавливаемую опцию.

Принцип договорных опций (команд) позволяет согласовать возможности представления информации (изменение набора символов, режима эха, и т.д.) на терминальных устройствах.

Одна из сторон инициирует запрос об эффективности какой-либо опции. Другая сторона подтверждает (опция немедленно вступает в силу) или отвергает запрос.

Таблица 3.1
NVT-символы настройки для опций переговоров

Символ	**Десят. знач.**	**Двоичное знач.**	**Смысловое значение**
WILL	251	11111011	1. Предложение для запуска 2. Принятие запроса на запуск
WONT	252	11111100	1. Отказ запросу на запуск 2. Предложение для отключения 3. Принятие запроса на отключение
DO	253	11111101	1. Одобрение предложения на запуск 2. Требование на запуск
DONT	254	11111110	1. Неодобрение предложения на запуск 2. Одобрение предложения на блокировку 3. Требование на отключение
IAC	255	11111111	Интерпретация (следующего символа) как управления (Interpret (the next character) as control)

Симметрия связи позволяет серверу и клиенту в течение одной сессии меняться местами:

- при взаимодействии на удалённом хосте возможности ввода и отображения информации определяются конкретным физическим терминалом, и договорной процесс сводится к заказу терминальной программой характеристик этого терминала;
- при обмене информацией между двумя терминальными программами в режиме «терминал-терминал» каждая из сторон может выступать инициатором изменения принципов представления информации (принцип прямого действия).

Симметричность синтаксиса согласования может потенциально привести к бесконечному циклу согласования.

Пример 3.1. Повтор каждого символа, посланного серверу (рис. 3.1)

Рис. 3.1. Пример эхо-опции NVT

Клиент должен послать запрос (**IAC**, **DO** и **ECHO**) серверу, используя команду **DO**. Сервер принимает запрос и возможную опцию, о чем информирует клиента посылкой трёх символов одобрения: **IAC**, **WILL** и **ECHO**.

Пример 3.2. Переговоры о субопциях (рис. 3.2, табл. 3.2)

Таблица 3.2

NVT-символы настройки подопций

Символ	**Десят. знач.**	**Двоичное знач.**	**Смысловое значение**
SE	240	11110000	Конец подопции
SB	250	11111010	Начало подопции

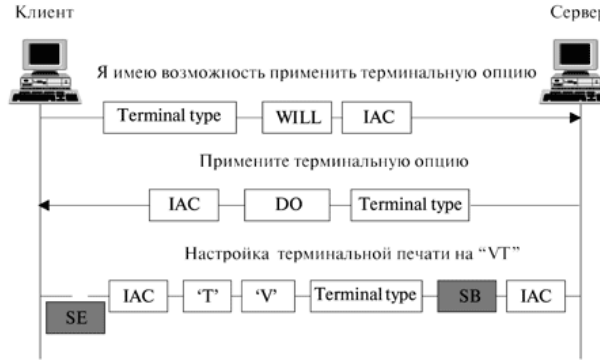

Рис. 3.2. Переговоры о субопции NVT

3.1.2. telnetd-процесс

telnetd — это сервер, обслуживающий протокол TELNET.

Основная задача telnetd — прослушивание TCP-порта.

При поступлении запроса на обслуживание telnetd назначает каждому удалённому клиенту **псевдотерминал (pty)** в качестве *стандартного файла ввода (stdin)*, *стандартного файла вывода (stdout)* и *стандартного файла ошибок (stderr)*.

При установке взаимодействия с удалённым клиентом telnetd обменивается командами настройки.

3.1.3. Telnet-интерфейс

Telnet-интерфейс позволяет организовать взаимодействие пользователя с устройством для работы по протоколу TELNET:

- в режиме командной строки (command mode) — команды сразу достигают удалённого хоста;
- в режиме удалённого терминала (input mode) — команды буферизуются, а затем пакетом отправляются на удалённый хост.

Таким образом:

- telnet-соединение — TCP-соединение, используемое для передачи данных с различной управляющей информацией;
- идеологически telnet — основа всех протоколов прикладного уровня TCP/IP;
- фактически — для передачи управляющих сообщений используется урезанный (без поддержки терминала) протокол telnet.

Принцип работы протоколов прикладного уровня посредством telnet:
- подключение к определённому порту сервиса;
- установление через этот порт сессии telnet;
- передача команд рассматриваемого протокола посредством управляющих сообщений telnet внутри сессии telnet.

Таблица 3.3
Основные команды режима командной строки telnet

Команда	**Назначение**
open host [port]	Начать telnet-сессию с машиной host по порту port
close	Завершить telnet-сессию и вернуться в командный режим
quit	Завершить работу telnet
z	Заморозить telnet-сессию и перейти в режим интерпретатора команд локальной системы.
mode type	Если значение type line, то используется буферизованный обмен данными, если character — обмен небуферизованный
? [command]	Список команд или описание конкретной команды
send argument	Ввод команд и сигналов протокола TELNET, указываемых в качестве аргумента

3.2. Кратное введение в криптографию. Шифрование

Определение 3.1. *Шифрование* — обратимое преобразование данных с целью их сокрытия от посторонних.

Определение 3.2. *Ключ шифрования* — секретная кодовая последовательность, используемая в процессе преобразования информации.

Ключи шифрования:
- симметричные;
- асимметричные.

Асимметричные ключи шифрования:
- используется пара ключей — закрытый и открытый;

– один ключ пары шифрует, другой — расшифровывает;
– открытый ключ может быть доступен каждому, второй (секретный) ключ должен храниться в надёжном месте;
– сообщение может быть кем угодно зашифровано с помощью открытого ключа, но открыть его (расшифровать и прочитать) сможет только хозяин закрытого ключа;
– наоборот, сообщение может быть сертифицировано секретным ключом, а ассоциированный открытый ключ выступает в качестве идентификатора хозяина закрытого ключа, позволяя корректно расшифровать сообщение.

Симметричные ключи шифрования:
– для шифрования и расшифровки данных используется один ключ;
– является потенциально небезопасным;
– вкладывается внутрь сообщения, зашифрованного асимметричным ключом.

3.2.1. Алгоритм асимметричного шифрования RSA

Алгоритм RSA:
– безопасность алгоритма основана на трудоёмкости разложения на множители (факторизации) больших чисел;
– открытый и закрытый ключи являются функциями двух больших простых чисел, разрядностью 100–200 десятичных цифр;
– восстановление открытого текста по шифрованному тексту и открытому ключу равносильно разложению числа на два больших простых множителя:
  1) для генерации двух ключей применяются два больших случайных простых числа p и q одинаковой длины;
  2) рассчитывается произведение $n = pq$;
  3) случайным образом выбирается ключ шифрования e так, чтобы e и $(p-1)(q-1)$ являлись взаимно простыми числами;
  4) вычисляется ключ расшифрования d:

$$d = e^{-1}(\mathrm{mod}(p-1)(q-1)),$$

  где d и n также взаимно простые числа.

Числа e и n — это открытый ключ, а число d — закрытый. Числа p и q могут быть отброшены, но они не должны быть раскрыты.

Формула шифрования:
– при шифровании сообщение m разбивается на цифровые блоки m_i, размерами меньше n (для двоичных данных выбирается самая большая степень числа 2, меньшая n);
– зашифрованное сообщение c будет состоять из блоков c_i, причём длина блока c_i равна длине блока m_i;

– формула шифрования имеет вид:

$$c_i = {m_i}^e \bmod n.$$

Формула расшифрования:

– для каждого зашифрованного блока c_i вычисляется:

$$m_i = {c_i}^d \bmod n;$$

– так как

$${c_i}^d = ({m_i}^e)^d = {m_i}^{k(p-1)(q-1)+1} = m_i,$$

все по mod$n$, то формула восстанавливает сообщение.

3.2.2. Алгоритм симметричного шифрования DES

Алгоритм симметричного шифрования DES:

– предназначен для шифрования данных 64-битовыми блоками;
– представляет собой комбинацию двух методов шифрования: *рассеивания* и *перемешивания*;
– к тексту применяется *подстановка*, а за ней — *перестановка*, зависящая от ключа.

Схема алгоритма:

– после первоначальной перестановки 64-битовый блок разбивается на правую и левую половины по 32 бита;
– затем выполняется 16 раундов одинаковых действий (рис. 3.3), называемых функцией f, в которых данные объединяются с ключом;
– после 16-го раунда правая и левая половины объединяются, и алгоритм завершается заключительной перестановкой (обратной к первоначальной);
– на каждом раунде биты ключа сдвигаются, а затем из 56 битов ключа выбираются 48 битов;
– правая половина данных увеличивается до 48 бит путём перестановки с расширением, складывается операцией XOR с 48 битами смещённого и переставленного ключа, проходит через 8 S-блоков, образуя 32 новых бита, и переставляется снова. Эти четыре операции выполняются функцией f;
– результат исполнения функции f складывается с левой половиной с помощью ещё одной операции XOR;
– в итоге появляется новая правая половина, а старая правая половина становится новой левой. Эти действия повторяются 16 раз, образуя 16 раундов алгоритма DES.

Если обозначить через B_i результат i-й итерации, L_i и R_i — левую и правую половины B_i, K_i — 48-битовый ключ для раунда i, а f — функцию, выполняющую все подстановки, перестановки и операцию

XOR с ключом, то раунд можно представить так:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i).$$

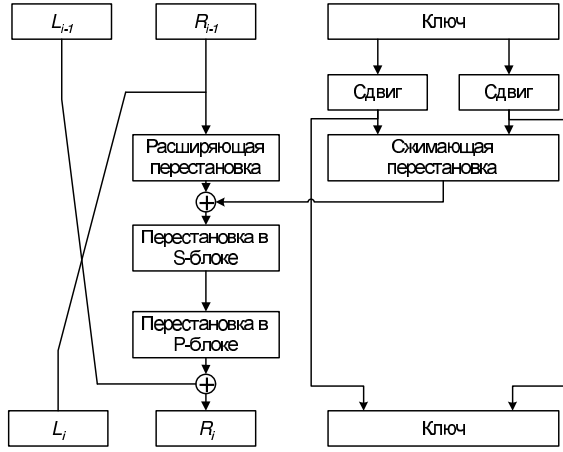

Рис. 3.3. Один раунд алгоритма DES

Расшифрование DES.
Алгоритм DES позволяет использовать для шифрования и расшифрования блока одну и ту же функцию. Единственное отличие состоит в том, что ключи должны использоваться в обратном порядке, т.е. если в раундах зашифрования использовались ключи $K_1, K_2, \ldots, K_{16}$, то ключами расшифрования будут $K_{16}, K_{15}, \ldots, K_1$.

3.2.3. Хэш-функция

Контроль целостности передаваемых или хранимых данных в основном производится путём расчёта некоторой контрольной суммы данных.

Криптографически стойкие контрольные суммы вычисляются как результат применения к исходному тексту **хэш-функции**.

Свойства хэш-функций:
– необратимость;
– рассеивание;
– стойкость к коллизиям.

Хэш-функции — односторонняя (необратимая) функция.

Определение 3.3. *Односторонняя функция* — функция, определённая, например, на множестве натуральных чисел и не требующая для вычисления своего значения больших вычислительных ресурсов, но вычисление обратной функции (т.е. по известному значению функции восстановить значение аргумента) оказывается невозможно теоретически или вычислительно.

Коллизией хэш-функции H называется ситуация, при которой существуют два различных текста T_1 и T_2, но $H(T_1) = H(T_2)$.

Требование стойкости к коллизиям означает, что для криптографически-«хорошей» хэш-функции для заданного текста T_1 вычислительно невозможно найти текст T_2, вызывающий коллизию.

Свойство рассеивания требует, чтобы минимальные изменения текста, подлежащего хешированию, вызывали максимальные изменения в значении хэш-функции.

Основные применяемые на сегодняшний день алгоритмы, реализующие хэш-функции:

- MD2, MD4, MD5, SHA и его вариант SHA1;
- российский алгоритм, описанный стандартом ГОСТ Р 34.11–94.

Чаще используются MD5, SHA1 и в России ГОСТ Р 34.11–94.

3.3. Протокол SSH

SSH (Secure Shell) — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов).

- SSH шифрует весь трафик, включая передаваемые пароли, и допускает выбор различных алгоритмов шифрования.
- SSH позволяет безопасно передавать в незащищённой среде практически любой другой сетевой протокол (таким образом, можно не только удалённо работать на компьютере через командную оболочку, но и передавать по шифрованному каналу звуковой поток или видео).
- SSH может использовать сжатие передаваемых данных для последующего их шифрования.

SSH-туннель — туннель, создаваемый посредством SSH-соединения и используемый для шифрования туннелированных данных. Используется для того, чтобы обезопасить передачу данных в Интернете. Особенность состоит в том, что незашифрованный трафик какого-либо протокола шифруется на одном конце SSH-соединения и расшифровывается на другом.

- SSH-сервер обычно прослушивает 22 TCP-порт.

– Для аутентификации сервера SSH использует алгоритм Диффи–Хеллмана
– Для аутентификации клиента — шифрование с открытым ключом; для шифрования передаваемых данных — более быстрое симметричное шифрование.
– Среди алгоритмов шифрования с открытым ключом чаще всего используются RSA и DSA. Из симметричных алгоритмов — AES, Blowfish и 3DES.
– Целостность переданных данных проверяется с помощью CRC32 в SSH-1 или HMAC-SHA1/HMAC-MD5 в SSH-2.

Команды ssh:

– `ssh user@host` — подключиться к host как user;
– `ssh -p port user@host` — подключиться к host на порт port как user;
– `ssh-copy-id user@host` — добавить ключ на host для user, чтобы включить логин без пароля и по ключам.

Возможности SSH

– Жёсткая проверка и достоверность, исключение большинства узких сетевых мест (IP, routing, DNS spoofing).
– Все соединения прозрачны и автоматически шифруются. Для обмена ключей используется метод RSA и шифр для кодирования сессии (сеанса работы).
– Обеспечение безопасности сессий X11. Автоматическая установка DISPLAY на серверной машине и перенаправление любых X11-соединений через безопасные (шифруемые) каналы. Автоматически создаётся поддельная информация Xauthority и отправляется удалённой машине; локальный клиент автоматически проверяет входящие соединения X11 и заменяет подставные данные авторизации на реальные (никогда не передавая удалённой машине реальных данных).
– Произвольные TCP/IP порты могут быть перенаправлены через шифруемые каналы в обоих направлениях.
– Никакого доверия сети. Минимальное доверие удалённой стороне, DNS. Аутентификация RSA не доверяет ничему, кроме закрытых ключей шифрования (privat key).
– Клиент RSA проверяет серверную часть в начале каждого соединения, чтобы избежать различных сетевых атак и узких мест в сети.
– Распространение host authentication ключей может быть выполнено администратором автоматически, когда происходит первое подсоединение к машине (ключи, полученные во время первого соединения, будут сохранены и использованы для аутентикации в будущем), или вручную каждым пользователем для своих потребностей.
– Любой пользователь может создать любое количество ключей RSA-

аутентикации для собственных нужд. Каждый пользователь имеет файл со списком открытых RSA-ключей (public key), для которых проверены соответствующие закрытые ключи (private key) и приняты как аутентикационные. Пользовательские ключи аутентикации обычно имеют размер 1024 бит.
- Программа сервера имеет свой серверный RSA-ключ, который автоматически создаётся каждый час (server key). Этот ключ никогда не сохраняется в файле. Обмен сессионными ключами кодируется с использованием server key и server host key. Наличие отдельного server key делает невозможным дешифрацию перехватываемого сеанса, позже ибо через час server key снова изменится. Временной интервал регенерации server key может быть изменён с 1-часа на иной интервал. Обычная длина server key — 768 бит.
- Агент аутентикации, запущенный на локальной станции пользователя, может быть использован для хранения RSA authentication keys. SSH автоматически перенаправляет данные на вход authentication agent поверх любого соединения, и поэтому нет необходимости сохранять RSA authentication keys на других машинах сети кроме локальной-пользовательской. Протоколы аутентикации используются только для проверки того, что пользовательский агент имеет определённый ключ.
- Клиентская часть может быть сконфигурирована как с системной стороны, так и с пользовательской; большинство возможностей клиентской части может быть перенастроено под соответствующие нужды.
- Полная замена rlogin, rsh и rcp.

3.3.1. SSH-1

- Клиент посылает серверу запрос на установление SSH-соединения и создание нового сеанса.
- Соединение будет принято сервером, если он получил сообщения на установление соединения и готов к открытию нового сеанса связи.
- После этого клиент и сервер обмениваются информацией о поддерживаемых версиях протоколов.
- Соединение будет продолжено, если будет найдено соответствие между протоколами и получено подтверждение о готовности обеих сторон продолжить соединение по данному протоколу.
- Сервер посылает клиенту постоянный открытый и временный серверный ключи. Клиент использует эти ключи для зашифровки сессионного ключа. Несмотря на то что временный ключ посылается прямым текстом, сессионный ключ по-прежнему безопасный.
- Сессионный ключ шифруется временным ключом и открытым ключом сервера, и, таким образом, только сервер может его расшифровать. На этом этапе и клиент, и сервер обладают сессионным

ключом и, следовательно, готовы к безопасному сеансу передачи зашифрованных пакетов.
– Аутентификация сервера происходит исходя из его возможности расшифровки сессионного ключа, который зашифрован публичным ключом сервера. Аутентификация клиента может происходить различными способами, в том числе DSA, RSA, OpenPGP или по паролю.
– Сессия продолжается до тех пор, пока и клиент, и сервер способны аутентифицировать друг друга. Установленное соединение по протоколу SSH-1 позволяет защитить передаваемые данные стойким алгоритмом шифрования, проверкой целостности данных и сжатием.

3.3.2. SSH-2

Основное различие между протоколами SSH-1 и SSH-2 заключается в том, что протокол SSH-2 разделяет все функции протокола SSH между тремя протоколами (протоколом транспортного уровня, протоколом аутентификации и протоколом соединения), в то время как протокол SSH-1 представляет собой один неделимый протокол.

Протокол транспортного уровня предоставляет возможность шифрования и сжатия передаваемых данных, а также реализует систему контроля целостности данных.

Протокол соединения позволяет клиентам устанавливать многопоточное соединение через оригинальный SSH-туннель, таким образом снижая нагрузку, которую создают клиентские процессы.

Протокол аутентификации отделён от протокола транспортного уровня, так как не всегда бывает необходимым использование системы аутентификации. В случае, если нужна аутентификация, процесс защищается оригинальным безопасным каналом, установленным через протокол транспортного уровня.

Сам по себе, протокол транспортного уровня является достаточным для установления защищённого соединения, он является основой протокола SSH-2, и протоколы соединения и аутентификации основаны на нем.

Глава 4. Служба имён доменов DNS. Службы NetBIOS, WINS

4.1. Система доменных имён DNS

Определение 4.1. *Система доменных имён (Domain Name System, DNS)* — распределённая система (распределённая база данных), ставящая в соответствие доменному имени хоста (компьютера или другого сетевого устройства) IP-адрес и наоборот.

Один IP-адрес может иметь множество имён (виртуальный хостинг — поддержка на одном узле нескольких веб-сайтов). Одному имени может быть сопоставлено множество IP-адресов (позволяет создавать балансировку нагрузки).

Характеристики DNS:

- **распределённость хранения информации**: каждый узел сети хранит только те данные, которые входят в его зону ответственности и (возможно) адреса корневых DNS-серверов;
- **кеширование информации**: узел может хранить некоторое количество данных не из своей зоны ответственности для уменьшения нагрузки на сеть;
- **иерархическая структура**: все узлы объединены в дерево, каждый узел может самостоятельно определять работу нижестоящих узлов или делегировать их другим узлам;
- **резервирование**: несколько серверов (разделённые физически и логически) отвечают за хранение и обслуживание своих узлов (зон), что обеспечивает сохранность данных и продолжение работы даже в случае сбоя одного из узлов.

Определение 4.2. *Зона* — логический узел в дереве имён.

Определение 4.3. *Домен* — название зоны в системе доменных имён (DNS) Интернета, выделенной какой-либо стране, организации или для иных целей.

Структура доменного имени отражает порядок следования зон в иерархическом виде.

Доменное имя читается слева направо от младших доменов к доменам высшего уровня (в порядке повышения значимости). Корневым доменом всей системы является точка «.» (часто опускают), следом идут домены первого уровня (географические или тематические), затем — домены второго уровня, третьего и т.д. Например, сайт `www.telesys.pfu.edu.ru` принадлежит домену `pfu.edu.ru` (рис. 4.1).

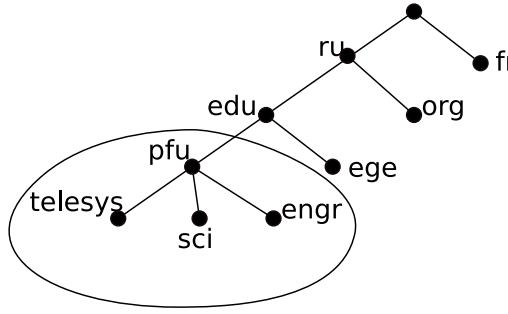

Рис. 4.1. Домен pfu.edu.ru

Определение 4.4. *Поддомен (subdomain)* — имя подчинённой зоны.

Такое деление может достигать глубины 127 уровней, а каждая метка может содержать до 63 символов, пока общая длина вместе с точками не достигнет 254 символов.

Домен содержит все узлы сети, доменные имена которых в него входят. Узлы сети связаны логически, зачастую по географическому или организационному признаку и совсем необязательно сетью, адресом или типом используемого оборудования.

Определение 4.5. *DNS-сервер* — специализированное ПО для обслуживания DNS.

В качестве серверов доменных имён чаще всего используются различные версии BIND (Berkeley Internet Name Domain).

Определение 4.6. *DNS-клиент* — специализированная библиотека (или программа) для работы с DNS.

В ряде случаев DNS-сервер выступает в роли DNS-клиента. Клиент решает следующие задачи:
- опрашивание DNS-серверов;
- интерпретация полученных ответов (RR-записей или сообщений об ошибках);
- возврат информации в программу, которая её запросила.

Определение 4.7. *Ответственность (authoritative)* — признак размещения зоны на DNS-сервере.

Ответы DNS-сервера могут быть: *авторитативными (ответственными, authoritative)*, когда сервер заявляет, что сам отвечает за

зону, и *неавторитативными (неответственные, non-authoritative)*, когда сервер обрабатывает запрос и возвращает ответ других серверов.

DNS-серверы обычно обладают полной информацией по определённым сегментам (или зонам) пространства доменных имён, которая загружается из файла либо может быть получена от других серверов имён (т.е. *сервер имён является авторитативным для конкретной зоны*).

Делегирование поддоменов включает передачу ответственности за какую-то часть домена другой организации — различные DNS-серверы назначаются авторитативными в делегируемых поддоменах.

Хранимая зоной информация после делегирования включает уже не информацию по делегированному поддомену, а информацию о серверах имён, являющихся для этого поддомена авторитативными.

Определение 4.8. *DNS-запрос (DNS query)* — запрос от клиента (или сервера) серверу.

Запрос может быть *рекурсивным* — сервер опрашивает серверы (в порядке убывания уровня зон в имени), пока не найдёт ответ или не обнаружит, что домен не существует; или *нерекурсивным* — возвращает данные о зоне, которая находится в зоне ответственности DNS-сервера, получившего запрос, или возвращает адреса корневых серверов (рис. 4.2).

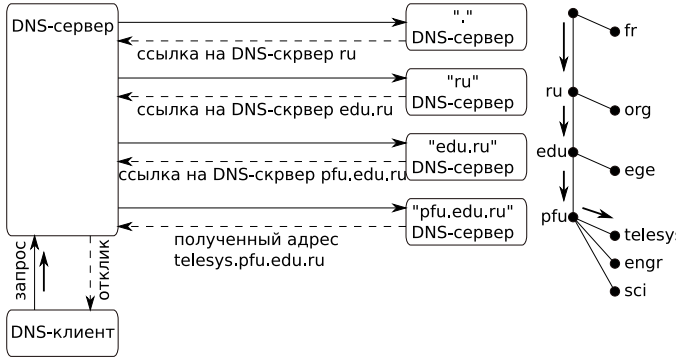

Рис. 4.2. Разрешение имени telesys.pfu.edu.ru в сети Интернет

Родовые домены высшего уровня (generic top-level domains, gTLDs):

– com — коммерческие организации;
– edu — образовательные организации;
– gov — правительственные организации;

– mil — военные организации;
– net — в прошлом организации, обеспечивающие работу сетевой инфраструктуры, такие как NSFNET (nsf.net) и UUNET (uu.net); с 1996 г. доступен для коммерческих организаций;
– org — в прошлом некоммерческие организации; с 1996 г. доступен для любых организаций;
– int — международные организации, такие как NATO (nato.int).

С 2001 г. в gTLDs входят домены name, biz, info, pro.

Спецификация DNS определяет **два типа DNS-серверов**:
– *первичный мастер-сервер (primary master)* — производит загрузку данных для зоны из файла на машине-сервере;
– *дополнительный*, или *вторичный, мастер-сервер (secondary master)* — получает данные зоны от другого DNS-сервера, который является авторитативным для этой зоны и называется его *мастером (master server)*.

Когда запускается вторичный сервер, он устанавливает связь с мастером и, в случае необходимости, получает зональные данные. Этот процесс называется **передачей**, или **трансфером, зоны (zone transfer)**.

Определение 4.9. *Файлы данных зоны* — файлы, из которых первичные DNS-серверы производят чтение зональных данных.

Вторичные DNS-серверы также могут загружать зональные данные из файлов.

В файле описания зоны используются:
– *директивы управления (control entries)*;
– *записи описания ресурсов (resource records, RR)*.

Директивы управления:
– директива `$ORIGIN` определяет текущее имя домена (например, в случае, когда в описание зоны требуется включить запись описания хоста из другой зоны);
– директива `$INCLUDE` используется для того, чтобы в файл описания зоны можно было включить содержание другого файла (рекомендуется при описании больших зон, разбивать их на небольшие фрагменты).

```
[<comment>]
$ORIGIN [<comment>]
$INCLUDE [] [<comment>]
```

В квадратные скобки [ ] заключены необязательные параметры, а в угловые скобки < > — сущности.

RR-записи описывают все узлы сети в зоне и помечают делегирование поддоменов.

Типы записи описания ресурсов:

– SOA-запись — указывает на авторитативность для зоны;
– NS-запись — перечисляет DNS-серверы зоны;
– A — отображение имён узлов в адреса;
– PTR — отображение адресов в имена узлов;
– CNAME — каноническое имя (для псевдонимов);
– MX — отображение имён почтовых серверов.

Формат записи SOA:

```
[zone] [ttl] IN SOA origin contact (
                    serial refresh retry expire minimum)
```

– *zone* — имя зоны;
– *ttl* — время кеширования (в SOA всегда пустое, определяется директивой управления `$TTL`);
– IN — класс данных Internet;
– *origin* — доменное имя primary master сервера зоны;
– *contact* — почтовый адрес лица, осуществляющего администрирование зоны (так как символ `@` имеет особый смысл при описании зоны, то вместо него в почтовом адресе используется символ «.»);
– *serial* — серийный номер файла зоны (учёт изменений файла описания зоны)[1] ;
– *refresh* — интервал времени, после которого slave сервер обязан обратиться к master серверу с запросом на верификацию своего описания зоны;
– *retry* — интервал времени, после которого slave сервер должен повторить попытку синхронизировать описание зоны с master сервером;
– *expire* — интервал времени, после которого slave сервер должен прекратить обслуживание запросов к зоне, если он не смог в течение этого времени верифицировать описание зоны, используя информацию с master сервера;
– *minimum* — время негативного кеширования (negative caching), т.е. время кеширования ответов, которые утверждают, что установить соответствие между доменным именем и IP-адресом нельзя.

[1] Серийный номер должен быть положительным целым числом и увеличиваться каждый раз, когда в файл зоны вносятся изменения (RFC1982). Увеличение серийного номера показывает вторичным серверам, что зона изменена и что им необходимо обновить у себя зону.

Если вы измените серийный номер так, что после сохранения файла зоны он останется неизменным или станет меньше, чем был ранее, то вторичные серверы не будут перечитывать данные с первичного сервера, так как будут считать, что данные не изменились.

При задании серийного номера вручную обычно пользуются форматом даты ISO 8601, дополнительно добавляя две цифры для версии (вряд ли вы за день измените файл более 100 раз). Таким образом, формат серийного номера обычно имеет вид ГГГГММДДВВ.

NS-записи обычно следуют сразу за записью SOA в файле описания зоны и указывают на серверы, которые ответственны за эту зону.

Формат записи NS:

```
[domain][ttl] IN NS [server]
```

– *domain* — имя домена, для которого сервер, указанный последним аргументом записи NS, поддерживает описание зоны;
– *server* — доменное имя сервера.

Записи NS указывают как на master, так и на slave серверы. Обычно primary master записывают первым, а резервные серверы указывают вслед за ним.

Основное назначение **адресной записи** — установить соответствие между доменным именем машины и IP-адресом.

Адресная запись имеет следующий формат:

```
[host][ttl] IN A [address]
```

– *host* — доменное имя хоста;
– *address* — IP-адрес машины.

Задача поиска доменного имени по IP-адресу является обратной к прямой задаче — поиску IP-адреса по доменному имени. Прямая задача решается в DNS при помощи записей типа *A (Address)*. Обратная же — при помощи записей-указателей типа *PTR (Pointer)*, которые совместно с записями SOA и NS составляют описание так называемой **обратной зоны**.

Формат PTR-записи имеет следующий вид:

```
[name][ttl] IN PTR [host]
```

– *name* — номер (не реальный IP-адрес машины, а имя в специальном домене in-addr.arpa или в одной из его зон);
– *host* — доменное имя хоста.

Домен IN-ADDR.ARPA.

Основное назначение **домена ARPA (Address and Routing Parameter Area Domain)** — обеспечить отображение численных величин, определяемых протоколами межсетевого обмена, в пространство имён.

В настоящее время в ARPA выделено три поддомена:
– in-addr.arpa — для отображения IP-адресов IPv4 в пространство доменных имён;
– ip6.arpa — для отображения IP-адресов IPv6 в пространство доменных имён;
– e164.arpa — для отображения телефонных номеров формата E.164.

Имена в домене IN-ADDR.ARPA образуют иерархию цифр, которые соответствуют IP-адресам, и записываются эти имена в обратном порядке относительно написания IP-адреса. Например, хост с адресом 194.226.43.1 должен быть описан в домене in-addr.arpa как

```
1.43.226.194.in-addr.arpa
```

Файл конфигурации named.

Пример конфигурации **прямой зоны**:

```
zone "sci.pfu.edu.ru" in {
        type master;
        file "sci.pfu.edu.ru";
};
```

Пример конфигурации **обратной зоны**:

```
zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "localhost.rev";
};
```

Запись **Mail eXchanger (MX)** определяет хост, который отвечает за доставку почты в определённый домен.

Запись MX может быть определена как для всей зоны, так и для отдельно взятой машины и имеет следующий **формат**:

```
[name] [ttl] IN MX [preference] [host]
```

– *name* — имя машины или домена, на который может отправляться почта;
– *preference* — приоритет почтового сервера, имя которого (поле *host*) указано последним аргументом в поле данных MX-записи.

Запись **CNAME** определяет синонимы для реального (канонического) доменного имени машины, которое определено в записи типа A (Address).

Формат записи CNAME:

```
[nickname] [ttl] IN     CNAME   [host]
```

Поле *nickname* определяет синоним для канонического имени, которое задаётся в поле *host*.

Например,

```
$ORIGIN sci.pfu.edu.ru.
olga    IN      A       144.206.192.2
www     IN      CNAME   olga.sci.pfu.edu.ru.
gopher  IN      CNAME   olga.sci.pfu.edu.ru.
```

Фрагмент файла `sci.pfu.edu.ru` **описания зоны:**

```
\@   IN     SOA    ns.sci.pfu.edu.ru. user.sci.pfu.edu.ru. (
                   1   ;Порядковый номер
                   3h  ;Обновление через 3 часа
                   1h  ;Повторение попытки через 1 час
                   1w  ;Устаревание через 1 неделю
                   1h );Отрицательное TTL в 1 час
     IN     NS     ns.sci.pfu.edu.ru.
ns   IN     A      192.168.0.1
$ORIGIN sci.pfu.edu.ru.
; Zone sci.pfu.edu.ru
ns   IN     A      192.168.0.1 ; name server
www  IN     A      192.168.0.2 ; web server
```

Символ `@` в записи SOA указывает на то, что текущим *именем домена* является sci.pfu.edu.ru.

Первое имя после SOA — имя первичного master-сервера DNS зоны. Второе имя — адрес электронной почты человека, управляющего зоной.

Запись *описания сервера доменных имён (NS)* относится к домену sci.pfu.edu.ru, т.е. *авторитативным сервером* для домена sci.pfu.edu.ru будет ns.sci.pfu.edu.ru.

Далее определяется *адрес хоста с именем* ns.sci.pfu.edu.ru (необязательно указывать имя целиком).

`$ORIGIN` определяет имя текущей зоны.

4.2. Протокол NetBIOS

NetBIOS — интерфейс, не зависящий от фирмы-производителя, между ЭВМ типа IBM/PC:
- имеет собственную DNS, которая может взаимодействовать с интернетовской;
- в качестве транспортных протоколов использует TCP и UDP;
- использует для службы имён — порт 137, для службы дейтаграмм — порт 138, а для сессий — порт 139;
- имеет собственную систему команд (call, listen, hang up, send, receive, session status, reset, cancel, adapter status, unlink, remote program load) и примитивов для работы с дейтаграммами (send datagram, send broadcast datagram, receive datagram, receive broadcast datagram).

Сессия начинается с NetBIOS-запроса, задания IP-адреса и определения TCP-порта удалённого объекта, далее следует обмен NetBIOS-сообщениями, после чего сессия закрывается. Сессия осуществляет обмен информацией между двумя NetBIOS-приложениями. Длина сообщения — от 0 до 131071 байта. Допустимо одновременное осуществление нескольких сессий между двумя объектами.

Оконечные узлы NetBIOS:
- широковещательные: b-узлы;
- точка-точка: p-узлы;
- смешанного типа: m-узлы.

IP-адрес может ассоциироваться с одним из указанных типов:
- **b-узлы** устанавливают связь со своим партнёром посредством широковещательных запросов;
- **p- и m-узлы** для этой цели используют **NetBIOS сервер имён (NBNS)** и **сервер распределения дейтаграмм (NBDD)**.

4.3. NetBeui

NetBeui (NetBios extended user interface) — улучшенная версия протокола NetBIOS:

– используется операционными системами LAN manager, LAN server, Windows for Workgroups и Windows NT;
– стандартизован формат пакетов NetBios, добавлены некоторые новые функции;
– базируется на протоколе LLC2, вводит стандарт на формат кадра NetBIOS (NDF) и использует NetBIOS в качестве интерфейса более высокого уровня;
– обладает высоким быстродействием и служит для объединения небольших локальных сетей (20–200 ЭВМ) друг с другом или с главной ЭВМ;
– имеет ограничение — отсутствие внутренней маршрутизации.

4.4. WINS

Определение 4.10. *Windows Internet Name Service (WINS)* — служба сопоставления NetBIOS-имён компьютеров с IP-адресами узлов.

Сервер WINS осуществляет регистрацию имён, выполнение запросов и освобождение имён.

Протокол WINS разработан компанией MicroSoft для операционной среды Windows и предназначен для расширения возможностей NetBIOS. В локальной сети Microsoft имя компьютера — это имя в стиле протокола NetBIOS, заданное во время инсталляции Windows NT.

Два аспекта функционирования WINS:
– *регистрация* — фиксация уникального имени для каждого компьютера сети;
– *разрешение* — получение адреса по имени.

Режимы работы NetBIOS поверх TCP/IP определяют способ идентификации сетевых ресурсов и доступа к ним:
– **b-node** — для разрешения имён используются широковещательные сообщения;
– **p-node** — для разрешения имён используется взаимодействие типа «точка-точка» с сервером имён WINS, который представляет собой улучшенный вариант сервера NetBIOS Name Server (NBNS);
– **m-node** — для разрешения имён сначала используется режим b-node, а затем p-node, если широковещательный запрос не смог разрешить имя;
– **h-node** — для разрешения имён сначала используется режим p-node, а затем b-node, если сервер имён недоступен или в его базе данных отсутствует запрашиваемое имя.

В сети Windows NT тип режима разрешения имени назначается сервером DHCP (Dynamic Host Control Protocol). Если в сети имеются серверы WINS, то протокол NetBIOS поверх TCP/IP разрешает имена путём взаимодействия с сервером WINS. Если серверы WINS

отсутствуют, то протокол NetBIOS поверх TCP/IP использует для разрешения имён широковещательный режим b-node. Протокол NetBIOS поверх TCP/IP в среде Windows NT может также использовать файлы LMHOSTS и службу DNS для разрешения имён в зависимости от того, как сконфигурирован стек TCP/IP на конкретном компьютере.

Способы разрешения имён на хосте посредством серверов WINS:

- протокол WINS *не поддерживается*:
  - компьютер регистрирует своё имя путём широковещательной рассылки пакетов *name registration request (запрос регистрации имени)* по локальной подсети с помощью дейтаграмм протокола UDP;
  - чтобы найти адрес конкретного компьютера, компьютер, не поддерживающий протокол WINS, широковещательно распространяет пакеты *name query request (запрос разрешения имени)* по локальной подсети;
  - если на запрос не приходит ответ, то используются данные из файла LMHOSTS;
- протокол WINS *поддерживается*:
  - во время конфигурации стека TCP/IP имя компьютера регистрируется в сервере WINS, а IP-адрес сервера WINS хранится локально;
  - запросы разрешения имени посылаются сначала на сервер WINS, включая запросы от удалённых клиентов, которые в этом случае маршрутизируются через IP-маршрутизатор;
  - если имя не найдено в базе данных WINS, а клиентский компьютер сконфигурирован как h-node, то он использует широковещательные пакеты *name query request* тем же способом, что и компьютер, не поддерживающий службу WINS;
  - если все предыдущие методы не привели к успеху, просматривается файл LMHOSTS.

Глава 5. Протокол передачи файлов. Основные модули службы FTP. Схема функционирования: управляющий сеанс и сеанс передачи данных. Команды взаимодействия FTP-клиента и FTP-сервера

5.1. Протокол FTP

Протокол FTP (File Transfer Protocol, RFC959) обеспечивает файловый обмен между удалёнными пользователями.

Согласно RFC959, FTP был разработан для следующих целей:

1) поддержка распространения файлов (компьютерных программ и/или данных);
2) поощрение концепции неявного использования удалённых компьютеров;
3) предоставление пользователям единого интерфейса обмена данными;
4) организация надёжного и эффективного обмена данными.

FTP представляет собой классическую клиент-серверную архитектуру. Неявное использование удалённых хостов — концепция, подразумевающая создание унифицированного интерфейса между пользовательской и удалённой системой.

В качестве транспортного протокола используется TCP и порты:

- 20 — передача данных;
- 21 — передача команд управления.

Схема взаимодействия по протоколу FTP (рис. 5.1):

- по запросу клиента формируется канал управления;
- по команде клиента сервером формируется информационный канал;
- по управляющему каналу клиент может посылать команды;
- сервер воспринимает, интерпретирует команды от клиента и передает отклики.
- **Канал управления**:
  - используется для передачи команд от клиента и откликов от сервера;
  - может быть закрыт только после завершения информационного обмена;
  - использует протокол Telnet.
- **Информационный канал**:
  - используется для передачи данных;

– на протяжении всей FTP-сессии может формироваться и ликвидироваться по мере необходимости.

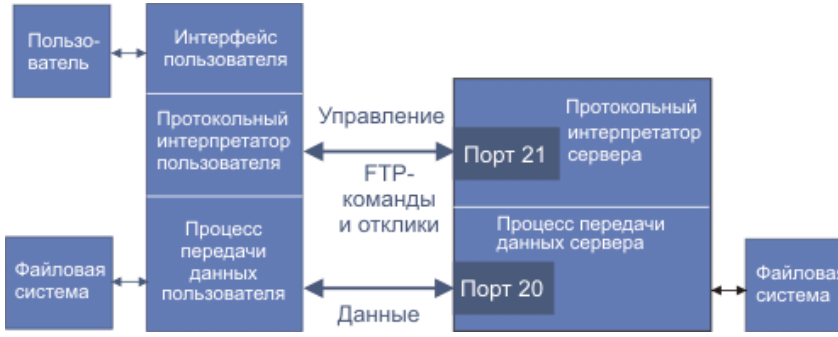

Рис. 5.1. Схема взаимодействия по протоколу FTP

Типичная последовательность действий при взаимодействии по протоколу FTP (рис. 5.2):
– идентификация пользователя (ввод имени и пароля);
– выбор рабочего каталога;
– выбор режима обмена данными;
– выполнение команд обмена данными;
– завершение процедуры.

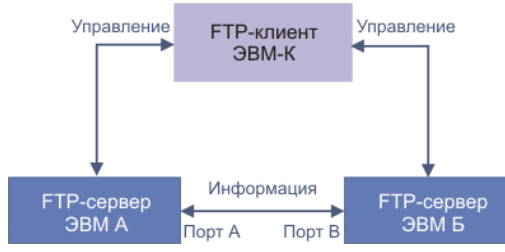

Рис. 5.2. Организация обмена данными между двумя FTP-серверами

Различают **активный** и **пассивный** режимы работы. Действия клиента и сервера в **активном** режиме (рис. 5.3):
1) клиент устанавливает связь и посылает запрос на 21 порт сервера с порта N ($N < 1024$);
2) сервер посылает ответ на порт N;

3) сервер устанавливает канал для передачи данных между своим 20 портом и $N + 1$ портом клиента.

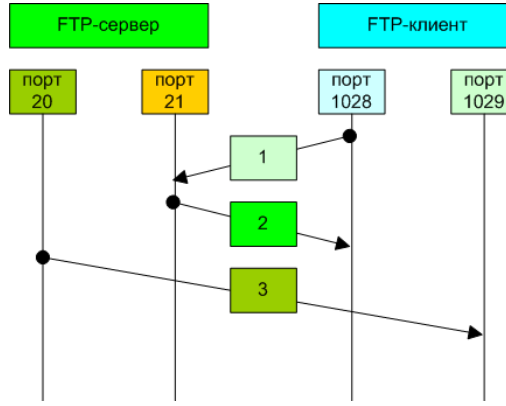

Рис. 5.3. Активный режим FTP

Данный режим работы выгоден для сервера, но не выгоден клиенту:

– существует вероятность того, что порт $N + 1$ уже занят другим приложением;
– соединение может быть сброшено, если указанный порт на стороне клиента закрыт межсетевым экраном.

Действия клиента и сервера в **пассивном** режиме (рис. 5.4):

1) клиент устанавливает связь и посылает запрос (PASV) на 21 порт сервера с порта N ($N < 1024$);
2) сервер посылает ответ и сообщает номер порта для канала данных P ($P > 1024$) на порт N ($N < 1024$) клиента;
3) клиент устанавливает связь для передачи данных по порту $N + 1$ на порт сервера P ($P > 1024$).

Пассивный режим выгоден для клиента, но не выгоден серверу поскольку соединение будет установлено на P порту, который не является зарезервированным системой, то запрос может быть блокирован либо межсетевым экраном, либо самим приложением.

Режимы представления данных на клиенте FTP:

– ASCII — передача текстовой информации (используется по умолчанию и присутствует во всех реализациях FTP);
– EBCDIC — используется только, когда оба узла поддерживают кодировку EBCDIC;

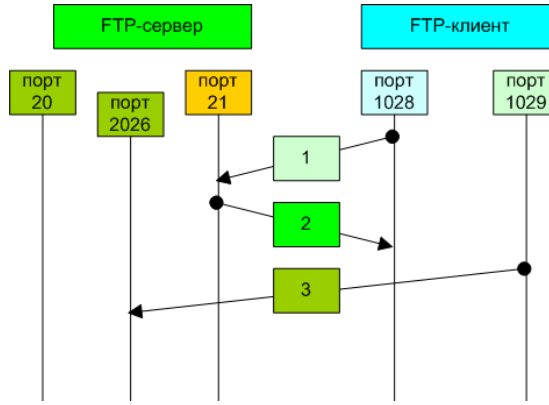

Рис. 5.4. Пассивный режим FTP

– IMAGE — передача бинарных файлов (данные передаются в виде последовательных 8-битных блоков; если блок не полон, то оставшееся место дописывается нулями; разработан для эффективного хранения и передачи файлов);
– LOCAL — клиент задает размер блоков, в которых он желает получать данные. Необходимо в том случае, если системы используют различную длину машинного слова.

EBCDIC (Extended Binary Coded Decimal Interchange Code) — расширенный двоично-десятичный код обмена информацией. Разработан IBM для использования в своих мейнфреймах. Существует шесть версий EBCDIC, причём несовместимых между собой.

В дополнение к режимам представления данных FTP также поддерживает указание их структуры.

Структуры данных FTP:
– **файловая** — используется по умолчанию, представляет собой упорядоченную последовательность данных;
– **запись** — файл состоит из упорядоченной последовательности записей;
– **страницы** — файл состоит из независимых, проиндексированных страниц.

Три режима обмена данными в FTP:
– **потоковый** — данные передаются как единый поток байт (используется по умолчанию);
– **блочный** — данные передаются блоками (в заголовок очередной порции данных записывается ее порядковый номер);
– **сжатый** — реализуется простейший механизм сжатия.

Основные команды для работы с FTP:
- CDUP — подняться на уровень выше в директории;
- CWD — сменить директорию;
- LIST — выдать содержимое директории;
- PASV — войти в пассивный режим;
- PORT — войти в активный режим;
- GET — получить указанный файл;
- MGET — получить набор файлов;
- STOR — сохранить файл на сервер;
- TYPE — задать режимы обмена IMAGE, ASCII или EBCDIC;
- HELP — вывод справки о возможных командах;
- QUIT — выход.

Этапы выполнения команд обмена:
- формирование канала под управлением клиента (так как именно клиент выдал команду `get`, `dir`, `put` и т.д.);
- клиент выбирает произвольный номер порта и осуществляет для него процедуру `passive open`;
- клиент посылает номер порта серверу по каналу управления (порт 21), используя команду `PORT`;
- сервер получает номер порта по каналу управления и выдает команду `active open` в указанный порт ЭВМ клиента (сервер для канала данных всегда использует порт с номером 20).

Коды диагностики протокола FTP:
- 1** — позитивный предварительный отклик. Сигнализирует о том, что операция начата, но еще не завершена;
- 2** — сигнал успешного завершения процедуры. Можно вводить другую команду;
- 3** — положительный промежуточный отклик. Команда принята, но для продолжения нужна дополнительная информация;
- 4** — негативный отклик. Команда не воспринята, но можно попробовать исполнить ее еще раз;
- 5** — команда не выполнена и не может быть выполнена в принципе.

Коды для второй цифры отклика:
- *0* — синтаксис верен, но команда не имеет смысла;
- *1* — необходима дополнительная информация;
- *2* — отклик, связанный с управлением канала информации;
- *3* — отклик команд аутентификации;
- *4* — функция не определена;
- *5* — отклик характеризует состояние файловой системы.

Некоторые коды откликов:
- 150 — статус файла правилен, подготавливается открытие канала;
- 220 — слишком много подключений к FTP-серверу (можете попробовать позднее); в некоторых версиях указывает на успешное завершение промежуточной процедуры;

– 221 — благополучное завершение по команде quit;
– 226 — закрытие канала, обмен завершен успешно;
– 227 — сервер возвращает ip-адрес и номер порта;
– 230 — пользователь идентифицирован, продолжайте;
– 250 — запрос прошел успешно.

Пример 5.1. Работа с сервером FTP

```
220 FTP server ready.
USER Tassadar
230 Login successful.
PASV
227 Entering Passive Mode (10,10,10,3,233,92)
LIST
150 Here comes the directory listing.
226 Directory send OK.
CWD incoming
250 Directory successfully changed.
PASV
227 Entering Passive Mode (10,10,10,3,207,56)
STOR Princess_Mononoke.avi
150 Ok to send data.
226 File receive OK.
QUIT
221 Goodbye.
```

5.2. Протокол TFTP

TFTP (Trivial FTP, RFC-1350, RFC-783, RFC-906, STD0033) — упрощённая версия FTP.

Особенности TFTP:

– не имеет системы безопасности и идентификации;
– используется транспортный протокол UDP (порт 69), а не TCP;
– обычно передача осуществляется блоками по 512 байт с ожиданием подтверждения приема каждого пакета (протокол «стой-и-жди»).

Применение TFTP:

– при загрузке операционной системы в бездисковые рабочие станции (например, X-терминалы);
– для загрузки конфигурационных файлов в маршрутизатор;
– непосредственное исполнение команды TFTP (TFTP имя_ЭВМ), хотя эта процедура не дает каких-либо серьезных преимуществ перед FTP (кроме скорости обмена).

5.3. Поиск в FTP-архивах

Тематические FTP-серверы:
- ftp.microsoft.com — FTP-сервер Microsoft, где содержатся программы для ее операционных систем;
- oak.oakland.edu — программное обеспечение для MS-DOS, MAC и других операционных систем;
- ftp.sure.net — архив различных документов по Интернету;
- ocf.berkley.edu — библиотека документов по сетевым технологиям.

Archie — база данных по содержимому анонимных FTP-серверов.
Некоторые из Archie-серверов:
- archie.th-darmstadt.de — 130.3.22.60;
- archie.unipi.it — 131.114.21.10;
- archie.switch.ch — 130.59.1.40.

Другие системы FTP-поиска:
- **система «Мамонт»** (http://www.mmnt.ru) — одна из крупнейших в Интернете систем для поиска в FTP-архивах. Отличается большим количеством проиндексированных архивов.
- **Google**. Чтобы найти необходимые данные, в строке поиска достаточно набрать, например:

  ```
  index of ftp/mp3
  ```

  На данный запрос система вернет список всех известных ей FTP-архивов с mp3-файлами.
- **FileSearch** — ищет файлы на FTP-серверах по именам самих файлов и каталогов.

Глава 6. Понятие электронной почты

Основные почтовые протоколы в Интернете:
- SMTP (Simple Mail Transport Protocol);
- POP (Post Office Protocol);
- IMAP (Internet Mail Access Protocol).

Доставка почты от отправителя к получателю проходит три стадии (рис. 6.1):
1. электронная почта проходит через пользовательского агента на локальный сервер;
2. электронная почта доставляется удаленному серверу, но не к удаленному агенту пользователя;
3. удаленный агент пользователя применяет протокол POP3 или IMAP4, чтобы запустить почтовый ящик и получить почту.

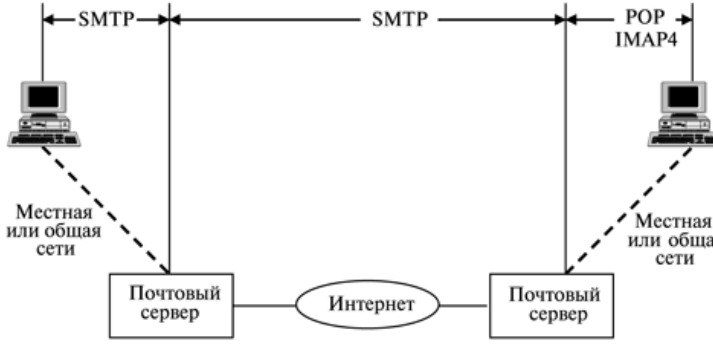

Рис. 6.1. Доставка электронной почты

6.1. Простой протокол передачи почты SMTP

Простой протокол передачи почты (Simple Mail Transfer Protocol, SMTP) описывает систему команд и соглашений, обеспечивающих обмен почтовыми сообщениями между пользователями одной и той же или различных компьютерных сетей.

SMTP поддерживает посылку сообщений:
- одному или более получателям;

– включающих в себя текст, голосовые сообщения, видео или графические материалы;
– для пользователей сетей, не входящих в Интернет.

6.1.1. Команды и отклики SMTP

SMTP использует **команды** и **отклики** для передачи сообщения между почтовыми агентами (MTA) клиента и сервера.

Таблица 6.1

Команды SMTP

Ключевое слово	Аргумент(ы)
HELLO	Имя хоста отправителя
MAIL FROM	Отправитель сообщения
RCPT TO	Предназначенный получатель сообщения
DATA	Содержание (тело сообщения)
QUIT	Завершение соединения
RSET	Прерывание текущего действия
VRFY	Проверяемое имя получателя
NOOP	Проверка состояния получателя
TURN	Смена положения отправителя и получателя (в настоящее время не используется)
EXPN	Почтовый список расширения
HELP	Имя команды
SEND FROM	Предназначенный получатель сообщения
SMOL FROM	Предназначенный получатель сообщения
SMAL FROM	Предназначенный получатель сообщения

Команды SMTP (табл. 6.1):
– **HELLO** — используется клиентом для идентификации самого себя; *аргумент* — доменное имя хоста клиента.
– **MAIL FROM** — используется клиентом для идентификации отправителя сообщения; *аргумент* содержит адрес отправителя электронной почты (локальную часть и доменное имя).

```
MAIL FROM: berlin@sut.ru
```

– **RCPT** — используется клиентом для идентификации получателя сообщения; *аргумент* — адрес электронной почты получателя (если имеется много получателей, команда повторяется).

```
RCPT TO: jenifer@edu.com
```

– **DATA** — используется для посылки сообщения; все строки, следующие за символами, рассматриваются как почтовое сообщение.
```
DATA
Текст сообщения
```
– **QUIT** — команда, завершающая соединение.
– **RSET** — команда, прерывающая текущее действие; накопленная информация об отправителе и получателе удаляется.
– **VRFY** — команда, используемая для верификации адреса получателя, который посылается как аргумент.
```
VRFY: jenifer@edu.com
```
– **NOOP** — используется клиентом для проверки состояния получателя и требует ответа от получателя.
– **TURN** — позволяет отправителю и получателю перейти в положение, при котором отправитель становится получателем, и наоборот.
– **EXPN** — запрашивает хост получателя, чтобы расширить список, как это указано в аргументах, и вернуть адреса почтовых ящиков получателей, которые включаются в этот лист.
```
EXPN: x,y,z
```
– **HELP** — запрашивает у получателя информацию о команде, содержащейся в аргументе.
```
HELP: mail
```
– **SEND FROM** — показывает, что почта может быть доставлена терминалу получателя, а не в почтовый ящик (если получатель не зарегистрирован, то почта возвращается обратно); *аргументом* является адрес отправителя.
```
SEND FROM: berlin@edu.com
```
– **SMOL FROM** — показывает, что почта может быть доставлена терминалу или в почтовый ящик получателя (если получатель зарегистрирован, то доставка осуществляется в терминал; иначе — только в почтовый ящик); *аргумент* — адрес отправителя.
```
SMOL FROM: berlin@edu.com
```
– **SMAL FROM** — показывает, что почта может быть доставлена терминалу получателя и в почтовый ящик (если получатель зарегистрирован, почта будет доставлена в почтовый ящик и терминал; иначе — только в почтовый ящик); *аргумент* — адрес отправителя.
```
SMAL FROM: berlin@edu.com
```

Отклики посылаются от сервера к клиенту и содержат три десятичных кода, которые могут дополняться текстовой информацией:
– **2yz** (положительное подтверждение завершения) — требуемая команда успешно завершена, и можно передавать следующую команду;
– **3yz** (положительное промежуточное подтверждение) — требуемая команда принята, но получатель нуждается в большей информации для завершения обработки;

– **4yz** (отрицательное переходное подтверждение) — требуемая команда должна быть отклонена, но состояние ошибки временное, команда может быть передана повторно;
– **5yz** (отрицательное постоянное подтверждение завершения) — требуемая команда должна быть отклонена, и не может быть передана повторно.

Вторые и третьи цифры сообщают дополнительные детали об отклике.

Положительное подтверждение завершения:
– 211 — системное состояние или отклик на справку;
– 214 — справочное сообщение;
– 220 — готовность к обслуживанию;
– 221 — завершение обслуживания передающего канала;
– 250 — требуемая команда завершена;
– 251 — пользователь не местный: сообщение должно быть передано далее.

Положительное промежуточное подтверждение:
– 354 — начало ввода почты.

Отрицательное переходное подтверждение:
– 421 — обслуживание не доступно;
– 450 — почтовый ящик недоступен;
– 451 — команда прервана: местная ошибка;
– 452 — команда прервана: недостаточно памяти.

Отрицательное постоянное подтверждение завершения:
– 500 — синтаксическая ошибка: неопознанная команда;
– 501 — синтаксическая ошибка в параметрах или аргументах;
– 502 — команда не выполнена;
– 503 — неправильная последовательность команд;
– 504 — команда временно не выполнена;
– 550 — команда не выполнена: почтовый ящик недоступен;
– 551 — пользователь не местный;
– 552 — требуемое действие прервано: переполнение местной памяти;
– 553 — требуемое действие не принято к исполнению: недопустимое имя почтового ящика;
– 554 — неудавшийся переход.

6.1.2. Процесс передачи почтовых сообщений

Процесс передачи почтовых сообщений осуществляется в три фазы: *установление соединения, передача почты* и *подключение оконечного устройства*.

Установление соединения

1. Сервер посылает код 220 (Готов к обслуживанию), чтобы сказать клиенту, что он готов принять почту. Если сервер не готов, то он посылает код 421 (Обслуживание не готово).

2. Клиент посылает сообщение HELLO, чтобы идентифицировать себя, используя доменное имя адреса.
3. Сервер отвечает кодом 250 (Требуемая команда завершена) или другим кодом в зависимости от ситуации.

Передача сообщения

1. Клиент посылает сообщение MAIL FROM, чтобы представить отправителю почтовый адрес отправителя (имя почтового ящика и доменное имя).
2. Сервер отвечает кодом 250 или другим кодом.
3. Клиент посылает сообщение RCPT TO (получатель), которое включает почтовый адрес получателя.
4. Сервер отвечает кодом 250 или другим сообщением.
5. Клиент посылает сообщение DATA, чтобы инициализировать передачу сообщений.
6. Сервер отвечает кодом 354 (Начало ввода почты) или другим сообщением.
7. Клиент посылает содержание сообщения в виде последовательности строк, завершаемых двумя символами конца строки (возврат каретки и продвижение на другую линию).
8. Сервер отвечает кодом 250 или соответствующим кодом.

Окончание соединения

1. Клиент посылает команду QUIT.
2. Сервер отвечает кодом 221 или соответствующим другим кодом.
3. После фазы окончания соединения TCP-соединение должно быть завершено.

6.2. Multipurpose Internet Mail Extensions

Определение 6.1. Многоцелевое расширение интернет-почты (Multipurpose Internet Mail Extensions, MIME) — дополняющий SMTP-протокол, позволяющий передавать сообщения, используя SMTP-данные, которые не имеют вид ASCII.

MIME преобразовывает данные, отличающиеся от ASCII, к виду ASCII и доставляет их клиенту SMTP через Интернет. Сервер SMTP на приёмной стороне получает данные в виде ASCII и доставляет к MIME, чтобы преобразовать данные в первоначальный вид.

MIME имеет пять заголовков для определения параметров преобразования:

– MIME–Version (MIME-версия);
– Content–Type (содержание – тип) — тип данных в используемом информационном блоке сообщения;

- Content–Transfer–Encoding (содержание – передача – кодирование) — определяет метод кодирования сообщения для передачи в виде нулей и единиц;
- Content–Id (содержание – идентификатор) — идентифицирует полное сообщение;
- Content–Description (содержание – описание) — определяет, является ли информационный блок изображением, аудио- или видеоинформацией.

Типы и подтипы MIME:

- текст — неформатированный текст;
- из многих частей:
  - смешанный — упорядоченные типы данных;
  - параллельный — неупорядоченные типы данных;
  - обзорный — упорядоченные сообщения;
  - альтернативный — разрозненные сообщения;
- сообщение:
  - RFC822 — сообщение;
  - частичный — фрагмент сообщения;
  - внешний блок — ссылка на другое сообщение;
- изображение:
  - JPEG;
  - GIF;
  - MPEG;
- аудио / видео;
- прикладной текст:
  - PostScript;
  - поток октетов.

Типы кодирования в заголовке Content–Transfer–Encoding:

- 7 бит — NVT ASCII-символы и короткие линии;
- 8 бит — не-ASCII-символы и короткие линии;
- двоичный — не-ASCII-символы с не лимитированной длиной линии;
- базовый 64 — 6-битовые блоки данных, закодированные по 8 бит ASCII-символами;
- предназначенный для печати — не-ASCII-символы, закодированные как последовательность знаков ASCII.

6.3. POP3

Post Office Protocol, Version 3 (POP3) — один из протоколов почтового доступа, состоящий из клиента POP3, устанавливаемого на компьютере получателя, и POP3-сервера, устанавливаемого на почтовом сервере.

Схема работы POP3:
- почтовые сообщения принимаются почтовым сервером и сохраняются там, пока на рабочей станции клиента не будет запущено приложение POP3;
- приложение устанавливает соединение с сервером и забирает сообщения оттуда;
- почтовые сообщения на сервере стираются.

Принцип работы POP3:
- POP3-сервер прослушивает TCP-порт 110;
- по установлении связи POP3-сервер посылает клиенту уведомление (например, +OK POP3 server ready) и сессия переходит в фазу авторизации;
- в процессе авторизации клиент должен представить себя серверу, передав имя и пароль;
- если авторизация успешно завершена, сессия переходит в состояние транзакции (TRANSACTION);
- производиться обмен командами и откликами;
- при получении от клиента команды QUIT сессия переходит в состояние UPDATE, при этом все ресурсы освобождаются и TCP-связь разрывается.

Команды POP3 состоят из ключевых слов (3–4 символа), за которыми могут следовать *аргументы*.
- Каждая команда завершается парой символов CRLF.
- Как ключевые слова, так и аргументы могут содержать только печатаемые ASCII-символы.
- В качестве разделителя используются символы пробела.
- Каждый аргумент может содержать до 40 символов.

Сигнал отклика в POP3 содержит *индикатор состояния* и *ключевое слово*, за которым может следовать дополнительная информация.
- Отклик завершается кодовой последовательностью CRLF.
- Длина отклика не превышает 512 символов, включая CRLF.
- Существует два индикатора состояния: *положительный* — «+OK» и *отрицательный* — «-ERR».
- Отклики на некоторые команды могут содержать несколько строк.

В состоянии транзакции клиент может посылать серверу последовательность POP3-команд, на каждую из которых сервер должен послать отклик:
- **STAT** — используется для просмотра состояния текущего почтового ящика; в ответ возвращается строка, содержащая количество и общий размер в байтах сообщений, которые клиент может получить с POP3-сервера;
- **LIST [msg]** — в ответ возвращается информационная строка, содержащая номер и размер сообщения, или список информационных строк обо всех сообщениях в данном почтовом ящике;

- **RETR msg** — используется для передачи клиенту запрашиваемого сообщения;
- **DELE msg** — используется для маркировки сообщения на удаление, которое реально удаляется только после закрытия транзакции на стадии UPDATE;
- **NOOP** — используется для проверки состояния соединения с POP3-сервером;
- **RSET** — используется для отката транзакции внутри сессии;
- **TOP msg n** — POP3-сервер по запросу клиента отправляет заголовок сообщения, затем пустую строку и далее требуемое количество строк сообщения;
- **USER name** — передаёт серверу имя пользователя;
- **PASS string** — передаёт серверу пароль почтового ящика;
- **QUIT** — закрытие POP3-сессии.

6.4. IMAP

Протокол **IMAP4 (Internet Message Access Protocol)** позволяет клиентам получать доступ и манипулировать сообщениями электронной почты на сервере.

Отличием протокола IMAP4 от протокола POP3 является то, что IMAP4 поддерживает работу с системой каталогов (папок) удалённых сообщений так же, как если бы они располагались на локальном компьютере.

IMAP4 позволяет:

- клиенту создавать, удалять и переименовывать почтовые ящики;
- проверять наличие новых сообщений и удалять старые;
- читать из почтового ящика только сообщения, удовлетворяющие определённым условиям или их части, менять атрибуты сообщений и перемещать отдельные сообщения.

Протокол IMAP4 работает поверх транспортного протокола, который обеспечивает надёжный и достоверный канал передачи данных между клиентом, и сервером IMAP4.

При работе по TCP, IMAP4 использует 143-й порт.

Принцип передачи данных IMAP4:

- клиент и сервер обмениваются приветствиями;
- клиент отправляет на сервер команды и данные;
- сервер передаёт клиенту ответы на обработку команд и данных
- после завершения обмена канал закрывается.

Каждая команда клиента начинается с **идентификатора (тега)** команды. Тег, как правило, представляет собой короткую строку, состоящую из букв и цифр. Тег является уникальным идентификатором команды клиента. Ответы сервера или следующие команды клиента могут ссылаться на данную команду по её тегу.

Строки данных, передаваемые с сервера в ответ на команду клиента, могут содержать не тег, а **символ «*»**. Это означает, что они являются промежуточными строками потока данных ответа, а идентификатор их команды содержится в последней строке потока.

Если сервер обнаружил ошибку в команде, он отправляет **уведомление BAD** клиенту с тегом неправильной команды.

Если команда успешно обработана, то возвращается **уведомление OK** с тегом команды.

Если команда вернула отрицательный результат, например, в случае невозможности выполнить данную команду, то возвращается **уведомление NO** с тегом невыполненной команды.

Каждое сообщение в почтовой системе для работы с IMAP имеет **уникальный идентификатор**, по которому можно получить доступ к нему. Уникальный идентификатор UID представляет собой 32-битное число, которое идентифицирует сообщение в данной папке. Каждому сообщению, попавшему папку, присваивается максимальное число из UID-сообщений, попавших в данную папку ранее. Уникальные идентификаторы сообщений сохраняются от сессии к сессии и могут использоваться, например, для синхронизации каталогов мобильных пользователей.

Все сообщения в данном почтовом ящике **последовательно нумеруются**. При добавлении в почтовый ящик нового сообщения ему присваивается номер на 1 больше количества сообщений в почтовом ящике. При удалении какого-либо сообщения из данной папки порядковые номера всех сообщений пересчитываются. Поэтому порядковый номер сообщения может меняться во время сессии.

Помимо числовых идентификаторов сообщениям назначаются **флаги**:

- **Seen** — данное сообщение было прочитано;
- **Answered** — на сообщение был дан ответ;
- **Deleted** — сообщение помечено на удаление;
- **Draft** — формирование данного сообщения не завершено;
- **Recent** — сообщение «только что» поступило в почтовый ящик, т.е. данная сессия — первая, которая может прочитать это сообщение.

Одни флаги могут быть действительны для данного сообщения постоянно от сессии к сессии, другие — только для данной сессии.

Команды IMAP:

- **LOGIN** — передаёт серверу пароль и идентификатор пользователя для регистрации пользователя в системе (аргумент — строка с идентификатором и паролем клиента);
- **SELECT** — выбор каталога (папки) сообщений, с которым будет работать клиент (аргумент — имя почтового каталога);
- **EXAMINE** — открывает заданный почтовый ящик исключительно на чтение;

- **STATUS** — запрос статуса какой-либо папки (параметры — имя папки, тип запрашиваемой информации);
- **LIST** — получить список папок (подкаталогов), находящихся в определённой папке и доступных клиенту (аргументы — имя каталога, список подкаталогов, маска имён подкаталогов);
- **FETCH** — прочитать любое сообщение или определённую группу сообщений, часть сообщения или определённые атрибуты сообщения (аргументы — порядковый номер сообщения, критерии запроса (описание вида возвращаемой информации));
- **STORE** — сохранить сообщение с другими флагами, добавить или удалить флаги сообщения (аргументы — номера сообщений, идентификатор операции, перечень флагов);
- **SEARCH** — организовать поиск сообщений по определённым критериям;
- **APPEND** — добавить сообщение в каталог;
- **COPY** — копирует сообщения с заданными порядковыми номерами в указанный каталог.

Глава 7. Простой протокол управления сетью SNMP

Определение 7.1. *Простой протокол управления сетью (Simple Network Management Protocol, SNMP)* — структура для управления устройствами в сети Интернет (например, мостами, маршрутизаторами и другими сетевыми устройствами) с использованием набора протоколов TCP/IP.

7.1. Концепция SNMP

Концепция SNMP:
- используется концепция менеджера (управляющий хост) и агента (устройства Интернета);
- одна или несколько станций менеджера управляют набором агентов;
- SNMP освобождает задачи управления и от физических характеристик управляемых устройств, и от основной технологии организации сети.

Менеджеры и агенты
- Станция управления (менеджер) является хостом, который выполняет SNMP-программу клиента.
- Управляемая станция (агент) является, например, маршрутизатором (или хостом), выполняющим SNMP-программу сервера.
- Управление достигается с помощью простого взаимодействия между менеджером и агентом.
- Агент сохраняет характеристики информации в базе данных.
- Менеджер имеет доступ к содержимому базы данных.

Управление с SNMP базируется на трёх основных идеях:
1) менеджер проверяет агента, запрашивая информацию, отражающую поведение агента;
2) менеджер вынуждает агента выполнить задачу повторно, переустановив значения базы данных агента;
3) агент вносит вклад в процесс управления, предупреждая менеджера о необычной ситуации.

Управление компонентами сети Интернет осуществляется при взаимодействии трёх элементов:
1) структуры управляющей информации (Structure of Management Information, **SMI**);
2) базы управляющей информации (Management Information Base, **MIB**);
3) протокола **SNMP**.

SMI определяет правила формального описания объектов, но *не задаёт* конкретную архитектуру набора объектов: сколько объектов управляется данным объектом или какой объект используется каким типом.

MBI представляет информационную базу для SNMP об управляемых и контролируемых объектах сети, в которой информация об объектах сформирована по категориям.

Для каждого управляемого объекта протокол MIB должен определить число объектов, присвоить им имена согласно правилам, определённым SMI, и присвоить тип каждому из названных объектов.

SNMP:

- определяет состояния устройств сети и сетевого трафика;
- определяет содержание формата пакета, который будет послан от менеджера агенту и наоборот;
- интерпретирует и высылает результаты статистики характеристик управляемых объектов (часто с помощью другого программного обеспечения управления);
- отвечает за чтение и изменение передаваемых значений в процессе управления.

Общий процесс управления (рис. 7.1):

1) менеджер станции (SNMP-клиент) посылает сообщение к агенту (SNMP-серверу), чтобы получить от агента число UDP-дейтаграмм;
2) MIB находит объект, чтобы сохранить полученное в сообщении число пользовательских дейтаграмм UDP;
3) SMI с помощью других вложенных в него протоколов кодирует имя объекта;
4) SNMP создаёт сообщение, называемое GetRequest, и инкапсулирует его в закодированное сообщение.

7.2. Структура управляющей информации (SMI)

Функции SMI:

- присвоить имена объектам;
- определить тип данных, которые могут быть сохранены в объекте;
- показать, как кодировать данные для передачи по сети.

Атрибуты SMI, отвечающие за обработку объекта:

- имя — иерархический идентификатор объекта;
- тип данных — тип сохраняемых в объекте данных;
- метод кодирования данных.

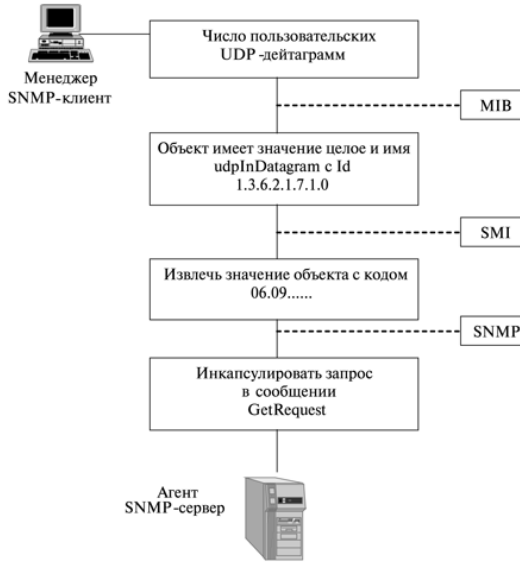

Рис. 7.1. Общий процесс управления SNMP

7.2.1. Имя

SMI использует идентификатор объекта, который является иерархическим и основан на структуре дерева. Структура дерева начинается с корня, не имеющего имени (рис. 7.2). Каждый объект определяется последовательностью целых чисел, разделённых точками:
`iso.org/dot/internet.mgmt.mib-2 <--> 1.3.6.1.2.1`

7.2.2. Тип данных

Для обработки типов данных SMI использует фундаментальные ASN.1-определения, которые могут быть дополнены новыми определениями.

SMI использует две категории типов данных: *простые* (табл. 7.1) и *структурированные*. При этом SMI определяет **два вида структурированных типов данных**: *sequence (последовательности)* и *sequence of (последовательности из)*.

Тип данных **sequence (последовательности)** — комбинация простых типов данных, необязательно одного типа (аналог структуры).

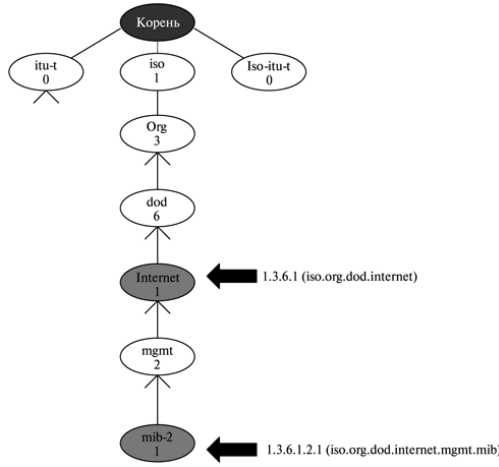

Рис. 7.2. Идентификация объекта SMI

Тип данных **sequence of (последовательность из)** — комбинация из простых типов данных одного типа или комбинации последовательного типа данных одного типа (аналог массива).

7.2.3. Метод кодирования данных

Для кодирования данных, передаваемых по сети, SMI использует стандарт **BER (Basic Encoding Rules)**. Каждая часть данных кодируется в формате тройки: **тег**, **длина** и **значение** (рис. 7.3).

Рис. 7.3. Формат кодирования данных в SMI

Тег — однобайтовое поле, определяющее тип данных и состоящее из трёх подполей: *класс* (2 бита), *формат* (1 бит) и *номер* (5 битов).

Таблица 7.1

Тип данных SMI

Тип	Размер	Описание
INTEGER	4 байта	Целое со значением между 0 и 231-1
Integer 32	4 байта	Целое со значением между 0 и 231-1
Unsigned32	4 байта	Значения без знака между 0 и 231
OCTET STRING	Переменный	Строка байтов не более 65 535 байт длины
OBJECT IDENTIFIER	Переменный	Идентификатор объекта
IPAdress	4 байта	IP-адрес, состоящий из четырёх байт
Counter32	4 байта	Целое, значение которого может быть увеличено от 0 до 232; когда оно достигает максимального значения, свёртывается в 0
Counter64	8 байтов	64-битовый счётчик
Gauge32	4 байта	32-битовый счётчик (counter32), но достигающий максимального значения и не сворачивающийся в 0
TimeTics	4 байта	Считает значение, в котором записано время в 1/100 секунды
BITS		Строка бит
Opaque	Переменный	Неинтерпретируемая строка

Подполе *класс* определяет область действия данных. Определены *четыре класса*: универсальный (00) (типы данных из ASN.1), прикладной (01) (типы данных, которые добавлены SMI), контекстно-определённый (10) (типы данных имеют значения, которые могут измениться от одного протокола к другому) и частный (11) (типы данных определяются поставщиком).

Подполе *формат* указывает, являются ли данные *простыми (0)* или *структурированными (1)*.

Подполе *номер* делит простые или структурированные данные на подгруппы.

Поле *длина* — один или более байт. Если поле длины — 1 байт, то старший знаковый бит должен быть равен 0, а другие 7 бит определяют

данные. Если поле длины больше чем 1 байт, то старший знаковый бит первого байта должен быть равным 1, а другие 7 бит первого байта определяют число байт, нужных для определения длины.

Поле *значение* кодирует значение данных в соответствии с правилами, определёнными в правилах кодирования (например, ANSI).

7.3. База управляющей информации (MIB)

Каждый агент имеет свою собственную базу MIB, являющуюся отображением всех объектов, которыми может управлять менеджер.

Объекты в MIB разбиты по категориям на 10 групп: система (sys), интерфейс (if), адрес трансляции (at), ip, icmp, tcp, udp, egp, trans и snmp (рис. 7.4).

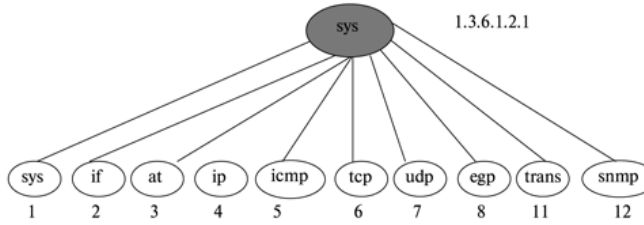

Рис. 7.4. Дерево объектов идентификации MIB

В группу *система* входит название и версия оборудования, операционной системы, сетевого программного обеспечения и пр.

В группу *интерфейсы* входит число поддерживаемых интерфейсов, тип интерфейса, работающего под IP (Ethernet, LAPB etc.), размер дейтаграмм, скорость обмена, адрес интерфейса.

IP-группа включает в себя время жизни дейтаграмм, информацию о фрагментации, маски субсетей и т.д.

В *TCP-группу* входит алгоритм повторной пересылки, максимальное число повторных пересылок и пр.

Чтобы организовать доступ любой простой переменной, используется групповой id, сопровождаемый id переменной.

Чтобы описать представителя или содержимое каждой переменной, надо добавить суффикс экземпляра — ноль.

Чтобы идентифицировать таблицу, также используется id, определяется вход (последовательность) в таблице и каждый объект входа (переменные в листе дерева). Чтобы обратиться к определённому образцу (строке) таблицы, надо добавить индекс к вышеупомянутым id. Индексы базируются на значении одной или более областей во входах.

Идентификаторы объекта (включая идентификаторы образца) находятся в **лексикографическом порядке**.

Таблицы упорядочиваются в соответствии с правилами строки-столбца, и это означает, что нужно идти от столбца к столбцу, а в каждом столбце нужно идти от вершины к основанию.

Лексикографическое упорядочение делается менеджером для того, чтобы организовать доступы к набору переменных один за другим, задавая первую переменную.

7.4. Протокол SNMP

Команды SNMP:

- GetRequest (PDU 0) — посылается от менеджера к агенту, чтобы извлечь значение переменной или набора переменных;
- GetNextRequest (PDU 1) — посылается от менеджера к агенту, чтобы извлечь значение переменной; найденное значение — значение объекта, следующего после определённого в таблице Objectid в PDU (используется, чтобы извлечь значения входов в таблице);
- GetBulkRequest (PDU 2) — посылается от менеджера к агенту, чтобы извлечь большое количество данных;
- SetRequest (PDU 3) — посылается от менеджера к агенту, чтобы установить (сохранить) значение в переменной;
- Response (PDU 4) — посылается от агента к менеджеру в ответ на GetRequest или GetNextRequest и содержит значение(я) переменной(ых), которую запрашивает менеджер;
- Trap (PDU 5) — посылается от агента к менеджеру, чтобы сообщить о событии, например, если агент перезагружен, он информирует менеджера и сообщает время перезагрузки;
- Inform Request (PDU 6) — посылается от одного менеджера другому удалённому менеджеру, чтобы получить значение некоторых переменных от агентов, которые управляются удалённым менеджером;
- SNMPv3-Trap (PDU 7) — отклик на событие (расширение по отношению v1 и v2);
- Report (PDU 8) — разработан, чтобы сообщить о некоторых типах ошибок между менеджерами.

SNMP не посылает отдельные PDU, а включает PDU в сообщение (рис. 7.5).

Сообщение в SNMPv3 состоит из четырёх элементов: *версия, заголовок, параметры защиты* и *данные* (которые включают кодируемый PDU). Поскольку длина этих элементов отличается от сообщения к сообщению, SNMP применяет основные правила кодирования — BER, чтобы кодировать каждый элемент.

Версия определяет текущую версию.

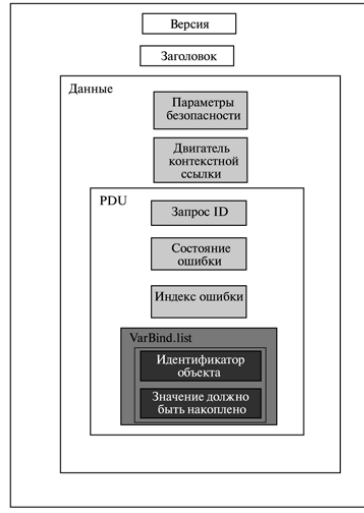

Рис. 7.5. SNMP-сообщение

Заголовок содержит значения для идентификации сообщения, максимальный размер сообщения (максимальный размер ответа), флаг сообщения (один октет типа данных OCTET STRING, где каждый бит определяет тип защиты, тип секретности или идентификации либо другую информацию) и модели обеспечения безопасности (определение протокола защиты).

Параметр защиты сообщения используется для создания дайджеста сообщения.

Данные содержат PDU. Если данные зашифрованы, то есть информация об источнике шифровки (программе-менеджере, которая зашифровала сообщение) и контекст шифровки (тип кодирования), сопровождаемый зашифрованным PDU. Если данные не зашифрованы, то они состоят только из PDU.

Формат SNMP-сообщений, вкладываемых в UDP-дейтаграммы, приведён на рис. 7.6.

Поле *Версия* содержит значение, равное номеру версии SNMP минус один.

Поле *Пароль* (community — определяет группу доступа) включает в себя последовательность символов, которая является пропуском при взаимодействии менеджера и объекта управления.

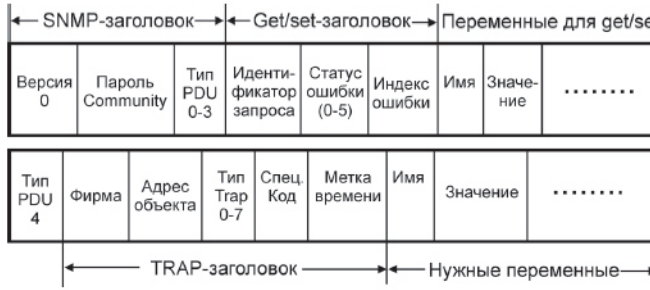

Рис. 7.6. Формат SNMP-сообщений, вкладываемых в UDP-дейтаграммы

Для запросов GET, GET-next и SET значение идентификатора запроса устанавливается менеджером и возвращается объектом управления в отклике GET, что позволяет связывать в пары запросы и отклики.

Поле *Фирма (enterprise)* — sysobjectid объекта.

Поле *Статус ошибки* характеризуется целым числом, присланным объектом управления (табл. 7.2).

Поле *Индекс ошибки (error index)* является указателем переменной и устанавливается объектом управления не равным нулю для ошибок badvalue, readonly и nosuchname.

Для *типа TRAP* 0-4 поле *Специальный код* должно быть равно нулю (табл. 7.3).

Поле *Временная метка* содержит число сотых долей секунды (число тиков) с момента инициализации объекта управления.

7.5. Нотация ASN.1

ASN.1 представляет собой язык описания типов данных и их значений. В общем виде такое описание имеет вид:

```
data type value name | data type identifier ::= data value
или {data type identifier (data value)}
```

Коме того, ASN.1 является языком программирования, который служит для описания MIB и может использоваться для модификации существующей или создания новой базы данных MIB для SNMP.

Список наиболее распространённых ключевых слов ASN.1 для описания MIB SNMP: BEGIN, IDENTIFIER, END, OCTETS, STRING, SEQUENCE, INTEGER, STRING, OBJECT,OF, NULL, DEFINITIONS DESCRIPTION.

Таблица 7.2

Коды ошибок SNMP

Статус ошибки	Имя ошибки	Описание
0	Noerror	Все в порядке
1	Toobig	Объект не может уложить отклик в одно сообщение
2	Nosuchname	В операции указана неизвестная переменная
3	badvalue	В команде set использована недопустимая величина или неправильный синтаксис
4	Readonly	Менеджер попытался изменить константу
5	Generr	Прочие ошибки

Объекты MIB создаются макросами ASN.1. Макрос OBJECT-TYPE имеет формат:

```
data type value name OBJECT-TYPE
SYNTAX data type identifier
UNITS
ACCESS
STATUS
DESCRIPTION
REFERENCE
INDEX
DEFVAL
::= {data value}
```

– SYNTAX — определяет тип данных (простой, структурированный и т.д);
– ACCESS — задаёт уровень доступа к объекту (read only, read & write);
– STATUS — определяет статус описания объекта (текущий, устаревший и пр.);
– DESCRIPTION — описывает роль или функцию управляемого объекта и способы его применения;
– REFERENCE — опционное описание, характеризующее наследование объекта (указывает на родительский объект);
– INDEX — работает в случае, когда объект содержит список или таблицу, и позволяет указать позицию в списке или таблице;

Таблица 7.3

Коды TRAP SNMP

Тип TRAP	**Имя TRAP**	**Описание**
0	Coldstart	Установка начального состояния объекта
1	Warmstart	Восстановление начального состояния объекта
2	Linkdown	Интерфейс выключился. Первая переменная в сообщении идентифицирует интерфейс
3	Linkup	Интерфейс включился. Первая переменная в сообщении идентифицирует интерфейс
4	Authenticationfailure	От менеджера получено snmp-сообщение с неверным паролем (community)
5	EGPneighborloss	EGP-партнёр отключился. Первая переменная в сообщении определяет ip-адрес партнёра
6	Entrprisespecific	Информация о TRAP содержится в поле *Специальный код*

– DEFVAL — опционная характеристика, указывающая значение объекта по умолчанию.

Базовые правила обозначений метасинтаксиса ASN.1 приведены в табл. 7.4.

ASN.1 имеет четыре разновидности типов: простые типы, не имеющие компонент, структурные типы, имеющие компоненты, помеченные (tagged) типы, которые получаются из других типов, а также прочие типы, которые включают в себя типы CHOICE и ANY. Типам и значениям могут присваиваться имена с помощью оператора (::=). Эти имена в дальнейшем могут использоваться для определения других типов и величин.

Все типы ASN.1 кроме CHOICE и ANY имеют метки, которые состоят из класса и неотрицательного кода метки (табл. 7.5). Типы ASN.1 тождественны, если их числовые метки совпадают.

Существует четыре класса меток:

Таблица 7.4
Базовые правила обозначений метасинтаксиса ASN.1

Обозначение	Описание
n	(полужирный курсив) обозначает переменную
[ ]	(квадратные скобки, напечатанные полужирным шрифтом) указывают на то, что терм является опционным
{ }	(фигурные скобки, напечатанные полужирным шрифтом) группируют родственные термы
|	(вертикальная черта, напечатанная полужирным шрифтом) выделяет альтернативные значения
...	(многоточие, напечатанное полужирным шрифтом) обозначает повторения
=	(знак равенства, напечатанный полужирным шрифтом) описывает терм как субтерм

– universal — для типов, значения которых являются неизменными для всех приложений;
– application — для типов со значением, специфическим для приложений, таких как служба каталогов X.500; типы двух разных приложений могут иметь одну и ту же метку и разные значения;
– private для типов, которые являются специфическими для данного предприятия;
– content-specific для типов со значением, специфическим для данного структурного типа.

В ASN.1 определено четыре структурированных типа:
– SEQUENCE — упорядоченный набор из одного или более типов;
– SEQUENCE OF — упорядоченный набор из нуля или более представителей данного типа;
– SET — неупорядоченный набор из одного или более типов;
– SET OF — неупорядоченный набор из представителей данного типа.

Структурированные типы могут иметь компоненты, в том числе со значениями по умолчанию.

Существуют типы, помеченные явно и неявно. Неявно помеченные типы получаются из других типов путём изменения метки. Явно поме-

Таблица 7.5

Типы и их метки ASN.1

Тип	**Комментарий**	**Цифровая метка (шестнадцатеричная)**
INTEGER	Любое целое число	02
BIT STRING	Произвольная строка бит	03
OCTET STRING	Произвольная последовательность октетов	04
NULL	0	05
OBJECT IDENTIFIER	Последовательность целых компонент, идентифицирующих объект	06
SEQUENCE and SEQUENCE OF		10
SET and SET OF		11
PrintableString	Последовательность печатных символов	13
IA5String	Произвольная строка символов IA5 (ASCII)	16
UTCTime	Универсальное время (по Гринвичу; GMT)	17

ченные типы получаются из других типов путём добавления внешней метки. Помеченный явно тип — структурированный тип, состоящий из одного компонента основного типа.

Глава 8. Язык гипертекстовой разметки HTML: основные теги, таблицы стилей

8.1. Структура документа HTML

Определение 8.1. *HyperText Markup Language (HTML)* — язык разметки гипертекста.

Определение 8.2. *Гипертекстовый документ* — текстовый файл, имеющий специальные метки, называемые **тегами**, которые впоследствии опознаются браузером и используются им для отображения содержимого файла на экране компьютера.

Определение 8.3. *Гиперссылка* — специальная конструкция языка HTML, позволяющая щелчком мыши перейти к просмотру другого документа.

Для файлов, содержащих HTML-документы, приняты расширения **.htm** или **.html**.

Определение 8.4. *Тег* — специальная конструкция языка HTML, используемая для разметки документа и управляющая его отображением.

Прописные и строчные буквы при записи тегов не различаются. В большинстве случаев теги используются парами. Пара состоит из открывающего (start tag) и закрывающего (end tag) тегов. Для некоторых атрибутов значение может не указываться. У закрывающего тега атрибутов не бывает. Последовательность символов, составляющая текст, может состоять из пробелов, табуляций, символов перехода на новую строку, символов возврата каретки, букв, знаков препинания, цифр, и специальных символов (например, +, #, $,), за исключением символов, имеющих в HTML специальный смысл: < (меньше), > (больше), & (амперсенд) и "(двойная кавычка).

Синтаксис открывающего и закрывающего тегов:

```
<имя_тега [атрибуты]>
</имя_тега>
```

Имя закрывающего тега отличается от имени открывающего лишь тем, что перед ним ставится наклонная черта.

Атрибуты тега записываются в следующем формате:

```
имя["значение"]
```

Прямые скобки, используемые в описании синтаксиса, означают, что данный элемент может отсутствовать. Кавычки при задании значения аргумента не обязательны и могут быть опущены.

Структура документа HTML:
- информация о типе документа (**Document Type Definition, DTD**);
- заголовок документа (ограничен элементом HEAD);
- тело документа (ограничен элементом BODY)

Пример 8.1.
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
       ...
  </head>
  <body>
       ...
  </body>
</html>
```

Элемент `<!DOCTYPE>` предназначен для указания типа текущего документа — **DTD (Document Type Definition)**; имеет следующий синтаксис:

```
<!DOCTYPE [Элемент верхнего уровня] [Публичность]
 "[Регистрация]//[Организация]//[Тип] [Имя]//[Язык]"
 "[URL]">
```

Параметры:
- *элемент верхнего уровня* — указывает элемент верхнего уровня в документе (для HTML — тег `<html>`);
- *публичность* — указывает, каким является объект: *публичным* (значение PUBLIC) или *системным ресурсом* (значение SYSTEM), например, локальным файлом (для HTML/XHTML — значение PUBLIC);
- *регистрация* — сообщает, что разработчик DTD зарегистрирован в ISO, принимает одно из двух значений: *плюс (+)* — разработчик зарегистрирован в ISO и *минус (−)* — разработчик не зарегистрирован;
- *организация* — уникальное название организации, разработавшей DTD (официально HTML/XHTML публикует W3C, что и указываетя в `<!DOCTYPE>`);
- *тип* — тип описываемого документа (для HTML/XHTML — DTD);
- *имя* — уникальное имя документа для описания DTD;
- *язык* — язык, на котором написан текст для описания объекта. Содержит две буквы, пишется в верхнем регистре (для документа HTML/XHTML — английский язык (EN));
- *URL* — адрес документа с DTD.

Пример 8.2.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
```

Заголовок документа содержит информацию о документе (метаданные) и имеет следующий синтаксис:

```
<head>
   ...
</head>
```

Задача заголовка — представление необходимой информации для браузера и сервера HTTP. Содержимое тега `<head>` не отображается напрямую на веб-странице, за исключением тега `<title>`. Внутри `<head>` допускается размещать элементы `<base>`, `<basefont>`, `<bgsound>`, `<link>`, `<meta>`, `<script>`, `<style>`, `<title>`:

Тег `<title>` определяет заголовок документа.

Тег `<base>` определяет полный адрес документа, используется для корректного определения относительных адресов в документе.

Тег `<basefont>` предназначен для задания шрифта, размера и цвета текста по умолчанию. Указанные значения будут использоваться во всем документе за исключением тега `<font>`, в котором можно переопределить параметры оформления текста.

Тег `<bgsound>` определяет музыкальный файл, который будет проигрываться на веб-странице при ее открытии. Громкость, продолжительность звучания музыки и другие характеристики определяются с помощью атрибутов тега, а также могут управляться через скрипты.

Тег `<link>` устанавливает связь с внешним документом, например, со стилевым файлом или файлом, содержащим шрифты.

Атрибуты `<link>`:

- charset — кодировка связываемого документа;
- href — путь к связываемому файлу;
- media — определяет устройство, для которого следует применять стилевое оформление;
- rel — определяет отношения между текущим документом и файлом, на который делается ссылка;
- sizes — указывает размер иконок для визуального отображения;
- type — MIME-тип данных подключаемого файла.

Синтаксис:

```
<LINK [REL=тип_отношения]
      [HREF=URL]
      [TYPE=тип_содержания]
>
```

Пример 8.3.

```
<LINK REL=stylesheet href="../css/style.css"
      TYPE="text/css">
```

Тег <meta> определяет метатеги, которые используются для хранения информации предназначенной для браузеров и поисковых систем.

Атрибуты <meta>:
- charset — задает кодировку документа;
- content — устанавливает значение атрибута, заданного с помощью *name* или *http-equiv*;
- http-equiv — предназначен для конвертирования метатега в заголовок HTTP;
- name — имя метатега, также косвенно устанавливает его предназначение.

Синтаксис:

```
<META [name=имя]
[HTTP-EQUIV=имя_HTTP-оператора]
CONTENT=текст
>
```

Пример 8.4. Запрет кэширования

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

Пример 8.5. Запретить хранение документа после пересылки

```
<META HTTP-EQUIV="Cache-Control" CONTENT="no-store">
```

Тег <script> предназначен для описания скриптов, может содержать ссылку на программу или ее текст на определённом языке.

Атрибуты <script>:
- defer — откладывает выполнение скрипта до тех пор, пока вся страница не будет загружена полностью;
- language — устанавливает язык программирования, на котором написан скрипт;
- src — адрес скрипта из внешнего файла для импорта в текущий документ;
- type — определяет тип содержимого тега <script>.

Синтаксис:

```
<SCRIPT [TYPE=тип_языка_сценариев]>
  JavaScript / VBScript - код
</SCRIPT>
```

или

```
<SCRIPT [TYPE=тип_языка_сценариев] [SRC=URL]>
</SCRIPT>
```

Тег <style> применяется для определения стилей элементов вебстраницы.

Атрибуты <style>:
- media — определяет устройство вывода, для работы с которым предназначена таблица стилей;

– type — сообщает браузеру, какой синтаксис использовать, чтобы правильно интерпретировать стили.

Синтаксис:

```
<STYLE TYPE=тип_описания_стилей>
  описание стиля/стилей
</STYLE>
```

Тело документа

Элемент <body> предназначен для хранения содержания веб-страницы (контента), отображаемого в окне браузера:

– иерархических контейнеров и заставок;
– заголовков;
– блоков (параграфы, списки, формы, таблицы, картинки и т.п.);
– горизонтальных отчеркиваний и адресов;
– текста, разбитого на области действия стилей (подчеркивание, выделение, курсив);
– математических описаний, графики и гипертекстовых ссылок.

Синтаксис:

```
<body>
  ...
</body>
```

Атрибуты <body>:

– background — задает фоновый рисунок на веб-странице;
– bgcolor — цвет фона веб-страницы;
– link — цвет ссылок на веб-странице;
– alink — цвет активной ссылки;
– vlink — цвет посещенных ссылок;
– text — цвет текста в документе;
– bgproperties— определяет, прокручивать фон совместно с текстом или нет;
– scroll — устанавливает, отображать полосы прокрутки или нет;
– bottommargin — отступ от нижнего края окна браузера до контента;
– topmargin — отступ от верхнего края окна браузера до контента;
– leftmargin — отступ по горизонтали от левого края окна браузера до контента;
– rightmargin — отступ от правого края окна браузера до контента;

Пример 8.6.

```
<HTML>
  <BODY BACKGROUND="image.jpg"  BGCOLOR="black"
        TEXT="#FFFFFF" LINK="#FF0000" VLINK="#656565"
        MARGINHEIGHT="25" TOPMARGIN="25"
        LEFTMARGIN="45" MARGINWIDTH="45">
    Текст документа.
  </BODY>
</HTML>
```

Универсальные атрибуты (применяются практически ко всем тегам, поэтому выделены в отдельную группу, чтобы не повторять их для всех тегов):
- accesskey — позволяет получить доступ к элементу с помощью заданного сочетания клавиш;
- contenteditable — сообщает, что элемент доступен для редактирования пользователем;
- contextmenu — устанавливает контекстное меню для элемента;
- class — определяет имя класса, которое позволяет связать тег со стилевым оформлением;
- dir — задает направление и отображение текста (слева направо или справа налево);
- id — указывает имя стилевого идентификатора;
- hidden — скрывает содержимое элемента от просмотра;
- lang — браузер использует значение параметра для правильного отображения некоторых национальных символов;
- spellcheck — указывает браузеру, проверять или нет правописание и грамматику в тексте;
- style — применяется для определения стиля элемента с помощью правил CSS;
- tabindex — устанавливает порядок получения фокуса при переходе между элементами с помощью клавиши Tab;
- title — описывает содержимое элемента в виде всплывающей подсказки.

Теги управления разметкой
- **Заголовки разного уровня**, показывающие относительную важность раздела документа
  Синтаксис:
  ```
  <h1>Заголовок первого уровня</h1>
  ...
  <h6>Заголовок шестого уровня</h6>
  ```
  Атрибут:
  - align — определяет выравнивание заголовка. Возможные значения атрибута: justify, left, right, center
- **Текстовый абзац** Синтаксис:
  ```
  <p>Текст</p>
  ```
  Атрибут:
  - align — определяет выравнивание текста. Возможные значения атрибута: justify, left, right, center
- **Принудительный перевод строки** Синтаксис:
  ```
  Текст<br>
  Текст<br>
  ```
  Атрибут:
  - clear — сообщает браузеру, как обрабатывать следующую строку, если текст обтекает плавающий элемент.

– **Списки:**
  – нумерованный список;
```
<OL>
  <LI>элемент нумерованного списка</LI>
  <LI>элемент нумерованного списка</LI>
</OL>
```
  Атрибуты `<ol>`:
  – type — вид маркера списка;
  – reversed — нумерация в списке по убыванию (3, 2, 1);
  – start — задаёт число, с которого будет начинаться нумерованный список.

  Атрибуты `<li>`:
  – type — вид маркера нумерованного или маркированного списка;
  – value — число, с которого будет начинаться нумерованный список;
  – маркированный список (атрибут: type — вид маркера списка):
```
<UL>
  <LI>элемент маркированного списка</LI>
  <LI>элемент маркированного списка</LI>
</UL>
```
  – список определений:
```
<DL>
  <DT>Термин</DT> <DD>Определение</DD>
</DL>
```
– **Комментарии**

  Синтаксис:
```
<!-- текст -->
```
  Содержимое комментариев браузер не выводит на экран.
– **Гипертекстовые ссылки**

  Синтаксис:
```
<a href="URL">...</a>
<a name="идентификатор">...</a>
```
  Атрибуты `<a>`:
  – accesskey — активация ссылки с помощью комбинации клавиш;
  – charset — указывает кодировку текста, на который ведет ссылка;
  – coords — устанавливает координаты активной области;
  – href — задает адрес документа, на который следует перейти;
  – hreflang — идентифицирует язык текста по ссылке;
  – media — указывает тип носителя документа, на который ведёт ссылка;
  – name — устанавливает имя якоря внутри документа;
  – rel — отношения между ссылаемым и текущим документами;
  – rev — отношения между текущим и ссылаемым документами;
  – shape — задает форму активной области ссылки для изображений;

- tabindex — определяет последовательность перехода между ссылками при нажатии на кнопку Tab;
- target — имя окна или фрейма, куда браузер будет загружать документ;
- title — добавляет всплывающую подсказку к тексту ссылки;
- type — указывает MIME-тип документа, на который ведёт ссылка.

Пример 8.7.

```
<A HREF="http://www.example.com">
  Пример гипертекстовой ссылки
</A>
```

- **Графика**
  Тег `<img>` предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG или PNG.

  Синтаксис:
  ```
  <img src="URL" alt="альтернативный текст">
  ```

  Пример 8.8.
  ```
  <img src="myimage.gif" alt="Картинка">
  ```

  Атрибуты `<img>`:
  - align — определяет, как рисунок будет выравниваться по краю и способ обтекания текстом;
  - alt — альтернативный текст для изображения;
  - border — толщина рамки вокруг изображения;
  - height — высота изображения;
  - hspace — горизонтальный отступ от изображения до окружающего контента;
  - ismap — говорит браузеру, что картинка является серверной картой-изображением;
  - longdesc — указывает адрес документа, где содержится аннотация к картинке;
  - src — путь к графическому файлу;
  - vspace — вертикальный отступ от изображения до окружающего контента;
  - width — ширина изображения;
  - usemap — ссылка на тег `<map>`, содержащий координаты для клиентской карты-изображения.

- **Средства описания таблиц**
  Элемент `<table>` служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов `<tr>` и `<td>`. Внутри `<table>`

допустимо использовать следующие элементы: `<caption>`, `<col>`, `<colgroup>`, `<tbody>`, `<td>`, `<tfoot>`, `<th>`, `<thead>` и `<tr>`.

Синтаксис:

```
<table>
  <tr>
    <td>...</td>
  </tr>
</table>
```

Атрибуты `<table>`:

- align — определяет выравнивание таблицы;
- background — задает фоновый рисунок в таблице;
- bgcolor — цвет фона таблицы;
- border — толщина рамки в пикселях;
- bordercolor — цвет рамки;
- cellpadding — отступ от рамки до содержимого ячейки;
- cellspacing — расстояние между ячейками;
- cols — число колонок в таблице;
- frame — сообщает браузеру, как отображать границы вокруг таблицы;
- height — высота таблицы;
- rules — сообщает браузеру, где отображать границы между ячейками;
- summary — краткое описание таблицы;
- width — ширина таблицы.

- **Пользовательские формы**
  - Тег `<form>` устанавливает форму на веб-странице.
  - Форма предназначена для обмена данными между пользователем и сервером.
  - Документ может содержать любое количество форм, но одновременно на сервер может быть отправлена только одна форма.
  - Данные форм должны быть независимы друг от друга.

Синтаксис:

```
<form action="URL">
  ...
</form>
```

Атрибуты `<form>`:

- accept-charset — устанавливает кодировку, в которой сервер может принимать и обрабатывать данные;
- action — адрес программы или документа, который обрабатывает данные формы;
- autocomplete — включает автозаполнение полей формы;
- enctype — способ кодирования данных формы;
- method — метод протокола HTTP;
- name — имя формы;

– novalidate — отменяет встроенную проверку данных формы на корректность ввода;
– target — имя окна или фрейма, куда обработчик будет загружать возвращаемый результат.

Тег `<input>` является одним из разносторонних элементов формы и позволяет создавать разные элементы интерфейса и обеспечить взаимодействие с пользователем. Главным образом `<input>` предназначен для создания текстовых полей, различных кнопок, переключателей и флажков.

Синтаксис:

`<input атрибуты>`

Атрибуты `<input>`:

– accept — устанавливает фильтр на типы файлов, которые можно отправить через поле загрузки файлов;
– accesskey — переход к элементу с помощью комбинации клавиш;
– align — определяет выравнивание изображения;
– alt — альтернативный текст для кнопки с изображением;
– autocomplete — включает или отключает автозаполнение;
– border — толщина рамки вокруг изображения;
– checked — предварительно активированный переключатель или флажок;
– disabled — блокирует доступ и изменение элемента;
– form — связывает поле с формой по её идентификатору;
– formaction — определяет адрес обработчика формы;
– formenctype — устанавливает способ кодирования данных формы при их отправке на сервер;
– formmethod — сообщает браузеру, каким методом следует передавать данные формы на сервер;
– formnovalidate — отменяет встроенную проверку данных на корректность;
– formtarget — определяет окно или фрейм, в которое будет загружаться результат, возвращаемый обработчиком формы;
– list — указывает на список вариантов, которые можно выбирать при вводе текста;
– max — верхнее значение для ввода числа или даты;
– maxlength — максимальное количество символов разрешенных в тексте;
– min — нижнее значение для ввода числа или даты;
– multiple — позволяет загрузить несколько файлов одновременно;
– name — имя поля, предназначено для того, чтобы обработчик формы мог его идентифицировать;
– pattern — устанавливает шаблон ввода;
– placeholder — выводит подсказывающий текст;
– readonly — устанавливает, что поле не может изменяться пользователем;
– required — обязательное для заполнения поле;

– size — ширина текстового поля;
– src — адрес графического файла для поля с изображением;
– step — шаг приращения для числовых полей;
– tabindex — определяет порядок перехода между элементами с помощью клавиши Tab;
– type — сообщает браузеру, к какому типу относится элемент формы;
– value — значение элемента.

Пример 8.9.

```
<form name="test" method="post" action="input1.php">
 <p><b>Ваше имя:</b><br>
  <input name="nick" type="text" size="40">
 </p>
 <p><b>Каким браузером в основном пользуетесь:</b><Br>
  <input type="radio" name="browser" value="ie">
    Internet Explorer<Br>
  <input type="radio" name="browser" value="opera"> Opera<Br>
  <input type="radio" name="browser" value="firefox"> Firefox<Br>
 </p>
 <p>Комментарий<Br>
  <textarea name="comment" cols="40" rows="3"></textarea></p>
 <p><input type="submit" value="Отправить">
  <input type="reset" value="Очистить"></p>
</form>
```

Тег `<select>` позволяет создать элемент интерфейса в виде раскрывающегося списка, а также список с одним или множественным выбором

Синтаксис:

```
<select>
  <option>Пункт 1</option>
  <option>Пункт 2</option>
</select>
```

Атрибуты `<select>`:

– accesskey — позволяет перейти к списку с помощью некоторого сочетания клавиш;
– autofocus — устанавливает, что список получает фокус после загрузки страницы;
– disabled — блокирует доступ и изменение элемента;
– form — связывает список с формой;
– multiple — позволяет одновременно выбирать сразу несколько элементов списка;
– name — имя элемента для отправки на сервер или обращения через скрипты;
– required — список обязателен для выбора перед отправкой формы;

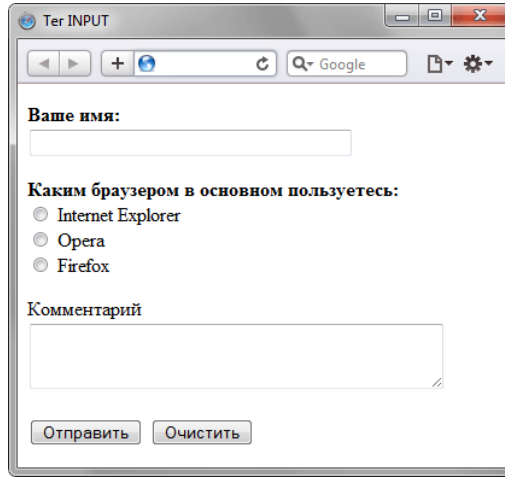

Рис. 8.1. Вид элементов формы в браузере

- size — количество отображаемых строк списка;
- tabindex — определяет последовательность перехода между элементами при нажатии на клавишу Tab

– **Работа с фреймами**
  - Определяет структуру фреймов на веб-странице.
  - Фреймы разделяют окно браузера на отдельные области, расположенные вплотную друг к другу.
  - В каждую из областей загружается самостоятельная веб-страница определяемая с помощью тега `<frame>`.
  - Тег `<frameset>` заменяет собой элемент `<body>` на веб-странице.

  Синтаксис:

```
<frameset>
  <frame>
</frameset>
```

  Атрибуты `<frameset>`:
  - border — толщина границы между фреймами;
  - bordercolor — цвет линии границы;
  - cols — устанавливает ширину или пропорции фреймов в виде колонок;
  - frameborder — определяет, отображать рамку вокруг фрейма или нет;
  - framespacing — аналог атрибута `border`, задает ширину границы;
  - rows — задает размер или пропорции фреймов в виде строк.

  Атрибуты `<frame>`:

- bordercolor — цвет линии границы;
- frameborder — определяет, отображать рамку вокруг фрейма или нет;
- name — задает уникальное имя фрейма;
- noresize — определяет, можно изменять размер фрейма пользователю или нет;
- scrolling — способ отображения полосы прокрутки во фрейме;
- src — путь к файлу, предназначенному для загрузки во фрейме.

8.2. Каскадные таблицы стилей CSS

Cascading Style Sheets (CSS) выдает браузеру инструкции о том, как отобразить определенный элемент — оформление, размещение пробелов и позиционирование.

Строка

```
<link rel="stylesheet" href="style.css" type="text/css">
```

ссылается на внешний файл с описанием стилей под именем `style.css`.

```
body {
  font-family: Arial, Verdana,  sans-serif; /* Шрифты*/
  font-size: 11pt; /* Размер основного шрифта в пунктах  */
  background-color: #f0f0f0; /* Цвет фона веб-страницы */
  color: #333; /* Цвет основного текста */
}
h1 {
  color: #a52a2a; /* Цвет заголовка */
  font-size: 24pt; /* Размер шрифта в пунктах */
  font-family: Georgia, Times, serif; /* Семейство шрифтов */
}
p {
  text-align: justify; /* Выравнивание по ширине */
  margin-left: 60px; /* Отступ слева в пикселах */
  margin-right: 10px; /* Отступ справа в пикселах */
  border-left: 1px solid #999; /* Параметры линии слева */
  border-bottom: 1px solid #999; /* Параметры линии снизу */
  padding-left: 10px; /* Отступ от линии слева до текста  */
  padding-bottom: 10px; /* Отступ от линии снизу до текста  */
}
```

Типы стилей:

- *стиль браузера* — оформление, которое по умолчанию применяется к элементам веб-страницы браузером;
- *стиль автора* — стиль, который добавляет к документу его разработчик;
- *стиль пользователя* — стиль, который может включить пользователь сайта через настройки браузера.

Преимущества стилей:

- разграничение кода и оформления;
- разное оформление для разных устройств:
  с помощью стилей можно определить вид веб-страницы для разных устройств вывода: монитора, принтера, смартфона, КПК и др.;
- расширенные по сравнению с HTML способы оформления элементов;
- ускорение загрузки сайта:
  при хранении стилей в отдельном файле, он кэшируется и при повторном обращении к нему извлекается из кэша браузера;
- единое стилевое оформление множества документов;
- централизованное хранение.

Способы добавления стилей на страницу

- *Связанные стили*
  описание селекторов и их значений располагается в отдельном файле, как правило, с расширением css, а для связывания документа с этим файлом применяется тег `<link>`
- *Глобальные стили*
  свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы в контейнере `<style>`
- *Внутренние стили*
  является по существу расширением для одиночного тега, используемого на текущей веб-странице; для определения стиля используется атрибут `style`, а его значением выступает набор стилевых правил

Базовый синтаксис CSS

```
Селектор { свойство1: значение; свойство2: значение; }
```

В качестве селектора может выступать любой тег HTML, для которого определяются правила форматирования, такие как: цвет, фон, размер и т.д.

Синтаксис:

```
Тег { свойство1: значение; свойство2: значение; ... }
```

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега.

Синтаксис:

```
Тег.Имя класса {
  свойство1: значение;
  свойство2: значение; ...
}
```

Можно также использовать классы и без указания тега. При такой записи класс можно применять к любому тегу.

Синтаксис:

```
.Имя класса {
  свойство1: значение;
  свойство2: значение; ...
}
```

Пример 8.10.

```
P { /* Обычный абзац */
 text-align: justify; /* Выравнивание текста по ширине */
}
P.cite { /* Абзац с классом cite */
 color: navy; /* Цвет текста */
 margin-left: 20px; /* Отступ слева */
 border-left: 1px solid navy; /* Граница слева от текста*/
 padding-left: 15px; /* Расстояние от линии до текста */
}
```

Идентификатор (называемый также «ID-селектор») определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты.

Синтаксис:

```
#Имя идентификатора {
  свойство1: значение;
  свойство2: значение;
... }
```

Обращение к идентификатору происходит аналогично классам, но в качестве ключевого слова у тега используется параметр `id`, значением которого выступает имя идентификатора.

Пример 8.11.

```
#help {
 position: absolute; /* Абсолютное позиционирование */
 left: 160px; /* Положение элемента от левого края */
 top: 50px; /* Положение от верхнего края */
 width: 225px; /* Ширина блока */
 padding: 5px; /* Поля вокруг текста */
 background: #f0f0f0; /* Цвет фона */
}
```

Как и при использовании классов, идентификаторы можно применять к конкретному тегу.

Синтаксис:

```
Тег#Имя идентификатора {
  свойство1: значение;
  свойство2: значение;
... }
```

Пример 8.12.

В CSS-файле:

```
P {
 color: green; /* Зеленый цвет текста */
 font-style: italic; /* Курсивное начертание текста */
}
P#opa {
 color: red; /* Красный цвет текста */
 border: 1px solid #666; /* Параметры рамки */
 background: #eee; /* Цвет фона */
 padding: 5px; /* Поля вокруг текста */
}
```

В html-документе (рис. 8.2):

```
<body>
   <p>Обычный параграф</p>
   <p id="opa">Параграф необычный</p>
</body>
```

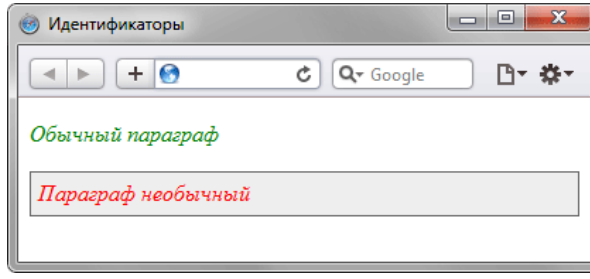

Рис. 8.2. Вид текста после применения стиля

Псевдоклассы определяют динамическое состояние элементов, которое изменяется со временем или с помощью действий пользователя, а также положение в дереве документа.

Пример: текстовая ссылка, которая меняет свой цвет при наведении на нее курсора мыши.

Синтаксис:

```
Селектор:Псевдокласс { Описание правил стиля }
```

Условно все псевдоклассы делятся на три группы:

– *определяющие состояние элементов* — распознают текущее состояние элемента и применяют стиль только для этого состояния;
– *имеющие отношение к дереву элементов* — применяются к первому дочернему элементу селектора, который расположен в дереве элементов документа;
– *указывающие язык текста* — позволяют изменять стиль оформления иностранных текстов, а также некоторые настройки.

Псевдоклассы, определяющие состояние элементов:

– `:active` — действует при активации пользователем элемента, например, ссылка становится активной, если навести на нее курсор и щелкнуть мышкой;
– `:link` — применяется к непосещенным ссылкам, т.е. таким ссылкам, на которые пользователь еще не нажимал;
– `:focus` — применяется к элементу при получении им фокуса, например, для текстового поля формы фокусом будет установленный в него курсор;
– `:hover` — ктивизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит;
– `:visited` — применяется к посещенным ссылкам.

Псевдоклассы, имеющие отношение к дереву документа:

– `:first-child` — применяется к первому дочернему элементу селектора, который расположен в дереве элементов документа.

Пример 8.13.

```
P {
  text-indent: 1em; /* Отступ первой строки */
}
P:first-child {
 text-indent: 0; /* Для первого абзаца отступ убираем */
}
```

Псевдоклассы, задающие язык текста:

– `:lang` — определяет язык, который используется в документе или его фрагменте.

Пример 8.14.

```
/* Вид кавычек для немецкого языка */
q:lang(de) {
 quotes: "\201E" "\201C";
}
/* Вид кавычек для английского языка */
q:lang(en) {
 quotes: "\201C" "\201D";
}
/* Вид кавычек для русского и французского языка */
q:lang(fr), q:lang(ru) {
```

```
 quotes: "\00AB" "\00BB";
}
```

Каскадирование

Под **каскадом** в данном случае понимается одновременное применение разных стилевых правил к элементам документа — с помощью подключения нескольких стилевых файлов, наследования свойств и других методов.

Приоритеты браузеров, которыми они руководствуются при обработке стилевых правил (чем выше в списке находится пункт, тем ниже его приоритет, и наоборот):

- стиль браузера;
- стиль пользователя;
- стиль автора;
- стиль автора с добавлением `!important`;
- стиль пользователя с добавлением `!important`.

Ключевое слово `!important` играет роль в том случае, когда пользователи подключают свою собственную таблицу стилей.

Глава 9. XML. Синтаксические правила построения, структура XML-документа. DTD и XDR-схемы

9.1. Синтаксические правила построения XML-документа

Определение 9.1. *XML (eXtensible Markup Language)* — текстовый формат, предназначенный для хранения структурированных данных, для обмена информацией между программами, а также для создания на его основе специализированных языков разметки.

Достоинства XML:
- человеко-ориентированный формат документа;
- поддержка Unicode;
- возможность описать основные структуры данных — записи, списки и деревья;
- самодокументируемый формат, который описывает структуру и имена полей так же, как и значения полей;
- строго определённый синтаксис и требования к анализу;
- широко используется для хранения и обработки документов;
- формат основан на международных стандартах;
- иерархическая структура XML подходит для описания практически любых типов документов;
- представляет собой простой текст, свободный от лицензирования и каких-либо ограничений;
- не зависит от платформы;
- является подмножеством SGML, для которого накоплен большой опыт работы и созданы специализированные приложения.

Недостатки XML:
- синтаксис XML избыточен:
  - размер XML-документа существенно больше бинарного представления тех же данных (примерно в 10 раз);
  - размер XML-документа существенно больше, чем в альтернативных текстовых форматах передачи данных;
  - избыточность XML приводит к росту стоимости хранения, обработки и передачи данных;
  - вместо XML можно использовать значительно более простые и производительные решения;
- пространства имён XML сложно использовать и реализовывать в XML-парсерах;
- XML не содержит встроенной поддержки типов данных (нет понятий «целое число», «строка», «булевы значения» и т.д.);

– иерархическая модель данных, предлагаемая XML, ограничена по сравнению с реляционной моделью и объектно-ориентированными графами.

Основные синтаксические правила построения XML-документов:

– XML-документ содержит один и только один корневой элемент, содержащий все остальные элементы;
– *дочерние элементы*, содержащиеся в корневом элементе, должны быть правильно вложены;
– *имена* элементов подчиняются правилам:
  – имя начинается с буквы, знака подчёркивания или двоеточия;
  – после первого символа в имени могут быть буквы, цифры, знаки переноса, подчёркивания, точка или двоеточие;
  – имена не могут начинаться с буквосочетания XML.

XML-документ имеет следующую **структуру**.

– Первая строка XML-документа называется *объявлением XML*. Это необязательная строка, указывающая версию стандарта XML (обычно 1.0). Также здесь может быть указана кодировка символов и внешние зависимости.
– *Комментарий* может быть размещён в любом месте дерева документа. XML-комментарии размещаются внутри пары тегов `<!--` и `-->`. Два дефиса (`--`) не могут быть применены внутри комментария.
– XML-документ состоит из вложенных элементов, некоторые имеют атрибуты и содержимое.
– *Элемент* обычно состоит из открывающего и закрывающего тегов, обрамляющих текст и другие элементы.
– *Открывающий тег* — имя элемента в угловых скобках.
– *Закрывающий тег* — имя элемента, перед которым стоит косая черта, заключённое в угловые скобки.
– *Содержимым элемента* называется всё, что расположено между открывающим и закрывающим тегами, включая текст и другие (вложенные) элементы.
– Элемент может иметь *атрибуты* — пары `имя=значение`, добавляемые внутрь открывающего тега после названия элемента.
– *Значения атрибутов* всегда заключаются в кавычки (одинарные или двойные), одно и то же имя атрибута не может встречаться дважды в одном элементе. Не рекомендуется использовать разные типы кавычек для значений атрибутов одного тега.
– Для обозначения элемента без содержания, называемого *пустым элементом*, необходимо применять особую форму записи, состоящую из одного тега, в котором после имени элемента ставится косая черта.

Отображение XML-документа в веб-браузере — без использования CSS или XSL XML-документ отображается как простой текст.

Наиболее распространены **три способа преобразования XML-документа** в отображаемый пользователю вид:
- применение стилей CSS;
- применение преобразования XSLT;
- применение обработчика XML-документа, разработанного на каком-либо языке программирования.

9.2. Схемы DTD и XDR

Определение 9.2. *Схема* чётко определяет имя и структуру корневого элемента, включая спецификацию всех его дочерних элементов.

Схемы описывают:
- что именно является разметкой;
- значение разметки.

Можно задать, какие элементы и в каком количестве обязательны, а какие — не обязательны, определить, какие элементы содержат атрибуты, допустимые значения этих атрибутов, в том числе значения по умолчанию.

Спецификации описания схем:
- *DTD (Document Type Definition)* — язык определения типа документов.
- *XDR (XML Data Reduced)* — диалект XML, разработанный Microsoft.
- *XSD (язык определения схем XML)* — рекомендована консорциумом W3C.

9.2.1. DTD

Схема DTD предоставляет шаблон разметки документа, в котором указываются наличие, порядок следования и расположение элементов и их атрибутов в документе XML (рис. 9.1).

Типы элементов документа:
- данные:
  `<!ELEMENT имя (#PCDATA)>`
  содержит только текстовые данные;
- другие элементы:
  `<!ELEMENT имя (дочерний элемент 1, дочерний элемент 2)>`
  содержит только дочерние элементы;
- смешанное:
  `<!ELEMENT имя (#PCDATA, дочерний элемент)*>`
  содержит комбинацию текстовых данных и дочерних элементов;
- EMPTY:
  `<!ELEMENT имя EMPTY>`
  ничего не содержит;

– ANY:
`<!ELEMENT имя ANY>`
может содержать текстовые данные или дочерние элементы.

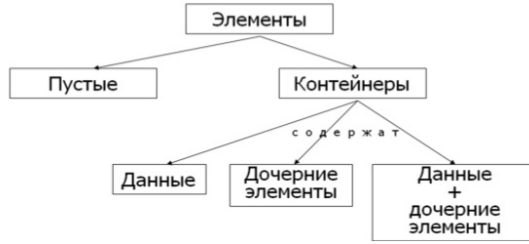

Рис. 9.1. Модель содержимого для XML-документа

Таблица 9.1
Индикаторы вхождения последовательностей

Символ	Пример	Описание
,	`(a, b, c)`	Последовательное использование элементов списка
|	`(a | b | c)`	Используется один из элементов списка
	`date`	Используется один и только один элемент
?	`subject?`	Необязательное использование (0 или 1 раз)
+	`paragraph+`	Используется 1 или несколько раз
*	`brother*`	Используется 0 или несколько раз

Атрибуты, находящиеся внутри тегов документа, описываются отдельно с помощью синтаксиса:

```
<!ATTList
имя_элемента   имя_атрибута1 (тип)   значение_по_умолчанию
...
имя_элемента   имя_атрибутаN (тип)   значение_по_умолчанию >
```

Атрибут в DTD может иметь один из трёх **типов**:
– строка;
– маркированный атрибут;
– атрибут с перечислением.

Кроме типа атрибута можно также задавать и его **модальность**:
- `#REQUIRED` — атрибут обязательно должен быть указан;
- `#FIXED` — значение атрибута не должно отличаться от указанного;
- `#IMPLIED` — необязательное значение.

Типы маркированных атрибутов:
- ID — уникальный идентификатор элемента (начинается с буквы, двоеточия или подчёркивания);
- IDREF — ссылка на элемент, содержащий атрибуты ID;
- ENTITIES — ссылка на внешний элемент;
- NMTOKEN — содержит буквы, цифры, точки, знаки подчёркивания, переносы, двоеточия, но не пробелы.

Пример 9.1. DTD схема, описывающая структуру электронного почтового ящика:

```
<!ELEMENT mailbox (message*)>
<!ELEMENT message (head, body)>
<!ATTLIST message uid CDATA #REQUIRED>
<!ELEMENT head ( from,to+, subject?, CC*, notify?) >
<!ELEMENT from (#PCDATA)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT subject (#PCDATA)>
<!ELEMENT CC (#PCDATA)>
<!ELEMENT notify EMPTY>
<!ELEMENT body (#PCDATA)>
```

XML-документ, удовлетворяющий данной схеме:

```
<?xml version="1.0" ?>
<!DOCTYPE mailbox SYSTEM "mailbox.dtd">
<mailbox>
  <message uid="1">
   <head>
    <from>user1@sci.pfu.edu.ru</from>
    <to>user2@sci.pfu.edu.ru</to>
    <subject>Re:</subject>
   </head>
   <body>What's up! </body>
  </message>
  <message uid="2">
   <head>
    <from>user3@sci.pfu.edu.ru</from>
    <to>user2@sci.pfu.edu.ru</to>
    <subject>Remind</subject>
    <CC>user1@sci.pfu.edu.ru </CC>
    <notify/>
   </head>
```

```
   <body>Remind me about meeting.</body>
  </message>
</mailbox>
```

Объявление схемы

– объявление внутренней схемы:
```
<!DOCTYPE корневой_элемент [
<!ELEMENT корневой_элемент (модель содержания)>
]>
```
– объявление внешней схемы:
```
<!DOCTYPE корневой_элемент SYSTEM "name.DTD">
```

Недостатки DTD-схем:

– не являются экземплярами XML:
  – требуется изучение совершенно другого языка;
  – нельзя легко расширить или преобразовать к другим языкам разметки — HTML или DHTML;
– не предоставляют контроль за типами данных;
– не обеспечивают поддержку пространств имен XML.

9.2.2. XDR

Корневым элементом в схеме XDR всегда является **элемент Schema**:

```
<Schema name="имя_схемы"
    xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:dt="urn:schemas-microsoft-com:datatypes">
    <-- Объявления других элементов -->
</Schema>
```

XML Schema определяет:

– элементы, входящие в документ;
– атрибуты, допустимые в документе;
– дочерние элементы;
– порядок дочерних элементов;
– количество дочерних элементов;
– может ли элемент быть пустым или содержать текст;
– типы элементов и атрибутов;
– фиксированные значения и значения по умолчанию элементов и атрибутов.

Элемент **ElementType** имеет синтаксис:

```
<ElementType content="{empty | texOnly | eltOnly | mixed}">
  dt:type "datatype"
  model="{open | closed}"
  name = "idref"
  order="{one | seq | many}"
>
```

Атрибуты ElementType:

- `content` — содержание элемента; допустимые значения: `empty` (пустой элемент), `eltOnly` (может быть только контейнером для других элементов), `textOnly` (только текстовые данные), `mixed` (смешанные данные);
- `dt:type` — тип данных элемента **model**; может принимать значения:
  - Open — разрешено использовать элементы, не определённые в схеме;
  - Closed — запрещено использовать элементы, не определённые в схеме;
- `name` — имя элемента;
- `order` — порядок следования дочерних элементов в экземпляре XML; допустимые значения:
  - one — предполагается наличие одного документа;
  - many — любое количество элементов в любом порядке;
  - seq — элементы указываются в строго заданном порядке.

Дочерние элементы для ElementType:

- `AttributeType` — определяет атрибут;
- `attribute` — определяет сведения о дочернем элементе `AttributeType`;
- `element` — объявляет дочерний элемент;
- `group` — определяет порядок следования элементов;
- `description` — обеспечивает описание элемента ElementType;
- `datatype` — обеспечивает тип данных элемента ElementType.

Для **объявления атрибутов** используется синтаксис:

```
<AttributeType
    default="default-value"
    dt:type="primitive-type"
    dt:values="enumerated-values"
    name="idref"
    required="{yes|no}"
>
```

Атрибуты `AttributeType`:

- `default` — значение по умолчанию;
- `dt:type` — один из следующих типов: `entity`, `entities`, `enumeration`, `id`, `idref`, `nmtoken`, `nmtokens`, `notation`, `string`;
- `dt:values` — допустимые значения;
- `name` — имя атрибута;
- `required` — указывает на обязательное наличие атрибута в описании.

Синтаксис для описания элемента `attribute`:

```
<attribute
    default="default-value"
    type="attribute-type"
```

```
    [required="{yes|no}"
]>
```

Возможные значения элемента `attribute`:

- `default` — значение по умолчанию;
- `type` — имя элемента `AttributeType`, определённого в данной схеме; должно соответствовать атрибуту `name` элемента `AttributeType`;
- `required` — указывает на обязательное наличие атрибута в описании.

Индикаторы вхождения в схемах XDR имеют синтаксис:

```
<element
  type="element-type"
  [minOccur="{0|1}"]
  [maxOccur="{1|*}"]
>
```

Элемент **group** имеет синтаксис:

```
<group order="(one|seq|many)"
   minOccur="(0|1)" maxOccur="(1|*)">
      <element type="ElementType/">
      <element type="ElementType/">
      <element type="ElementType/">
      <element type="ElementType/">
</group>
```

Пример 9.2. Описание структуры электронного почтового ящика.

XML-документ:

```
<?xml version="1.0" encoding="Windows-1251"?>
<mail>
  <to>user1@domain.ru</to>
  <from>user2@domain.ru </from>
  <subject>Встреча</subject>
  <body>Позвони мне завтра утром</body>
</mail>
```

Структура документа (используется XML Schema):

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.sci.pfu.edu.ru"
xmlns=" http://www.sci.pfu.edu.ru"
elementFormDefault="qualified">
<xs:element name="mail">
    <xs:complexType>
      <xs:sequence>
  <xs:element name="to" type="xs:string"/>
  <xs:element name="from" type="xs:string"/>
  <xs:element name="subject" type="xs:string"/>
```

```
  <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Ссылка на схему в XML-документе:

```
<?xml version="1.0"?>
<mail
  xmlns="http://www.sci.pfu.edu.ru "
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.sci.pfu.edu.ru mail.xsd">
  <to>user1@domain.ru</to>
  <from>user2@domain.ru</from>
  <subject>Встреча</heading>
  <body>Позвони мне завтра утром</body>
</mail>
```

Элемент
`xmlns:xs="http://www.w3.org/2001/XMLSchema"`
указывает на то, что элементы и типы данных, используемые в схеме входят в пространство `http://www.w3.org/2001/XMLSchema`, причём, все элементы и типы данных из этого пространства имен должны иметь префикс xs.

Фрагмент
`targetNamespace="http://www.sci.pfu.edu.ru"`
указывает на то, что элементы, определяемые в схеме, входят в пространство `http://www.sci.pfu.edu.ru`.

Фрагмент:
`xmlns="http://www.sci.pfu.edu.ru"`
указывает, что пространством имен по умолчанию является `http://www.sci.pfu.edu.ru`

Фрагмент
`elementFormDefault="qualified"`
указывает на то, что любые элементы, объявленные в схеме, должны принадлежать пространству имен

Строка
`xsi:schemaLocation="http://www.sci.pfu.edu.ru mail.xsd"`
указывает на местоположение файла схемы

Определение простых элементов

Синтаксис для определения простого элемента:

```
<xs:element name="xxx" type="yyy"/>
```

где xxx — имя элемента, yyy — тип данных элемента.

Встроенные типы данных элементов:

– xs:string

– xs:decimal
– xs:integer
– xs:boolean
– xs:date
– xs:time

Пример 9.3. Фрагмент XML-документа

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

описывается в схеме следующим образом:

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Следующие фрагменты:

```
<xs:element name="color" type="xs:string" default="red"/>
<xs:element name="color" type="xs:string" fixed="red"/>
```

описывают значение элемента по умолчанию и фиксированное значение соответственно.

Описание атрибута:

```
<xs:attribute name="xxx" type="yyy"/>
```

где xxx — имя атрибута, а yyy — тип данных атрибута.

Встроенные типы данных для атрибутов:

– xs:string
– xs:decimal
– xs:integer
– xs:boolean
– xs:time

XML-элемент с атрибутом:

```
<lastname lang="EN">Smith</lastname>
```

описывается соответствующей схемой:

```
<xs:attribute name="lang" type="xs:string"/>
```

Значение атрибута по умолчанию:

```
<xs:attribute name="lang" type="xs:string" default="EN"/>
```

Фиксированное значение атрибута:

```
<xs:attribute name="lang" type="xs:string" fixed="EN"/>
```

Описание обязательных атрибутов (по умолчанию все атрибуты являются необязательными):

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

Ограничения на содержимое

Для задания допустимых значений величин XML-элементов и атрибутов можно использовать **ограничения**.

Ограничения на значения XML-элементов называются **фасетами**.

Список возможных ограничений:

- enumeration — определяет список допустимых значений;
- fractionDigits — указывает максимальное число десятичных позиций, должно быть неотрицательным;
- length — указывает точное число символов или элементов в списке, должно быть неотрицательным;
- maxExclusive — указывает верхнюю границу числовых значений;
- maxInclusive — указывает верхнюю границу числовых значений;
- maxLength — указывает максимальное число символов или элементов в списке, должно быть неотрицательным;
- minExclusive — указывает нижнюю невключаемую границу числовых значений;
- minInclusive — указывает нижнюю включаемую границу числовых значений;
- minLength — указывает минимальное число символов или элементов в списке; должно быть неотрицательным;
- pattern — определяет точно последовательность символов (шаблон), которая допустима;
- totalDigits — определяет точное число цифр, должно быть неотрицательным;
- whiteSpace — указывает, как обрабатывать неотображаемые символы (пробел, табуляция и др.)

Пример 9.4.

```
<xs:element name="password">
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z0-9]{8}"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
```

Описание сложных элементов

При описании документов, имеющих сложную иерархическую структуру, можно сначала определить все элементы и атрибуты, а затем ссылаться на них.

Пример 9.5.

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

Сложный элемент в схеме можно определить следующим образом:

```
<xs:element name="firstname" type="xs:string"/>
<xs:element name="lastname" type="xs:string"/>
<xs:element name="employee">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="firstname"/>
   <xs:element ref="lastname"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Несколько элементов могут ссылаться на один и тот же сложный тип:

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

С помощью специальных индикаторов можно указывать, как элементы могут использоваться в документах.

Всего используется семь индикаторов:

- индикатор порядка (Order):
  - All
  - Choice
  - Sequence
- индикатор вхождения (Occurrence):
  - maxOccurs
  - minOccurs
- групповые индикаторы (Group):
  - Group name
  - attributeGroup name

Пример 9.6.

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name"
          type="xs:string" maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Глава 10. Введение в Jscript, VBScript, JavaScript

10.1. Введение в JScript

Определение 10.1. *JScript* — интерпретируемый, объектно-ориентированный язык программирования.

Ограничения JScript:

– не позволяет разрабатывать самостоятельные приложения;
– сценарии на JScript могут выполняться только при помощи интерпретатора, в частности веб-браузером.

Типы данных:

– число — целое, с плавающей запятой;
– строка — объявляется при помощи двойных кавычек `"string"` или апострофов `'string'` или NaN (не число);
– объект;
– логический тип — допускает значения `true` и `false`;
– null — переменная без значения;
– undefined — тип не определён.

Операторы:

– условные выражения `if` и `if...else`;
– ИЛИ *, И `&&`;
– циклы `for`, `for...in`, `while`, `do...while` и `switch`;
– оператор завершения `break`;
– оператор продолжения `continue`.

Функции:

– встроенные;
– определяемые;

Определение функции состоит из *объявления параметров* и *блока инструкций* JScript.

Объект в JScript — совокупность *методов* и *свойств*.

Объекты:

– встроенные;
– созданные;
– браузерные.

Можно обратиться к любой части объекта или массива (его свойствам и методам) либо по имени, либо по индексу.

10.2. Кратко о VBScript

Определение 10.2. *Visual Basic Scripting Edition (VBScript)* — сценарный язык программирования, интерпретируемый компонентом Windows Script Host.

VBScript широко используется при создании скриптов в операционных системах семейства Microsoft Windows.

Синтаксис VBScript является упрощённой версией синтаксиса языка Visual Basic.

Области использования VBScript:
- автоматизация администрирования систем Windows;
- серверный программный код в страницах ASP;
- клиентские сценарии в браузере Internet Explorer.

10.3. Java-апплеты

Определение 10.3. *Java-апплет* — программа, написанная на языке Java и откомпилированная в байт-код.

Java-апплет:
- выполняется в браузере с использованием виртуальной Java-машины (JVM);
- используется для предоставления интерактивных возможностей веб-приложений, которые не возможны в HTML;
- платформонезависим, может выполняться браузерами на многих операционных платформах.

Код апплета загружается с веб-сервера, и браузер либо вставляет апплет в веб-страницу, либо открывает отдельное окно с собственным пользовательским интерфейсом апплета.

Апплет может быть внедрён в веб-страницу с помощью использования HTML тега `<applet>` или тега `<object>`.

10.4. Программное взаимодействие с HTML-документами на основе DOM API

Определение 10.4. *DOM (Document Object Model)* — объектная модель документа — не зависящий от платформы и языка программный интерфейс, позволяющий программам получать доступ к содержимому документов, а также изменять содержимое, структуру и вид документов.

В рамках DOM:
- любой документ представляется в виде дерева узлов;

– каждый узел представляет собой элемент, атрибут, текстовый, графический или любой другой объект;
– узлы между собой находятся в отношении «родитель–потомок».

Каждый DOM-элемент является объектом и предоставляет свойства для манипуляции своим содержимым, доступа к родителям и потомкам.

Пример 10.1.

```
<html>
  <head>
    <title>Заголовок</title>
  </head>
  <body>
     Документ
   </body>
</html>
```

Самый внешний тег — `<html>`. Внутри `<html>` находятся два узла: `<head>` и `<body>` — они становятся дочерними узлами для `<html>`.

HTML-теги образуют узлы-элементы (element node) DOM-дерева. Текст представлен текстовыми узлами (text node). И то и другое — равноправные узлы дерева DOM.

Доступ к DOM начинается с `document`.

Войти в корень дерева:

– `document.documentElement` — свойство ссылается на DOM-объект для тега HTML;
– `document.body` — соответствует тегу BODY.

Из узла-родителя можно получить все дочерние элементы:

– псевдомассив `childNodes` хранит все дочерние элементы, включая текстовые;
– свойство `children` перечисляет только дочерние узлы, соответствующие тегам;
– свойства `firstChild` и `lastChild` обеспечивают быстрый доступ к первому и последнему потомку;
– свойство `parentNode` ссылается на родительский узел;
– свойства `previousSibling` и `nextSibling` дают доступ к левому и правому соседу.

DOM-узлы имеют три основных свойства:

– *тип* `nodeType`;
– *имя* `nodeName`, `tagName`;
– *содержимое узла* `innerHTML` (для не-элементов содержимое хранится в свойстве `nodeValue`).

Пример 10.2. На экран выводится всё содержимое `document.body`, а затем заменяется на другой контент:

```
<!DOCTYPE HTML>
<html>
<body>
  <p>Параграф</p>
  <div>Div</div>
  <script>
    // читаем текущее содержимое
    alert( document.body.innerHTML );
    // заменяем содержимое
    document.body.innerHTML = 'Новый BODY!';
  </script>
</body>
</html>
```

Узлы DOM являются HTML-элементами, у которых есть атрибуты. Доступ к атрибутам осуществляется при помощи стандартных методов:

```
elem.hasAttribute(name) // проверяет наличие атрибута
elem.getAttribute(name) // получает значение атрибута
elem.setAttribute(name, value) // устанавливает атрибут
elem.removeAttribute(name) // удаляет атрибут
```

Методы для создания узлов:

```
document.createElement(tag) // создает элемент
document.createTextNode(value) // создает текстовый узел
elem.cloneNode(deep) // клонирует элемент,
                     // если deep == true, то со всеми потомками.
```

Вставка и удаление узлов:

```
parent.appendChild(elem)
parent.insertBefore(elem, nextSibling)
parent.removeChild(elem)
parent.replaceChild(elem, currentElem)
```

Пример 10.3.

```
<script>
function createMessage(title, body) {
  var container = document.createElement('div');
  container.innerHTML = '<div class="message"> \
    <h1>' + title + '</h1> \
    <div class="content">' + body + '</div> \
    <input class="ok" type="button" value="OK"> \
  </div>';
  return container.firstChild;
}
var messageElem = \
  createMessage('Привет, Мир!', 'Я - элемент DOM!')
document.body.appendChild(messageElem);
</script>
```

10.5. Введение в JavaScript

Сценарий JavaScript — фрагмент кода, расположенный между дескрипторами `<SCRIPT>` и `</SCRIPT>`:

```
Текст HTML-документа
<SCRIPT>
  Код сценария
</SCRIPT>
Текст HTML-документа
```

Переменные:

- могут хранить данные любых типов: числа, строки текста, логические значения, ссылки на объекты, нулевое значение `null`, значение `NaN` (сообщает о недопустимости операции);
- объявляются с помощью ключевого слова `var`.

После объявления в переменную можно записать данные:

```
var message;
message = 'Привет'; // сохраним в переменной строку
```

В дальнейшем данные доступны при обращении по имени:

```
alert(message); // выведет содержимое переменной
```

`alert` выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «OK».

Синтаксис:

```
alert(сообщение)
```

Пример 10.4.

```
alert("Привет");
```

Функция `prompt` выводит модальное окно с заголовком `title`, полем для ввода текста, заполненным строкой по умолчанию `default` и кнопками OK/CANCEL:

```
result = prompt(title, default);
```

Пользователь должен либо что-то ввести и нажать OK, либо отменить ввод кликом на CANCEL или нажатием ESC на клавиатуре.

Пример 10.5.

```
var years = prompt('Сколько вам лет?', 100);
alert('Вам ' + years + ' лет!')
```

Функция `confirm` выводит окно с вопросом `question` с двумя кнопками: OK и CANCEL:

```
result = confirm(question);
```

Результатом будет `true` при нажатии OK и `false` — при CANCEL(Esc).

Пример 10.6.

```
var isAdmin = confirm("Вы - администратор?");
alert(isAdmin);
```

Основные операторы:

– присваивание;
– арифметические операторы;
– конкатенация строк;
– приоритет;
– инкремент/декремент: ++, --;
– побитовые операторы;
– вызов операторов с присваиванием;
– оператор запятая.

Оператор присваивания выглядит как знак равенства =

```
var i = 1 + 2;
alert(i); // 3
```

Он вычисляет выражение, которое находится справа, и присваивает результат переменной.

Арифметические операторы:

– плюс +
– минус -
– умножить *
– поделить /
– остаток от деления %

Пример 10.7.

```
var i = 2;
i = (2 + i) * 3 / i;
alert(i); // 6
```

Конкатенация строк

Если применить оператор + к строкам, то он их объединяет:

```
var a = "my" + "string";
alert(a); // mystring
```

Если хотя бы один аргумент является строкой, то второй будет также преобразован к строке.

Любой другой арифметический оператор наоборот, всегда преобразует аргументы к числам.

```
alert( 2 + '1' ); // "21"
alert( '1' - 2 ); // -1
alert( 6 / '2'); // 3
```

Приоритет

Если в выражении есть несколько операторов, то порядок их выполнения определяется **приоритетом**.

Каждый оператор получает *числовой приоритет*, и тот, у кого число меньше — выполнится раньше.

Существует целая **таблица приоритетов**. Отрывок из таблицы:

```
...       ...       ...
5       умножение       *
5       деление         /
6       сложение        +
6       вычитание       -
16      присвоение      =
...       ...       ...
```

Инкремент/декремент: ++, --

Определяют увеличение или уменьшение переменной на единицу:

```
var i = 2;
i++;      // делает i = i + 1, но короче.
alert(i); // 3

var k = 2;
k--;      // делает k = k - 1, но короче.
alert(k); // 1
```

Побитовые операторы:

- AND (и): &
- OR (или): |
- XOR(побитовое исключающее или): ^
- NOT (не): ~
- LEFT SHIFT (левый сдвиг): <<
- RIGHT SHIFT (правый сдвиг): >>
- ZERO-FILL RIGHT SHIFT (правый сдвиг с заполнением нулями): >>>

Вызов операторов с присваиванием

Применить оператор к переменной и сохранить результат в ней же, например:

```
n = n + 5;
d = d * 2;
```

Этот синтаксис можно сократить при помощи совмещённых операторов: +=, -=, *=, /=, >>=, <<=, >>>=, &=, |=, ^=:

```
i = 2;
i += 5; // i=7, то же что i = i + 5
i *= 2; // i=14, то же что i = i * 2
```

Оператор запятая

Запятая позволяет перечислять выражения, разделяя их запятой. Каждое из них вычисляется и отбрасывается, за исключением последнего, которое возвращается.

```
// три операции в одной строке
for(a = 1, b = 3, c = a * b; a < 10; a++) {
 ...}
```

Операторы сравнения:
– больше/меньше: `a > b`, `a < b`
– больше/меньше или равно: `a >= b`, `a <= b`
– равно `a == b`
– не равно `!=`

Логические значения:
– true — имеет смысл «да», «верно», «истина»;
– false — означает «нет», «неверно», «ложь».

Логические операторы:
– `||` (ИЛИ)
– `&&` (И)
– `!` (НЕ)

Синтаксис:
```
result = a || b;
result = a && b;
result = !a;
```

Циклы

Цикл `while`:
```
while(условие) {
  // код, тело цикла
}
```

Цикл `do..while`:
```
do {
  // тело цикла
} while(условие);
```

Цикл `for`:
```
for(начало; условие; шаг) {
 // ... тело цикла ...
}
```

Функции

Синтаксис:
```
function f(arg1,arg2,...)
{
/* тело функции */
}
```

Пример 10.8. Объявление и вызов функции:
```
function showMessage() {
  alert('Привет всем присутствующим!');
}
showMessage();
```

Функция может содержать локальные переменные, объявленные через `var`. Такие переменные видны только внутри функции:

```
function showMessage() {
  var message = 'Привет, я - Вася!';
 // локальная переменная
  alert(message);
}

showMessage(); // 'Привет, я - Вася!'

alert(message); // <-- будет ошибка,
//т.к. переменная видна только внутри
```

В функцию можно передавать переменные:

```
function showMessage(title, text) {
  title = "*** " + title + " ***";  // (1)
  alert(title + '\n\n' + text);
}

var title = 'От Маши'; // (2)
var msg = 'Привет'
showMessage(title, msg);
// параметры возьмутся из переменных
```

Результат работы функции может быть передан во внешний код:

```
function calcD(a, b, c) {
   return b*b - 4*a*c;
}

var test = calcD(-4, 2, 1);
alert(test); // 20
```

Объекты:

- содержат *свойства* и *методы*;
- идентифицируются именами;
- формы, изображения, гипертекстовые ссылки и другие компоненты веб-страницы, HTML-документ, отображаемый в окне браузера, окно браузера и даже сам браузер являются объектами;
- в процессе работы JavaScript сценарий обращается к объектам, получает информацию и управляет ими.

С помощью *переопределённых объектов* можно реализовать массив, описать дату и время, выполнить математические вычисления и решить некоторые другие задачи.

Объект `document` описывает HTML-документ, отображаемый в окне браузера.

Пример 10.9. Вывод строки текста в окне браузера.

```
<HTML>
  <HEAD>
    <TITLE>Сценарий JavaScript</TITLE>
  </HEAD>
```

```
  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
      document.write("Проверка сценария JavaScript");
    </SCRIPT>
  </BODY>
</HTML>
```

Имена чувствительны к регистрам символов!

Пустой объект (пустой ассоциативный массив) может быть создан одним из двух синтаксисов:

```
o = new Object();
o = {};   // пустые фигурные скобки
```

Объект может содержать в себе любые значения, которые называются *свойствами объекта*. Доступ к свойствам осуществляется *по имени* (иногда говорят *по ключу*).

Пример 10.10. Объект person для хранения информации о человеке.

```
var person = {};
```

Основные операции с объектами:

– присвоение свойства по ключу;
– чтение свойства по ключу;
– удаление свойства по ключу.

Для обращения к свойствам используется запись «через точку», вида `объект.свойство` или через квадратные скобки `объект['свойство']`.

Пример 10.11.

```
var person = {};

// 1. присвоение
// запишем свойство с именем 'name' и значением 'Вася'
person.name = 'Вася';
// запишем свойство с именем 'age' и значением 25
person.age = 25;

// 2. чтение
// вывести значения
alert(person.name + ': ' + person.age);

// 3. удаление
// удалить значение с именем name
delete person.name;
```

Глава 11. Методические указания

11.1. Цели и задачи дисциплины

Целью дисциплины является знакомство слушателей с современными прикладными протоколами Интернета, изучение методов организации межсетевого взаимодействия, которые используются для передачи данных пользователей, а также при управлении сетями связи.

В процессе преподавания дисциплины решаются следующие задачи:

- изучение технических аспектов протоколов прикладного уровня и протоколов, используемых в WWW;
- изучение механизмов передачи электронной почты и данных сети Интернет, элементы обеспечения сетевой безопасности.

Также в рамках курса рассматриваются основы веб-программирования и создания ресурсов Интернета.

11.2. Компетенции для направления 010300 — Фундаментальная информатика и информационные технологии

11.2.1. Место дисциплины в структуре ООП

Цикл, к которому относится дисциплина: вариативная часть профессионального цикла Б.3, дисциплина по выбору.

11.2.2. Требования к входным знаниям, умениям и компетенциям студента

Требуется пройти обучение по дисциплинам: Архитектура вычислительных систем, Операционные системы, Компьютерные сети, Сетевые технологии.

Знать:

- концепции, базовые алгоритмы, принципы разработки и функционирования современных операционных систем (ПК-20);
- теоретические и методические основы, функциональные возможности конфигурирования и использования операционных систем и сетей телекоммуникаций (ПК-25).

Владеть:

- основными методами, способами и средствами получения, хранения, переработки информации, навыками работы с компьютером как средством управления информацией (ОК-12);

– методами и навыками использования и конфигурирования операционных систем, платформенных окружений (ПК-24).

Дисциплины, для которых данная дисциплина является предшествующей: Администрирование локальных сетей, Информационная безопасность, Курсовая работа, Выпускная квалификационная работа.

11.2.3. Требования к результатам освоения дисциплины

Процесс изучения дисциплины направлен на формирование следующих компетенций: ОК: 12, 13; ПК: 1, 2, 4, 6–8, 20, 23–29.

11.2.3.1. В результате изучения дисциплины студент должен

знать:

– концепции, базовые алгоритмы, принципы разработки и функционирования современных операционных систем (ПК-20);
– теоретические и методические основы, функциональные возможности, следующих предметных областей: Архитектура и организация компьютеров, Конфигурирование и использование операционных систем, Разработка и принципы сетевых технологий (ПК-25);
– теоретические основы и общие принципы использования следующих профессиональных областей: Системное администрирование, Управление информационными коммуникациями (П-26);

уметь:

– работать с информацией в глобальных компьютерных сетях (ОК-13);
– применять в профессиональной деятельности современные электронные библиотеки и коллекции, сетевые технологии, библиотеки и пакеты программ, современные профессиональные стандарты сетевых технологий (ПК-1);
– профессионально решать задачи производственной и технологической деятельности с учетом современных достижений науки и техники, включая создание информационных ресурсов глобальных сетей, образовательного контента (ПК-2);
– применять в исследовательской и прикладной деятельности фундаментальные концепции и системные методологии, международные и профессиональные стандарты в области сетевых технологий (ПК-4);
– осуществлять целенаправленный поиск информации о новейших научных и технологических достижениях в сети Интернет, взаимодействовать и сотрудничать с профессиональными сетевыми сообществами и международными консорциумами, отслеживать динамику развития направлений области сетевых технологий (ПК-6);

– применять на практике международные и профессиональные стандарты информационных технологий, современные парадигмы и методологии, инструментальные средства, относящиеся к сетевым технологиям (ПК-7);
– квалифицированно применять в профессиональной деятельности современные электронные библиотеки и коллекции, сетевые технологии, библиотеки и пакеты программ, современные профессиональные стандарты информационных технологий (ПК-27);
– решать задачи производственной и технологической деятельности на высоком профессиональном уровне, включая создание информационных ресурсов глобальных сетей (ПК-28);
– оценивать процессы жизненного цикла информационных систем, сервисов систем информационных технологий (ПК-29);

владеть:

– основными методами, способами и средствами получения, хранения, переработки информации, навыками работы с компьютером как средством управления информацией (ОК-12);
– базовыми сетевыми технологиями, эффективно применять их для решения научно-технических и прикладных задач, связанных с развитием и использованием информационных и сетевых технологий (ПК-8);
– методами и навыками использования и конфигурирования сетевых технологий (ПК-23);
– методами и навыками использования и конфигурирования операционных систем и платформенных окружений (ПК-24).

11.3. Компетенции для направления 010200 — Математика и компьютерные науки

11.3.1. Место дисциплины в структуре ООП

Цикл, к которому относится дисциплина: вариативная часть профессионального цикла Б.3, дисциплина по выбору.

11.3.2. Требования к входным знаниям, умениям и компетенциям студента

Требуется пройти обучение по дисциплинам:

Архитектура компьютеров, Операционные системы, Компьютерные сети, Сетевые технологии.

уметь:

– быстро находить, анализировать и грамотно контекстно обрабатывать научно-техническую, естественнонаучную и общенаучную информацию, приводя ее к проблемно-задачной форме (ОК-10);

– определять общие формы, закономерности, инструментальные средства отдельной предметной области (ПК-1);
– грамотно пользоваться языком предметной области (ПК-7);

владеть:

– способностью и постоянной готовностью совершенствовать и углублять свои знания, быстро адаптироваться к любым ситуациям;
– фундаментальной подготовкой в области компьютерных наук, готовностью к использованию полученных знаний в профессиональной деятельности (ОК-11);
– значительными навыками самостоятельной работы с компьютером, программирования, использования методов обработки информации и численных методов решения базовых задач (ОК-12);
– базовыми знаниями в области информатики и современных информационных технологий, навыками использования программных средств и навыками работы в компьютерных сетях, умением создавать базы данных и использовать ресурсы Интернета (ОК-13);
– способностью к анализу и синтезу информации, полученной из любых источников (ОК-14);
– навыками контекстной обработки информации (ПК-14).

Дисциплины, для которых данная дисциплина является предшествующей: Администрирование локальных сетей, Информационная безопасность, Курсовая работа, Выпускная квалификационная работа.

11.3.3. Требования к результатам освоения дисциплины

Процесс изучения дисциплины направлен на формирование следующих компетенций: ОК: 7, 8, 11-15; ПК: 1-3, 7-10, 14, 22-23.

11.3.3.1. В результате изучения дисциплины студент должен

знать:

– и понимать корректные постановки классических задач прикладных протоколов Интернета и www (ПК: 9, 10).

уметь:

– анализировать и синтезировать информацию, полученную из любых источников (ОК-14);
– определять общие формы, закономерности, инструментальные средства предметной области прикладных протоколов Интернета (ПК-1);
– понять поставленную задачу (ПК-2);
– формулировать результат (ПК-3);

– грамотно пользоваться языком предметной области прикладных протоколов Интернета (ПК-7);
– ориентироваться в постановках задач прикладных протоколов Интернета и www (ПК-8);
– увидеть прикладной аспект в решении научной задачи, грамотно представить и интерпретировать результат (ПК-22);
– проанализировать результат и скорректировать математическую модель, лежащую в основе задачи (ПК-24).

владеть:

– значительными навыками самостоятельной научно-исследовательской работы (ОК-7);
– способностью и постоянной готовностью совершенствовать и углублять свои знания, умением быстро адаптироваться к любым ситуациям (ОК-8);
– фундаментальной подготовкой в области компьютерных наук, готовностью к использованию полученных знаний в профессиональной деятельности (ОК-11);
– значительными навыками самостоятельной работы с компьютером, программирования, использования методов обработки информации и численных методов решения базовых задач (ОК-12);
– базовыми знаниями в области информатики и современных информационных технологий, навыками использования программных средств, работы в компьютерных сетях и использования ресурсов Интернета (ОК-13);
– способностью к письменной и устной коммуникации на русском языке (ОК-15);
– навыками контекстной обработки информации (ПК-14).

11.4. Компетенции для направления 080500 — Бизнес-информатика

11.4.1. Место дисциплины в структуре ООП

Цикл, к которому относится дисциплина: вариативная часть профессионального цикла Б.3, дисциплина по выбору.

11.4.2. Требования к входным знаниям, умениям и компетенциям студента

Требуется пройти обучение по дисциплинам: Архитектура вычислительных систем, Операционные системы, Сети и системы телекоммуникаций, Сетевые технологии.

Знать:

– концепции, базовые алгоритмы, принципы разработки и функционирования современных операционных систем;
– теоретические и методические основы, функциональные возможности конфигурирования и использования операционных систем и сетей телекоммуникаций.

Владеть:

– основными методами, способами и средствами получения, хранения, переработки информации, навыками работы с компьютером как средством управления информацией.

Дисциплины, для которых данная дисциплина является предшествующей: Информационная безопасность, Курсовая работа, Выпускная квалификационная работа.

11.4.3. Требования к результатам освоения дисциплины

Процесс изучения дисциплины направлен на формирование следующих компетенций: ОК: 13, 16, ПК: 7.

11.4.3.1. В результате изучения дисциплины студент должен

знать:

– теоретические основы и общие принципы использования web-технологий;

уметь:

– управлять контентом предприятия и Интернет-ресурсов, процессами создания и использования информационных сервисов (контент-сервисов) (ПК-7);
– применять в профессиональной деятельности современные сетевые технологии, библиотеки и пакеты программ, современные профессиональные стандарты сетевых технологий;
– профессионально решать задачи производственной и технологической деятельности с учетом современных достижений науки и техники, включая: создание информационных ресурсов глобальных сетей;
– применять в исследовательской и прикладной деятельности фундаментальные концепции и системные методологии, международные и профессиональные стандарты в области сетевых технологий;
– осуществлять целенаправленный поиск информации о новейших научных и технологических достижениях в сети Интернет;
– применять на практике международные и профессиональные стандарты информационных технологий, современные парадигмы и методологии, инструментальные средства, относящиеся к сетевым технологиям;

– квалифицированно применять в профессиональной деятельности современные сетевые технологии, библиотеки и пакеты программ, современные профессиональные стандарты информационных технологий;
– решать задачи производственной и технологической деятельности на высоком профессиональном уровне, включая создание информационных ресурсов глобальных сетей;
– оценивать процессы жизненного цикла информационных систем, сервисов систем информационных технологий.

Владеть:

– навыками работы с компьютером как средством управления информацией, способностью работать с информацией в глобальных компьютерных сетях (ОК-13);
– способностью работать с информацией из различных источников (ОК-16);
– основными методами, способами и средствами получения, хранения, переработки информации, навыками работы с компьютером как средством управления информацией;
– базовыми сетевыми технологиями, способностью эффективно применять их для решения научно-технических задач и прикладных задач, связанных с развитием и использованием информационных и сетевых технологий;
– методами и навыками использования и конфигурирования сетевых технологий;
– методами и навыками использования и конфигурирования операционных систем и платформенных окружений.

11.5. Компетенции для направления 010400 — Прикладная математика и информатика

11.5.1. Место дисциплины в структуре ООП

Цикл, к которому относится дисциплина: вариативная часть профессионального цикла Б.3, дисциплина по выбору.

11.5.2. Требования к входным знаниям, умениям и компетенциям студента

Требуется пройти обучение по дисциплинам: Архитектура компьютеров, Операционные системы, Компьютерные сети, Сетевые технологии.

Уметь:

– работать с информацией в глобальных компьютерных сетях (ОК-12);

– использовать в научной и познавательной деятельности, а также в социальной сфере профессиональные навыки работы с информационными и компьютерными технологиями (ОК-14);
– работать с информацией из различных источников, включая сетевые ресурсы сети Интернет, для решения профессиональных и социальных задач (ОК-15);
– осуществлять целенаправленный поиск информации о новейших научных и технологических достижениях в сети Интернет и из других источников (ПК-6);
– собирать, обрабатывать и интерпретировать данные современных научных исследований, необходимых для формирования выводов по соответствующим научным, профессиональным проблемам (ПК-7).

Владеть:

– навыками работы с компьютером как средством управления информацией (ОК-11);
– способностью приобретать новые научные и профессиональные знания, используя современные образовательные и информационные технологии (ПК-2).

Дисциплины, для которых данная дисциплина является предшествующей: Администрирование локальных сетей, Информационная безопасность, Курсовая работа, Выпускная квалификационная работа.

11.5.3. Требования к результатам освоения дисциплины

Процесс изучения дисциплины направлен на формирование следующих компетенций: ОК: 11, 12, 14, 15; ПК: 2, 6, 7.

11.5.3.1. В результате изучения дисциплины студент должен

знать:

– общие принципы построения архитектур современных сетей телекоммуникаций;
– эталонную модель взаимодействия открытых систем;
– основные протоколы современных сетей телекоммуникаций;

уметь:

– работать с информацией в глобальных компьютерных сетях (ОК-12);
– использовать в научной и познавательной деятельности, а также в социальной сфере профессиональные навыки работы с информационными и компьютерными технологиями (ОК-14);

– работать с информацией из различных источников, включая сетевые ресурсы сети Интернет, для решения профессиональных и социальных задач (ОК-15);
– приобретать новые научные и профессиональные знания, используя современные образовательные и информационные технологии (ПК-2);
– осуществлять целенаправленный поиск информации о новейших научных и технологических достижениях в сети Интернет и из других источников (ПК-6);
– собирать, обрабатывать и интерпретировать данные современных научных исследований, необходимые для формирования выводов по соответствующим научным, профессиональным, социальным и этическим проблемам (ПК-7);

владеть:

– навыками работы с компьютером как средством управления информацией (ОК-11);
– методами администрирования IP-сетей.

11.6. Объём дисциплины и виды учебной работы

Общая трудоемкость дисциплины составляет четыре зачётных единицы (см. табл. 11.1).

11.7. Содержание дисциплины

11.7.1. Содержание разделов дисциплины

1. **Интернет. Введение в понятие протоколов прикладного уровня.**
   Предпосылки возникновения Интернета. Структура Интернета, схема административного устройства. Координирующие сетевые организации. Документация, регламентирующая развитие и эксплуатацию сети Интернет. Адресация в Интернете: URI, URL. Представление информации в Интернете. Методология поиска информации. Поисковые машины.
2. **Сетевые службы. Прикладные протоколы Интернета.**
   (a) Обзор протоколов прикладного уровня различных стеков.
   (b) Служба имен доменов DNS. Службы WINS, NetBIOS.
   (c) Протокол обмена гипертекстовой информацией (HTTP). Схема функционирования и область применения. Формат HTTP-сообщений. Динамические веб-страницы. Шлюзы прикладного уровня. Взаимодействие HTTP-сервера с внешними программами.

Таблица 11.1
Объём дисциплины и виды учебной работы

Вид учебной работы	**Всего часов**	**Семестры**
	144	5
Аудиторные занятия (всего)	72	72
В том числе:		
Лекции	36	36
Практические занятия (ПЗ)	-	-
Семинары (С)	-	-
Лабораторные работы (ЛР)	36	36
Самостоятельная работа (всего)	72	72
В том числе:	-	-
Курсовой проект (работа)	-	-
Реферат	-	-
Другие виды самостоятельной работы:		
Самостоятельная проработка дополнительных материалов по дисциплине	72	72
Вид промежуточной аттестации		экзамен
Общая трудоемкость, час	144	144
Зач. ед.	4	4

(d) Эмуляция удаленного терминала и удаленный доступ к ресурсам сети. Протоколы TELNET и SSH.
(e) Протокол передачи файлов. Основные модули службы FTP. Схема функционирования: управляющий сеанс и сеанс передачи данных. Команды взаимодействия FTP-клиента и FTP-сервера. Поиск в FTP-архивах.
(f) Электронная почта. Терминология: агент пересылки почты (MTA), агент доставки почты (MDA), почтовый агент пользователя (MUA), Mail retrieve agent (MRA). Почтовые серверы. Пользовательские агенты. Принципы передачи и форматы сообщений электронной почты. Протокол SMTP. Протоколы POP3 и IMAP. Шифрование почтовых сообщений: S/MIME, Open PGP.
(g) Протокол SNMP. Понятия SMI, MIB. Нотация ASN.1.

3. **Языки разметки. Веб.**
   (a) Языки HTML: основные теги, таблицы стилей.
   (b) XML. Синтаксические правила построения XML-документа. Структура XML-документа. Структурирование информации средствами языка XML. DTD-схемы. Недостатки DTD-схем. XDR-схемы. Элементы и атрибуты XDR схем.
   (c) Язык SGML и его подмножества. Структурирование информации средствами языка SGML. XHTML. Таблицы стилей.
4. **Разработка Веб-контента.**
   Введение в Jscript: типы данных, операторы, функции и объекты. Краткая характеристика VBScript. Java-апплеты. Программное взаимодействие с HTML-документами на основе DOM API. Введение в JavaScript: изучение основ языка и его применения для автоматизации процесса разметки и добавления интерактивных возможностей веб-страниц.

11.7.2. Разделы дисциплины и междисциплинарные связи с обеспечиваемыми (последующими) дисциплинами

Таблица 11.2

Разделы дисциплины и междисциплинарные связи с обеспечиваемыми (последующими) дисциплинами

№ п/п	Наименование обеспечиваемых (последующих) дисциплин	№№ разделов данной дисциплины, необходимых для изучения обеспечиваемых (последующих) дисциплин			
		1	2	3	4
1.	Администрирование локальных сетей		+	+	+
2.	Информационная безопасность	+	+		+
3.	Курсовая работа		+	+	+
4.	Выпускная квалификационная работа			+	+

11.7.3. Разделы дисциплин и виды занятий

Таблица 11.3

Разделы дисциплин и виды занятий

№ п/п	Наименование раздела дисциплины	Лекц.	Прак. зан.	Лаб. зан.	Сем.	СРС	Всего час.
1.	Интернет. Введение в понятие протоколов прикладного уровня	4		2		8	14
2.	Сетевые службы. Прикладные протоколы Интернет.	14		10		12	36
3.	Языки разметки. Веб.	12		12		28	52
4.	Разработка веб-контента	6		12		24	42
	Итого:	36		36		72	144

11.8. Лабораторный практикум

Таблица 11.4

Лабораторный практикум

№ п/п	№ раздела дисциплины	Наименование лабораторных работ	Трудоёмкость (час.)
1.	1.	1.1. Структура документации Интернета. Представление информации в Интернете. 1.2. Методология поиска информации. Поисковые машины. Ключевые слова. Организация поисковых запросов. Получение результата в условиях неопределенности запроса.	2

Таблица 11.4

Лабораторный практикум (окончание)

№ п/п	№ раздела дисциплины	Наименование лабораторных работ	Трудоёмкость (час.)
2.	2.	2.1. Правила настройки серверов имен в различных средах. Настройка DNS-сервера. 2.2. Методы и правила настройки HTTP-сервера. Вызов программ шлюзами. Типовые серверы и организация взаимодействия между исходными серверами и серверами посредниками. 2.3. Методы и правила настройки почтового сервера и почтового агента. 2.4. SNMP. Практические сведения о применении MIB. Методы и правила кодирования в синтаксисе ASN.1.	10
3.	3.	3.1. Создание простейших веб-страниц. Применение таблиц, кадров, таблиц стилей. 3.2. Структурирование информации средствами языка XML. DTD-схемы. 3.3. Структурирование информации средствами языка SGML. XHTML. Таблицы стилей.	12
4.	4.	4.1. Разработка веб-контента. 4.2. Применение JavaScript для автоматизации процесса разметки и добавления интерактивных возможностей веб-страниц.	12

11.9. Практические занятия (семинары)

Практические занятия (семинары) не предусмотрены.

11.10. Примерная тематика курсовых проектов (работ)

Курсовые проекты (работы) не предусмотрены.

11.11. Учебно-методическое и информационное обеспечение дисциплины

– **Основная литература:**
  1. Кулябов Д.С., Королькова А.В. Архитектура и принципы построения современных сетей и систем телекоммуникаций. — М.: РУДН, 2008.
  2. Таненбаум Э. Компьютерные сети. 4-е изд. — СПб.: Изд-во «Питер», 2007.
– **Дополнительная литература:**
  1. Семенов Ю. А. Алгоритмы телекоммуникационных сетей. Часть 1. Алгоритмы и протоколы каналов и сетей передачи данных. Интернет-университет информационных технологий. — М.: ИНТУИТ.ру, БИНОМ, Лаборатория знаний, 2007, 640 с. — `http://www.intuit.ru/department/network/algoprotnet/`
  2. Семенов Ю. А. Алгоритмы телекоммуникационных сетей. Часть 2. Протоколы и алгоритмы маршрутизации в INTERNET. Интернет-университет информационных технологий. —М.: ИНТУИТ.ру, БИНОМ, Лаборатория знаний, 2007, 832 с. — `http://www.intuit.ru/department/network/pami/`
  3. Семенов Ю. А. Алгоритмы телекоммуникационных сетей. Часть 3. Процедуры, диагностика, безопасность. Интернет-университет информационных технологий. — М.: ИНТУИТ.ру, БИНОМ, Лаборатория знаний, 2007, 512 с. — `http://www.intuit.ru/department/network/pdsi/`
  4. Олифер В. Г., Олифер Н. А. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 4-е изд. — СПб: Питер, 2010. — 944 с.
  5. Сычев А.В. Web-технологии. Интернет-университет информационных технологий. —М.: ИНТУИТ.ру, БИНОМ, Лаборатория знаний. — `http://www.intuit.ru/department/internet/webtechno/`
  6. Основы XML.Интернет-университет информационных технологий. — М.: ИНТУИТ.ру, БИНОМ, Лаборатория знаний. — `http://www.intuit.ru/department/internet/xml/`
  7. Адамс Д.Р., Флойд К.С. Основы работы с XHTML и CSS. Интернет-университет информационных технологий. —М.: ИНТУИТ.ру, БИНОМ, Лаборатория знаний. — `http://www.intuit.ru/department/internet/xhtml/`
  8. Храмцов П.Б. Введение в JavaScript и CGI. ИНТУИТ.ру, БИНОМ, Лаборатория знаний. — `http://www.intuit.ru/department/internet/vinjscgi/`
  9. Джо Барнс Практикум по программированию на JavaScript. Интернет-университет информационных технологий. — М.: ИНТУИТ.ру, БИНОМ, Лаборатория знаний. — `http://www.intuit.ru/department/internet/jspractics/`

– **Программное обеспечение**: ОС Linux, ОС Windows, flash, java, Virtualbox, Drupal, WEB-редактор.
– **Базы данных, информационно-справочные и поисковые системы** не требуются.

11.12. Материально-техническое обеспечение дисциплины

– Дисплейные классы компьютерного центра.
– Учебные лаборатории кафедры систем телекоммуникаций:
  – ауд. 110: проектор DMS800 с интерактивной доской Board 1077, ноутбук Toshiba Satellite 17/300GB Intel Core2 2.4 GHz (10 шт.);
  – ауд. 114: проектор DMS800 с интерактивной доской Board 1077, ноутбук Toshiba Satellite 17/300GB Intel Core2 2.4 GHz (10 шт.).

11.13. Методические рекомендации по организации изучения дисциплины

Учебным планом на изучение дисциплины отводится один семестр. Промежуточный контроль знаний предусматривает: проведение контрольной работы в середине семестра, подготовку и сдачу лабораторных работ. В качестве итогового контроля знаний предусмотрен экзамен.

11.13.1. Примерный перечень вопросов текущего промежуточного контроля знаний

1. Определение, область применения, примеры протоколов прикладного уровня.
2. World Wide Web. История, причины возникновения, современное состояние. Схема административного устройства сети Интернет.
3. Разновидности протокола FTP.
4. Функции основных файлов настройки типового DNS-сервера под управлением Linux.
5. Основные функции протокола SNMP. Концепция MIB. Базовые правила обозначений метасинтаксиса ASN.1.
6. Основные протоколы передачи электронной почты. Концепция MIME. Схема типовой SMTP-системы. Основные функциональные элементы систем электронной почты.
7. Упрощенная модель протокола FTP. Режимы обмена данными по протоколу FTP.

11.13.2. Примерный перечень вопросов итоговых семестровых испытаний

1. Схема задействования сервера-посредника в протоколе HTTP. Структуры ресурса и объекта в протоколе HTTP.
2. Понятия URI, URL и URN.
3. Основные методы доступа по протоколу HTTP.
4. Организация работы с CGI переменными окружения. Форма и схема передачи запроса шлюзу CGI.
5. Взаимосвязь основных языков разметки. Концепция теговой модели документа.
6. Основные особенности SGML, по сравнению с прочими языками разметки. Понятие структуры текста в языке SGML.
7. Назначение и основные функции языка XML. Сравнение языков разметки SGML и XML. Принципы структурирования XML-документа.
8. Концепция теговой модели документа в интерпретации HTML. Классификация групп тегов HTML. Оформление таблиц с помощью тегов HTML. Оформление кадров с помощью тегов HTML.
9. Целевые фреймы HTML. Механизм представления текста с помощью CSS.
10. Сравнительная характеристика языков XML и HTML.

Глоссарий

DNS-запрос (DNS query) запрос от клиента (или сервера) серверу. 43
DNS-клиент специализированная библиотека (или программа) для работы с DNS. 42
DNS-сервер специализированное ПО для обслуживания DNS. 42

HyperText Markup Language (HTML) язык разметки гипертекста. 81

Java-апплет программа, написанная на языке Java и откомпилированная в байт-код. 112
JScript интерпретируемый, объектно-ориентированный язык программирования. 111

Visual Basic Scripting Edition (VBScript) сценарный язык программирования, интерпретируемый компонентом Windows Script Host. 112

Windows Internet Name Service (WINS) служба сопоставления NetBIOS-имён компьютеров с IP-адресами узлов. 49
World Wide Web (WWW) глобальное информационное пространство, основанное на физической инфраструктуре сети Интернет и протоколе передачи данных HTTP (HyperText Transfer Protocol). 6

XML (eXtensible Markup Language) текстовый формат, предназначенный для хранения структурированных данных, для обмена информацией между программами, а также для создания на его основе специализированных языков разметки. 99

Веб-браузер специальная программа, основная функция которой — отображение гипертекста, являющегося основным методом представления информации в веб. 6
Гиперссылка специальная конструкция языка HTML, позволяющая щелчком мыши перейти к просмотру другого документа. 81
Гипертекстовый документ текстовый файл, имеющий специальные метки, называемые **тегами**, которые впоследствии опознаются браузером и используются им для отображения содержимого файла на экране компьютера. 81
Индексатор программа-анализатор веб-страниц; анализу подвергаются текст, служебные html-теги, заголовки, особенности стилистики и структурные формы. 15

Интернет (Internet) глобальная телекоммуникационная сеть информационных и вычислительных ресурсов. 6

Информационно-поисковая система (IRS) система, предназначенная для обеспечения поиска и отображения документов, представленных в базах данных. 15

Ключ шифрования секретная кодовая последовательность, используемая в процессе преобразования информации. 33

Многоцелевое расширение интернет-почты (MIME) дополняющий SMTP протокол, позволяющий передавать сообщения, используя SMTP-данные, которые не имеют вид ASCII. 62

Односторонняя функция функция, определенная, например, на множестве натуральных чисел и не требующая для вычисления своего значения больших вычислительных ресурсов, но вычисление обратной функции (т.е. по известному значению функции восстановить значение аргумента) оказывается невозможно теоретически или вычислительно. 37

Поисковый робот программа, являющаяся составной частью поисковой системы и предназначенная для перебора страниц Интернета с целью занесения информации о них в базу данных поисковика. 15

Простой протокол управления сетью (SNMP) структура для управления устройствами в сети Интернет (например, мостами, маршрутизаторами и другими сетевыми устройствами) с использованием набора протоколов TCP/IP. 68

Ранжирование построение поисковой машиной списка релевантных запросу сайтов, расположенных в соответствии с весом и ценностью запрашиваемой информации. 16

Релевантность степень соответствия найденной в поисковой системе информации запросу пользователя. 16

Система доменных имён (Domain Name System, DNS) распределённая система (распределённая база данных), ставящая в соответствие доменному имени хоста (компьютера или другого сетевого устройства) IP адрес и наоборот. 41

Тег специальная конструкция языка HTML, используемая для разметки документа и управляющая его отображением. 81

Терм ключевое слово, устойчивое словосочетание. 17

Файлы данных зоны файлы, из которых первичные DNS-серверы производят чтение зональных данных. 44

Шифрование обратимое преобразование данных с целью их сокрытия от посторонних. 33

Предметный указатель

Список иллюстраций

Список таблиц

Используемая литература

1. Коннолли Д. «Распутывание» URI, URL и URN. Присвоение имен и проблема постоянства. — DeveloperWorks. IBM. — http://www.ibm.com/developerworks/ru/library/x-urlni/index.html?S_TACT=105AGX99&S_CMP=GR01.
2. Ландэ Д. В., Снарский А. А., Безсуднов И. В. Интернетика. Навигация в сложных сетях: модели и алгоритмы. — М.: URSS, 2009.
3. Поисковые системы, каталоги и интернет-бизнес. Статьи и новости о электронной коммерции. — http://www.iskati.com/index.php.
4. Храмцов П. Информационно-поисковые системы Internet // Открытые системы. — 1996. — № 3. — http://www.publish.ru/os/1072579/text/178885.html.
5. Крюков Д. В. Поисковая система «Turtle». Физиология и анатомия. — Stack Technologies Ltd. — http://www.turtle.ru/db/architecture/turtle.html.
6. Тактаев С. А. Поиск информации в компьютерных сетях: новые подходы. — ThePromo.ru. — http://www.thepromo.ru/articleview.php?id=47.
7. Блинков И. Архитектура Google 2011. — Insight IT. — 2011. — http://www.insight-it.ru/masshtabiruemost/arkhitektura-google-2011/.
8. Кальченко Д. Интеллектуальные агенты семантического Web'а // КомпьютерПресс. — 2004. — № 10. — http://www.compress.ru/article.aspx?id=12195&iid=468.
9. Публикации о невидимом Интернете. — http://ci-razvedka.com/Articles.html.
10. Исследование и разработка системы метаданных для электронных информационных ресурсов и сервисов в фундаментальной науке: Отчёт / А. Б. Антопольский, В. И. Ауссем, С. А. Блау, А. И. Жежель / Лаборатория разработки и внедрения информационных технологий. НПБ им. К.Д. Ушинского. Отчет о результатах работ по гранту РФФИ № 04-07-90087. — 2004. — http://rd.feb-web.ru/antopolsky-04.htm.
11. Кантор И. Современный учебник JavaScript. — 2012. — http://learn.javascript.ru/.
12. Каскадные таблицы стилей второго уровня. Спецификация CSS2. — 1998. — http://www.opennet.ru/docs/RUS/CSS2/.
13. Спецификация HTML 4.0. — 1997. — http://www.opennet.ru/docs/RUS/HTML4r/.
14. Extensible Markup Language (XML, Открытый язык разметки) 1.0. — 1998. — http://www.opennet.ru/docs/RUS/XML/.
15. Учебник CSS. — HTML.net. — 2010. — http://ru.html.net/tutorials/css/.

16. Учебник HTML. — HTML.net. — 2010. — http://ru.html.net/tutorials/html/.
17. Учебник PHP. — HTML.net. — 2010. — http://ru.html.net/tutorials/php/.
18. Востров А. Создание Интернета. — seoded.ru. — 2008. — http://www.seoded.ru/istoriya/internet-history/voznik-interneta-of.html.
19. Востров А. Возникновение Интернета, неофициальная версия. — seoded.ru. — 2008. — http://www.seoded.ru/istoriya/internet-history/voznik-interneta-neof.html.
20. Семенов Ю. А. Алгоритмы телекоммуникационных сетей. Часть 3. Процедуры, диагностика, безопасность. — М.: Интернет-университет информационных технологий — ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2007. — http://www.intuit.ru/department/network/pdsi/.
21. Берлин А. Н. Основные протоколы интернет. — М.: Интернет-университет информационных технологий — ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2008. — http://www.intuit.ru/department/network/internetprot/.
22. Храмцов П. Б. Система доменных имен. — 2004. — http://citforum.ru/internet/dns/khramtsov/.
23. BIND. — Internet Systems Consortium. — http://www.isc.org/software/bind.
24. DNS и BIND. Руководство для системных администраторов. — http://dnsbind.ru/.
25. Агапов А. Работа с почтовым ящиком через Telnet. — ABOUTMAIL.RU: всё о почте. — 2007. — http://www.aboutmail.ru/articles/rfc/work-with-telnet.html.
26. Соколов М. Работаем с почтой... вручную. — ABOUTMAIL.RU: всё о почте. — 2006. — http://www.aboutmail.ru/articles/rfc/mail-manually.html.
27. Описание протокола POP3. — ABOUTMAIL.RU: всё о почте. — 2006. — http://www.aboutmail.ru/articles/rfc/pop3.html.
28. Описание протокола передачи почты SMTP. — ABOUTMAIL.RU: всё о почте. — 2006. — http://www.aboutmail.ru/articles/rfc/smtp.html.
29. Описание протокола IMAP. — ABOUTMAIL.RU: всё о почте. — 2006. — http://www.aboutmail.ru/articles/rfc/imap.html.
30. Wobus J. Почтовые протоколы. — ABOUTMAIL.RU: всё о почте. — 2006. — http://www.aboutmail.ru/articles/rfc/mapi-protocols.html.
31. Учебник FTP. — www.XServer.ru: On-Line библиотека. — http://www.xserver.ru/computer/protokol/ftp/1/index.shtml.
32. Тимачев С. М. Обзор HTTP. — www.XServer.ru: On-Line библиотека. — http://www.xserver.ru/computer/protokol/http/1/.
33. HyperText Transfer Protocol — протокол обмена WWW-серверов. —

www.XServer.ru: On-Line библиотека. — http://www.xserver.ru/computer/protokol/http/3/.
34. Язык XML. Стилевые таблицы XSL. — www.XServer.ru: On-Line библиотека. — http://www.xserver.ru/computer/langprogr/xml/4/.
35. Печерский А. Язык XML — практическое введение. — www.XServer.ru: On-Line библиотека. — http://www.xserver.ru/computer/langprogr/xml/6/.
36. PHP — система разработки скриптов. — www.XServer.ru: On-Line библиотека. — http://www.xserver.ru/computer/langprogr/php/8/.
37. Сычев А. В. Web-технологии: Учебный курс. — Воронежский государственный университет, 2009. — http://window.edu.ru/resource/423/61423.
38. Кулябов Д. С. Защита информации в компьютерных сетях: Учебно-методическое пособие. Часть 1. — РУДН, 2004. — http://window.edu.ru/resource/898/64898.
39. Чукарин А. В. Прикладные протоколы Интернет и WWW: Учебно-методическое пособие. — РУДН, 2006. — http://window.edu.ru/resource/897/64897.

Учебное издание

Анна Владиславовна Королькова
Дмитрий Сергеевич Кулябов

Прикладные протоколы Интернет и www

курс лекций

Учебное пособие

Тематический план 2012 г., № 20

Технический редактор *Н. А. Ясько*

Издание подготовлено в авторской редакции
Компьютерная вёрстка *А. В. Королькова*, *Д. С. Кулябов*

Подписано в печать 6.08.2012 г. Формат 60×84/16. Печать офсетная.
Усл. печ. л. 8,60. Тираж 100 экз. Заказ № 368.

Российский университет дружбы народов
115419, ГСП-1, г. Москва, ул. Орджоникидзе, д. 3

Типография РУДН
115419, ГСП-1, г. Москва, ул. Орджоникидзе, д. 3, тел. 952-04-41