Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»

Кафедра электронных приборов

ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Учебное пособие
для студентов направления «Электроника и микроэлектроника»
(специальность «Электронные приборы и устройства»

2012

Шандаров Евгений Станиславович

Информационные системы: учебное пособие для студентов направления «Электроника и микроэлектроника» (специальность «Электронные приборы и устройства» / А.А. Колегов; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Томский государственный университет систем управления и радиоэлектроники, Кафедра электронных приборов. -- Томск: ТУСУР, 2012. - 100 с.

Цель преподавания дисциплины состоит в формировании у студентов знаний по основам построения информационных систем с использованием технологий Интернет, навыков создания самостоятельных информационных систем.

Предназначено для студентов очной и заочной форм, обучающихся по направлению «Электроника и микроэлектроника» (специальность «Электронные приборы и устройства») по дисциплине «Информационные технологии в электронике».

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Томский государственный университет систем управления и
радиоэлектроники»

Кафедра электронных приборов

УТВЕРЖДАЮ
Зав.кафедрой ЭП
_____С.М. Шандаров
«___» _____ 2012 г.

ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Учебное пособие
для студентов направления «Электроника и микроэлектроника»
(специальность «Электронные приборы и устройства»

Разработчик
_____Е.С. Шандаров
«____»_____2012 г

2012

Содержание

1. Понятие информационных систем. Базовые принципы построения информационных систем на основе технологий Интернет

Современные информационные системы - это комплексные решения , интегрирующие телекоммуникационные и информационные технологии, различные среды передачи информации и ее носители. Информационные системы окружают нас повсюду, они помогают бухгалтерам в начислении зарплаты и расчете налогов, продают авиабилеты и позволяют вносить платежи за мобильную связь, обеспечивают бесперебойную и безаварийную работу технологических установок и даже пытаются управлять целыми производственными комплексами.

При проектировании и создании современных информационных систем, в дальнейшем ИС, могут быть использованы совершенно различные подходы. В данном пособии мы постараемся рассмотреть круг вопросов, связанных с проектированием и созданием информационных систем на базе технологий Интернет. Как нам представляется, именно этот подход на сегодняшний день является наиболее прогрессивным и динамично развивающимся.

Попробуем определить несколько базовых принципов построения информационных систем на основе технологий Интернет.

В первую очередь отметим, что любая ИС содержит в себе следующий набор обязательных компонентов:

- интерфейс пользователя
- подсистема обработки пользовательской информации
- подсистема хранения данных
- канал передачи данных между центром процессинга и приложением пользователя

Технологии Интернет предлагают нам обширный инструментарий для реализации любого из вышеуказанных компонентов. Для обслуживания канала передачи данных воспользуемся семейством протоколов TCP/IP, в качестве

подсистемы обработки пользовательской информации возьмем HTTP-сервер Apache с подключенным языком написания скриптов PHP. С задачей хранения структурированных данных прекрасно справится один из многочисленных SQL-серверов. В качестве пользовательского интерфейса возможно использование как специально написанного программного обеспечения на языке высокого уровня, так и набор стандартных компонентов, предлагаемый нам любым Интернет-броузером и языком гипертекстовой разметки документов HTML.

При этом наша информационная система будет доступна пользователю из любой точки земного шара (во всяком случае если там есть Интернет или работает спутниковая связь), обеспечит высокую скорость обработки информации (благодаря технологии распределенных систем) и сможет защитить наши данные от постороннего вмешательства.

В чем на наш взгляд состоят основные преимущества использования технологий Интернет для построения современных информационных систем?

Выделим несколько ключевых причин

- открытые стандарты
- поддержка любой вычислительной платформы
- возможность использования программного обеспечения с открытым кодом
- относительно простая и эффективная организация многопользовательской работы
- относительно простой способ решения задачи актуализации программного обеспечения
- доступность информационной системы из любой точки мира

Безусловно, для проектирования и создания эффективных информационных систем необходимо изучить основные компоненты технологий Интернет.

2. Интернет. Краткое историческое введение

Около 20 лет назад Министерство Обороны США создало сеть, которая явилась предтечей Интернет, - она называлась ARPAnet. ARPAnet была экспериментальной сетью, - она создавалась для поддержки научных исследований в военно-промышленной сфере, - в частности, для исследования методов построения сетей, устойчивых к частичным повреждениям, получаемым, например, при бомбардировке авиацией и способных в таких условиях продолжать нормальное функционирование. Это требование дает ключ к пониманию принципов построения и структуры Интернет. В модели ARPAnet всегда была связь между компьютером-источником и компьютером-приемником (станцией назначения). Сеть *a priori* предполагалась ненадежной: любая часть сети может исчезнуть в любой момент.

На связывающиеся компьютеры - не только на саму сеть - также возложена ответственность обеспечивать налаживание и поддержание связи. Основной принцип состоял в том, что любой компьютер мог связаться как равный с равным с любым другим компьютером.

Передача данных в сети была организована на основе протокола Internet - IP. Протокол IP - это правила и описание работы сети. Этот свод включает правила налаживания и поддержания связи в сети, правила обращения с IP-пакетами и их обработки, описания сетевых пакетов семейства IP (их структура и тому подобное). Сеть задумывалась и проектировалась так, чтобы от пользователей не требовалось никакой информации о конкретной структуре сети. Для того, чтобы послать сообщение по сети, компьютер должен поместить данные в некий «конверт», называемый, например, IP, указать на этом «конверте» конкретный адрес в сети и передать получившиеся в результате этих процедур пакеты в сеть.

Эти решения могут показаться странными, как и предположение о «ненадежной» сети, но уже имеющийся опыт показал, что большинство этих реше-

ний вполне разумно и верно. Пока Международная Организация по Стандартизации (Organization for International Standartization - ISO) тратила годы, создавая окончательный стандарт для компьютерных сетей, пользователи ждать не желали. Активисты Internet начали устанавливать IP-программное обеспечение на все возможные типы компьютеров. Вскоре это стало единственным приемлемым способом для связи разнородных компьютеров. Такая схема понравилась правительству и университетам, которые проводят политику покупки компьютеров у различных производителей. Каждый покупал тот компьютер, который ему нравился и вправе был ожидать, что сможет работать по сети совместно с другими компьютерами.

Примерно 10 лет спустя после появления ARPAnet появились Локальные Вычислительные Сети (LAN), например,такие как Ethernet и др. Одновременно появились компьютеры, которые стали называть рабочими станциями. На большинстве рабочих станций была установлена Операционная Система UNIX. Эта ОС имела возможность работы в сети с протоколом Internet (IP). В связи с возникновением принципиально новых задач и методов их решения появилась новая потребность: организации желали подключиться к ARPAnet своей локальной сетью. Примерно в то же время появились другие организации, которые начали создавать свои собственные сети, использующие близкие к IP коммуникационные протоколы. Стало ясно, что все только выиграли бы, если бы эти сети могли общаться все вместе, ведь тогда пользователи из одной сети смогли бы связываться с пользователями другой сети.

Одной из важнейших среди этих новых сетей была NSFNET, разработанная по инициативе Национального Научного Фонда (National Science Foundation - NSF). В конце 80-х годов NSF создал пять суперкомпьютерных центров, сделав их доступными для использования в любых научных учреждениях. Было создано всего лишь пять центров потому, что они очень дороги даже для богатой Америки. Именно поэтому их и следовало использовать кооперативно. Возникла проблема связи: требовался способ соединить эти центры

и предоставить доступ к ним различным пользователям. Сначала была сделана попытка использовать коммуникации ARPAnet, но это решение потерпело крах, столкнувшись с бюрократией оборонной отрасли и проблемой обеспечения персоналом.

Тогда NSF решил построить свою собственную сеть, основанную на IP технологии ARPAnet. Центры были соединены специальными телефонными линиями с пропускной способностью 56 Kbps. Однако, было очевидно, что не стоит даже и пытаться соединить все университеты и исследовательские организации непосредственно с центрами, так как проложить такое количество кабеля не только очень дорого, но практически невозможно. Поэтому решено было создавать сети по региональному принципу. В каждой части страны заинтересованные учреждения должны были соединиться со своими ближайшими соседями. Получившиеся цепочки подсоединялись к суперкомпьютеру в одной из своих точек, таким образом суперкомпьютерные центры были соединены вместе. В такой топологии любой компьютер мог связаться с любым другим, передавая сообщения через соседей.

Это решение было успешным, но настала пора, когда сеть уже более не справлялась с возросшими потребностями. Совместное использование суперкомпьютеров позволяло подключенным общинам использовать и множество других вещей, не относящихся к суперкомпьютерам. Неожиданно университеты, школы и другие организации осознали, что получили в свое распоряжение море данных и мир пользователей. Поток сообщений в сети (трафик) нарастал все быстрее и быстрее пока, в конце концов, не перегрузил управляющие сетью компьютеры и связывающие их телефонные линии. В 1987 году контракт на управление и развитие сети был передан компании Merit Network Inc., которая занималась образовательной сетью Мичигана совместно с IBM и MCI. Старая физически сеть была заменена более быстрыми (примерно в 20 раз) телефонными линиями. Были заменены на более быстрые и сетевые управляющие машины.

Процесс совершенствования сети идет непрерывно. Однако, большинство этих перестроек происходит незаметно для пользователей. Включив компьютер, вы не увидите объявления о том, что ближайшие полгода Internet не будет доступна из-за модернизации. Возможно даже более важно то, что перегрузка сети и ее усовершенствование создали зрелую и практичную технологию. Проблемы были решены, а идеи развития проверены в деле.

Важно отметить то, что усилия NSF по развитию сети привели к тому, что любой желающий может получить доступ к сети. Прежде Internet была доступна только для исследователей в области информатики, государственным служащим и подрядчикам. NSF способствовал всеобщей доступности Internet по линии образования, вкладывая деньги в подсоединение учебного заведения к сети, только если то, в свою очередь, имело планы распространять доступ далее по округе. Таким образом, каждый студент четырехлетнего колледжа мог стать пользователем Internet.

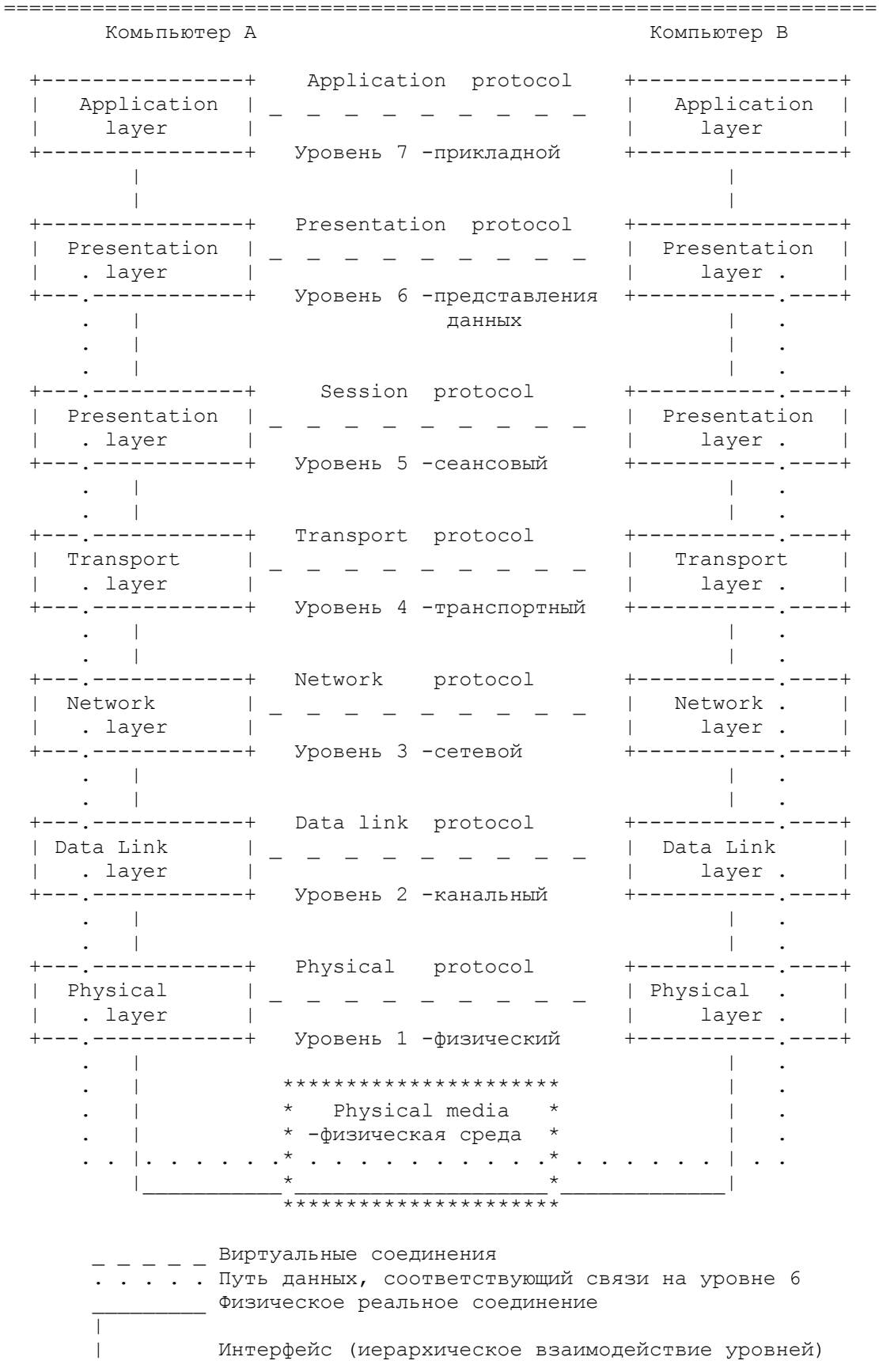
И потребности продолжают расти. Большинство таких колледжей на Западе уже подсоединено к Internet, предпринимаются попытки подключить к этому процессу средние и начальные школы. Выпускники колледжей прекрасно осведомлены о преимуществах Internet и рассказывают о них своим работодателям. Вся эта деятельность приводит к непрерывному росту сети, к возникновению и решению проблем этого роста, развитию технологий и системы безопасности сети.

3. Работа Интернет. Организация, структура, методы

Эталонная модель ISO OSI. Структура функционирования сети

Современные сети построены по многоуровневому принципу. Чтобы организовать связь двух компьютеров, требуется сначала создать свод правил их взаимодействия, определить язык их общения, то есть определить, что означают посылаемые ими сигналы и так далее. Эти правила и определения называются протоколом. Для работы сетей необходимо запастись множеством различ-

ных протоколов: например, управляющих физической связью, установлением связи по сети, доступом к различным ресурсам и т.д. Многоуровневая структура спроектирована с целью упростить и упорядочить это великое множество протоколов и отношений. Продемонстрируем принципы этой структуры на диаграмме. На рисунке показана общепринятая семиуровневая структура согласно ISO. Эта модель известна как «эталонная модель ISO OSI». Она позволяет составлять сетевые системы из продуктов - модулей программного обеспечения - выпущенных разными производителями.


Комьпьютер А
Компьютер В
Application protocol
Application layer
Уровень 7 -прикладной
Presentation protocol
Presentation layer
Уровень 6 -представления данных
Session protocol
Уровень 5 -сеансовый
Transport protocol
Transport layer
Уровень 4 -транспортный
Network protocol
Network layer
Уровень 3 -сетевой
Data link protocol
Data Link layer
Уровень 2 -канальный
Physical protocol
Physical layer
Уровень 1 -физический
Physical media
-физическая среда
Виртуальные соединения
Путь данных, соответствующий связи на уровне 6
Физическое реальное соединение
Интерфейс (иерархическое взаимодействие уровней)

Взаимодействие уровней в этой модели - субординарное. Каждый уровень может реально взаимодействовать только с соседними уровнями (верхним и нижним), виртуально - только с аналогичным уровнем на другом конце линии.

Под реальным взаимодействием мы подразумеваем непосредственное взаимодействие, непосредственную передачу информации, например, пересылку данных в оперативной памяти из области, отведенной одной программе, в область другой программы. При непосредственной передаче данные остаются неизменными все время. Под виртуальным взаимодействием мы понимаем опосредованное взаимодействие и передачу данных; здесь данные в процессе передачи могут уже определенным, заранее оговоренным образом видоизменяться.

Такое взаимодействие аналогично схеме цепи посылки письма одним директором фирмы другому. Например, директор некоторой фирмы пишет письмо редактору газеты. Директор пишет письмо на своем фирменном бланке и отдает этот листок секретарю. Секретарь запечатывает листок в конверт, надписывает конверт, наклеивает марку и передает почте. Почта доставляет письмо в соответствующее почтовое отделение. Это почтовое отделение связи непосредственно доставляет письмо получателю - секретарю редактора газеты. Секретарь распечатывает конверт и, по мере надобности, подает редактору. Ни одно из звеньев цепи не может быть пропущено, иначе цепь разорвется: если отсутствует, например, секретарь, то листок с письменами директора так и будет пылиться на столе у секретаря.

Здесь мы видим, как информация (лист бумаги с текстом) передается с верхнего уровня вниз, проходя множество необходимых ступеней - стадий обработки. Обрастает служебной информацией (пакет, адрес на конверте, почтовый индекс; контейнер с корреспонденцией; почтовый вагон, станция назначения почтового вагона и так далее), изменяется на каждой стадии обработки и постепенно доходит до самого нижнего уровня - уровня почтового транспорта (автомобильного, железнодорожного, воздушного,...), которым реально перево-

зится в пункт назначения. В пункте назначения происходит обратный процесс: вскрывается контейнер и извлекается корреспонденция, считывается адрес на конверте и почтальон несет его адресату (секретарю), который восстанавливает информацию в первоначальном виде, - достает письмо из конверта, прочитывает его и определяет его срочность, важность, и в зависимости от этого передает информацию выше. Директор и редактор, таким образом, виртуально имеют прямую связь. Ведь редактор газеты получает в точности ту же информацию, которую отправил директор, а именно - лист бумаги с текстом письма. Начальствующие персоны совершенно не заботятся о проблемах пересылки этой информации. Секретари также имеют виртуально прямую связь: секретарь редактора получит в точности то же, что отправил секретарь директора, а именно - конверт с письмом. Секретарей совершенно не волнуют проблемы почты, пересылающей письма. И так далее.

Аналогичные связи и процессы имеют место и в эталонной модели ISO OSI. Физическая связь реально имеет место только на самом нижнем уровне (аналог почтовых поездов, самолетов, автомобилей). Горизонтальные связи между всеми остальными уровнями являются виртуальными, реально они осуществляются передачей информации сначала вниз, последовательно до самого нижнего уровня, где происходит реальная передача, а потом, на другом конце, обратная передача вверх последовательно до соответствующего уровня.

Модель ISO OSI предписывает очень сильную стандартизацию вертикальных межуровневых взаимодействий. Такая стандартизация гарантирует совместимость продуктов, работающих по стандарту какого-либо уровня, с продуктами, работающими по стандартам соседних уровней, даже в том случае, если они выпущены разными производителями. Количество уровней может показаться избыточным, однако же, такое разбиение необходимо для достаточно четкого разделения требуемых функций во избежание излишней сложности и создания структуры, которая может подстраиваться под нужды конкретного пользователя, оставаясь в рамках стандарта.

Дадим краткий обзор уровней эталонной модели ISO OSI.

Уровень 0 связан с физической средой - передатчиком сигнала и на самом деле не включается в эту схему, но весьма полезен для понимания. Этот почетный уровень представляет посредников, соединяющих конечные устройства: кабели, радиолинии и так далее. Кабелей существует великое множество различных видов и типов: экранированные и неэкранированные витые пары, коаксиальные, на основе оптических волокон и т.д. Так как этот уровень не включен в схему, он ничего и не описывает, только указывает на среду.

Уровень 1 - физический. Включает физические аспекты передачи двоичной информации по линии связи. Детально описывает, например, напряжения, частоты, природу передающей среды. Этому уровню вменяется в обязанность поддержание связи и прием-передача битового потока. Безошибочность желательна, но не требуется.

Уровень 2 - канальный. Связь данных. Обеспечивает безошибочную передачу блоков данных (называемых кадрами (frame)) через уровень 1, который при передаче может искажать данные. Этот уровень должен определять начало и конец кадра в битовом потоке, формировать из данных, передаваемых физическим уровнем, кадры или последовательности , включать процедуру проверки наличия ошибок и их исправления. Этот уровень (и только он) оперирует такими элементами, как битовые последовательности, методы кодирования, маркеры. Он несет ответственность за правильную передачу данных (пакетов) на участках между непосредственно связанными элементами сети. Обеспечивает управление доступом к среде передачи. В виду его сложности, канальный уровень подразделяется на два подуровня: MAC (Medium Access Control) - Управление доступом к среде и LLC (Logical Link Control) - Управление логической связью (каналом). Уровень MAC управляет доступом к сети (с передачей маркера в сетях Token Ring или распознаванием конфликтов (столкновений передач) в сетях Ethernet) и управлением сетью. Уровень LLC, действующий над уровнем MAC, и есть собственно тот уровень, который посылает и получа-

ет сообщения с данными.

Уровень 3 - сетевой. Этот уровень пользуется возможностями, предоставляемыми ему уровнем 2, для обеспечения связи двух любых точек в сети. Любых, необязательно смежных. Этот уровень осуществляет проводку сообщений по сети, которая может иметь много линий связи, или по множеству совместно работающих сетей, что требует маршрутизации, то есть определения пути, по которому следует пересылать данные. Маршрутизация производится на этом же уровне. Выполняет обработку адресов, а также и демультиплексирование.

Основной функцией программного обеспечения на этом уровне является выборка информации из источника, преобразование ее в пакеты и правильная передача в точку назначения.

Есть два принципиально различных способа работы сетевого уровня. Первый - это метод виртуальных каналов. Он состоит в том, что канал связи устанавливается при вызове (начале сеанса (session) связи), по нему передается информация, и по окончании передачи канал закрывается (уничтожается). Передача пакетов происходит с сохранением исходной последовательности, даже если пакеты пересылаются по различным физическим маршрутам, то есть виртуальный канал динамически перенаправляется. При этом пакеты данных не включают адрес пункта назначения, так как он определяется во время установления связи.

Второй - метод дейтаграмм . Дейтаграммы - независимые , они включают всю необходимую для их пересылки информацию. В то время, как первый метод предоставляет следующему уровню (уровню 4) надежный канал передачи данных, свободный от искажений (ошибок) и правильно доставляющий пакеты в пункт назначения, второй метод требует от следующего уровня работы над ошибками и проверки доставки нужному адресату.

Уровень 4 - транспортный. Регламентирует пересылку пакетов сообщений между процессами, выполняемыми на компьютерах сети. Завершает организацию передачи данных: контролирует на сквозной основе поток данных,

проходящий по маршруту, определенному третьим уровнем: правильность передачи блоков данных, правильность доставки в нужный пункт назначения, их комплектность, сохранность, порядок следования. Собирает информацию из блоков в ее прежний вид. Или же оперирует с дейтаграммами, то есть ожидает отклика-подтверждения приема из пункта назначения, проверяет правильность доставки и адресации, повторяет посылку дейтаграммы, если не пришел отклик. В рамках транспортного протокола предусмотрено пять классов качества транспортировки и соответствующие процедуры управления. Этот же уровень должен включать развитую и надежную схему адресации для обеспечения связи через множество сетей и шлюзов. Другими словами, задачей данного уровня является довести до ума передачу информации из любой точки в любую во всей сети.

Транспортный уровень скрывает от всех высших уровней любые детали и проблемы передачи данных, обеспечивает стандартное взаимодействие лежащего над ним уровня с приемом-передачей информации независимо от конкретной технической реализации этой передачи.

Уровень 5 - сеансовый. Координирует взаимодействие связывающихся пользователей: устанавливает их связь, оперирует с ней, восстанавливает аварийно оконченные сеансы. Этот же уровень ответственен за картографию сети - он преобразовывает региональные (доменные) компьютерные имена в числовые адреса , и наоборот. Он координирует не компьютеры и устройства, а процессы в сети, поддерживает их взаимодействие - управляет сеансами связи между процессами прикладного уровня.

Уровень 6 - уровень представления данных. Этот уровень имеет дело с синтаксисом и семантикой передаваемой информации, то есть здесь устанавливается взаимопонимание двух сообщающихся компьютеров относительно того, как они представляют и понимают по получении передаваемую информацию. Здесь решаются, например, такие задачи, как перекодировка текстовой информации и изображений, сжатие и распаковка, поддержка сетевых файловых си-

стем (NFS), абстрактных структур данных и так далее.

Уровень 7 - прикладной. Обеспечивает интерфейс между пользователем и сетью, делает доступными для человека всевозможные услуги. На этом уровне реализуется, по крайней мере, пять прикладных служб: передача файлов, удаленный терминальный доступ, электронная передача сообщений, служба справочника и управление сетью. В конкретной реализации определяется пользователем (программистом) согласно его насущным нуждам и возможностям его кошелька, интеллекта и фантазии. Имеет дело, например, с множеством различных протоколов терминального типа, которых существует более ста.

Следует понимать, что подавляющее большинство современных сетей в силу исторических причин лишь в общих чертах, приближенно, соответствуют эталонной модели ISO OSI. Рассмотрим некоторые из указанных уровней в их проекции на Internet подробнее. Следует не забывать, что проекция искажена в силу несоответствия технологии Internet стандарту эталонной модели - перемешаны некоторые уровни и функции уровней

Уровни работы сети

Пересылка битов

Пересылка битов происходит на физическом уровне схемы ISO OSI. Увы, здесь всякая попытка краткого и доступного описания обречена на провал. Требуется введение огромного количества специальных терминов, понятий, описаний процессов на физическом уровне и так далее. И потом, существует столь великое разнообразие приемо-передатчиков и передающих сред, - трудно даже и обозреть этот океан технологий. Для понимания работы сетей этого и не требуется. Считайте, что просто имеется труба, по которой из конца в конец перекачиваются биты. Именно биты, безо всякого деления на какие-либо группы (байты, декады и тому подобное).

Пересылка данных

Об организации блочной, символьной передачи, обеспечении надежности пересылки поговорим на других уровнях модели ISO OSI. То есть функции ка-

нального уровня в Internet распределены по другим уровням, но не выше транспортного. В этом смысле Internet не совсем соответствует стандарту ISO. Здесь канальный уровень занимается только разбиением битового потока на символы и кадры и передачей полученных данных на следующий уровень. Обеспечением надежности передачи он себя не утруждает.

Сети коммутации пакетов

Настала пора поговорить об Internet именно как о сети, а не паутине линий связи и множестве приемо-передатчиков. Казалось бы, Internet вполне аналогична телефонной сети, и модель телефонной сети достаточно адекватно отражает ее структуру и работу. В самом деле, обе они электронные, обе позволяют вам устанавливать связь и передавать информацию. И Internet тоже состоит, в первую очередь, из выделенных телефонных линий. Но увы! Картина эта неверна и приводит ко многим заблуждениям относительно работы Internet, ко множеству недоразумений. Телефонная сеть - это так называемая сеть с коммутацией каналов, то есть когда вы делаете вызов, устанавливается связь и на все время сеанса связи имеется физическое соединение с абонентом. При этом вам выделяется часть сети, которая для других уже не доступна, даже если вы молча дышите в трубку, а другие абоненты хотели бы поговорить по действительно неотложному делу. Это приводит к нерациональному использованию очень дорогих ресурсов - линий связи. Internet же является сетью с коммутацией пакетов, что принципиально отличается от сети с коммутацией каналов.

Для Internet более подходит модель, которая поначалу может не внушать доверия: почта, обыкновенная государственная почтовая служба. Почта является сетью пакетной связи. Нет никакой выделенной вам части этой сети. Ваше послание перемешивается с посланиями других пользователей, кидается в контейнер, пересылается в другое почтовое отделение, где снова сортируется. Хотя технологии сильно разнятся, почта является прекрасным и наглядным примером сети с коммутацией пакетов. Модель почты удивительно точно отра-

жает суть работы и структуры Internet. Ею мы и будем пользоваться далее.

Протокол Internet (IP)

По проводу можно переслать биты только из одного его конца в другой. Internet же умудряется аккуратно передавать данные в различные точки, разбросанные по всему миру. Как она это делает? Забота об этом возложена на сетевой (межсетевой) уровень в эталонной модели ISO OSI.

Различные части Internet - составляющие сети - соединяются между собой посредством компьютеров, которые называются «узлы»; так Сеть связывается воедино. Сети эти могут быть Ethernet, Token Ring, сети на телефонных линиях, пакетные радиосети и тому подобные. Выделенные линии и локальные сети суть аналоги железных дорог, самолетов почты и почтовых отделений, почтальонов. Посредством их почта движется с места на место. Узлы - аналоги почтовых отделений, где принимается решение, как перемещать данные («пакеты») по сети, точно так же, как почтовый узел намечает дальнейший путь почтового конверта. Отделения или узлы не имеют прямых связей со всеми остальными. Если вы отправляете конверт из города Асино (Томская область) в Уфу (Башкирия), конечно же, почта не станет нанимать самолет, который полетит из ближайшего к Асино аэропорта (Богашево) в Уфу, просто местное почтовое отделение отправляет послание на подстанцию в нужном направлении, та в свою очередь, дальше в направлении пункта назначения на следующую подстанцию; таким образом письмо станет последовательно приближаться к пункту назначения, пока не достигнет почтового отделения, в ведении которого находится нужный объект и которое доставит сообщение получателю. Для работы такой системы требуется, чтобы каждая подстанция знала о наличествующих связях и о том, на какую из ближайших подстанций оптимально следует передать адресованный туда-то пакет. Примерно также и в Internet: узлы выясняют, куда следует ваш пакет данных, решают куда его дальше отправить и отправляют.

На каждой почтовой подстанции определяется следующая подстанция, куда будет далее направлена корреспонденция, то есть намечается дальнейший путь (маршрут) - этот процесс называется маршрутизацией. Для осуществления маршрутизации каждая подстанция имеет таблицу, где адресу пункта назначения (или индексу) соответствует указание почтовой подстанции, куда следует посылать далее этот конверт (бандероль,). Их сетевые аналоги называются таблицами маршрутизации. Эти таблицы рассылаются почтовым подстанциям централизовано соответствующим почтовым подразделением. Время от времени рассылаются предписания по изменению и дополнению этих таблиц. В Internet, как и любые другие действия, составление и модификация, таблиц маршрутизации (этот процесс тоже является частью маршрутизации и называется так же) определяются соответствующими правилами - протоколами ICMP (Internet Control Message Protocol), RIP (Routing Internet Protocol) и OSPF (Open Shortest Path First). Узлы, занимающиеся маршрутизацией, называются маршрутизаторами.

А откуда сеть знает, куда назначен ваш пакет данных? От вас. Если вы хотите отправить письмо и хотите, чтобы ваше письмо достигло места назначения, вы не можете просто кинуть листочек бумаги в ящик. Вам следует уложить его в стандартный конверт и написать на нем не «на деревню дедушке», как Ванька Жуков, а адрес получателя в стандартной форме. Только тогда почта сможет правильно обработать ваше письмо и доставить его по назначению. Аналогично в Internet имеется набор правил по обращению с пакетами - протоколы. Протокол Internet (IP) берет на себя заботы по адресации или по подтверждению того, что узлы понимают, что следует делать с вашими данными по пути их дальнейшего следования. Согласно нашей аналогии, протокол Internet работает так же как правила обработки почтового конверта. В начало каждого вашего послания помещается заголовок, несущий информацию об адресате, сети. Чтобы определить, куда и как доставить пакет данных, этой информации достаточно.

Адрес в Internet состоит из 4 байт. При записи байты отделяются друг от друга точками: 123.45.67.89 или 3.33.33.3 . В действительности адрес состоит из нескольких частей. Так как Internet есть сеть сетей, начало адреса говорит узлам Internet, частью какой из сетей вы являетесь. Правый конец адреса говорит этой сети, какой компьютер или хост должен получить пакет (хотя реально не все так просто, но идея такова). Каждый компьютер в Internet имеет в этой схеме уникальный адрес, аналогично обычному почтовому адресу, а еще точнее - индексу. Обработка пакета согласно адресу также аналогична. Почтовая служба знает, где находится указанное в адресе почтовое отделение, а почтовое отделение подробно знает подопечный район. Internet знает, где искать указанную сеть, а эта сеть знает, где в ней находится конкретный компьютер. Для определения, где в локальной сети находится компьютер с данным числовым IP-адресом, локальные сети используют свои собственные протоколы сетевого уровня. Например, Ethernet для отыскания Ethernet-адреса по IP-адресу компьютера, находящегося в данной сети, использует протокол ARP - протокол разрешения(в смысле различения) адресов.

Числовой адрес компьютера в Internet аналогичен почтовому индексу отделения связи. Первые цифры индекса говорят о регионе (например, 45 - это Башкирия, 63 – Томская область и т.д.), последние две цифры - номер почтового отделения в городе, области или районе. Промежуточные цифры могут относиться как к региону, так и к отделению, в зависимости от территориального деления и вида населенного пункта. Аналогично существует несколько типов адресов Internet (типы: A, B, C, D, E), которые по-разному делят адрес на поля номера сети и номера узла, от типа такого деления зависит количество возможных различных сетей и машин в таких сетях.

По ряду причин (особенно, - практических, из-за ограничений оборудования) информация, пересылаемая по сетям IP, делится на части (по границам байтов), раскладываемые в отдельные пакеты. Длина информации внутри пакета обычно составляет от 1 до 1500 байт. Это защищает сеть от монополизации

каким-либо пользователем и предоставляет всем примерно равные права. Поэтому же, если сеть недостаточно быстра, чем больше пользователей ее одновременно пользует, тем медленнее она будет общаться с каждым.

Протокол IP является дейтаграммным протоколом, то есть IP-пакет является дейтаграммой. Это совершенно не укладывается в модель ISO OSI, в рамках которой уже сетевой уровень способен работать по методу виртуальных каналов.

Одно из достоинств Internet состоит в том, что протокола IP самого по себе уже вполне достаточно для работы (в принципе). Это совершенно неудобно, но, при достаточных аскетичности, уме и упорстве удастся проделать немалый объем работы. Как только данные помещаются в оболочку IP, сеть имеет всю необходимую информацию для передачи их с исходного компьютера получателю. Работа вручную с протоколом IP напоминает нам суровые времена доперсональной компьютерной эры, когда пользователь всячески угождал ЭВМ, укрощая свои тело, дух и эстетические чувства. Об удобстве пользователя никто и не собирался думать, потому что машинное время стоило во много раз дороже человеческого. Но сейчас в аскетизме надобности уже нет. Поэтому следует построить на основе услуг, предоставляемых IP, более совершенную и удобную систему. Для этого сначала следует разобраться с некоторыми жизненно важными проблемами, которые имеют место при пересылке информации:

- Большая часть пересылаемой информации длиннее 1500 символов. Если бы почта пересылала только почтовые карточки и отказывалась бы от пересылки чего-либо большего, мы бы, например, лишились увлекательнейшего литературного жанра - эпистолярного. Не говоря уже о том, что практической пользы от такой почты было бы очень немного;
- Возможны и неудачи. Почта, нередко бывает, письма теряет; сеть тоже, бывает, теряет пакеты или искажает в пути информацию в них. В отличие

от почты, Internet может с честью выходить из таких затруднительных положений;

- Пакеты могут приходить в последовательности, отличной от начальной. Пара писем, отправленных друг за другом на днях, не всегда приходит к получателю в том же порядке; то же верно и для Internet.

Таким образом, следующий уровень Internet должен обеспечить способ пересылки больших массивов информации и позаботиться об «искажениях», которые могут возникать по вине сети.

Протокол управления передачей (TCP) и протокол пользовательских дейтаграмм (UDP)

Transmission Control Protocol - это протокол, тесно связанный с IP, который используется в аналогичных целях, но на более высоком уровне - транспортном уровне эталонной модели ISO OSI. Часто эти протоколы, по причине их тесной связи, именуют вместе, как TCP/IP. Термин «TCP/IP» обычно означает все, что связано с протоколами TCP и IP. Он охватывает целое семейство протоколов, прикладные программы и даже саму сеть. В состав семейства входят протоколы TCP, UDP, ICMP, telnet, FTP и многие другие. TCP/IP - это технология межсетевого взаимодействия, технология internet. Сеть, которая использует технологию internet, называется internet.

Сам протокол TCP занимается проблемой пересылки больших объемов информации, основываясь на возможностях протокола IP. Как это делается? Вполне здраво можно рассмотреть следующую ситуацию. Как можно переслать книгу по почте, если та принимает только письма и ничего более? Очень просто: разодрать ее на страницы и отправить страницы отдельными конвертами. Получатель, руководствуясь номерами страниц, легко сможет книгу восстановить. Этим же простым и естественным методом и пользуется TCP.

TCP делит информацию, которую надо переслать, на несколько частей.

Нумерует каждую часть, чтобы позже восстановить порядок. Чтобы пересылать эту нумерацию вместе с данными, он обкладывает каждый кусочек информации своей обложкой - конвертом, который содержит соответствующую информацию. Это и есть TCP-конверт. Получившийся TCP-пакет помещается в отдельный IP-конверт и получается IP-пакет, с которым сеть уже умеет обращаться.

Получатель (TCP-модуль (процесс)) по получении распаковывает IP-конверты и видит TCP-конверты, распаковывает и их и помещает данные в последовательность частей в соответствующее место. Если чего-то не достает, он требует переслать этот кусочек снова. В конце концов информация собирается в нужном порядке и полностью восстанавливается. Вот теперь этот массив пересылается выше к пользователю (на диск, на экран, на печать).

В действительности, это слегка утрированный взгляд на TCP. В реальности пакеты не только теряются, но и могут искажаться при передаче из-за наличия помех на линиях связи. TCP решает и эту проблему. Для этого он пользуется системой кодов, исправляющих ошибки. Существует целая наука о таких кодировках. Простейшим примером такового служит код с добавлением к каждому пакету контрольной суммы (и к каждому байту бита проверки на четность). При помещении в TCP-конверт вычисляется контрольная сумма, которая записывается в TCP-заголовок. Если при приеме заново вычисленная сумма не совпадает с той, что указана на конверте, значит что-то тут не то, - где-то в пути имели место искажения, так что надо переслать этот пакет по-новой, что и делается.

Для ясности и полноты картины, необходимо сделать здесь важное замечание: Модуль TCP разбивает поток байтов на пакеты, не сохраняя при этом границ между записями. То есть, если один прикладной процесс делает 3 записи в -порт, то совсем не обязательно, что другой прикладной процесс на другом конце виртуального канала получит из своего -порта именно 3 записи, причем именно таких (по разбиению), что были переданы с другого конца. Вся инфор-

мация будет получена исправно и с сохранением порядка передачи, но она может уже быть разбита по другому и на иное количество частей. Не существует зависимости между числом и размером записываемых сообщений с одной стороны и числом и размером считываемых сообщений с другой стороны. TCP требует, чтобы все отправленные данные были подтверждены принявшей их стороной. Он использует ожидания (таймауты) и повторные передачи для обеспечения надежной доставки. Отправителю разрешается передавать некоторое количество данных, не дожидаясь подтверждения приема ранее отправленных данных. Таким образом, между отправленными и подтвержденными данными существует окно уже отправленных, но еще не подтвержденных данных. Количество байт, которое можно передавать без подтверждения, называется размером окна. Как правило, размер окна устанавливается в стартовых файлах сетевого программного обеспечения. Так как TCP-канал является , то есть данные могут одновременно передаваться в обоих направлениях, то подтверждения для данных, идущих в одном направлении, могут передаваться вместе с данными, идущими в противоположном направлении. Приемники на обеих сторонах виртуального канала выполняют управление потоком передаваемых данных для того, чтобы не допускать переполнения буферов.

Таким образом, протокол TCP обеспечивает гарантированную доставку с установлением логического соединения в виде байтовых потоков. Он освобождает прикладные процессы от необходимости использовать ожидания и повторные передачи для обеспечения надежности. Наиболее типичными прикладными процессами, использующими TCP, являются ftp и telnet.

Большие возможности TCP даются не бесплатно, реализация TCP требует большой производительности процессора и большой пропускной способности сети. Когда прикладной процесс начинает использовать TCP, то начинают общаться модуль TCP на машине пользователя и модуль на машине сервера. Эти два оконечных модуля TCP поддерживают информацию о состоянии соединения - виртуального канала. Этот виртуальный канал потребляет ресурсы обоих

оконечных модулей TCP. Канал этот, как уже указывалось, является дуплексным. Один прикладной процесс пишет данные в TCP-порт, откуда они модулями соответствующих уровней по цепочке передаются по сети и выдаются в TCP-порт на другом конце канала, и другой прикладной процесс читает их отсюда - из своего TCP-порта. эмулирует (создает видимость) выделенную линию связи двух пользователей. Гарантирует неизменность передаваемой информации. Что входит на одном конце, выйдет с другого. Хотя в действительности никакая прямая линия отправителю и получателю в безраздельное владение не выделяется (другие пользователи могут пользовать те же узлы и каналы связи в сети в промежутках между пакетами этих), но извне это, практически, именно так и выглядит.

Как бы хорошо это не звучало, но это не панацея. Как уже отмечалось, установка TCP-виртуального канала связи требует больших расходов на инициирование и поддержание соединения и приводит к задержкам передачи. Если вся эта суета - излишество, лучше обойтись без нее. Если все данные, предназначенные для пересылки, умещаются в одном пакете, и если вас не особенно заботит надежность доставки, то можно обойтись без TCP.

Имеется другой стандартный протокол транспортного уровня, который не отягощен такими накладными расходами. Этот протокол называется UDP - User Datagram Protocol - протокол пользовательских дейтаграмм. Он используется вместо TCP. Здесь данные помещаются не в TCP, а в UDP-конверт, который также помещается в IP-конверт. Этот протокол реализует дейтаграммный способ передачи данных.

Дейтаграмма - это пакет, передаваемый через сеть независимо от других пакетов без установления логического соединения и подтверждения приема. Дейтаграмма - совершенно самостоятельный пакет, поскольку сама содержит всю необходимую для ее передачи информацию. Ее передача происходит безо всякого предварения и подготовки. Дейтаграммы, сами по себе, не содержат средств обнаружения и исправления ошибок передачи, поэтому при передаче

данных с их помощью следует принимать меры по обеспечению надежности пересылки информации. Методы организации надежности могут быть самыми разными, обычно же используется метод подтверждения приема посылкой эхо-отклика при получении каждого пакета с дейтаграммой.

UDP проще TCP, поскольку он не заботится о возможной пропаже данных, пакетов, о сохранении правильного порядка данных и так далее. UDP используется для клиентов, которые посылают только короткие сообщения и могут просто заново послать сообщение, если отклик подтверждения не придет достаточно быстро. Предположим, что вы пишите программу, которая просматривает базу данных с телефонными номерами где-нибудь в другом месте сети. Совершенно незачем устанавливать TCP связь, чтобы передать 33 или около того символов в каждом направлении. Вы можете просто уложить имя в UDP-пакет, запаковать это в IP-пакет и послать. На другом конце прикладная программа получит пакет, прочитает имя, посмотрит телефонный номер, положит его в другой UDP-пакет и отправит обратно. Что произойдет, если пакет по пути потеряется? Ваша программа тогда должна действовать так: если она ждет ответа слишком долго и становится ясно, что пакет затерялся, она просто повторяет запрос, то есть посылает еще раз то же послание. Так обеспечивается надежность передачи при использовании протокола UDP.

В отличие от TCP, данные, отправляемые прикладным процессом через модуль UDP, достигают места назначения как единое целое. Например, если процесс-отправитель производит 3 записи в UDP-порт, то процесс-получатель должен будет сделать 3 чтения. Размер каждого записанного сообщения будет совпадать с размером соответствующего прочитанного. Протокол UDP сохраняет границы сообщений, определяемые прикладным процессом. Он никогда не объединяет несколько сообщений в одно целое и не делит одно сообщение на части.

Альтернатива TCP-UDP позволяет программисту гибко и рационально использовать предоставленные ресурсы, исходя из своих возможностей и по-

требностей. Если нужна надежная доставка, то лучше может быть TCP. Если нужна доставка дейтаграмм, то - UDP. Если нужна эффективная доставка по длинному и ненадежному каналу передачи данных, то лучше использовать TCP. Если нужна эффективность на быстрых сетях с короткими соединениями, лучше всего будет UDP. Если потребности не попадают ни в одну из этих категорий, то выбор транспортного протокола не ясен. Прикладные программы, конечно, могут устранять некоторые недостатки выбранного протокола. Например, если вы выбрали UDP, а вам необходима надежность, то прикладная программа должна обеспечить надежность сама, как описано выше: требовать подтверждения, пересылки утерянных или увечных пакетов и так далее. Если вы выбрали TCP, а вам нужно передавать записи, то прикладная программа должна вставлять метки в поток байтов так, чтобы можно было различить записи.

Создание сети с человеческим лицом. Прикладное обеспечение

И вот мы имеем возможность передавать информацию между различными точками в сети. Вот теперь мы можем начать работать над созданием дружественного интерфейса Internet, позаботиться об удобстве для пользователя. Для этого мы напишем программное обеспечение, которое будет понимать язык команд, выдавать сообщения об ошибках, подсказки, использовать для адресации сетевых компьютеров при общении с пользователем имена, а не числа и так далее. В модели ISO OSI на это работают уровни выше транспортного, то есть сеансовый, представления данных и прикладной. Вся эта деятельность направлена на повышение уровня удобства работы в сети, на создание систем, позволяющих пользоваться предоставляемыми возможностями обычному пользователю сети.

Ведь большинство пользователей совсем не волнует ни наличие надежного потока битов между машинами, ни пропускная способность этих линий или тонкости и особенности используемой технологии, ни даже экзотичность этой технологии. Они хотят использовать этот битовый поток для дела, как то: пере-

слать файл, добраться до каких-то данных или просто поиграть в игру. Приложения - это части программного обеспечения. Их создают на основе сервиса TCP или UDP. Приложения позволяют пользователю достаточно просто справиться с возникшей задачей, не погружаясь в пучину технической информации о конкретной сети, о протоколах и так далее.

Прикладное обеспечение разнится очень сильно. Приложения могут быть от самодельной программы до патентованных продуктов, поставляемых различными фирмами (DEC, Microsoft и т.п.). Существует три стандартных Internet -приложения: удаленный доступ, передача файлов, электронная почта (e-mail); наряду с ними используются другие широко распространенные нестандартные приложения.

Предоставление услуг Internet построено по схеме «клиент - сервер». Предоставление услуг осуществляется совместной работой двух процессов: на компьютере пользователя и на компьютере-сервере. Процесс на компьютере пользователя называется клиентом, а на компьютере-сервере - сервером. Клиент и сервер являются, по сути, частями одной программы, взаимодействующие по виртуальной связи в сети. Сервер по указаниям клиента выполняет соответствующие действия, например, пересылает клиенту файл. Для предоставления услуги совершенно необходимо наличие двух этих модулей - клиента и сервера, и их одновременная согласованная работа. Взаимодействие клиента и сервера описывается соответствующими стандартными протоколами, поэтому клиент и сервер могут быть выпущены совершенно разными производителями и работать на разнородных компьютерах. Поэтому же существует небольшая проблема нестандартности интерфейса клиента непосредственно уже с пользователем. Это взаимодействие может иметь совершенно различную форму: интерактивную, командную и так далее. Системы команд могут различаться. Но от этого сами возможности не изменяются, поскольку клиент и сервер всегда взаимодействуют одинаково - согласно протоколу.

Так как прикладным обеспечением снабжают по большей части через ло-

кальные сети, в разговоре о приложениях возникает вышеупомянутая проблема: команды, сообщения, справки, подсказки и тому подобное в разных локальных сетях могут в той или иной степени отличаться. Об этом не следует забывать при чтении руководств пользователя: сообщения могут отличаться, но смысл их будет такой же, то же касается и команд. Даже если они слегка отличаются, не стоит волноваться, большинство приложений имеет разумную систему подсказок и описание набора команд, где вы детально и конкретно сможете разузнать все, что вам понадобится.

Системы сетевых адресов

Региональная Система Имен

Числовые адреса хороши для связи машин, люди же предпочитают имена. Очень непросто разговаривать, используя машинную адресацию (как бы это звучало: «192.112.36.5 обещает вскоре...»?), еще труднее запомнить эти адреса. Поэтому компьютерам в Internet для удобства пользователей были присвоены собственные имена. Тогда описанный разговор принимает вид: «NIC обещает вскоре...». Все приложения Internet позволяют пользоваться системными именами вместо числовых адресов.

Как мы уже упоминали, для понимания полезно использовать почтовую аналогию. Сетевые численные адреса вполне аналогичны почтовой индексации. Машины, сортирующие корреспонденцию на почтовых узлах, ориентируются именно по индексам, и только если с индексами выходит какая-то несуразность, передают почту на рассмотрение людям, которые по адресу могут определить правильный индекс почтового отделения места назначения. Людям же приятнее и удобнее иметь дело с географическими названиями - это аналоги доменных имен.

Конечно, такое именование имеет свои собственные проблемы. Прежде всего, следует убедиться, что никакие два компьютера, включенные в сеть, не имеют одинаковых имен. Должно также обеспечить преобразование имен в чи-

словые адреса, для того чтобы машины (и программы) могли понимать нас, пользующихся именами: техника по-прежнему общается на языке цифр.

В начале Internet размерами напоминала курилку, и иметь дело с именами было довольно просто. NIC создал регистратуру. Можно было послать запрос и в ответ высылали список имен и адресов. Этот файл, называется «host file» (файл рабочих ЭВМ), регулярно распространялся по всей сети - рассылался всем машинам. Имена были простыми словами, все были единственными. Если вы использовали имя, ваш компьютер просматривал этот файл и подставлял вместо имени реальный числовой адрес. Так же, как работает телефонный аппарат со встроенным списком абонентов. Все было легко, просто и замечательно. Всем хватало простых имен, в курилке был один Джон, один Марк, один Анатолий.

Но по мере развития и расширения Internet возрастало количество пользователей, хостов, а потому увеличивался и упомянутый файл. Возникали значительные задержки при регистрации и получении имени новым компьютером, стало затруднительно изыскивать имена, которые еще никто не использовал, слишком много сетевого времени затрачивалось на рассылку этого огромного файла всем машинам, в нем упомянутым. Стало очевидно, - чтобы справиться с такими темпами изменений и роста сети, нужна распределенная оперативная система, опирающаяся на новый принцип. Таковая была создана, ее назвали «доменной системой имен» - DNS, а способ адресации - способом адресации по доменному принципу. DNS иногда еще называют региональной системой наименований.

Структура региональной системы имен

Доменная система имен - это метод назначения имен путем передачи сетевым группам ответственности за их подмножество имен. Каждый уровень этой системы называется доменом. Домены в именах отделяются друг от друга точками: ed.tusur.ru, db.ed.tomsk.ru, www.yandex.ru, www.microsoft.com. В име-

ни может быть различное количество доменов, но практически их не больше пяти. По мере движения по доменам слева направо в имени, количество имен, входящих в соответствующую группу возрастает.

Первым в имени стоит название рабочей машины - реального компьютера с IP адресом. Это имя создано и поддерживается группой (например, компьютер www (это web-сервер на UNIX компьютере в учебной классе кафедры Электронных приборов) в группе ed (кафедра Электронных приборов), к которой он относится. Группа входит в более крупное подразделение tusur (Томский государственный университет систем управления и радиоэлектроники, ТУСУР), которое в свою очередь, является частью национальной сети (Российской Федерации, домен ru). Для США наименование страны по традиции опускается, там самыми крупными объединениями являются сети образовательных (edu), коммерческих (com), государственных (gov), военных (mil) учреждений, а также сети других организаций (org) и сетевых ресурсов (net).

Группа может создавать или изменять любые ей подлежащие имена. Если ed решит поставить другой компьютер, например, Windows NT, и назвать его winnt, он ни у кого не должен спрашивать разрешения, все, что от него требуется, - это добавить новое имя в соответствующую часть соответствующей всемирной базы данных, и, рано или поздно, каждый, кому потребуется, узнает об этом имени. Аналогично, если в ТУСУРе решат создать новую группу, например, schools, они (домен tusur) могут это сделать также, ни у кого на то не спрашивая никакого соизволения. И тогда, если каждая группа придерживается таких простых правил и всегда убеждается, что имена, которые она присваивает, единственны во множестве ее непосредственных подчиненных, то никакие две системы, где бы те ни были в сети Internet, не смогут заиметь одинаковых имен.

Эта ситуация совершенно аналогична ситуации с присвоением географических названий - организацией почтовых адресов. Названия всех стран различаются. Различаются названия всех областей, республик в Федерации, и эти названия утверждаются в государственном масштабе из центра (конечно, обычно

сами регионы заботятся об уникальности своих названий, поэтому здесь царит полная демократия: как республика хочет, так она и называется). В республиках - субъектах федерации - решают вопросы о названиях районов и округов, в пределах одной республики они различаются. Аналогично далее с городами и улицами городов. В разных городах могут быть улицы с одинаковыми названиями: почему бы не быть во всех городах Союза по улице Ленина или Мира? Это улицы разных городов, и их не перепутать. В пределах же одного населенного пункта улицы всенепременно имеют разные названия, причем именование этих улиц целиком и полностью под ответственностью и началом соответствующего центрального органа данного населенного пункта (мэрии, сельсовета, горсовета). Таким образом, почтовый адрес на основе географических и административных названий однозначно определяет точку назначения.

Поскольку Internet - сеть мировая, требовался также способ передачи ответственности за имена внутри стран им самим. Сейчас принята двухбуквенная кодировка государств. Это оговорено в RFC 822. Так, например, домен Канада называется ca, Россия - ru, США - us и т.д. США также включили в эту систему структурирования для всеобщности и порядка. Всего же кодов стран почти 300, из которых около 100 имеет компьютерную сеть того или иного рода. Единый каталог Internet находится у SRI International (Менло-Парк, Калифорния, США) - государственной организации.

Поиск адреса по доменному имени

Теперь вы знаете, как соотносятся домены и создаются имена. Возможно, вы теперь озадачены: а как использовать эту замечательную систему? Автоматически. Вам надо лишь употребить имя на компьютере, который понимает, как обращаться с DNS. Вам никогда не придется самим разыскивать адрес, соответствующий этому имени, или подавать специальную команду для его поиска (в UNIX - команда nslookup). Вы, конечно, можете это проделать - для собственного удовольствия, но зачем, ведь этого совсем не требуется. Все компьютеры

Internet способны пользоваться доменной системой. И работающий в сети компьютер всегда знает свой собственный сетевой адрес.

Когда вы пользуетесь именем, например, ms.tusur.ru, компьютер должен преобразовать его в адрес. Для этого он начинает запрашивать помощь у DNS-серверов. Это узлы, рабочие машины, обладающие соответствующей базой данных, в число обязанностей которых входит обслуживание такого рода запросов. DNS-сервер начинает обработку имени с правого его конца и двигается по нему влево, то есть сначала производится поиск адреса в самой большой группе (домене), потом постепенно сужает поиск. Но для начала опрашивается на предмет наличия у него нужной информации местный узел. Здесь возможны три случая:

- Местный сервер знает адрес, потому, что этот адрес содержится в его части всемирной базы данных. Например, если вы подсоединены к сети Томского государственного университета систем управления и радиоэлектроники (tusur), то ваш местный сервер должен обладать информацией о всех компьютерах локальной сети этого института (ms, muma, ed и т.д.);
- Местный сервер знает адрес, потому, что кто-то недавно уже запрашивал тот же адрес. Когда запрашивается адрес, сервер DNS придерживает его у себя в памяти некоторое время, как раз на случай, если кто-нибудь еще захочет попозже того же адреса - это повышает эффективность системы;
- Местный сервер адрес не знает, но знает как его выяснить.

Как местный сервер может разузнать запрошенный адрес? В его прикладном или системном программном обеспечении имеется информация о том, как связаться с корневым сервером. Это сервер, который знает адреса серверов имен высшего уровня (самых правых в имени), здесь это уровень государств (ранга домена ru). У него запрашивается адрес компьютера, ответственного за зону ru. Местный DNS-сервер связывается с этим более общим сервером и

запрашивает у него адрес сервера, ответственного за домен tusur.ru. Теперь уже запрашивается этот сервер и у него запрашивается адрес рабочей машины ms.

На самом деле, для повышения эффективности, поиск начинается не с самого верха, а с наименьшего домена, в который входите и вы, и компьютер, имя которого вы запросили. Например, если ваш компьютер имеет имя trailers.tomsk.ru, то опрос начнется (если имя не выяснится сразу) не со всемирного сервера, чтобы узнать адрес сервера группы ru, а сразу с группы ru, что сразу сокращает поиск и по объему, и по времени.

Этот поиск адреса совершенно аналогичен поиску пути письма без надписанного почтового индекса. Как определяется этот индекс? Все регионы пронумерованы - это первые цифры индекса. Письмо пересылается на центральный почтамт этого региона, где имеется справочник с нумерацией районов этого региона - это следующие цифры индекса. Теперь письмо идет на центральный почтамт соответствующего района, где уже знают все почтовые отделения в подопечном районе. Таким образом по географическому адресу определяется почтовый индекс, ему соответствующий. Также определяется и адрес компьютера в Internet, но путешествует не послание, а запрос вашего компьютера об этом адресе. И в отличие от случая с почтой, информация об адресе доходит до вас, как если бы районный почтамт места назначения отправлял вам письмо, любезно уведомляя вас на будущее об индексе, которого вы не изволили знать.

Замечания по региональной системе имен

Распространено несколько заблуждений, с которыми вы можете столкнуться, имея дело с именами. Приведем несколько верных утверждений в качестве опорных, чтобы вывести вас из заблуждений, или предостеречь от них:

- Части доменного имени говорят о том, кто ответственен за поддержку этого имени, то есть в чьем подчинении-ведении оно находится. Они могут вообще ничего не сообщать о владельце компьютера, соответствую-

щего этому IP-адресу, или даже (несмотря на коды стран), где же эта машина находится. Вполне можно иметь в Антарктиде машину с именем www.tusur.ru (главный web-сервер ТУСУРа, город Томск). Это совершенно ненормально, но никаким законам не противоречит.

- Части доменного имени даже не всегда указывают локальную сеть, в которой расположен компьютер. Часто доменные имена и сети перекрываются, и жестких связей между ними нет: две машины одного домена могут не принадлежать одной сети. Например, системы mx.ed.tusur.ru и ms.ed.tusur.ru могут находиться в совершенно разных сетях. И еще раз: доменные имена указывают на ответственного за домен.
- У машины может быть много имен. В частности, это верно для машин, предоставляющих какие-либо услуги, которые в будущем могут быть перемещены под опеку другой машины. Когда эти службы будут перемещены, то имя, под которым эта машина выступала в качестве такого сервера, будет передано новой машине-серверу вместе с услугами, - для внешних пользователей ничего не изменится. То есть они будут продолжать пользоваться этой службой, запрашивая ее по тому же имени, независимо от того, какой компьютер на самом деле занимается обслуживанием. Имена, по смыслу относящиеся к службе, называются «каноническими именами» или «именами» (cnames). В Internet они встречаются довольно часто.
- Для связи имена необязательны. Как-нибудь вам придет сообщение: «адресат неизвестен», что означает, что Internet не может преобразовать использованное вами имя в число, - имя более недееспособно в том виде, в котором его знает ваш компьютер. Однажды заполучив числовой эквивалент имени, ваша система перестает использовать для связи на машинном уровне доменную форму адреса.
- Запоминать лучше имена, а не числовые адреса. Некоторым кажется, что система имен это «еще одно звено в цепи, которое может выйти из

строя». Но адреса привязаны к конкретным точкам сети. Если компьютер, предоставляющий некие услуги, переносится из одного здания в другое, его сетевое расположение, а значит и адрес, скорее всего изменятся. Имя же менять не надо и не следует. Когда администратор присваивает новый адрес, ему нужно только обновить запись имени в базе данных так, чтобы имя указывало на новый адрес. Так как имя работает по-прежнему, вас совершенно не должно заботить то, что компьютер расположен уже в другом месте.

Региональная система имен, возможно, и выглядит сложно, но это одна из тех составляющих, делающих общение с сетью более простым и удобным. Несомненное преимущество доменной системы состоит в том, что она разбивает громадье Internet на набор вполне обозримых и управляемых частей. Хотя сеть включает миллионы компьютеров, все они поименованы, и именование это организовано в удобной рациональной форме, что упрощает работу.

4. Сервера World Wide Web (WWW)

Распределенная информационная гипертекстовая система World Wide Web (WWW) является одним из самых популярных, если не самым популярным, ресурсом Internet. Простота поддержки баз данных Web и проста использования программ доступа к ресурсам Web привели к тому, что скорость установки Web-серверов такова, что их количество удваивается каждые 62 дня. Интегрированные мультипротокольные интерфейсы World Wide Web объединяют в себе не только средство просмотра Web, но и доступ к FTP-архивам и средство работы с электронной почтой.

Практически, мультипротокольные программы типа Netscape Navigator, Internet Explorer, Opera и другие стали стандартным интерфейсом доступа в Сеть. Кроме этого, для разработки самих страниц Web не требуется какого-то изощренного программного обеспечения. Достаточно иметь обычный текстовый редактор и уже можно разрабатывать не только информационные страни-

цы, но и стандартные формы ввода информации. Все это подвигло разработчиков программного обеспечения заговорить о технологии World Wide Web, как о технологии, способной удовлетворить множеству требований разнообразных задач, которые встречаются в организации информационной системы корпорации. После того, как Sun объявила о возможности использования в World Wide Web мобильных кодов Java, то последние сомнения о возможности организации корпоративного информационного сервиса на основе технологии World Wide Web отпали, и вся концепция получила название Intranet.

Разберем историю проекта, архитектуру программного обеспечения и протоколы World Wide Web более подробно.

История развития, отцы-основатели, современное состояние

Что же предлагал Тим Бернерс-Ли в 1989 году и что из этого получилось? В документах «World Wide Web: Proposal for HyperText Project», направленных руководству CERN, он считал, что информационная система, построенная на принципах гипертекста, должна объединить все множество информационных ресурсов CERN, которое состояло из базы данных отчетов, компьютерной документации, списков почтовых адресов, информационной реферативной системы, наборов данных результатов экспериментов и т.п. Гипертекстовая технология должна была позволить легко «перепрыгивать» из одного документа в другой.

Проект делился на две фазы, или, как у нас принято говорить, очереди. Первая очередь (продолжительностью в три месяца) должна была показать жизнеспособность идеи проекта. В течении этого этапа работ предполагалось разработать программы-интерфейсы для работы в алфавитно-цифровом режиме и программу-интерфейс для Macintosh и NeXT, работающую в графическом режиме, сервер для доступа к ресурсам Usenet, сервер для доступа к информационно-поисковой системе CERN, гипертекстовый сервер и программу-шлюз между Internet и DECnet.

В последующие три месяца (вторая очередь) предполагалось разработать средства подготовки гипертекстовых документов, полноэкранную программу просмотра для VM/XA, X-Window-интерфейс и систему автоматической нотификации просматриваемых материалов.

Кроме программного обеспечения предполагалось разработать общий протокол обмена информацией в сети, метод отображения текста на экране компьютера, создать набор базовых документов, иллюстрирующих работу системы, который мог бы пополняться за счет документов пользователей, обеспечить поиск по ключевым словам в этом наборе документов.

Любопытно, что из проекта в обязательном порядке исключались всякие исследования, связанные с конвертированием информации из форматов каких-либо редакторов в форматы данных системы, возможностью работы с видео- и аудио-информацией, все работы, связанные с защитой информации от несанкционированного доступа.

На всю эту полугодовую работу автор просил 4-х разработчиков и одного программиста, и для каждого из них отдельное рабочее место (компьютер того типа, для которого разработчик будет писать программное обеспечение). Кроме этого требовалось приобрести коммерческое программное обеспечение, которое было бы полезно при разработке системы (Guide, KMS, FrameMaker).

Как видно, запросы были невелики, и в октябре 1990 года проект стартовал. Уже в ноябре был реализован прототип системы для NeXT, к рождеству «задышал» line mode browser, разработке которого придавалось особое значение, так как он открывал доступ к системе через telnet, а в марте его можно было уже демонстрировать. Через год в Internet был установлен анонимный telnet для доступа в систему. Первое сообщение об WWW было послано в телеконференции: alt.hypertext, com.sys.next, comp.text.sgml и comp.mail. multimedia, в августе 1991 года.

По современным меркам результаты, которых достигли разработчики к 1991 году выглядят довольно скромно, если не вдаваться в суть работы и огра-

ничиться только внешним ее проявлением. Сообщество Internet получило еще одну программу, работающую в режиме командной строки. Прошло еще целых полтора года до того момента, когда программа Mosaic, разработанная Марком Андресеном (Mark Andressen) из Национального Центра Суперкомпьютерных Приложений (NCSA), и построенная на принципах WWW, обеспечили бурный рост популярности «паутины» в Internet.

NCSA начала проект по разработке интерфейса в World Wide Web месяц спустя после объявления CERN. Одна из задач NCSA - это разработка доступных некоммерческих программ, с другой стороны NCSA изучает новые технологии на предмет их коммерческого применения в будущем. World Wide Web безусловно подходила под эти два параметра. Кроме того, спецификации WWW производили впечатление добротно выполненной академической работы с обзором литературы по данному вопросу, обилием ссылок и обоснованностью принятых решений. Мультипротокольный переносимый интерфейс в WWW, создание которого начала Группа Разработки Программного Обеспечения NCSA, был назван Mosaic. Пробная версия программы была закончена в первой половине 1993 года, а в августе 1993 была анонсирована альфа-версия для Internet.

Следует отметить, что сам проект Mosaic внес огромный вклад в развитие спецификаций World Wide Web, существенно обогатив различные компоненты системы. Разработчики Mosaic ввели в стандарты WWW большое количество новшеств. Агрессивная политика команды NCSA привела к тому, что многие программы-интерфейсы, разработанные в рамках ранних стандартов, постепенно стали отмирать, не выдержав конкуренции. Для самого NCSA это закончилось тем, что лидер команды, Марк Андресен, покинул в марте 1994 года NCSA и организовал коммерческую корпорацию Netscape. С этого момента начался новый этап борьбы, но теперь между старыми коллегами. Netscape активно навязывает свои стандарты, что приводит к тому, что документы, подготовленные с расширениями Netscape неправильно отображаются Mosaic, а документы с

расширенными возможностями NCSA могут вообще не отображаться Netscape.

Следует отметить, что проект NCSA преследовал большие цели, нежели просто программу-интерфейс в WWW. С самого начала Mosaic разрабатывалась как программа с возможностями доступа к ресурсам Internet посредством различных протоколов, в число которых входили FTP, telnet, NNTP, SMTP. Однако вначале предполагалось, что делаться это будет за счет вызова внешних, относительно Mosaic, программ. В настоящее время Netscape сам поддерживает, кроме перечисленных, протоколы доступа в Gopher и Wais. Последнее позволяет использовать Netscape, впрочем как и Mosaic, для работы вне рамок World Wide Web.

Mosaic на некоторое время затмила разработки CERN. Однако эта группа имела хорошо продуманную стратегию развития системы, которая включала в себя следующие основные моменты: разработка и поддержка стандартов спецификаций системы, разработка библиотеки свободно распространяемых мобильных кодов системы, полного комплекта средств, обеспечивающих разработку и реализацию компонентов системы на любом типе компьютера в сети, подготовка набора справочных и демонстрационных документов о состоянии сети и направлениях ее развития. Данная стратегия позволила распространять программное обеспечение, разработанное в рамках проекта в Internet, а наличие line mode broser'a позволила открыть возможности WWW для огромной аудитории пользователей алфавитно-цифровых устройств, подключенных в сеть. Некоторое время NCSA лидировала и по числу установок серверов, однако в настоящее время CERN обеспечил себе паритет и в этой области. Правда, и здесь не обошлось без "накладок". Так, форматы файлов конфигурации программы imagemap, обеспечивающей работу с графическими гипертекстовыми ссылками, у этих двух серверов различны.

Другим показателем успешного развития работ является образование W3-консорциума. Консорциум образован после подписания соглашения между Масачусетским Технологическим Институтом (MIT, USA) и Национальным

Институтом Информатики и Автоматики (INRA, France) с согласия CERN. Если не вдаваться в подробности, то смысл этого соглашения заключается в том, что все программное обеспечение аккумулируется в MIT, участники имеют право copyright на все разработанное программное обеспечение и спецификации. Программное обеспечение распространяется свободно. За представителем MIT закрепляется должность директора, а за представителем INRA - должность зам. директора. Взносы полноправных участников W3C составляют $50.000 в год, а ассоциированных членов - $5.000 в год, соглашение заключено на три года начиная с 1 октября 1994 года. Любопытно, что организации с годовым оборотом, превышающим $50 миллионов, обязаны регистрироваться как полноправные члены, и что консорциум надеется получать прибыль, превышающую $1,5 миллиона, так как предусмотрен порядок использования средств сверх этой суммы. Средства до этого предела используются на развитие системы и исследования.

Образование Netscape Corporation и W3C легко объяснимы с точки зрения роста популярности WWW. В марте 1993 года трафик World Wide Web составлял 0,1% от общего трафика сети NSF, сентябре 1993 года он уже составил 1,0% от общего трафика сети NSF. В октябре 1993 года количество зарегистрированных серверов WWW равнялось 500, а к июню 1994 года оно достигло 1500 и продолжало стремительно расти. В настоящий момент число web-серверов не поддается исчислению.

Следующим важным этапом развития технологии World Wide Web стало появление весной 1995 года языка программирования Java, анонсированного компанией Sun Microsystems. Если быть более точным, то прямое отношение к World Wide Web имеет не сам язык, а мобильные коды и возможность их интерпретации программами просмотра Web. Создав свой броузер (программу просмотра) HotJava, Sun смогла продемонстрировать, что идеология интерпретации языка разметки документов может быть расширена. В страницы теперь можно стало встраивать фрагменты программ, которые после передачи по сети активировались на компьютере пользователя, расширяя тем самым концепцию

распределенных вычислений.

К этому времени кроме Java появились еще и языки управления сценариями просмотра документов, самым известным из которых стал JavaScript. Тем самым, к середине 1996 года технология World Wide Web превратилась в полноценную гипертекстовую технологию, которая стала позволять решать большинство из тех задач, до которых доросли локальные гипертекстовые системы.

Учитывая все сказанное выше, попытаемся подробно остановиться на особенностях World Wide Web и отдельных ее компонентах, спецификациях и способах наращивания системы за счет внешнего программного обеспечения, существующем программном обеспечении и особенностях его функционирования на различных компьютерных платформах. Этим вопросам и будут посвящены следующие несколько разделов.

Понятие гипертекста

В предыдущем разделе речь шла об истории и основных вехах развития World Wide Web. В последнее время часто приходится слышать, что WWW - это очень просто. Однако, за этой кажущейся простотой скрывается хорошо продуманная сложная система. При этом следует заметить, что система бурно развивается. Познакомимся более подробно с WWW.

В 1989 году, когда Т.Бернерс-Ли предложил свою систему, в мире информационных технологий наблюдался повышенный интерес к новому и модному в то время направлению - гипертекстовым системам. Идея гипертекстовой информационной системы состоит в том, что пользователь имеет возможность просматривать документы (страницы текста) в том порядке, в котором ему это больше нравится, а не последовательно, как это принято при чтении книг. Поэтому Т. Нельсон и определил гипертекст как нелинейный текст. Достигается это путем создания специального механизма связи различных страниц текста при помощи гипертекстовых ссылок, то есть у обычного текста есть ссылки

типа "следующий-предыдущий", а у гипертекста можно построить еще сколь угодно много других ссылок. Любимыми примерами специалистов по гипертексту являются энциклопедии, Библия, системы типа "Help".

Простой, на первый взгляд, механизм построения ссылок оказывается довольно сложной задачей, так как можно построить статические ссылки, динамические ссылки, ассоциированные с документом в целом или только с отдельными его частями, то есть контекстные ссылки. Дальнейшее развитие этого подхода приводит к расширению понятия гипертекста за счет других информационных ресурсов, включая графику, аудио- и видео-информацию, до понятия гипермедиа. Тем, кто интересуется более подробно различными схемами и способами разработки гипертекстовых систем, стоит обратиться к специальной литературе.

Основные компоненты технологии World Wide Web

К 1989 году гипертекст представлял новую, многообещающую технологию, которая имела относительно большое число реализаций с одной стороны, а с другой стороны делались попытки построить формальные модели гипертекстовых систем, которые носили скорее описательный характер и были навеяны успехом реляционного подхода описания данных. Идея Т.Бернерс-Ли заключалась в том, чтобы применить гипертекстовую модель к информационным ресурсам, распределенным в сети, и сделать это максимально простым способом. Он заложил три краеугольных камня системы из четырех существующих ныне, разработав:

- язык гипертекстовой разметки документов HTML (HyperText Markup Language);
- универсальный способ адресации ресурсов в сети URL (Universal Resource Locator);
- протокол обмена гипертекстовой информацией HTTP (HyperText Transfer

Protocol).

Позже команда NCSA добавила к этим трем компонентам четвертый:

- универсальный интерфейс шлюзов CGI (Common Gateway Interface).

Java не включается в этот список намеренно, т.к. область применения этого языка гораздо шире чем простое "оживление" World Wide Web.

Идея HTML - пример чрезвычайно удачного решения проблемы построения гипертекстовой системы при помощи специального средства управления отображением. На разработку языка гипертекстовой разметки существенное влияние оказали два фактора: исследования в области интерфейсов гипертекстовых систем и желание обеспечить простой и быстрый способ создания гипертекстовой базы данных, распределенной на сети.

В 1989 году активно обсуждалась проблема интерфейса гипертекстовых систем, т.е. способов отображения гипертекстовой информации и навигации в гипертекстовой сети. Значение гипертекстовой технологии сравнивали со значением книгопечатания. Утверждалось, что лист бумаги и компьютерные средства отображения/воспроизведения серьезно отличаются друг от друга, и поэтому форма представления информации тоже должна отличаться. Наиболее эффективной формой организации гипертекста были признаны контекстные гипертекстовые ссылки, а кроме того, было признано деление на ссылки, ассоциированные со всем документом в целом и отдельными его частями.

Самым простым способом создания любого документа является его набивка в текстовом редакторе. Опыт создания хорошо размеченных для последующего отображения документов в CERN'е был - трудно найти физика, который не пользовался бы системой TeX или LaTeX. Кроме того к тому времени существовал стандарт языка разметки - Standard Generalised Markup Language (SGML).

Следует также принять во внимание, что согласно своим предложениям Т.Бернерс-Ли предполагал объединить в единую систему имеющиеся информационные ресурсы CERN, и первыми демонстрационными системами должны были стать системы для NeXT и VAX/VMS.

Обычно гипертекстовые системы имеют специальные программные средства построения гипертекстовых связей. Сами гипертекстовые ссылки хранятся в специальных форматах или даже составляют специальные файлы. Такой подход хорош для локальной системы, но не для распределенной на множестве различных компьютерных платформ. В HTML гипертекстовые ссылки встроены в тело документа и хранятся как его часть. Часто в системах применяют специальные форматы хранения данных для повышения эффективности доступа. В WWW документы - это обычные ASCII- файлы, которые можно подготовить в любом текстовом редакторе. Таким образом, проблема создания гипертекстовой базы данных была решена чрезвычайно просто.

В качестве базы для разработки языка гипертекстовой разметки был выбран SGML (Standard Generalised Markup Language). Следуя академическим традициям, Бернерс-Ли описал HTML в терминах SGML (как описывают язык программирования в терминах формы Бекуса-Наура). Естественно, что в HTML были реализованы все разметки, связанные с выделением параграфов, шрифтов, стилей и прочего, так как реализация для NeXT подразумевала графический интерфейс. Важным компонентом языка стало описание встроенных и ассоциированных гипертекстовых ссылок, встроенной графики и обеспечение возможности поиска по ключевым словам.

С момента разработки первой версии языка (HTML 1.0) прошло уже много лет. За это время произошло довольно серьезное развитие языка. Почти вдвое увеличилось число элементов разметки, оформление документов все больше приближается оформлению качественных печатных изданий, развиваются средства описания нетекстовых информационных ресурсов и способы взаимодействия с прикладным программным обеспечением. Совершенствуется ме-

ханизм разработки типовых стилей. Фактически, в настоящее время HTML развивается в сторону создания стандартного языка разработки интерфейсов как локальных, так и распределенных систем.

Вторым краеугольным камнем WWW стала универсальная форма адресации информационных ресурсов. Universal Resource Identification (URI) представляет собой довольно стройную систему, учитывающую опыт адресации и идентификации e-mail, Gopher, WAIS, telnet, ftp и т.п. Но реально из всего, что описано в URI, для организации баз данных в WWW требуется только Universal Resource Locator (URL). Без наличия этой спецификации вся мощь HTML оказалась бы бесполезной. URL используется в гипертекстовых ссылках и обеспечивает доступ к распределенным ресурсам сети. В URL можно адресовать как другие гипертекстовые документы формата HTML, так и ресурсы e-mail, telnet, ftp, Gopher, WAIS, например. Различные интерфейсные программы по-разному осуществляют доступ к этим ресурсам. Одни, как например Netscape, сами способны поддерживать взаимодействие по протоколам, отличным от протокола HTTP, базового для WWW, другие, как например Chimera, вызывают для этой цели внешние программы. Однако, даже в первом случае, базовой формой представления отображаемой информации является HTML, а ссылки на другие ресурсы имеют форму URL. Следует отметить, что программы обработки электронной почты в формате MIME также имеют возможность отображать документы, представленные в формате HTML. Для этой цели в MIME зарезервирован тип "text/html".

Третьим в нашем списке стоит протокол обмена данными в World Wide Web - HTTP (Hyper-Text Transfer Protocol). Данный протокол предназначен для обмена гипертекстовыми документами и учитывает специфику такого обмена. Так в процессе взаимодействия, клиент может получить новый адрес ресурса на сети (relocation), запросить встроенную графику, принять и передать параметры и т. п. Управление в HTTP реализовано в виде ASCII-команд. Реально, разработчик гипертекстовой базы данных сталкивается с элементами протокола

только при использовании внешних расчетных программ или при доступе к внешним, относительно WWW, информационным ресурсам, например базам данных.

Последняя составляющая технологии WWW - это уже плод работы группы NCSA - спецификация CGI (Common Gateway Interface). CGI была специально разработана для расширения возможностей WWW за счет подключения всевозможного внешнего программного обеспечения. Такой подход логично продолжал принцип публичности и простоты разработки и наращивания возможностей WWW. Если команда CERN предложила простой и быстрый способ разработки баз данных, то NCSA развила этот принцип на разработку программных средств. Надо заметить, что в общедоступной библиотеке CERN были модули, позволяющие программистам подключать свои программы к серверу HTTP, но это требовало использования этой библиотеки. Предложенный и описанный в CGI способ подключения не требовал дополнительных библиотек и буквально ошеломлял своей простотой. Сервер взаимодействовал с программами через стандартные потоки ввода/вывода, что упрощает программирование до предела. При реализации CGI чрезвычайно важное место заняли методы доступа, описанные в HTTP. И хотя реально используются только два из них (GET и POST), опыт развития HTML показывает, что сообщество WWW ждет развития и CGI по мере усложнения задач, в которых будет использоваться WWW-технология.

Архитектура построения системы

От описания основных компонентов перейдем к архитектуре взаимодействия программного обеспечения в системе World Wide Web. WWW построена по хорошо известной схеме "клиент-сервер".

Программа-клиент выполняет функции интерфейса пользователя и обеспечивает доступ практически ко всем информационным ресурсам Internet. В этом смысле она выходит за обычные рамки работы клиента только с сервером

определенного протокола, как это происходит в telnet, например. Отчасти, довольно широко распространенное мнение, что Opera или Netscape, которые безусловно являются WWW-клиентами, это просто графический интерфейс в Internet, является верным. Однако, как уже было отмечено, базовые компоненты WWW-технологии (HTML и URL) играют при доступе к другим ресурсам Mosaic не последнюю роль, и поэтому мультипротокольные клиенты должны быть отнесены именно к World Wide Web, а не к другим информационным технологиям Internet. Фактически, клиент - это интерпретатор HTML. И как типичный интерпретатор, клиент в зависимости от команд (разметки) выполняет различные функции. В круг этих функций входит не только размещение текста на экране, но и обмен информацией с сервером по мере анализа полученного HTML-текста, что наиболее наглядно происходит при отображении встроенных в текст графических образов. При анализе URL-спецификации или по командам сервера клиент запускает дополнительные внешние программы для работы с документами в форматах, отличных от HTML, например GIF, JPEG, MPEG, Postscript и т.п. Вообще говоря, для запуска клиентом программ независимо от типа документа была разработана программа Luncher, но в последнее время гораздо большее распространение получил механизм согласования запускаемых программ через MIME-типы.

Другую часть программного комплекса WWW составляет сервер протокола HTTP, базы данных документов в формате HTML, управляемые сервером, и программное обеспечение, разработанное в стандарте спецификации CGI. В настоящее время наиболее популярным HTTP-сервером является бесплатно распространяемый Apache.

База данных HTML-документов - это часть файловой системы, которая содержит текстовые файлы в формате HTML и связанные с ними графику и другие ресурсы. Особое внимание хотелось бы обратить на документы, содержащие элементы экранных форм. Эти документы реально обеспечивают доступ к внешнему программному обеспечению.

Прикладное программное обеспечение, работающее с сервером, можно разделить на программы-шлюзы и прочие. Шлюзы - это программы, обеспечивающие взаимодействие сервера с серверами других протоколов, например ftp, или с распределенными на сети серверами Oracle. Прочие программы - это программы, принимающие данные от сервера и выполняющие какие-либо действия: получение текущей даты, реализацию графических ссылок, доступ к локальным базам данных или просто расчеты.

Все, что было сказано до этого момента, можно отнести к классической схеме World Wide Web. В настоящее время следует говорить об изменении общей архитектуры.

Произошел возврат к модульной структуре сервера World Wide Web. Этот возврат был реализован в виде спецификации API. API - это спецификация разработки прикладных модулей, которые встраиваются в сервер, точнее редактируются совместно с модулями сервера. Применение во всех серверах многопотоковой технологии выполнения подзадач делает такой способ расширения возможностей сервера более экономичным с точки зрения ресурсов вычислительной установки, чем разработка CGI-скриптов.

Завершая обсуждение архитектуры World Wide Web хотелось бы еще раз подчеркнуть, что ее компоненты существуют практически для всех типов компьютерных платформ и свободно доступны в сети. Любой, кто имеет доступ в Internet, может создать свой WWW-сервер, или, по крайней мере, посмотреть информацию с других серверов.

Язык гипертекстовой разметки HTML

Язык гипертекстовой разметки HTML (HyperText Markup Language) был предложен Тимом Бернерсом-Ли в 1989 году в качестве одного из компонентов технологии разработки распределенной гипертекстовой системы World Wide Web.

Разработчики HTML пытались решить две задачи:

- дать дизайнерам гипертекстовых баз данных простое средство создания документов;
- сделать это средство достаточно мощным, чтобы отразить имевшиеся на тот момент представления об интерфейсе пользователя гипертекстовых баз данных.

Первая задача была решена за счет выбора таговой модели описания документа. Такая модель широко применяется в системах подготовки документов для печати. Примером такой системы является хорошо известный язык разметки научных документов TeX, предложенный Американским Математическим Обществом, и программы его интерпретации.

К моменту создания HTML существовал стандарт языка разметки печатных документов - SGML (Standard Generalized Markup Language), который и был взят в качестве основы HTML. Предполагалось, что такое решение поможет использовать существующее программное обеспечение для интерпретации нового языка. Однако, будучи доступным широкому кругу пользователей Internet, HTML зажил своей собственной жизнью. Вероятно, многие администраторы баз данных WWW и разработчики программного обеспечения для этой системы имеют довольно смутное представление о стандартном языке разметки SGML.

Вторым важным моментом, повлиявшим на судьбу HTML, стал выбор в качестве элемента гипертекстовой базы данных обычного текстового файла, который хранится средствами файловой системы операционной среды компьютера. Такой выбор был сделан под влиянием следующих факторов:

- такой файл можно создать в любом текстовом редакторе на любой аппаратной платформе в среде любой операционной системы.
- к моменту разработки HTML существовал американский стандарт для разработки сетевых информационных систем - Z39.50, в котором в каче-

стве единицы хранения указывался простой текстовый файл в кодировке LATIN1, что соответствует US ASCII.

Таким образом, гипертекстовая база данных в концепции WWW - это набор текстовых файлов, написанных на языке HTML, который определяет форму представления информации (разметка) и структуру связей этих файлов (гипертекстовые ссылки).

Такой подход предполагает наличие еще одной компоненты технологии - интерпретатора языка. В World Wide Web функции интерпретатора разделены между сервером гипертекстовой базы данных и интерфейсом пользователя.

Сервер, кроме доступа к документам и обработки гипертекстовых ссылок, осуществляет также препроцессорную обработку документов, в то время как интерфейс пользователя осуществляет интерпретацию конструкций языка, связанных с представлением информации.

К настоящему времени наиболее распространена третья версия языка - HTML 3.0. Если первая версия языка (HTML 1.0) была направлена на представление языка как такового, где описание его возможностей носило скорее рекомендательный характер, вторая версия языка (HTML 2.0) фиксировала практику использования конструкций языка, версия ++ (HTML++) представляла новые возможности, расширяя набор элементов HTML в сторону отображения научной информации и таблиц, а также улучшения стиля компоновки изображений и текста, то версия 3.0 призвана упорядочить все нововведения и согласовать их с существующей практикой. Кроме этого, в версии 3.0 снова делается попытка формализации интерфейса пользователя гипертекстовой распределенной системы.

Принципы построения и интерпретации HTML

Таговая модель описывает документ как совокупность элементов, каждый из которых окружен тагами. По своему значению таги близки к понятию скобок

"begin/end" в универсальных языках программирования, которые задают области действия имен локальных переменных и т. п. Таги определяют область действия правил интерпретации текстовых элементов документа. Типичным примером такого рода является таг стиля Italic, который определяет область отображения курсива.

Текст на языке HTML:

```
Текст следующий за словом "Italic" <I>отображается как
курсив</I>.
```

Текст отображаемый программой интерпретации:

Текст следующий за словом "Italic" *отображается как курсив*.

В приведенном выше примере элемент текста, который должен быть выделен курсивом, заключен между тагом начала стиля "Italic" - и тагом конца стиля - . Общая схема построения элемента текста в формате HTML может быть записана в следующем виде:

```
"элемент" := <"имя элемента" "список атрибутов">
содержание элемента
```

Конструкция перед содержанием элемента называется тагом начала элемента, а конструкция, расположенная после содержания элемента, - тагом конца элемента.

Структура гипертекстовой сети задается гипертекстовыми ссылками. Гипертекстовая ссылка - это адрес другого HTML документа, который тематически, логически или каким-либо другим способом связан с документом, в кото-

ром ссылка определена.

Для записи гипертекстовых ссылок в системе WWW была разработана специальная форма, которая называется Universe Resource Locator. Типичным примером использования этой записи можно считать следующий пример:

```
Этот текст содержит <A
HREF="http://ed.tusur.ru/altai/index.html">гипертекстовую
ссылку</A>.
```

В приведенном выше примере элемент "A", который в HTML называют якорем (anchor), использует атрибут "HREF", который обозначает гипертекстовую ссылку (HyperText REFerence), для записи этой ссылки в форме URL. Данная ссылка указывает на документ с именем "index.html" в директории "altai" на сервере "ed.tusur.ru", доступ к которому осуществляется по протоколу "http".

Гипертекстовые ссылки в HTML делятся на два класса: контекстные гипертекстовые ссылки и общие. Контекстные ссылки вмонтированы в тело документа, как это было продемонстрировано в предыдущем примере, в то время как общие ссылки связаны со всем документом в целом и могут быть использованы при просмотре любого фрагмента документа. Оба класса ссылок присутствуют в стандарте языка с самого его рождения, однако, первоначально наибольшей популярностью пользовались контекстные ссылки. Эта популярность привела к тому, что механизм использования общих ссылок практически полностью "атрофировался". Однако по мере стандартизации интерфейса пользователя и стилей представления информации разработчики языка снова вернулись к общим ссылкам и стремятся приспособить их к задачам управления этим интерфейсом.

Структура HTML-документа позволяет использовать вложенные друг в друга элементы. Собственно, сам документ - это один большой элемент с именем "HTML":

```
<HTML> Содержание документа </HTML>
```

Сам элемент HTML или гипертекстовый документ состоит из двух частей: заголовка документа (HEAD) и тела документа (BODY):

```
<HTML>
<HEAD>
Содержание заголовка
</HEAD>
<BODY>
Содержание тела документа
</BODY>
</HTML>
```

Рассмотрим пример классического документа:

```
<HTML>
<!--
Author: Evgeny Shandarov
Date:   November 18, 2006
-->
<HEAD>
<TITLE>This is a Baner</TITLE>
</HEAD>
<BODY BACKGROUND=back.gif VLINK=0000FF LINK=FF0000>
<CENTER>
<TABLE>
<TR><TD><IMG SRC="interne0.jpg"></TD>
```

```
<TD CENTER>
<H3>Информационные системы</H3>
<I>Кафедра Электронных приборов ТУСУР</I>
</TD></TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

Все, что расположено между <HTML> и </HTML> - это документ. Содержание элемента HEAD определяет заголовок документа, который состоит из двух элементов: TITLE и BASE. Вслед за заголовком начинается тело документа, которое содержит в своих первых строках некоторую вводную информацию и содержание документа, оформленное в виде списка.

Каждый документ в системе World Wide Web имеет свое имя, которое указывается в элементе TITLE заголовка документа. Его мы видим в первой строке интерфейса.

Контейнер BODY открывает тело документа. В качестве фона в этом элементе определена картинка back.gif. Эта картинка - "back.gif" - задана частичной формой спецификации URL, которая не задает полного адреса ресурса в сети.

Затем мы определили таблицу, состоящую из двух ячеек. В одной ячейке картинка, в то время как в другой - текстовый фрагмент. Текст определен как заголовок третьего уровня, который должен отображаться стилем Italic.

Протокол обмена гипертекстовой информацией (HyperText Transfer Protocol, HTTP)

HTTP - это протокол прикладного уровня, разработанный для обмена гипертекстовой информацией в сети Internet. Протокол используется одной из по-

пулярнейших систем Сети - Word Wide Web - с 1990 года.

Реальная информационная система требует гораздо большего количества функций, чем просто поиск. HTTP позволяет реализовать в рамках обмена данными набор методов доступа, базирующихся на спецификации универсального идентификатора ресурсов (Universal Resource Identifier), применяемого в форме универсального локатора ресурсов (Universe Resource Locator) или универсального имени ресурса (Universal Resource Name). Сообщения по сети при использовании протокола HTTP передаются в формате, схожим с форматом почтового сообщения Internet (RFC-822) или с форматом сообщений MIME (Multiperposal Internet Mail Exchange). HTTP используется для взаимодействия программ-клиентов с программами-шлюзами, разрешающими доступ к ресурсам электронной почты Internet (SMTP), спискам новостей (NNTP), файловым архивам (FTP), системам Gopher и WAIS. Протокол разработан для доступа к этим ресурсам посредством промежуточных программ-серверов (proxy), которые позволяют передавать информацию между различными информационными службами без потерь. Протокол реализует принцип "запрос/ответ". Запрашивающая программа - клиент - инициирует взаимодействие с отвечающей программой - сервером, и посылает запрос, включающий в себя метод доступа, адрес URI, версию протокола, похожее по форме на MIME сообщение с модификаторами типа передаваемой информации, информацию клиента, и, возможно, тело сообщения клиента. Сервер отвечает строкой состояния, включающей версию протокола и код возврата, за которой следует сообщение в форме, похожей на MIME. Данное сообщение содержит информацию сервера, метаинформацию и тело сообщения. Понятно, что в принципе, одна и та же программа может выступать и в роли сервера и в роли клиента (так собственно и происходит при использовании proxy-серверов).

При работе в Internet для обслуживания HTTP-запросов используется 80 порт TCP/IP. Практика использования протокола такова, что клиент устанавливает соединение и ждет ответа сервера. После отправки ответа сервер иниции-

рует разрыв соединения. Таким образом, при передаче сложных гипертекстовых страниц соединение может устанавливаться несколько раз. Остановимся более подробно на механизме взаимодействия и форме передаваемой информации.

Форма запроса клиента

Программа-клиент посылает после установления соединения запрос серверу. Этот запрос может быть в двух формах: в форме полного запроса и в форме простого запроса. Простой запрос содержит метод доступа и запрос ресурса. Например:

```
GET http://ed.tusur.ru/
```

В этой записи слово GET обозначает метод доступа GET, а http://ed.tusur.ru/ - это запрос ресурса. Клиенты, которые способны поддерживать версии протокола выше 0.9 должны пользоваться полной формой запроса. При использовании полной формы в запросе указываются строка запроса, несколько заголовков (заголовок запроса или общий заголовок) и, возможно, тело обозначения ресурса. В форме Бекуса-Наура общий вид полного запроса выглядит так:

```
<Полный запрос> := <Строка Запроса> (<Общий заголовок>|
<Заголовок запроса>|<Заголовок обозначения ресурса>)<сим-
вол новой строки>[<тело ресурса>]
```

Квадратные скобки здесь обозначают необязательные элементы заголовка. Строка запроса - это, практически, простой запрос ресурса. Отличие состоит в том, что в строке запроса можно указывать различные методы доступа и за запросом ресурса следует указывать версию протокола. Например, для вызова

внешней программы можно использовать следующую строку запроса:

```
POST http://ed.tusur.ru/cgi-bin/test HTTP/1.0
```

В данном случае используется метод POST и протокол версии 1.0.

Методы доступа

В настоящее время в практике World Wide Web реально используются только три метода доступа: POST, GET, HEAD.

GET - метод, позволяющий получить данные, заданные в форме URI в запросе ресурса. Если ссылаются на программу, то возвращается результат выполнения этой программы, но не текст программы. Дополнительные данные, которые надо передать для обработки, кодируются в запрос ресурса. Имеется разновидность метода GET - условный GET. При использовании этого метода сервер ответит на запрос только в том случае, если будут выполнены условия передачи. Это позволяет разгрузить сеть, избавив ее от передачи ненужной информации. Условие указывается в поле "if-Modified-Since" заголовка запроса. При использовании метода GET в поле тела ресурса возвращается собственно затребованная информация (текст HTML-документа, например).

HEAD - метод, аналогичный GET, но не возвращает тела ресурса. Используется для получения информации о ресурсе. Условного HEAD не существует. Данный метод используется для тестирования гипертекстовых ссылок.

POST - этот метод разработан для передачи большого объема информации на сервер. Им пользуются для аннотирования существующих ресурсов, посылки почтовых сообщений, работы с формами интерфейсов к внешним базам данных и внешним исполняемым программам. В отличии от GET и HEAD, в POST передается тело ресурса, которое и является информацией из поля форм или других источников ввода. В первых версиях протокола были определены и другие методы доступа (DELETE, например), но они не нашли должного при-

менения. Многие функции, которые возлагали на эти методы, можно успешно выполнять через POST.

Изменение числа методов доступа отражает практику использования HTTP. Однако, с исторической и методической точек зрения, первые версии протокола представляют несомненный интерес, особенно раздел, описывающий методы доступа. В версии 1993 года насчитывалось 13 различных методов доступа. Среди этих методов были такие, например, как:

- CHECKOUT - защита от несанкционированного доступа;
- PUT - замена содержания информационного ресурса;
- DELETE - удаление ресурса;
- LINK - создание гипертекстовой ссылки;
- UNLINK - удаление гипертекстовой ссылки;
- SPACEJUMP - переход по координатам;
- SEARCH - поиск.

Из этого списка видно, что протокол был действительно максимально ориентирован на работу с гипертекстовыми распределенными системами, причем не только с точки зрения потребителя, но и с точки зрения разработчика подобных систем. Однако, во-первых, как показывал опыт, практически не использовались методы доступа, связанные с изменением информации. Это объясняется прежде всего соображениями безопасности. Ни один администратор не позволит неизвестно кому менять информацию на его сервере. Во-вторых, методы SPACEJUMP и SEARCH были с успехом заменены на функционально аналогичные CGI-скрипты. Из этого не следует, что их возрождение невозможно, например, в язык гипертекстовой разметки вернулись ссылки, общие для всего документа, но пока их реализация в протоколе отложена. В-третьих, не нашли практической реализации методы установления/удаления ссылок LINK и UNLINK. Но необходимость в них растет и связана она с реорганизацией

сети. Многие информационные ресурсы меняют свои адреса, что вносит беспорядок в структуру сети, где и без этого начинающему пользователю трудно что-либо найти. Одним словом, вопрос о новых методах доступа все еще открыт, поэтому, видимо, спецификация HTTP еще не вышла на стадию RFC и остается в виде Internet Draft.

В обеих формах запроса важное место занимает форма запроса ресурса, которая кодируется в соответствии со спецификацией URI. Применительно к World Wide Web эта спецификация получила название URL. При обращении к серверу можно использовать как полную форму URL, так и упрощенную.

Полная форма содержит тип протокола доступа, адрес сервера ресурса, и адрес ресурса на сервере

Однако такой адрес реально нужен для работы через промежуточный сервер, так как тот может пересылать запросы. При обращении к первичному серверу клиент может опускать протокол и адрес, устанавливая взаимодействие с сервером по адресу, указанному в URL (в исходном документе), и порту 80, передавая только путь от корня сервера.

Здесь пока оставим обсуждение элементов запроса и обратимся к структуре ответа сервера. Дело в том, что элементы заголовков запроса и ответа одинаковые, и поэтому их имеет смысл обсудить после определения структуры ответа сервера.

Ответ сервера

Ответ сервера может быть, как и запрос, упрощенным или полным. При упрощенном ответе сервер возвращает только тело ресурса (например, текст HTML-документа). При полном ответе клиенту возвращается строка состояния (Status-Line), общий заголовок, заголовок ответа, заголовок ресурса и тело ресурса. В форме Бекуса-Наура полный ответ представляется следующим образом:

```
<Полный ответ> := <Строка состояния> (<Общий заголовок>|
<Заголовок ответа>|<Заголовок ресурса>) <символ новой
строки>[<тело ресурса>]
```

Строка состояния состоит из версии протокола, кода возврата и краткого описания этого кода. Например, она может выглядеть так:

```
HTTP/1.0 200 Success
```

Заголовок ответа сервера может состоять из адреса URI запрашиваемого ресурса, и/или наименования программы сервера, и/или кода идентификации для работы в защищенном режиме. Состав полей заголовка ресурса является общим и для запроса клиента и для ответа сервера, и состоит из разрешения на метод доступа, типа кодировки тела ресурса (содержания ресурса), длины тела ресурса, типа ресурса, время действия данной копии ресурса, времени последнего изменения ресурса и расширения заголовка.

Рассмотрим более подробно поля заголовка, обращая внимание на реальное применение каждого из них и возможность проявления ошибок при обработке этих полей разными программами (серверами и клиентами).

Если в заголовке ответа сервера указано предложение Location, то это значит, что требуемый ресурс находится по другому адресу и его надо запросить заново. Такая процедура называется перенаправлением. Перенаправление в заголовке будет выглядеть как:

```
Location: http://ed.tusur.ru/risk/riskform.html
```

Имя и версия сервера указываются в поле Server. При использовании сервера Церн данное поле будет выглядеть как:

```
Server: CERN/3.0 libwww/2.17
```

В заголовке может встречаться и поле контроля доступа WWW-Authenticate, которое определяет способ доступа к закрытым ресурсам. Например, это поле может выглядеть как:

```
WWW-Athenticate: Basic realm="WallyWorld"
```

Кроме рассмотренных выше полей, в заголовке могут встретиться и другие поля, которые определяют содержание тела передаваемого ресурса. Эти поля относятся к заголовку ресурса, но в ответе сервера они встречаются вперемешку с общими полями. Поле Allow определяет разрешенные методы доступа к ресурсу:

```
Allow: GET,HEAD
```

Поле Content-Encoding определяет тип кодирования передаваемого ресурса:

```
Content-Encoding: x-gzip
```

В данном случае указывается, что ресурс является заархивированным файлом в формате zip. Обычно ресурсы хранятся в виде, указанном в данном поле, и при их получении клиент должен обеспечить их преобразование в приемлемый для отображения вид. Это своего рода предварительная обработка данных, которая базируется обычно на MIME типах.

Другим полем, которое проявляет себя при работе в сети, порождая ошибки отображения ранними версиями программ просмотра или ранним версиями программ серверов, является поле Content-Length. Поле Content-Length указывает размер (количество байтов) передаваемого ресурса. Это поле указы-

вается как клиентом при работе по методу POST, так и сервером при ответе на запросы клиентов. При этом в ранних версиях программ-серверов может порождаться ошибка, вызванная возможностью вставки сервером некоторой информации в текст ресурса. Например, сервер Apache позволяет вставлять в текст HTML-документов фрагменты текста из других файлов при помощи выражения типа (технология Server Side Includes - SSI):

```
<!--#includes virtual="/include/commonheader.html" -->
```

В данном случае речь идет об общей заставке для всех документов некоторого раздела. На сервере в директории документов хранится файл одного размера, но после выполнения подстановки размер файла увеличивается, однако, сервер сообщает клиенту старый размер файла. Некоторые программы-клиенты, как правило устаревшие, неправильно обрабатывают такой ответ и информация не отображается.

Поле Content-Type определяет тип информации, передаваемой сервером. Наиболее часто используются типы text/plain - простой тест и text/html - документ в формате HTML. Для сокращения трафика по сети существует несколько полей, связанных со временем передачи информации и периодичностью ее изменения на серверах. Поле Date определяет время отправки сообщения. Информация из данного поля сохраняется в файле статистики сервера и может быть использована для анализа доступа к ресурсам сервера из сети.

Поле Expires определяет время годности ресурса для использования. Если время использования вышло, то ресурс не должен передаваться. Точнее, его не следует передавать и принимать. О поле if-Modified-Sience уже упоминалось. Оно предназначено для того, чтобы не передавать имеющиеся у клиента копии ресурса, если не были произведены его изменения. Поле Pragma используется при передаче сообщений с сервера на сервер. Реально известно только одно значение данного поля: no-cache, которое запрещает запоминать в буфере дан-

ные для последующего использования.

Поле Referer используется для того, чтобы указать, из какого ресурса была осуществлена ссылка на ресурс. Данное поле - это мечта любого администратора базы данных сети. При помощи информации из этого поля можно определить, в каких WWW-страницах прописан конкретный сервер. От этого зависит количество обращений к серверу, "качество" пользователей, время отклика на информацию, размещаемую на сервере. При необходимости можно связаться с администратором этого сервера и уведомить его об изменениях на вашем сервере.

Universal Resource Identifier - универсальный идентификатор. Спецификация универсального адреса информационного ресурса в сети

Из всех спецификаций World Wide Web только спецификация URI доведена до состояния RFC. За этим стандартом закреплен номер 1630. Выпущен этот документ в 1994 году и отражает состояние информационных ресурсов Internet на это время.

URI определяет способ записи (кодирования) адресов различных информационных ресурсов при обращении к ним из страниц WWW. Однако в последнее время данная спецификация стала встречаться и в почтовых сообщениях. При этом видимо предполагается, что пользователи почты должны использовать клиентов, поддерживающих этот формат сообщения (HTML). Реально, речь может идти о клиентах MIME.

Необходимость в URI была понятна разработчикам WWW с момента зарождения системы, так как предполагалось объединение в единую информационную среду средств, использующих различные способы идентификации информационных ресурсов. Первоначально это были FTP-архивы, информационно-поисковая система Alise, и справочная система ЦЕРН. Однако Бернерс-Ли подошел к делу основательно и разработал спецификацию, которая включала в себя обращения к FTP, Gopher, WAIS, Usenet, E-mail, Prospero, Telnet, Whois, X.

500 и, конечно, HTTP (WWW). В итоге была разработана универсальная спецификация, которая позволяет расширять список адресуемых ресурсов за счет появления новых схем.

Место применения URI - гипертекстовые ссылки, которые записываются в тагах <A HREF=URI> и <LINK HREF=URI>. Встраиваемые графические объекты также адресуются по спецификации URI в тагах <IMG SRC=URI> и <FIG SRC=URI>. Реализация URI для WWW называется URL (Uniform Resource Locator). Точнее, URL - это реализация схемы URI, отображенная на алгоритм доступа к ресурсам по сетевым протоколам. Существует еще и URN (Uniform Resource Name), которое отображает URI в пространство имен на сети. Появление URN связано с желанием адресовать части почтового сообщения MIME. Но здесь есть момент, который находится в стадии дебатов. Сообщение "живет" не более 5 дней. Если оно сохранено, то его можно превратить в другой информационный ресурс, например, WWW-страницу. Поэтому судьба URN еще не решена.

Принципы построения адреса WWW

В основу URI были заложены следующие принципы:

- Расширяемость - новые адресные схемы должны были легко вписываться в существующий синтаксис URI.
- Полнота - по возможности, любая из существовавших схем должна была описываться посредством URI.
- Читаемость - адрес должен был быть легко читаем пользователем, что вообще характерно для технологии WWW - документы вместе с ссылками могут разрабатываться в обычном текстовом редакторе.

Расширяемость была достигнута за счет выбора определенного порядка интерпретации адресов, который базируется на понятии "адресная схема".

Идентификатор схемы стоит перед остатком адреса, отделен от него двоеточием и определяет порядок интерпретации остатка.

Полнота и читаемость порождали коллизию, связанную с тем, что в некоторых схемах используется двоичная информация. Эта проблема была решена за счет формы представления такой информации. Символы, которые несут служебные функции, и двоичные данные отображаются в URI в шестнадцатеричном коде и предваряются символом "%".

Прежде, чем рассмотреть различные схемы представления адресов приведен пример простого адреса URI:

```
http://ed.tusur.ru/shandarov/index.html
```

Перед двоеточием стоит идентификатор схемы адреса - "http". Это имя отделено двоеточием от остатка URI, который называется "путь". В данном случае путь состоит из доменного адреса машины, на которой установлен сервер HTTP и пути от корня дерева сервера к файлу "index.html".

Кроме представленной выше полной записи URI, существует упрощенная. Она предполагает, что к моменту ее использования многие параметры адреса ресурса уже определены (протокол, адрес машины в сети, некоторые элементы пути). При таких предположениях автор гипертекстовых страниц может указывать только относительный адрес ресурса, то есть адрес относительно определенных базовых ресурсов.

Схемы адресации ресурсов Internet

В RFC-1630 рассмотрено 8 схем адресации ресурсов Internet и указаны две, синтаксис которых находится в стадии обсуждения.

Схема HTTP. Это основная схема для WWW. В схеме указывается ее идентификатор, адрес машины, TCP-порт, путь в директории сервера, поис-

ковый критерий и метка. Приведем несколько примеров URI для схемы HTTP:

```
http://ed.tusur.ru/shandarov/manifest.html
```

Это наиболее распространенный вид URI, применяемый в документах WWW. Вслед за именем схемы (http) следует путь, состоящий из доменного адреса машины и полного адреса HTML-документа в дереве сервера HTTP.

В качестве адреса машины допустимо использование и IP-адреса:

```
http://144.206.160.40/risk/risk.html
```

Если сервер протокола HTTP запущен на другой, отличный от 80 порт TCP, то это отражается в адресе:

```
http://144.206.130.137:8080/altai/index.html
```

При указании адреса ресурса возможна ссылка на точку внутри файла HTML. Для этого вслед за именем документа может быть указана метка внутри документа:

```
http://ed.tusur.ru/altai/volume4.html#first
```

Символ "#" отделяет имя документа от имени метки. Другая возможность схемы HTTP - передача параметров. Первоначально предполагалось, что в качестве параметров будут передаваться ключевые слова, но, по мере развития механизма CGI-скриптов, в качестве параметров стала передаваться и другая информация.

```
http://ed.tusur.ru/isindex.html?keyword1+keyword2
```

В данном примере предполагается, что документ "isindex.html" - документ с возможностью поиска по ключевым словам. При этом в зависимости от поисковой машины (программы, реализующей поиск) знак "+" будет интерпретироваться либо как "AND", либо как "OR". Вообще говоря, "+" заменяет " " (пробел) и относится к классу неотображаемых символов. Если необходимо передать такой символ в строке параметров, то следует передавать в шестнадцатеричном виде его ASCII-код.

```
http://ed.tusur.ru/isindex.html?keyword1%20keyword2
```

В данном случае имеется один параметр, в котором два слова разделены пробелом. Символ "%" обозначает начало ASCII-кода, который продолжается до первого символа, отличного от цифры.

При использовании HTML Forms параметры передаются как поименованные поля:

```
http://ed.tusur.ru/isindex.html?
field1=value1+field2=value
```

Значения "field1" и "field2" - это имена полей, а "value1" и "value" - их значения. При этом приведенному выше URI может соответствовать следующая HTML-форма:

```
<FORM ACTION=http://ed.tusur.ru/cgi-bin/test>
Введите значения полей:<BR>
Поле "field1":<INPUT NAME="filed1" VALUE="value1"><BR<
Поле "field2":<INPUT NAME="field2" VALUE="value2"><BR>
```

```
<HR>
</FORM>
```

Схема FTP. Данная схема позволяет адресовать файловые архивы FTP из программ-клиентов World Wide Web. При этом программа должна поддерживать протокол FTP. В данной схеме возможно указание не только имени схемы, адреса FTP-архива, но и идентификатора пользователя и даже его пароля. Наиболее часто данная схема используется для доступа к публичным архивам FTP:

```
ftp://ed.tusur.ru/pub/0index.txt
```

В данном случае записана ссылка на архив "ed.tusur.ru" с идентификатором "anonymous" или "ftp" (анонимный доступ). Если есть необходимость указать идентификатор пользователя и его пароль, то можно это сделать перед адресом машины:

```
ftp://nobody:password@ed.tusur.ru/users/local/pub
```

В данном случае эти параметры отделены от адреса машины символом "@", а друг от друга двоеточием. В некоторых системах можно указать и тип передаваемой информации, но данная возможность не стандартизирована. Стандарт рекомендует определять тип по характеру данных (текстовая информация - ASCII, двоичная - IMAGE). Следует также учитывать, что употребление идентификатора пользователя и его пароля не рекомендовано, так как данные передаются незашифрованными и могут быть перехвачены. Реальная защита в WWW осуществляется другими средствами и построена на других принципах.

Схема MAILTO. Данная схема предназначена для отправки почты по стандарту RFC-822 (стандарт почтового сообщения). Общий вид схемы выгля-

дит как:

```
mailto:shandarov@ed.tusur.ru
```

Схема TELNET. По этой схеме осуществляется доступ к ресурсу в режиме удаленного терминала. Обычно клиент вызывает дополнительную программу для работы по протоколу telnet. При использовании этой схемы необходимо указывать идентификатор пользователя, допускается использование пароля. Реально, доступ осуществляется к публичным ресурсам, и идентификатор и пароль являются общеизвестными, например, их можно узнать в базах данных Hytelnet.

```
telnet://guest:password@ed.tusur.ru
```

Схема FILE. WWW-технология используется как в сетевом, так и в локальном режимах. Для локального режима используют схему FILE.

```
file:///C|/text/html/index.htm
```

В данном примере приведено обращение к локальному документу на персональном компьютере с MS-DOS или MS-Windows. Следует заметить, что данная схема не может быть применена к CGI-скриптам. Очень часто, однако, пользователи пытаются применить file к скрипту, что является ошибкой. Любой скрипт может быть запущен только сервером HTTP, так как ему надо передавать параметры и данные. Клиент запускает только программы просмотра на основе MIME-типов из заголовка сообщений сервера или по расширению файла.

Существует еще несколько схем. Эти схемы практически не используются или находятся в стадии разработки, поэтому останавливаться на них мы не

будем.

Из приведенных выше примеров видно, что спецификация адресов ресурсов URI является довольно общей и позволяет проидентифицировать практически любой ресурс Internet. При этом число ресурсов может расширяться за счет создания новых схем. Они могут быть похожими на существующие, а могут и отличаться от них. Реальный механизм интерпретации идентификатора ресурса, опирающийся на URI, называется URL и пользователи WWW имеют дело именно с ним.

Common Gateway Interface - средство расширения возможностей технологии World Wide Web

Спецификация CGI была разработана в Центре Суперкомпьютерных Приложений Университета штата Иллинойс (NCSA). Работы над ней велись параллельно с Mosaic. С точки зрения общей архитектуры программного обеспечения World Wide Web, CGI определила все дальнейшее развитие системных средств. До появления этой спецификации все новые возможности реализовывались в виде модулей, включенных в библиотеку общих кодов ЦЕРН. Разработчики серверов должны были использовать эти коды для реализации программ или заменять их своими собственными аналогами. Это означало, что после компиляции сервера добавить в него новые возможности будет невозможно. CGI в корне изменила эту практику.

Главное назначение Common Gateway Interface - обеспечение единообразного потока данных между сервером и прикладной программой, которая запускается из-под сервера. CGI определяет протокол обмена данными между сервером и программой. Для тех, кто знаком с протоколом HTTP, может показаться, что CGI - это просто подмножество этого протокола. Однако это не так. Во-первых, CGI определяет порядок взаимодействия сервера с прикладной программой, в котором сервер выступает инициирующей стороной, во-вторых, CGI определяет механизм реального обмена данными и управляющими командами

в этом взаимодействии, что не определено в HTTP. Естественно, что такие понятия, как метод доступа, переменные заголовка, MIME, типы данных, заимствованы из HTTP и делают спецификацию прозрачной для тех, кто знаком с самим протоколом.

При описании различных программ, которые вызываются сервером HTTP и реализованы в стандарте CGI, используют следующую терминологию:

CGI-скрипт - программа, написанная в соответствии со спецификацией Common Gateway Interface. CGI-скрипты могут быть написаны на любом языке программирования (C, C++, PASCAL, FORTRAN и т.п.) или командном языке (shell, cshell, командный язык MS-DOS, Perl и т.п.).

Шлюз - это CGI-скрипт, который используется для обмена данными с другими информационными ресурсами Internet или приложениями-демонами. Обычная CGI-программа запускается сервером HTTP для выполнения некоторой работы, возвращает результаты серверу и завершает свое выполнение. Шлюз выполняется точно также, только, фактически, он инициирует взаимодействие в качестве клиента с третьей программой. Если эта третья программа является сервисом Internet, например, сервер Gopher, то шлюз становится клиентом Gopher, который посылает запрос по порту Gopher, а после получения ответа пересылает его серверу HTTP.

Аналогично происходит взаимодействие с серверами распределенных баз данных, например, Oracle.

Механизмы обмена данными

Собственно спецификация CGI описывает четыре набора механизмов обмена данными:

- через переменные окружения;
- через командную строку;
- через стандартный ввод;

- через стандартный вывод.

Переменные окружения. При запуске внешней программы сервер создает специфические переменные окружения, через которые передает приложению как служебную информацию, так и данные. Все переменные можно разделить на общие переменные окружения, которые генерируются при любой форме запроса, и запрос-ориентированные переменные.

К общим переменным окружения относятся:

- SERVER_SOFTWARE - определяет имя и версию сервера.
- SERVER_NAME - определяет доменное имя сервера.
- GATEWAY_INTERFACE - определяет версию интерфейса.

К запрос-ориентированным относятся:

- SERVER_PROTOCOL - протокол сервера. Вообще говоря, CGI разрабатывалась не только для применения в World Wide Web с протоколом HTTP, но и для других протоколов также, но широкое применение получила только в WWW.
- SERVER_PORT - определяет порт TCP, по которому осуществляется взаимодействие. По умолчанию для работы по HTTP используется 80 порт, но он может быть и переназначен при конфигурировании сервера.
- REQUEST_METHOD - определяет метод доступа к информационному ресурсу. Это важнейшая переменная в CGI. Разные методы доступа используют различные механизмы передачи данных. Данная переменная может принимать значения GET, POST, HEAD и т. п.
- PATH_INFO - передает программе путь, часть спецификации URL, в том виде, в котором она указана в клиентом. Реально это означает, что передается путь (адрес скрипта) в виде, указанном в HTML-документе.

- PATH_TRANSLATED - то же самое, что и PATH_INFO, но только после подстановки сервером определенных в его конфигурации вставок. Дело в том, что при конфигурировании сервера некоторым элементам (ветвям) дерева файловой системы можно назначить синонимы. Типичным примером такого сорта является назначение типа:

```
cgi-bin ------------> /usr/local/etc/httpd/cgi-bin
```

В данном случае справа указано стандартное место CGI скриптов для сервера NCSA, а слева - его синоним. При получении скриптом test управления, в переменной окружения PATH_INFO будет значение:

"/cgi-bin/test", а в PATH_TRANSLATED -
"/usr/local/etc/httpd/cgi-bin/test".

- SCRIPT_NAME - определяет адрес скрипта так, как он указан клиентом. Если не указаны параметры, то значение этой переменной будут совпадать с PATH_INFO, но если переменные указаны, то все, что следует за знаком "?" будет отброшено.

PATH_INFO ----------> "/cgi-bin/search?nuclear+isotop"
SCRIPT+NAME --------> "/cgi-bin/search"

- QUERY_STRING - переменная определяет содержание запроса к скрипту. Чрезвычайно важна при использовании метода доступа GET. Возвращаясь к примеру с адресами скрипта укажем, что в QUERY_STRING помещается все, что записано после символа "?".

QUERY_STRING -------> "nuclear+isotop"

При этом никакого преобразования строки запроса сервером не производится. Все манипулирования с содержанием QUERY_STRING возложены на скрипт.

Следующий набор переменных связан с идентификацией пользователя и его машины:

- REMOTE_HOST - доменный адрес машины, с которой осуществляется запрос.
- REMOTE_ADDR - IP-адрес запрашивающей машины.
- AUTH_TYPE - тип идентификации пользователя. Используется в случае если скрипт защищен от несанкционированного использования.
- REMOTE_USER - используется для идентификации пользователя.
- REMOTE_IDENT - данная переменная порождается сервером, если он поддерживает идентификацию пользователя по протоколу RFC-931. Рекомендовано использование этой переменной для первоначального использования скрипта.

Следующие две переменные определяют тип и длину передаваемой информации от клиента к серверу.

- CONTENT_TYPE - определяет MIME-тип данных, передаваемых скрипту. Используя эту переменную можно одним скриптом обрабатывать различные форматы данных.
- CONTENT_LENGTH - определяет размер данных в байтах, которые передаются скрипту. Данная переменная чрезвычайно важна при обмене данными по методу POST, т.к. нет другого способа определить размер данных, которые надо прочитать со стандартного ввода.

Возможна передача и других переменных окружения. В этом случае

перед именем указывается префикс "HTTP_". Отдельный случай представляют переменные, порожденные в заголовке HTML-документа в тагах META. Они передаются в заголовке сообщения и некоторые серверы могут порождать переменные окружения из этих полей заголовка.

Опции командной строки. Командная строка используется только при запросах типа ISINDEX. При HTML FORMS или любых других запросах неопределенного типа командная строка не используется. Если сервер определил, что к скрипту обращаются через ISINDEX-документ, то поисковый критерий выделяется из URL и преобразуется в параметры командной строки. При этом знаком разделения параметров является символ "+". Тип запроса определяется по наличию или отсутствию символа "=" в запросе. Если этот символ есть, то запрос не является запросом ISINDEX, если символа нет, то запрос принадлежит к типу ISINDEX. Параметры, выделенные из запроса, помещаются в массив параметров командной строки argv. При этом после из выделения происходит преобразование всех шестнадцатеричных символов в их ASCII-коды. Если число параметров превышает ограничения, установленные в командном языке, например в shell, то формирования командной строки не происходит и данные передаются только через QUERY_STRING. Вообще говоря, следует заранее подумать об объеме данных, передаваемом скрипту и выбрать соответствующий метод доступа. Размер переменных окружения тоже ограничен, и если необходимо передавать много данных, то лучше сразу выбрать метод POST, т.е. передачу данных через стандартный ввод.

Формат стандартного ввода. Стандартный ввод используется при передаче данных в скрипт по методу POST. Объем передаваемых данных задается переменной окружения CONTENT_LENGTH, а тип данных - переменной CONTENT_TYPE. Если из HTML-формы надо передать запрос типа: a=b&b=c, то CONTENT_LENGTH=7, CONTENT_TYPE=application/x-www-form-urlencoded, а первым символом в стандартном вводе будет символ "a". Следует всегда помнить, что конец файла сервером в скрипт не передается, а поэтому

завершать чтение следует по числу прочитанных символов. Позже мы разберем примеры скриптов и обсудим особенности их реализации в разных операционных системах.

Формат стандартного вывода. Стандартный вывод используется скриптом для возврата данных серверу. При этом вывод состоит из заголовка и собственно данных. Результат работы скрипта может передаваться клиенту без каких-либо преобразований со стороны сервера, если скрипт обеспечивает построение полного HTTP-заголовка, в противном случае сервер заголовок модифицирует в соответствии со спецификацией HTTP. Заголовок сообщения должен отделяться от тела сообщения пустой строкой. Обычно в скриптах указывают только три поля HTTP-заголовка:

```
Content-type, Location, Status.
```

Content-type указывается в том случае, когда скрипт сам генерирует документ "на лету" и возвращает его клиенту. В этом случае реального документа в файловой системе сервера не остается. При использовании такого сорта скриптов следует учитывать, что не все серверы и клиенты отрабатывают так, как представляется разработчику скрипта. Так, при указании Content-type: text/html, некоторые клиенты не реализуют сканирования полученного текста на предмет наличия в нем встроенной графики. Обычно в Content-type указывают текстовые типы text/plain и text/html.

Location используется для переадресации. Иногда переадресация помогает преодолеть ограничения сервера или клиента на обработку встроенной графики или серверной предобработки. В этом случае скрипт создает файл на диске и указывает его адрес в Location. Сервер, таким образом, передает реально существующий файл. В последнее время серверы стали буферизовать возвращаемые клиентам данные, что приводит к решению вопросов, связанных с повторным запуском скриптов для встраивания графики и разгрузки компьютера

с сервером HTTP.

Практика применения скриптов CGI

Применение скриптов широко практикуется в WWW. При их помощи, например, реализованы стеки графических гипертекстовых ссылок, встраивание даты в текст документов, встраивание ответов службы finger, доступ к базам данных и многое другое. Мы рассмотрим простейшие скрипты для распечатки параметров, передаваемых сервером, скрипты по обращению к shell, C скрипты, скрипты доступа к системе управления базами данных ingres и скрипт imagemap.

Простейшие скрипты и преобразование информации. Обсуждение начнем со скриптов, написанных на командном языке SHELL. Самый простой из них будет выглядеть как:

```
#!/bin/sh
echo Content-type: text/plain
echo
echo This is the result of script execution.
#The end of script
```

Первая строка определяет, что в качестве интерпретатора скрипта будет использован shell, вторая строка открывает заголовок сообщения, передаваемого скриптом серверу, и определяет тип передаваемой информации как обычный текст. Третья строка отделяет тело сообщения от его заголовка. В теле сообщения передается фраза из четвертой строки. Именно она и будет отображаться программой-интерфейсом пользователя.

В качестве следующего примера приведем скрипт, который отображает значения переменных окружения:

```
#!/bin/sh
```

```
echo Content-type: text/plain
echo
echo $REQUEST_METHOD
echo $QUERY_STRING
echo $CONTENT_TYPE
echo $CONTENT_LENGTH
#The end of script.
```

В данном скрипте пользователю будут возвращены значения указанных в строках команды echo переменных окружения.

Пользователю можно вернуть не только значения переменных окружения, но и результаты выполнения команд.

```
#!/bin/sh
echo Content-type: text/plain
echo
finger shandarov@ed.tusur.ru
#The end of script.
```

В результате выполнения этого скрипта пользователь получит информацию о пользователе shandarov с машины ed.tusur.ru.

При написании скриптов следует учитывать то, что сервер обычно стартует в момент, когда не все пути могут быть определены, поэтому при обращении к ресурсам следует указывать полные пути для этих ресурсов.

Общей проблемой, связанной с использованием скриптов, является проблема безопасности. Во-первых, скрипты очень часто не пишут сами, как, например, скрипт imagemap, а заимствуют. Понятно, что чужая программа может содержать ошибки. Поэтому лучше пользоваться библиотеками проверенных скриптов, которые рекомендованы, например, World Wide Web Consortium. Во-вторых, если пользователю разрешено иметь свои страницы, то он может получить возможность выполнять свои скрипты на сервере, что тоже приводит

к брешам в системе безопасности, особенно если это shell-скрипты. Существуют такие скрипты, которые требуют прав доступа к ресурсам машины. Эти права шире, чем права пользователя "nobody", например, при доступе к базам данных. Все эти моменты следует учитывать, как при написании скриптов, так и при разрешении использования различным группам пользователей.

Как правило, новые возможности WWW тестируются на скриптах, а затем, если эти возможности широко используются в практике, они включаются в стандарты различных компонентов системы и могут быть реализованы в новых возможностях серверов, как например imagemap и системы безопасности. Скрипты позволяют новым энтузиастам Сети приобщиться к сообществу и внести свою лепту в развитие системы.

Что такое cookie?

Cookie является решением одной из наследственных проблем HTTP спецификации. Эта проблема заключается в непостоянстве соединения между клиентом и сервером, как при FTP или Telnet сессии, т.е. для каждого документа (или файла) при передаче по HTTP протоколу посылается отдельный запрос. Включение cookie в HTTP протокол дало частичное решение этой проблемы.

Cookie это небольшая порция информации, которую сервер передает клиенту. Клиент (броузер) будет хранить эту информацию и передавать ее серверу с каждым запросом как часть HTTP заголовка. Некоторые cookie хранятся только в течение одной сессии, они удаляются после закрытия броузера. Другие, установленные на некоторый период времени, записываются в файл. Обычно этот файл называется 'cookie.txt'.

Что можно делать с помощью cookie?

Сами по себе cookies не могут делать ничего, это только лишь некоторая информация. Однако, сервер может реагировать на содержащуюся в cookies информацию. Например, в случае авторизованного доступа к чему либо через

WWW, в cookies сохраняется login и password в течение сессии, что позволяет не вводить их при запросе каждого запаролированного документа. Другой пример: cookies могут использоваться для построения персонализированных страниц. Чаще всего встречается такое - на некотором сервере Вас просят ввести свое имя, и каждый раз, когда Вы заходите на первую страницу этого сервера, Вам пишут что-то типа "Hello, your_name!". На использовании cookies также часто строят функцию оформления заказа в онлайновых магазинах, в частности, в Амазоне, такая своеобразная виртуальная корзина покупателя, как в обычном реальном супермаркете.

Формат и синтаксис Cookie

Полное описание поля Set-Cookie HTTP заголовка:

```
Set-Cookie: NAME=VALUE; expires=DATE; path=PATH;
domain=DOMAIN_NAME; secure
```

Минимальное описание поля Set-Cookie HTTP заголовка:

```
Set-Cookie: NAME=VALUE;
```

NAME=VALUE - строка символов, исключая перевод строки, запятые и пробелы. NAME-имя cookie, VALUE - значение.

expires=DATE - время хранения cookie, т.е. вместо DATE должна стоять дата в формате Wdy, DD-Mon-YYYY HH:MM:SS GMT, после которой истекает время хранения cookie. Если этот атрибут не указан, то cookie хранится в течение одного сеанса, до закрытия броузера.

domain=DOMAIN_NAME - домен, для которого значение cookie действительно. Например, domain=cit-forum.com. В этом случае значение cookie будет

действительно и для сервера cit-forum.com, и для www.cit-forum.com. Но не радуйтесь, указания двух последних периодов доменных имен хватает только для доменов иерархии "COM", "EDU", "NET", "ORG", "GOV", "MIL", и "INT". Для доменов иерархии "RU" придется указывать три периода.

Если этот атрибут опущен, то по умолчанию используется доменное имя сервера, с которого было выставлено значение cookie.

path=PATH - этот атрибут устанавливает подмножество документов, для которых действительно значание cookie. Например, указание path=/win приведет к тому, что значение cookie будет действительно для множества документов в директории /win/, в директории /wings/ и файлов в текущей директории с именами типа wind.html и windows.shtml

Если этот атрибут не указан, то значение cookie распространяется только на документы в той же директории, что и документ, в котором было установлено cookie.

secure - если стоит такой маркер, то информация cookie пересылается только через HTTPS (HTTP с использованием SSL). Если этот маркер не указан, то информация пересылается обычным способом.

Синтаксис HTTP заголовка для поля Cookie

Когда запрашивается документ с HTTP сервера, броузер проверяет свои cookie на предмет соответствия домену сервера и прочей информации. В случае, если найдены удовлетворяющие всем условиям значения cookie броузер посылает их в серверу в виде пары имя/значение:

```
Cookie: NAME1=OPAQUE_STRING1; NAME2=OPAQUE_STRING2 ...
```

В случае, если cookie принимает новое значение при имеющемся уже в броузере cookie с совпадающими NAME, domain и path, старое значение затирается новым. В остальных случаях новые cookies добавляются.

Использование expires не гарантирует сохранность cookie в течение заданного периода времени, поскольку клиент (броузер) может удалить запись вследствие нехватки выделенного места или каких-либо других лимитов.

Клиент (броузер) имеет следующие ограничения:

- всего может храниться до 300 значений cookies
- каждый cookie не может превышать 4Кбайт
- с одного сервера или домена может храниться до 20 значений cookie

Если ограничение 300 или 20 превышается, то удаляется первая по времени запись. При превышении 4К - корректность такого cookie страдает - отрезается кусок записи (с начала этой записи) равный превышению.

В случае кэширования документов, например, proxy-сервером, поле Set-cookie HTTP заголовка никогда не кэшируется.

Если proxy-сервер принимает ответ, содержащий поле Set-cookie в заголовке, предполагается, что поле таки доходит до клиента вне зависимости от статуса 304 (Not Modified) или 200 (OK).

Соответственно, если клиентский запрос содержит в заголовке Cookie, то он должен дойти до сервера, даже если установлен If-modified-since.

Примеры

Ниже приведено несколько примеров, иллюстрирующих использование cookies

Первый пример:

Клиент запрашивает документ и принимает ответ:

```
  Set-Cookie: CUSTOMER=WILE_E_COYOTE; path=/;
expires=Wednesday, 09-Nov-99 23:12:40 GMT
```

Когда клиент запрашивает URL с путем "/" на этом сервере, он посылает:

```
  Cookie: CUSTOMER=WILE_E_COYOTE
```

Клиент запрашивает документ и принимает ответ:

```
  Set-Cookie: PART_NUMBER=ROCKET_LAUNCHER_0001; path=/
```

Когда клиент запрашивает URL с путем "/" на этом сервере, он посылает:

```
Cookie: CUSTOMER=WILE_E_COYOTE;
PART_NUMBER=ROCKET_LAUNCHER_0001
```

Клиент получает:

```
  Set-Cookie: SHIPPING=FEDEX; path=/foo
```

Когда клиент запрашивает URL с путем "/" на этом сервере, он посылает:

```
  Cookie: CUSTOMER=WILE_E_COYOTE;
PART_NUMBER=ROCKET_LAUNCHER_0001
```

Когда клиент запрашивает URL с путем "/foo" на этом сервере, он посылает:

```
  Cookie: CUSTOMER=WILE_E_COYOTE;
PART_NUMBER=ROCKET_LAUNCHER_0001; SHIPPING=FEDEX
```

Второй пример:

Клиент принимает:

```
Set-Cookie: PART_NUMBER=ROCKET_LAUNCHER_0001; path=/
```

Когда клиент запрашивает URL с путем "/" на этом сервере, он посылает:

```
Cookie: PART_NUMBER=ROCKET_LAUNCHER_0001
```

Клиент принимает:

```
Set-Cookie: PART_NUMBER=RIDING_ROCKET_0023; path=/ammo
```

Когда клиент запрашивает URL с путем "/ammo" на этом сервере, он посылает:

```
Cookie: PART_NUMBER=RIDING_ROCKET_0023;
PART_NUMBER=ROCKET_LAUNCHER_0001
```

Комментарий: здесь мы имеем две пары имя/значение с именем "PART_NUMBER".

Это наследие из предыдущего примера, где значение для пути "/" прибавилось к значению для "/ammo".

Расширение технологии WWW. Язык программирования PHP

PHP был задуман примерно в конце 1994 года Расмусом Ледорфом (Rasmus Lerdorf). Ранние невыпущенные версии использовались на его домашней странице для того, чтобы следить за тем кто просматривал его интерактивное резюме. Первая используемая версия стала доступна где-то в начале 1995 и была известна как Personal Home Page Tools. Она состояла из очень упрощенно-

го движка синтаксического анализатора, который понимал только несколько специальных макрокоманд и ряд утилит, которые затем были в общем использовании на домашних страницах. Гостевые книги, счетчики и некоторые другие дополнения.

Довольно трудно дать какую-либо жесткую статистику, но отмечено, что к 1996 г. PHP/FI был использован по крайней мере на 15,000 веб-сайтах во всем мире. В середине 1997г. эта цифра выросла до более чем 50,000. В середине 1997г. также наблюдалось изменение в разработке PHP. Из частного любимого проекта Расмуса, которому способствовала горстка людей , это превратилось в намного более организованную рабочую группу. Синтаксический анализатор был заново переписан Зевом Сураски(Zeev Suraski) и Анди Гутмансом(Andi Gutmans), и этот новый синтаксический анализатор стал основой для PHP Версии 3.

Возможности PHP

HTTP-аутентификация средствами PHP

HTTP аутентификация в PHP доступна только при использовании модуля Apache. В модуле Apache PHP-скрипт, может использовать функцию Header() для отправки сообщения "Authentication Required" браузеру клиента, вызвав тем самым окно диалога Username/Password. Как только пользователь заполняет поля username и password, URL содержащий PHP-скрипт будет вызван заново с переменными $PHP_AUTH_USER, $PHP_AUTH_PW и $PHP_AUTH_TYPE содержащими введенную информацию. В данном случае обеспечивается только "Основная" аутификация.

Фрагмент примера сценария, который производит аутентификацию клиента на странице, должен быть следующим:

```
<?php
  if(!isset($PHP_AUTH_USER)) {
    Header("WWW-Authenticate: Basic realm=\"My Realm\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Text to send if user hits Cancel button\n";
    exit;
  } else {
    echo "Hello $PHP_AUTH_USER.<P>";
    echo "You entered $PHP_AUTH_PW as your password.<P>";
  }
?>
```

Вместо просто распечатывания $PHP_AUTH_USER и $PHP_AUTH_PW, Вы вероятно хотели бы проверить имя_пользователя и пароль для проверки правильности. Возможно, посылая запрос к базе данных, или, ища пользователя в dbm или текстовом файле.

Будьте внимательны при использовании браузера Internet Explorer. Он весьма придирчив к порядку заголовков. Отправка заголовка WWW-Authenticate перед заголовком HTTP/1.0 401 возможно даст аутентификацию в любом случае.

Чтобы предотвратить от записи кем-то сценарий , который определяет пароль для страницы, которая была опознана через традиционный внешний механизм, PHP_AUTH переменные не будут установлены, если допускается внешнее установление подлинности для той специфической страницы. В этом случае может быть использована переменная $REMOTE_USER чтобы идентифицировать внешне-опознанного пользователя.

Обратите внимание, однако, что вышеупомянутое не защищает от кого - то, кто может управлять не-аутентифицированным URL используя перехваченный пароль из аутентифицированных URL на том же самом сервере.

И Netscape и Internet Explorer очистит локальный кэш окна аутентификации после получения ответа сервера 401. Это эффективно как мера отключения пользователей("logout"), вынуждающая их повторно ввести их username и пароль. Некоторые используют это для отключения пользователя по истечении интервала времени("time out"), или обеспечивают кнопку "Log Out".

Поддержка file upload

PHP может принимать файлы, загруженные из любого браузера, отвечающего стандартам RFC-1867 (которыми являются, например, Netscape Navigator 3 или старше, Microsoft Internet Explorer 3 с исправлениями от Microsoft, или старше). Эта возможность позволяет людям загружать файлы. С PHP-аутификацией и функциями манипулирования файлами, вы имеете полный контроль над тем, кому позволять загружать файлы и что должно быть выполнено с файлом, если он был загружен.

Экран загрузки файла может быть организован созданием специальной формы, которая выглядит примерно так:

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_"
METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

URL должен указать на php html файл. Скрытое поле MAX_FILE_SIZE должно предшествовать полю ввода файла и означает максимально допустимый размер файла. Значение определяется в байтах. Для этого файла при успешной загрузке будут определены следующие переменные :

- $userfile - Временное имя файла под которым загруженный файл загружается в машину сервера.
- $userfile_name - Исходное имя файла в системе отправителя.
- $userfile_size - Размер загруженного файла в байтах.
- $userfile_type - Тип MIME файла, если броузер предоставил эту информацию. Например может быть "image/gif". Обратите внимание, что компонент вышеупомянутых переменных "$userfile" - это любое значение поля Name тега INPUT с TYPE=file обозначенное в форме загрузки. В приведенном выше примере формы загрузки мы назвали его "userfile".

По умолчанию файлы будут сохранены в заданном по умолчанию временном каталоге сервера. Его можно изменить, установкой переменной среды TMPDIR в среде, в которой PHP выполняется. Хотя, использование при ее установке обращения PutEnv () изнутри сценария PHP не будет работать.

Скрипт PHP, который получает загруженный файл, должен определить, что должно быть выполнено с загруженным файлом. Вы можете, например, использовать переменную $file_size, чтобы отбросить любые файлы, которые являются или слишком маленькими или слишком большими. Вы могли бы использовать переменную $file_type, чтобы отбросить любые файлы, которые не соответствуют некоторым критериям типа. В любом случае, вы должны или удалить файл из временного каталога или переместить эго в другое место.

Файл будет удален из временного каталога в конце запроса, если он не перемещен или переименован.

Поддержка HTTP cookie

PHP поддерживает HTTP cookies. Cookies - механизм для сохранения данных в удаленном браузере и, таким образом, - трэкинг или идентификация пользователей. Вы можете устанавливать файлы cookie используя функцию setcookie(). Cookies - часть HTTP заголовка, так что функция SetCookie() долж-

на вызваться прежде чем браузеру послан какая-нибудь информация для вывода.

Любой cookie, посланный Вам от клиента будет автоматически превращен в переменную PHP точно так же как данные методов GET и POST. Если вы желаете назначить множественные значения одиночному cookie - просто добавьте [] к имени cookie.

Поддержка баз данных

PHP поддерживает ряд различных баз данных, и в режиме работы в собственной системе команд и через ODBC, включая:

- Adabas D
- MySQL
- dBase
- Oracle
- Empress
- PostgreSQL
- FilePro
- Solid
- Informix
- Sybase
- InterBase
- Velocis
- mSQL
- Unix dbm

Разумеется, наибольшую популярность имеет «связка» PHP-MySQL.

Регулярные выражения

Регулярные выражения используются для сложного манипулирования строками в PHP. Функции, которые поддерживают регулярные выражения:

- ereg()
- ereg_replace()
- eregi()
- eregi_replace()
- split()

Все эти функции принимают строку регулярного выражения как их первый параметр. PHP использует расширенные регулярные выражения POSIX как определено в POSIX 1003.2. Для полного описания регулярных выражений POSIX см. соответствующие разделы руководства (regex), в каталоге regex дистрибутива PHP.

Пример регулярных выражений

```
ereg("abc",$string);
/* Возвращает 'истина', если "abc"
   найдено в строке $string. */

ereg("^abc",$string);
/* Возвращает 'истина', если "abc"
   найдено в начале строки $string. */

ereg("abc$",$string);
/*  Возвращает 'истина', если "abc"
   найдено в конце строки $string. */

eregi("(ozilla.[23]|MSIE.3)",$HTTP_USER_AGENT);
/*  Возвращает 'истина', если браузер клиента
   - Netscape 2, 3 или MSIE 3. */
```

```
    ereg("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)",
         $string,$regs);
    /* Помещает три слова - $regs[1], $regs[2] и
$regs[3], разделенные пробелом. */

    ereg_replace("^","<BR>",$string);
    /* Устанавливает тег <BR> в начало строки $string. */

    ereg_replace("$","<BR>",$string);
    /* Устанавливает тег <BR> в конец строки $string. */

    ereg_replace("\n","",$string);
    /* Отсекает символ "возврат каретки" в строке
$string. */
```

Обработка ошибок

В PHP есть 4 типа ошибок и предупреждений. Это:

- 1 - Нормальные Ошибки Функции(Normal Function Errors)
- 2 - Нормальные Предупреждения(Normal Warnings)
- 4 - Ошибки Синтаксического Анализатора(Parser Errors)
- 8 - Уведомления(Notices) : предупреждения, которые Вы можете проигнорировать но, которые могут подразумевать ошибки в вашем коде

Эти 4 типа комбинируются при определении ошибки, сообщая уровень. Ошибка по умолчанию, возвращает уровень 7, который является комбинацией 1 + 2 + 4, или все ошибки за исключением примечаний. Этот уровень может быть изменен в файле php3. ini директивой error_reporting. Он также может

быть установлен в вашем файле Apache httpd.conf директивой php3_error_reporting, или же это может быть произведено во времени выполнения сценария, с использованием функции error_reporting ().

Все выражения PHP могут также вызываться с префиксом "@", который выключает сообщение об ошибке, специфичное для этого выражения. Если ошибка произошла во время выполнения такого выражения, и допускается возможность track_errors, Вы можете найти сообщения об ошибках в глобальной переменной $php_errormsg.

Синтаксис и грамматика

Синтаксис PHP заимствован непосредственно из C, Java и Perl также повлияли на синтаксис данного языка.

Переход из HTML

Есть три способа выхода из HTML и перехода в "режим PHP кода":

```
1.  <? echo("простейший способ, инструкция обработки
SGML\n"); ?>

2.  <?php echo("при работе с XML документами делайте
так\n"); ?>

3.  <script language="php">
        echo ("некоторые редакторы (подобные FrontPage)
не любят обрабатывающие инструкции");
    </script>;

4.  <% echo("От PHP 3.0.4 можно факультативно применять
ASP-тэги"); %>
```

Разделение инструкций

Инструкции (утверждения) разделяются также как в C или Perl - точкой с запятой.

Закрывающий тэг (?>) тоже подразумевает конец утверждения, поэтому следующие записи эквивалентны:

```
<php
    echo "Это тест";
?>

<php echo "Это тест" ?>
```

HTML Формы (GET и POST)

Когда программой-обработчиком формы является PHP-скрипт, переменные этой формы автоматически доступны для данного скрипта PHP. Например, рассмотрим следующую форму:

```
<form action="foo.php3" method="post">
    Name: <input type="text" name="name"><br>
    <input type="submit">
</form>
```

При активизации формы PHP создаст переменную $name, значением которой будет то содержимое, которое было введено в поле Name: данной формы.

PHP также воспринимает массивы в контексте переменных формы, но

только одномерные. Вы можете, например, группировать взаимосвязанные переменные вместе или использовать это свойство для определения значений переменных при множественном выборе на входе:

Более сложные переменные формы:

```
<form action="array.html" method="post">
    Name: <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[]">
        <option value="warthog">Warthog
        <option value="guinness">Guinness
        </select>
    <input type="submit">
</form>
```

Если PHP-атрибут track_vars включен, через установку конфигурации track_vars или директивой <?php_track_vars?>, тогда переменные, активизированные посредством методов POST или GET, будут также находиться в глобальных ассоциативных массивах $HTTP_POST_VARS и $HTTP_GET_VARS соответственно.

Переменные окружения

PHP автоматически создает переменные окружения, как и обычные переменные.

```
echo $HOME;  /* Показывает переменную окружения HOME,
если она установлена. */
```

Хотя при поступлении информации механизмы GET, POST и Cookie также автоматически создают переменные PHP, иногда лучше явным образом прочитать переменную окружения, для того чтобы быть уверенным в получении ее правильной версии. Для этого может использоваться функция getenv(). Для установки значения переменной окружения пользуйтесь функцией putenv().

Таким образом, язык PHP является незаменимым инструментом при создании информационных систем с использованием технологий Интернет.

Список литературы

1. Компьютерные сети: Принципы, технологии, протоколы : Учебное пособие для вузов / В. Г. Олифер, Н. А. Олифер. - 3-е изд. - СПб. : Питер, 2008. - 957[3] с. - ISBN 978-5-469-00504-
2. Web-конструирование. HTML : учебное пособие / А. А. Дуванов. - СПб. : БХВ-Петербург, 2003. - 325[1] с. ISBN 5-94157-333-2 (в пер.)
3. UNIX для программистов и пользователей : пер. с англ. / Г. Гласс, К. Эйблс ; пер. А. Питько. - 3-е изд., перераб. и доп. - СПб. : БХВ-Петербург, 2004. - 820 с. - ISBN 5-94157-404-5
4. WEB-дизайн: Полное руководство : учебное пособие / С. В. Лебедев. - Харьков : Торнадо, 2001 ; Харьков : МТК-книга, 2001. - 736 с. : ил. - ISBN 966-635-063-7
5. Создание Web-страниц : Самоучитель: Пер. с англ. / Энди Шафран; Пер. М. Федорова. - СПб. : Питер, 2000. - 310[10] с. - ISBN 5-314-00072-5
6. Информационные системы : Учебник для вузов / Владимир Николаевич Петров. - СПб. : Питер, 2002. - 688 с. - ISBN 5-318-00561-6 (в пер.)

Шандаров Е.С.

Информационные системы

Учебное пособие

Усл. печ. л. Препринт
Томский государственный университет
систем управления и радиоэлектроники
634050, г.Томск, пр.Ленина, 40