Титульный лист мето
рекомендаций и ук

Форма
Ф СО ПГУ 7.18.3/40

Министерство образования и науки Республики Казахстан

Павлодарский государственный университет им. С. Торайгырова

Кафедра Информатики и информационных систем

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ И УКАЗАНИЯ

к изучению дисциплины WEB-технологии
для студентов специальности
5В070200- Информатика

Павлодар

Тема 1 Основные понятия Интернет.

Необходимость сети Интернет.

Передача данных стала фундаментальной частью вычислений. Сети, разбросанные по всему миру, собирают данные о таких разных предметах, как атмосферные условия, производство продуктов и воздушных перевозках. Группы создают электронные справочные списки, которые позволяют им получать информацию, интересную всем. Любители обмениваются программами для их домашних компьютеров. В научном мире сети данных стали необходимы, так как они позволяют ученым посылать программы и данные на удаленные суперкомпьютеры для обработки, получать результаты и обмениваться научной информацией с коллегами.

К сожалению, большинство сетей являются независимыми сущностями, созданными для удовлетворения потребностей одной группы людей. Пользователи выбирают аппаратную технологию, подходящую для их коммуникационных проблем. Более важно то, что нельзя создать универсальную сеть на основе одной аппаратной технологии, так как нет такой сети, которая удовлетворила бы все потребности. Некоторым пользователям нужна высокоскоростная сеть, соединяющая их машины, но такие сети не могут быть расширены на большие расстояния. Другим нужна более медленная сеть, которая будет соединять машины, находящиеся на расстоянии тысяч километров друг от друга.

Тем не менее, появилась новая технология, которая сделала возможным взаимное соединение большого числа разделенных физических сетей и заставила их работать как одно единое целое. Эта новая технология, называющаяся межсетевым обменом (internetworking), приспосабливает друг к другу различные аппаратные технологии, лежащие в основе физических сетей, с помощью добавления как физических соединений сетей, так и нового набора соглашений. Технология межсетевого обмена скрывает детали сетевого оборудования и позволяет компьютерам взаимодействовать вне зависимости от типа их физических соединений.

Технология межсетевого обмена является примером Взаимодействия Открытых Систем. Они называются открытыми потому, что в отличие от конкретных коммуникационных систем, продаваемых тем или иным производителем, ее спецификации доступны всем. Поэтому любой может создать программное обеспечение, необходимое для взаимодействия в объединенной сети. Более того, вся эта технология была разработана для того, чтобы упростить взаимодействие между машинами с различными аппаратными архитектурами, чтобы использовать почти любое оборудование сети с коммутацией пакетов, и чтобы позволить взаимодействие различных операционных систем.

Чтобы оценить значение межсетевой технологии, подумайте, как она повлияла на исследования. Представьте на минуту эффект взаимного соединения всех компьютеров, используемых учеными. Любой ученый может обменяться результатами эксперимента с любым другим ученым. Можно создать национальные центры данных, собирающие данные о природных явлениях и делающие их доступными для всех ученых. Компьютерные средства и программы, доступные в одном месте, могут использоваться учеными в других местах. В результате скорость, с которой осуществляются научные исследования, может резко возрасти.

TCP/IP

Правительственные агентства осознали важность и потенциал межсетевой технологии в будущем и стали финансировать исследования, которые сделали бы возможным создание национальной обьединенной сети.

Технология DARPA включает набор сетевых стандартов, описывающих детально процесс взаимодействия компьютеров, а также ряд соглашений при взаимодействии сетей и маршрутизации трафика. Официально называемый Связкой Межсетевых Протоколов

TCP/IP (а в обыденной речи - TCP/IP по именам двух основных стандартов), он может использоваться для взаимодействия компьютеров с помощью неограниченного числа сетей. Например, некоторые корпорации используют TCP/IP для связи сетей внутри их корпорации, даже если корпорация не имеет связи с внешними сетями. Другие группы используют TCP/IP для связи удаленных друг от друга мест.

Хотя технология TCP/IP интересна сама по себе, она особенно хороша из-за своей жизнеспособности, которую она продемонстрировала. Она стала базовой технологией для большого сообщества сетей, которые связывают большинство исследовательских институтов, включая университетские, объединенные или правительственные лаборатории. Национальный Научный Фонд(NSF), Министерство Энергетики(DOE), Министерство Обороны(DOD), Агентство по здравоохранению(HHS) и NASA взаимодействуют друг с другом, используя TCP/IP для соединения большого числа их исследовательских центров с центрами DARPA. Получившаяся сущность, известная как объединенный Интернет, Интернет DARPA/NSF, Интернет TCP/IP, или просто Интернет, позволяет исследователям всех связанных институтов разделять информацию с коллегами по всей стране так же легко, как если бы они были в соседней комнате. Таким образом, Интернет продемонстрировал жизнеспособность технологии TCP/IP и показал, как можно объединить большое количество разнообразных базовых сетевых технологий.

Средства Интернет

Нельзя говорить о технических деталях, лежащих в основе Интернета, не понимая средств, которые он обеспечивает.

Большая часть описания средств будет посвящена стандартам, называемым протоколами. Протоколы, такие как TCP и IP, дают формулы для передачи сообщений, описывают детали форматов сообщений и указывают, как обрабатывать ошибки. Самое важное то, что они позволяют нам рассматривать стандарты взаимодействия вне зависимости от того, на оборудовании какого производителя, они реализуются. По существу, протоколы являются для коммуникации тем, чем является языки программирования для вычислений. Язык программирования позволяет описать или понять вычисления, не зная системы команд конкретного ЦП. Аналогично, коммуникационный протокол позволяет нам описать или понять процесс передачи данных, не зная на каком оборудовании этот процесс выполняется.

Скрытие низкоуровневых деталей взаимодействия помогает улучшить производительность. Во-первых, программистам, работающим с высокоуровневыми протокольными абстракциями, не нужно знать или помнить множество деталей о конкретных параметрах оборудования. Они могут быстро создавать новые программы. Во-вторых, так как программы, разработанные, используя высокоуровневые абстракции, не ограничены архитектурой конкретной машины или конкретного сетевого оборудования, их не надо изменять при замене машины или изменении конфигурации. В-третьих, так как прикладные программы, построенные, используя высокоуровневые протоколы, независимы от используемого оборудования, они могут обеспечивать прямое взаимодействие различных машин. Программистам не нужно писать специальные версии прикладных программ для перемещения и трансляции данных для всех возможных пар типов машин. Мы увидим, что все сетевые средства описываются протоколами.

Следующие секции рассмотрят протоколы, используемые для описания средств прикладного уровня, а также протоколы, используемые при определении сетевых средств. Следующие главы опишут каждый из этих протоколов более детально.

Средства Интернета прикладного уровня.

С точки зрения пользователя, Интернет TCP/IP является набором прикладных программ, использующих сеть для выполнения полезных коммуникационных задач. Мы будем использовать термин взаимная работоспособность(interoperability) для описания способности различных вычислительных систем взаимодействовать при решении вычислительных задач. Мы утверждаем, что прикладные программы Интернета

показывают высокую степень взаимной работоспособности. Большинство пользователей, которые пользуются Интернетом, делают это, просто запуская прикладные программы , не понимая при этом технологии TCP/IP, структуры Интернета, и даже не зная пути, который проходят данные до назначения; они полагаются на то, что прикладные программы сами разберутся с этими деталями. Только программисты, пишущие такие прикладные программы, смотрят на Интернет как на сеть и понимают детали этой технологии.

Самые популярные и широко распространенные прикладные средства Интернета включают:

- ЭЛЕКТРОННУЮ ПОЧТУ. Электронная почта позволяет пользователю создать письмо и послать его человеку или группе людей. Другая часть этого приложения позволяет пользователю читать письма, которые он получил. Электронная почта была так успешна, что многие пользователи Интернета используют ее для обычной коммерческой переписки. Хотя существует много систем электронной почты, важно понимать, что использование TCP/IP делает доставку письма более надежной. Вместо того, чтобы полагаться на промежуточные машины при передаче письма, система предоставления письма в TCP/IP работает, напрямую соединяя машину отправителя с машиной получателя. Поэтому отправитель знает, что как только письмо покинуло его машину, оно успешно достигло места назначения.
- ПЕРЕДАЧУ ФАЙЛОВ. Хотя пользователи иногда и передают файлы, используя электронную почту, письмо предназначено для коротких, текстовых файлов. Протоколы TCP/IP включают прикладную программу передачи файлов, которая позволяет пользователям передавать или принимать довольно большие файлы программ или данных. Например, используя программу передачи файлов, можно скопировать с одной машины на другую большие обьемы данных, содержащие изображения со спутника, программы, написанные на Фортране или Паскале, или английский словарь. Эта система обеспечивает способ проверки личности пользователя или даже запрещение доступа. Как и письмо, передача файлов по Интернету TCP/IP надежна, так как две взаимодействующие машины делают это напрямую, не полагаясь на промежуточные машины для создания копий файла.
- УДАЛЕННЫЙ ДОСТУП. Являясь самым интересным приложением Интернета, удаленный доступ позволяет пользователю, находящемуся на одном компьютере, взаимодействовать с удаленной машиной и выполнять на ней интерактивный сеанс работы. Удаленный доступ позволяет создать впечатление, что терминал пользователя или его рабочая станция присоединены напрямую к удаленной машине, посылая каждый символ, нажатый на клавиатуре пользователя на удаленную машину и отображая каждый символ, возвращенный с удаленной машины, на экране терминала пользователя. Когда сеанс с удаленной машиной завершается, приложение возвращает пользователя в локальную систему.

Мы вернемся к каждому из этих приложений позднее, чтобы рассмотреть его более детально. Мы увидим, как они используют базовые протоколы TCP/IP и почему наличие стандартов прикладных протоколов помогло удостовериться, что они широко распространены.

Средства Интернета сетевого уровня.

Программист, который пишет прикладные программы, использующие протоколы TCP/IP, имеет совершенно другое представление об Интернете, чем пользователь, который просто запускает прикладные программы, такие как электронная почта. На сетевом уровне Интернет предоставляет два основных типа сервиса, который используют прикладные программы. И хотя на данном этапе несущественно понимание деталей этих средств, их нельзя опустить при любом обзоре TCP/IP:

- ДЕЙТАГРАММНОЕ СРЕДСТВО ДОСТАВКИ ПАКЕТОВ. Это средство, которое будет впоследствии детально обьяснено, образует основу всех других средств Интернета. Доставка без соединения (дейтаграмная доставка) является абстракцией

сервиса, который предоставляет большинство сетей с коммутацией пакетов. Это просто означает, что Интернет TCP/IP определяет маршрут передачи небольшого сообщения от одной машины к другой, основываясь только на адресной информации, находящейся в сообщении. Так как дейтаграмное средство маршрутизирует каждый пакет отдельно, оно не гарантирует надежной доставки пакетов в том порядке, в котором они были посланы. Так как это средство обычно напрямую отображается на лежащее в его основе оборудование, средство без соединения очень эффективно. Более того, использование доставки пакетов без соединения в качестве основы всех средств Интернета делает протоколы TCP/IP адаптируемыми к широкому диапазону сетевого оборудования.

- НАДЕЖНОЕ ПОТОКОВОЕ ТРАНСПОРТНОЕ СРЕДСТВО. Большинству приложений требуется нечто большее, чем простая доставка пакетов, так как они требуют от коммуникационного программного обеспечения автоматического восстановления при ошибках передачи, потере пакетов или сбоях промежуточных маршрутизаторов на пути между отправителем до получателем. Надежное транспортное средство обрабатывает эти ситуации. Оно позволяет приложению на одном компьютере устанавливать "соединение" с приложением на другом компьютере, а затем посылать большие обьемы данных по соединению, как если бы это было прямое аппаратное соединение. На самом деле, конечно, протоколы взаимодействия делят поток данных на маленькие сообщения и посылают их затем по одному, ожидая от получателя подтверждения приема.

Много сетей обеспечивает базовые средства, аналогичные описанным выше, поэтому кое-кто может удивиться:"Чем же отличаются средства TCP/IP от других?". Основными отличиями являются:

- НЕЗАВИСИМОСТЬ ОТ СЕТЕВОЙ ТЕХНОЛОГИИ. Хотя TCP/IP и основывается на удобной пакетной технологии, он независим от оборудования конкретного производителя. Обьединенный Интернет включает большое число сетевых технологий от сетей, предназначенных для работы в одном здании, до сетей, работающих на больших расстояниях. Протоколы TCP/IP определяют элемент передачи данных, называемый дейтаграммой, и описывают, как передавать дейтаграммы по конкретной сети.
- ВСЕОБЩАЯ СВЯЗНОСТЬ. Интернет TCP/IP позволяет любой паре компьютеров, присоединенных к нему, взаимодействовать друг с другом. Каждому компьютеру назначается адрес, который известен по всему Интернету. Каждая дейтаграмма содержит адреса отправителя и получателя. Промежуточные маршрутизаторы используют адрес получателя для того, чтобы принимать решение о дальнейшем маршруте дейтаграммы.
- МЕЖКОНЦЕВЫЕ ПОДТВЕРЖДЕНИЯ. Протоколы TCP/IP Интернета обеспечивают подтверждения между отправителем и получателем, а не между отправителем и промежуточными машинами на пути, даже когда две машины не связаны общей физической сетью.
- СТАНДАРТНЫЕ ПРИКЛАДНЫЕ ПРОТОКОЛЫ. Помимо базовых средств транспортного уровня(таких, как надежные потоковые соединения), протоколы TCP/IP включают стандарты для наиболее часто используемых приложений, таких как электронная почта, передача файлов и удаленный доступ. Поэтому при разработке прикладных программ, использующих TCP/IP, программисты часто могут обнаружить, что существующее программное обеспечение уже обеспечивает коммуникационные средства, которые им нужны.

Следующие темы рассмотрят в деталях средства, которые предлагаются программисту, а также большинство стандартов прикладных протоколов.

История создания сети Интернет

Частью того, что делает Интернет столь замечательным, является почти повсеместное его использование, а также его размеры и темпы роста обьединенного Интернета. DARPA начала работы в направлении разработки межсетевой технологии в середине 70-х, но архитектура и протоколы приняли форму, в которой они известны сейчас, лишь в 1977-1979 годах. В это время DARPA была известна как основное агентство, финансирующее исследования в области сетей с коммутацией пакетов, и внедрила множество новшеств в этой области в хорошо известную ARPANET. ARPANET использовала обычные выделенные линии точка-точка для соединения компьютеров, но DARPA также финансировала использование коммутации пакетов в радиосетях и спутниковых линиях связи. По существу растущее разнообразие аппаратных сетевых технологий вынудило DARPA изучить межсетевое взаимодействие и продвинуться по направлению к обьединенной сети.

Доступность результатов исследований, финансировавшихся DARPA, привлекла внимание нескольких исследовательских групп, особенно тех исследователей, кто уже имел опыт использования пакетной коммутации в ARPANET. DARPA собирало неформальные встречи исследователей для обмена идеями и обсуждения результатов экспериментов. С 1979 года в проект TCP/IP включилось так много исследователей, что DARPA образовало неформальный комитет для координации и управления разработкой протоколов и архитектур развивающегося обьединенного Интернета. Названная Группа по Конфигурации и Управлению Интернетом(ICCB), эта группа регулярно собиралась до 1983 года, когда она была реорганизована.

Обьединенный Интернет начал существовать с 1980 года, когда DARPA начала устанавливать на машинах, присоединенных к ее исследовательской сети, новый протоколы TCP/IP. ARPANET вскоре после создания стал магистральной сетью нового Интернета и был использован для большинства из ранних экспериментов с TCP/IP. Переход к технологии Интернета был завершен в январе 1983 года, когда секретариат МО США установил, что все компьютеры, присоединенные к глобальным сетям, используют TCP/IP. В это же самое время Оборонное Коммуникационное Агентство(DCA) разделило ARPANET на две отдельные сети, одна для дальнейших исследований и одна для военной связи. За исследовательской сетью осталось имя ARPANET, а военная часть, которая была несколько больше, получила название MILNET.

Для того чтобы заставить исследователей в университетах использовать новые протоколы, DARPA стала продавать их реализацию по низкой цене. В это время большинство университетских факультетов компьютерных наук использовали версию операционной системы UNIX, разработанную в программном отделении Берклиевского Университета в Калифорнии, чаще называемую Berkeley UNIX или BSD UNIX. Финансировав создание фирмой Bolt Beranek and NewMan, Inc. (BBN) реализации протоколов TCP/IP для UNIX и финансировав интеграцию этих протоколов в программные продукты, производимые отделением в Berkeley, DARPA смогла организовать взаимодействие с 90% всех компьютерных факультетов университетов. Новое программное обеспечение с протоколами появилось вовремя, так как многие факультеты сразу же приобретали еще компьютеры и соединяли их как локальные сети. Факультетам требовались протоколы взаимодействия, а других протоколов в то время не было в общем пользовании.

Берклиевское программное отделение стало популярным, так как оно предлагало не только базовые протоколы TCP/IP. Помимо стандартных прикладных программ TCP/IP, Беркли предлагало набор утилит для работы с сетью, которые напоминали средства UNIX, используемые на одной машине. Главное преимущество утилит Беркли заключалось в их сходстве со стандартным UNIXом. Например, опытный пользователь UNIX может быстро научиться пользоваться утилитой копирования удаленных файлов Беркли(rcp), так как он

ведет себя точно так, как утилита копирования файлов в UNIX, за исключением того, что она позволяет пользователям копировать файлы на удаленную машину или с нее.

Помимо набора служебных программ UNIX Беркли обеспечивает новую абстракцию операционной системы известную как порт(socket), которая позволяет прикладным программам получать доступ к коммуникационным протоколам. Являясь обобщением механизма UNIX для ввода-вывода, порт имеет опции для нескольких типов сетевых протоколов помимо TCP/IP. Ее принципы стали обсуждаться со времени ее разработки, и многие разработчики операционных систем предложили альтернативные варианты. Независимо от своих достоинств, введение абстракции порта было важным, так как позволяло программистам использовать протоколы TCP/IP с минимумом затрат. Поэтому, это стимулировало разработчиков экспериментировать с TCP/IP.

Успех технологии TCP/IP и Интернета в университетской среде вынудил другие группы тоже использовать его. Учитывая, что сетевое взаимодействие вскоре станет важной частью научных исследований, NSF принял активное участие в расширении Интернета TCP/IP среди ученых. Начиная с 1985 года, он начал претворять в жизнь программу создания сетей на основе его шести суперкомпьютерных центров. В 1986 он расширил деятельность в этом направлении, начав финансировать новую глобальную магистральную сеть, названную NSFNET, которая впоследствии связала все суперкомпьютерные центры между собой и ARPANET. Наконец, в 1986 NSF начал частично финансировать многие региональные сети, каждая из которых сейчас соединяет основные научно-исследовательские центры в этом районе. Все сети, финансировавшиеся NSF, используют протоколы TCP/IP, и все являются частью обьединенного Интернета.

За семь лет после своего создания Интернет обьединил сотни индивидуальных сетей, размещенных в США и Европе. Он соединил почти 20000 компьютеров в университетах, правительственных и частных исследовательских лабораториях. Как размер, так и использование Интернета продолжают расти быстрее, чем предполагалось. К концу 1987 года было установлено, что его рост достиг 15% в месяц и оставался таким последние два года. В 1990 году обьединенный Интернет включал более 3000 активных сетей и более чем 200000 компьютеров.

Использование протоколов TCP/IP и рост Интернета не ограничивались проектами, финансирующимися правительством. Основные компьютерные корпорации присоединилось к Интернету, так же как и множество других больших корпораций, включая: нефтяные компании, автомобильные концерны, электронные фирмы и телефонные компании. Вдобавок, многие компании используют протоколы TCP/IP в своих внутренних сетях, даже если они и не присоединены к обьединенному Интернету.

Быстрое расширение привело к проблемам диапазонов, непредусмотренным в исходном проекте, и заставило разработчиков найти технологии для управления большими, распределенными ресурсами. В исходном проекте, например, имена и адреса всех компьютеров, присоединенных к Интернету, хранились в одном файле, который редактировался вручную и затем распространялся по всему Интернету. Но в середине 1980 года стало ясно, что центральная база данных неэффективна. Во-первых, запросы на обновление файла скоро должны были превысить возможности людей, обрабатывавших их. Во-вторых, даже если существовал корректный центральный файл, не хватало пропускной способности сети, чтобы позволить либо частое распределение его по всем местам, либо оперативный доступ к нему из каждого места.

Были разработаны новые протоколы и стала использоваться система имен по всему обьединенному Интернету, которая позволяла любому пользователю автоматически определять адрес удаленной машины по ее имени. Известный как Доменная Система Имен, этот механизм основывается на машинах, называемых серверами имен, отвечающих на запросы об именах. Нет одной машины, содержащей всю базу данных об именах. Вместо этого, данные распределены по нескольким машинам, которые используют протоколы TCP/IP для связи между собой при ответе на запросы.

Рекомендуемая литература [2, 3, 4]

Тема 2 Протоколы сети Интернет и их стандартизация.

Многие из протоколов предшествовали Интернету, поэтому возникает вопрос: "Почему разработчики Интернета придумали новые протоколы, когда уже существует так много международных стандартов?»

Но связка протоколов TCP/IP не игнорировала международных стандартов. Она появилась просто потому, что существующие стандарты не удовлетворяли потребностям. Философия использования стандартов, когда они появляются, также означает, что когда появятся международные стандарты и обеспечат ту же самую взаимную работоспособность, что и TCP/IP, Интернет перейдет с TCP/IP на эти новые стандарты. Эти идеи согласуются с политикой федерального правительства, которое приняло Профиль Открытых Систем, который описывает использование межсетевой технологии МОС везде, где эта технология обеспечивает возможности, эквивалентные TCP/IP.

Важно понимать, что Интернет не является новым видом физической сети. На самом деле это метод взаимного соединения физических сетей и набор соглашений для использования сетей, которые позволяют компьютерам взаимодействовать друг с другом. В то время как аппаратная технология играет небольшую роль при концептуальном проектировании, важно понимать разницу между низкоуровневыми механизмами, обеспечиваемыми самим оборудованием, и высокоуровневыми средствами, которые обеспечивает программное обеспечение протоколов Интернета. Также важно понимать, как средства, обеспечиваемые технологией коммутации пакетов, влияют на наш выбор абстракций высокого уровня.

Два подхода к сетевому взаимодействию

Независимо от того, обеспечивают ли они соединение между компьютерами или между компьютерами и терминалами, коммуникационные сети могут быть разделены на два основных типа: с коммутацией каналов и коммутацией пакетов. Сети с коммутацией каналов работают, образуя выделенное соединение(канал) между двумя точками. Телефонная сеть США использует технологию с коммутацией каналов - телефонный вызов устанавливает канал от вызывающего телефона через локальную АТС, по линиям связи, к удаленной АТС, и, наконец, к отвечающему телефону. Пока существует канал, телефонное оборудование постоянно опрашивает микрофон, кодирует полученное значение в цифровой форме, и передает его по этому каналу к получателю. Отправителю гарантируется, что опросы будут доведены и воспроизведены, так как канал обеспечивает скорость 64 Кбит/с, которой достаточно для передачи оцифрованного голоса. Преимущество коммутации каналов заключается в ее гарантированной пропускной способности: как только канал создан, ни один сетевой процесс не уменьшит пропускной способности этого канала. Недостатком при коммутации каналов является ее стоимость: платы за каналы являются фиксированными и независимыми от трафика. Например, можно заплатить за телефонный вызов, даже если две разговаривающие стороны вообще ничего не говорили.

Сети с коммутацией пакетов, тип, обычно используемый при соединении компьютеров, используют совершенно другой подход. В сетях с коммутацией пакетов трафик сети делится на небольшие части, называемые пакетами, которые объединяются в высокоскоростных межмашинных соединениях. Пакет, который обычно содержит только несколько сотен байтов данных, имеет идентификатор, который позволяет компьютерам в сети узнавать, предназначен ли он им, и если нет, то помогает им определить, как послать его в указанное место назначения. Например, файл, передаваемый между двумя

машинами, может быть разбит на большое число пакетов, которые посылаются по сети по одному. Оборудование сети доставляет пакеты к указанному месту назначения, а сетевое программное обеспечение собирает пакеты опять в один файл. Главным преимуществом коммутации пакетов является то, что большое число соединений между компьютерами может работать одновременно, так как межмашинные соединения разделяются между всеми парами взаимодействующих машин. Недостатком ее является то, что по мере того как возрастает активность, данная пара взаимодействующих компьютеров получает все меньше сетевой пропускной способности. То есть, всякий раз, когда сеть с коммутацией пакетов становится перегруженной, компьютеры, использующие сеть, должны ждать, пока они не смогут послать следующие пакеты.

Несмотря на потенциальный недостаток не гарантируемой сетевой пропускной способности, сети с коммутацией пакетов стали очень популярными. Причинами их широкого использования являются стоимость и производительность. В связи с тем, что к сети может быть подключено большое число машин, требуется меньше соединений и стоимость остается низкой. Так как инженеры смогли создать высокоскоростное сетевое оборудование, с пропускной способностью обычно проблем не возникает. Так много компьютерных соединений использует коммутацию пакетов, что далее в книге термин СЕТЬ будет обозначать только сеть с коммутацией пакетов.

Глобальные сети

Сети с коммутацией пакетов, которые разрослись до больших географических размеров(например, континентальной части США), сильно отличаются от сетей, имеющих небольшие размеры(например, одну комнату). Чтобы помочь охарактеризовать различия в пропускной способности и способах использования, технологии коммутации пакетов часто делят на три большие категории: глобальные сети(WAN), городские сети(MAN) и локальные сети(LAN). Технологии WAN, иногда называемые long haul networks(буквально - сети дальних перевозок), позволяют взаимодействующим местам быть достаточно далеко друг от друга и предназначены для использования на больших расстояниях. Обычно WAN работают на более низких скоростях, чем другие технологии, и имеют гораздо большие паузы при соединении. Обычно скорости WAN лежат в диапазоне от 9.6 Кбит/с до 45 Мбит/с.

Самый новый вид сетевого оборудования, технологии MAN позволяют взаимодействовать в географических областях средних размеров и работают на скоростях от средних до высоких. Они получили такое имя из-за способности одной MAN занимать область размером с большой город. MAN работают с меньшими паузами, чем WAN, но не могут обеспечить взаимодействие на таких же больших расстояниях. Типичные MAN работают со скоростями от 56 Кбит/с до 100 Мбит/с.

Технологии LAN обеспечивают наивысшие скорости соединений между компьютерами, но не позволяют им занимать большие области. Например, типичная LAN занимает пространство, такое же как одно здание или небольшой университетский городок, и работает со скоростями от 4 Мбит/с до 2 Гбит/с.

Мы уже говорили о компромиссе между скоростью и расстоянием: технологии, обеспечивающие более высокие скорости взаимодействия, работают на более коротких расстояниях. Существуют и другие различия среди технологий в указанных выше трех категориях. В технологиях LAN каждый компьютер обычно содержит сетевое интерфейсное устройство, которое соединяет машину напрямую с сетевой средой передачи данных(например, медным проводом или коаксиальным кабелем). Часто сеть является пассивной, полагая, что электронные устройства в присоединенных компьютерах сами будут генерировать и получать необходимые электрические сигналы. В технологиях MAN сеть содержит активные коммутирующие элементы, которые приводят к появлению коротких задержек при направлении данных к их назначению. В технологиях WAN сеть обычно состоит из групп сложных маршрутизаторов пакетов, соединенных линиями связи. Сеть может быть расширена добавлением нового маршрутизатора и еще одной линии

связи. Присоединить компьютер к WAN значит соединить его с одним из маршрутизаторов пакетов. Эти маршрутизаторы вводят значительные паузы при маршрутизации траффика. Поэтому, чем больше становится WAN, тем больше времени ей надо для маршрутизации траффика.

Целью разработки сетевых протоколов является скрыть технологические различия между сетями, сделав соединение независимым от используемого оборудования. Следующие секции содержат шесть примеров сетевых технологий, используемых в Интернете, показывая при этом различия между ними. Следующие главы показывают, как программное обеспечение TCP/IP скрывает такие различия и делает коммуникационную систему независимой от базовой аппаратной технологии.

Стек протоколов TCP/IP

Стандарты TCP/IP опубликованы в серии документов, названных Request for Comment (RFC). Документы RFC описывают внутреннюю работу сети Internet. Некоторые RFC описывают сетевые сервисы или протоколы и их реализацию, в то время как другие обобщают условия применения. Стандарты TCP/IP всегда публикуются в виде документов RFC, но не все RFC определяют стандарты.

Стек был разработан по инициативе Министерства обороны США (Department of Defence, DoD) более 20 лет назад для связи экспериментальной сети ARPAnet с другими сателлитными сетями как набор общих протоколов для разнородной вычислительной среды. Сеть ARPA поддерживала разработчиков и исследователей в военных областях. В сети ARPA связь между двумя компьютерами осуществлялась с использованием протокола Internet Protocol (IP), который и по сей день является одним из основных в стеке TCP/IP и фигурирует в названии стека.

Большой вклад в развитие стека TCP/IP внес университет Беркли, реализовав протоколы стека в своей версии ОС UNIX. Широкое распространение ОС UNIX привело и к широкому распространению протокола IP и других протоколов стека. На этом же стеке работает всемирная информационная сеть Internet, чье подразделение Internet Engineering Task Force (IETF) вносит основной вклад в совершенствование стандартов стека, публикуемых в форме спецификаций RFC.

Если в настоящее время стек TCP/IP распространен в основном в сетях с ОС UNIX, то реализация его в последних версиях сетевых операционных систем для персональных компьютеров (Windows NT 3.5, NetWare 4.1, Windows 95) является хорошей предпосылкой для быстрого роста числа установок стека TCP/IP.

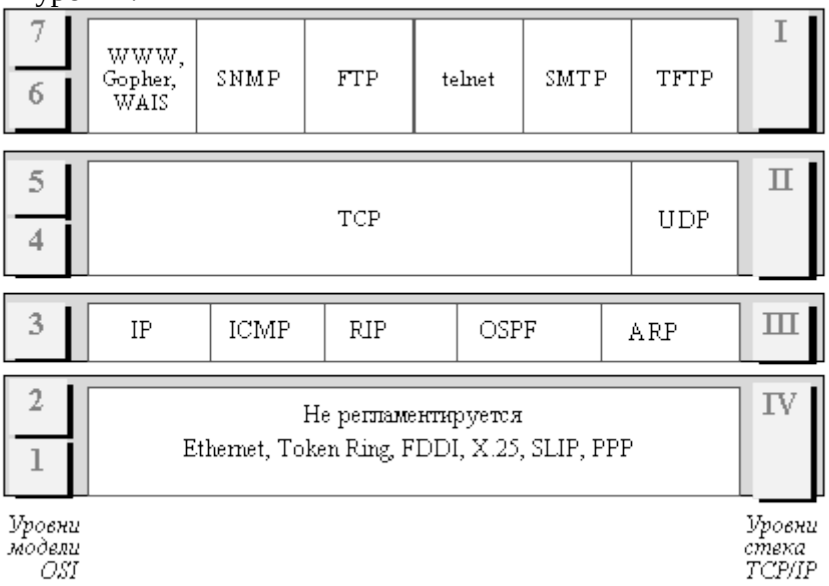
Итак, лидирующая роль стека TCP/IP объясняется следующими его свойствами:

- Это наиболее завершенный стандартный и в то же время популярный стек сетевых протоколов, имеющий многолетнюю историю.
- Почти все большие сети передают основную часть своего трафика с помощью протокола TCP/IP.
- Это метод получения доступа к сети Internet.
- Этот стек служит основой для создания intranet- корпоративной сети, использующей транспортные услуги Internet и гипертекстовую технологию WWW, разработанную в Internet.
- Все современные операционные системы поддерживают стек TCP/IP.
- Это гибкая технология для соединения разнородных систем как на уровне транспортных подсистем, так и на уровне прикладных сервисов.
- Это устойчивая масштабируемая межплатформенная среда для приложений клиент-сервер.

Структура стека TCP/IP. Краткая характеристика протоколов

Так как стек TCP/IP был разработан до появления модели взаимодействия открытых систем ISO/OSI, то, хотя он также имеет многоуровневую структуру, соответствие уровней стека TCP/IP уровням модели OSI достаточно условно.

Структура протоколов TCP/IP приведена на рисунке 2.1. Протоколы TCP/IP делятся на 4 уровня.



Самый нижний (***уровень IV***) соответствует физическому и канальному уровням модели OSI. Этот уровень в протоколах TCP/IP не регламентируется, но поддерживает все популярные стандарты физического и канального уровня: для локальных сетей это Ethernet, Token Ring, FDDI, Fast Ethernet, 100VG-AnyLAN, для глобальных сетей - протоколы соединений "точка-точка" SLIP и PPP, протоколы территориальных сетей с коммутацией пакетов X.25, frame relay. Разработана также специальная спецификация, определяющая использование технологии ATM в качестве транспорта канального уровня. Обычно при появлении новой технологии локальных или глобальных сетей она быстро включается в стек TCP/IP за счет разработки соответствующего RFC, определяющего метод инкапсуляции пакетов IP в ее кадры.

Следующий уровень (***уровень III***) - это уровень межсетевого взаимодействия, который занимается передачей пакетов с использованием различных транспортных технологий локальных сетей, территориальных сетей, линий специальной связи и т. п.

В качестве основного протокола сетевого уровня (в терминах модели OSI) в стеке используется протокол **IP**, который изначально проектировался как протокол передачи пакетов в составных сетях, состоящих из большого количества локальных сетей, объединенных как локальными, так и глобальными связями. Поэтому протокол IP хорошо работает в сетях со сложной топологией, рационально используя наличие в них подсистем и экономно расходуя пропускную способность низкоскоростных линий связи. Протокол IP является дейтаграммным протоколом, то есть он не гарантирует доставку пакетов до узла назначения, но старается это сделать.

К уровню межсетевого взаимодействия относятся и все протоколы, связанные с составлением и модификацией таблиц маршрутизации, такие как протоколы сбора маршрутной информации **RIP** (Routing Internet Protocol) и **OSPF** (Open Shortest Path First), а также протокол межсетевых управляющих сообщений **ICMP** (Internet Control Message Protocol). Последний протокол предназначен для обмена информацией об ошибках между маршрутизаторами сети и узлом - источником пакета. С помощью специальных пакетов ICMP сообщается о невозможности доставки пакета, о превышении времени жизни или продолжительности сборки пакета из фрагментов, об аномальных величинах параметров, об изменении маршрута пересылки и типа обслуживания, о состоянии системы и т.п.

Следующий уровень (***уровень II***) называется основным. На этом уровне функционируют протокол управления передачей **TCP** (Transmission Control Protocol) и протокол дейтаграмм пользователя **UDP** (User Datagram Protocol). Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования виртуальных соединений. Протокол UDP обеспечивает

передачу прикладных пакетов дейтаграммным способом, как и IP, и выполняет только функции связующего звена между сетевым протоколом и многочисленными прикладными процессами.

Верхний уровень (***уровень 1***) называется прикладным. За долгие годы использования в сетях различных стран и организаций стек TCP/IP накопил большое количество протоколов и сервисов прикладного уровня. К ним относятся такие широко используемые протоколы, как протокол копирования файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMTP, используемый в электронной почте сети Internet, гипертекстовые сервисы доступа к удаленной информации, такие как WWW и многие другие. Остановимся несколько подробнее на некоторых из них.

Протокол пересылки файлов **FTP** (File Transfer Protocol) реализует удаленный доступ к файлу. Для того, чтобы обеспечить надежную передачу, FTP использует в качестве транспорта протокол с установлением соединений - TCP. Кроме пересылки файлов протокол FTP предлагает и другие услуги. Так, пользователю предоставляется возможность интерактивной работы с удаленной машиной, например, он может распечатать содержимое ее каталогов. Наконец, FTP выполняет аутентификацию пользователей. Прежде, чем получить доступ к файлу, в соответствии с протоколом пользователи должны сообщить свое имя и пароль. Для доступа к публичным каталогам FTP-архивов Internet парольная аутентификация не требуется, и ее обходят за счет использования для такого доступа предопределенного имени пользователя Anonymous.

В стеке TCP/IP протокол FTP предлагает наиболее широкий набор услуг для работы с файлами, однако он является и самым сложным для программирования. Приложения, которым не требуются все возможности FTP, могут использовать другой, более экономичный протокол - простейший протокол пересылки файлов **TFTP** (Trivial File Transfer Protocol). Этот протокол реализует только передачу файлов, причем в качестве транспорта используется более простой, чем TCP, протокол без установления соединения - UDP.

Протокол **telnet** обеспечивает передачу потока байтов между процессами, а также между процессом и терминалом. Наиболее часто этот протокол используется для эмуляции терминала удаленного компьютера. При использовании сервиса telnet пользователь фактически управляет удаленным компьютером так же, как и локальный пользователь, поэтому такой вид доступа требует хорошей защиты. Поэтому серверы telnet всегда используют как минимум аутентификацию по паролю, а иногда и более мощные средства защиты, например, систему Kerberos.

Протокол **SNMP** (Simple Network Management Protocol) используется для организации сетевого управления. Изначально протокол SNMP был разработан для удаленного контроля и управления маршрутизаторами Internet, которые традиционно часто называют также шлюзами. С ростом популярности протокол SNMP стали применять и для управления любым коммуникационным оборудованием - концентраторами, мостами, сетевыми адаптерами и т.д. и т.п. Проблема управления в протоколе SNMP разделяется на две задачи.

Первая задача связана с передачей информации. Протоколы передачи управляющей информации определяют процедуру взаимодействия SNMP-агента, работающего в управляемом оборудовании, и SNMP-монитора, работающего на компьютере администратора, который часто называют также консолью управления. Протоколы передачи определяют форматы сообщений, которыми обмениваются агенты и монитор.

Вторая задача связана с контролируемыми переменными, характеризующими состояние управляемого устройства. Стандарты регламентируют, какие данные должны сохраняться и накапливаться в устройствах, имена этих данных и синтаксис этих имен. В стандарте SNMP определена спецификация информационной базы данных управления сетью. Эта спецификация, известная как база данных MIB (Management Information Base),

определяет те элементы данных, которые управляемое устройство должно сохранять, и допустимые операции над ними.

Рекомендуемая литература [2, 3, 4]

Тема 3 Инструментальные средства программирования в Интернет.

Средства программирования для Интернета можно условно разделить на группы, рис1.

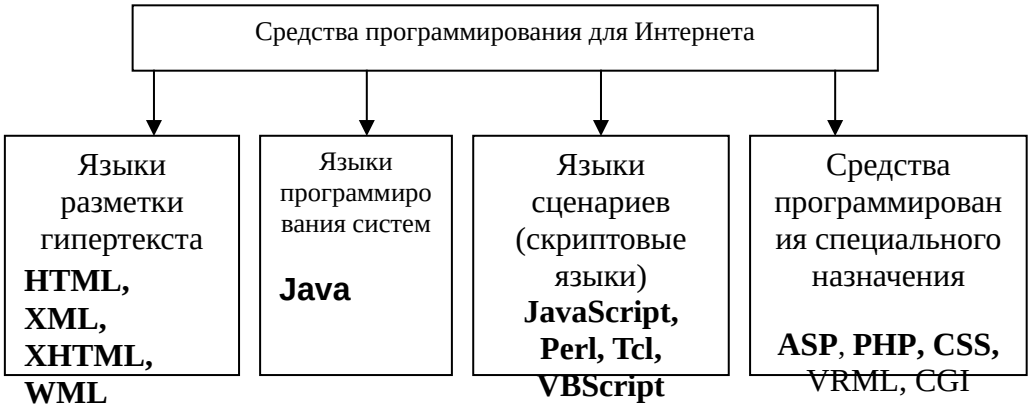


Рисунок 1 – Классификация средств программирования для Интернета

Языки разметки гипертекста.

HTML /HyperText Markup Language - язык разметки гипертекстов/, был необходим для статического размещения страниц во “Всемирной паутине”. Язык HTML был разработан Тимом Бернерс-Ли. В 1990-х годах он добился особенных успехов благодаря быстрому росту Web. В это время HTML был расширен и дополнен. В Web очень важно использование одних и тех же соглашений HTML авторами Web-страниц и производителями. Это явилось причиной совместной работы над спецификациями языка HTML. В каждой версии HTML предпринималась попытка отразить все большее число соглашений между работниками и пользователями этой индустрии, чтобы усилия авторов не были потрачены впустую, а их документы не стали бы нечитаемыми в короткий срок. Язык HTML разрабатывался с той точки зрения, что все типы устройств должны иметь возможность использовать информацию в Web: персональные компьютеры с графическими дисплеями с различным разрешением и числом цветов, сотовые телефоны, переносные устройства, устройства для вывода и ввода речи, компьютеры с высокой и низкой частотой и т.д. За основу модели разметки документов в HTML принята тэговая модель. *Тэговая модель* описывает документ как совокупность контейнеров, каждый из которых начинается и заканчивается тэгами. Т.е. документ HTML представляет собой не что иное, как обычный ASCII-файл, с добавленными в него управляющими HTML-кодами (тэгами).

XML /eXtensible Markup Language - расширенный язык разметки/, новый стандарт оформления самых разнообразных документов, в том числе и Web-страниц. Обладает расширенным набором управляющих символов, позволяющих создавать структуры любой вложенности, имеет возможности внутренней проверки правильности документа. Уникальность XML заключается в его неограниченной расширяемости в силу четкой структурированности данных, возможности определения своих тэгов и т.д.

XHTML - основанный на XML язык разметки гипертекста, максимально приближенный к текущим стандартам HTML. Применяется, как язык разметки, переходный от HTML к XML.

WML /Wireless Markup Language/ - основанный на XML язык разметки страничек, предназначенных для чтения на дисплеях сотовых телефонов и PDA. Отличается компактностью конструкций.

Языки программирования систем

Java - объектно-ориентированный и архитектурно нейтральный язык, обеспечивающий надежность и безопасность, обладает высокой производительностью в сочетании с многопоточностью и динамичностью. Был разработан Sun Microsystems в 1995 году. Ему нужны компиляторы и служебные файлы для функционирования. Язык Java потребовался для качественного скачка в создании интерактивных продуктов для сети Internet. Хотя имеет сходное с JavaScript звучание, на самом деле не имеет к последнему никакого отношения. Java - это не только язык, но и как бы виртуальный компьютер (виртуальная машина - ВМ), запускаемый поверх настоящего. Для Java ВМ пишут программы (как правило на языке Java). Результатом являются апплеты (applets) которые пересылаются по Internet вместе с HTML страничкой и запускаются на компьютере. Самый серьезный недостаток - чрезвычайно низкая эффективность таких программ по сравнению с настоящими. Преимущество - они могут выполняться под почти любой операционной системой, не только Windows.

Языки сценариев (скриптовые языки)

JavaScript - язык, программы на котором можно включать непосредственно в HTML код странички. Одно из самых распостраненных его применений - подсветка кнопок или пунктов меню при прохождении над ними курсора мышки. Все операции, которые можно исполнять в программе на JavaScript, описывают действия над хорошо известными и понятными объектами, которыми являются элементы рабочей области программы Netscape Navigator и контейнеры языка HTML. В JavaScript есть события - аналог программных прерываний. Эти события ориентированы на работу в World Wide Web, например, загрузка страницы в рабочую область Navigator'a или выбор гипертекстовой ссылки. Используя события, автор гипертекстовой страницы и программы ее отображающей может организовать просмотр динамических объектов.

Таблица сравнения JavaScript и Java:

JavaScript	Java
Не компилируемый клиентом.	Компилируемая клиентом перед запуском программы.
Объектный язык.	Объектно-ориентированный.
Внедренный в HTML - страницу.	Небольшие приложения, отделенные от HTML - страниц.
Переменные типы данных, не объявляются.	Переменные типы данных должны быть объявленны (строгий контроль типов).
Динамическое закрепление. Ссылки объекта, поверяются во время выполнения.	Статическое закрепление. Ссылки объекта, проверяются во время компиляции.

Jscript - был "разработан" Microsoft. Функционально это тоже самое, что и JavaScript. Названия разные из-за того, что JavaScript был уже запатентован Netscape к тому времени, как Microsoft решила встроить в свой браузер поддержку JavaScript. Вторая причина в том, что JScript не полностью следует спецификации Netscape.

VBscript /Visual Basic Scripting Edition/ – скриптовый язык, подмножество Visual Basic. В версии Microsoft Internet Explorer 3.0 помимо поддержки Java script появилась и поддержка VBscript. Между Java script и VBscript различий практически нет. Java script поддерживает только функции, VBscript поддерживает и функции и процедуры. VBscript отличие от Java script не чувствителен к регистру символов.

Perl - слово Perl является аббревиатурой выражения Practical Extraction and Report Language (практический язык извлечений и отчетов). Создатель языка - Ларри Уолл.

Мощные конструкции этого языка позволяют создавать (с минимальной затратой сил) некоторые очень эффективные специализированные решения и универсальные инструменты. Эти инструменты можно использовать и в дальнейшем, потому что написанные на Perl программы отличаются высокой переносимостью и готовностью к использованию. В язык введено много часто используемых функций работы со строками, массивами, всевозможными средствами преобразования данных, управления процессами и др. Perl, работает и как компилятор, и как интерпретатор.

Tcl - создатель языка Tcl и инструментария Tcl toolkit - Джон Остераут, основатель и руководитель компании Scriptics. Язык служит для автоматизации рутинных процессов и составления мощных команд для работы с абстрактными нетипизированными объектами. Не зависит от типа системы, позволяет создавать программы с графическим интерфейсом. В Tcl существенные характеристики шрифта можно использовать без каких-либо деклараций или преобразований.

Средства программирования специального назначения

CGI /Common Gateway Inteface - общий шлюзовый интерфейс/ - является стандартом интерфейса (связи) внешней прикладной программы с информационным сервером типа HTTP, Web-сервер. С помощью CGI можно создавать CGI-программы, называемые ***шлюзами***, которые во взаимодействии с такими прикладными системами, как система управления базой данных, электронная таблица, деловая графика и др., смогут выдать на экран пользователя динамическую информацию. Программа-шлюз запускается WWW сервером в реальном масштабе времени. Программа-шлюз может быть закодирована на языках C/C++, Fortran, Perl, TCL, Unix Schell, Visual Basic, Apple Script. Как выполнимый модуль, она записывается в поддиректорий с именем cgi-bin WWW сервера.

VRML /Virtual Reality Modelling Language/ - язык придуманный для отображения трехмерных объектов и пространств на WWW, а также взаимодействий с ними. Позволяет описывать в текстовом виде различные трехмерные сцены, освещение, тени, вращение в любом направлении, масштабирование и т.д. VRML-файл представляет собой обычный текстовый файл, интерпретируемый браузером. VRML требует специальной надстройки к браузеру - плагина (plug-in). Поскольку большинство браузеров не имеет встроенных средств поддержки VRML, для просмотра VRML-документов необходимо подключить вспомогательную программу - VRML-браузер. Один и тот же VRML-документ может выглядеть по-разному в разных браузерах. Многие разработчики браузеров добавляют нестандартные расширения VRML.

CSS /Cascading Style Sheets - таблицы каскадных стилей/ - язык, содержащий набор свойств для описания внешнего вида любых HTML документов. Для присвоения какому-либо элементу определенных характеристик нужно один раз описать этот элемент и определить это описание как стиль, а в дальнейшем просто указывать, что элемент, который надо оформить соответствующим образом, должен принять свойства описанного стиля. Можно сохранить описание стиля не в тексте странички, а в отдельном файле - это позволит использовать описание стиля на любом количестве страниц. Существует три вида таблиц стилей: Внутренние таблицы стилей, Глобальные таблицы стилей и Связанные таблицы стилей. Внутренние таблицы стилей (Inline Style Sheets) при помощи специального атрибута помещаются прямо в HTML-тэги. Глобальные (Global Style Sheets) определяют стиль элементов во всем документе. Связанные (Linked Style Sheets) могут быть использованы для нескольких документов сразу и хранятся во внешнем файле. Самую полную и свежую информацию можно найти на сайте http://www.w3c.org/style/.

ASP /Active Server Pages/ - это среда программирования, разработка Microsoft, которая обеспечивает возможность комбинирования HTML, скриптов и компонент для создания динамических Web-приложений, для обработки HTML запросов на сервере, т.е. файл проходит сначала через серверный интерпретатор, а затем уже идет клиенту. Внешне ASP функционирует также, как CGI. Аналогичным образом передаются параметры

(формат-строки запроса) и осуществляется вывод результатов. Однако производительность ASP оказывается гораздо выше, т. к. при каждом запросе не происходит отдельной загрузки ASP-интерпретатора. Создавать ASP-страницы можно в любом текстовом редакторе. Microsoft Visual InterDev 6.0 (входящий в состав Microsoft Visual Studio) является одним из лучших средств, которое позволяет не только быстро и эффективно создавать ASP-код, но и осуществлять расширенную отладку кода.

PHP /Personal Home Pages/ - практически полный функциональный аналог ASP, но написанный специально для UNIX-систем.

Рекомендуемая литература [5, 6]

Тема 4 Языки разметки гипертекста.

Популярность World Wide Web и неотъемлемой ее части, HTML, безусловно, стала причиной повышенного внимания к системам гипертекстовой разметки документов. Хотя понятие гипертекста было введено В.Бушем еще в 1945 году и, начиная с 60-х годов стали появляться первые приложения, использующие гипертекстовые данные, всплеск активности вокруг этой технологии начался лишь тогда, когда возникла реальная необходимость в механизме объединения множества информационных ресурсов, обеспечения возможности создания, просмотра нелинейного текста. И примером реализации этого механизма послужила паутина WWW.

Язык разметки документов - это набор специальных инструкций, называемых тэгами, предназначенных для формирования в документах какой-либо структуры и определения отношений между различными элементами этой структуры. Тэги языка, или, как их иногда называют, управляющие дескрипторы, в таких документах каким-то образом кодируются, выделяются относительно основного содержимого документа и служат в качестве инструкций для программы, производящей показ содержимого документа на стороне клиента. В самых первых системах для обозначения этих команд использовались символы “<” и “>”, внутри которых помещались названия инструкций и их параметры. Сейчас такой способ обозначения тэгов является стандартным.

Использование гипертекстовой разбивки текстового документа в современных информационных системах во многом связано с тем, что гипертекст позволяет создавать механизм нелинейного просмотра информации. В таких системах данные представляются не в виде непрерывного потока текстовой информации, а набором взаимосвязанных компонентов, переход по которым осуществляется при помощи гиперссылок.

HTML.

Самый популярный на сегодняшний день язык гипертекстовой разметки – HTML, был создан специально для организации информации, распределенной в сети Интернет, и является одной из ключевых составляющих технологии WWW. С использованием гипертекстовой модели документа способ представления разнообразных информационных ресурсов в сети стал более упорядочен, а пользователи получили удобный механизм поиска и просмотра нужной информации.

HTML является упрощенной версией стандартного общего языка разметки - SGML (Standart Generalised Markup Language), который был утвержден ISO в качестве стандарта еще в 80-х годах. Этот язык предназначен для создания других языков разметки, он определяет допустимый набор тэгов, их атрибуты и внутреннюю структуру документа. Контроль за правильностью использования дескрипторов осуществляется при помощи специального набора правил, называемых DTD- описаниями(более подробно о DTD мы поговорим чуть позже), которые используются программой клиента при разборе документа. Для каждого класса документов определяется свой набор правил, описывающих грамматику соответствующего языка разметки. С помощью SGML можно описывать структурированные данные, организовывать информацию, содержащуюся в

документах, представлять эту информацию в некотором стандартизованном формате. Но в виду некоторой своей сложности, SGML использовался, в основном, для описания синтаксиса других языков(наиболее известным из которых является HTML), и немногие приложения работали с SGML- документами напрямую.

Язык разметки гипертекста (HyperText Makeup Language - HTML) формулируется в терминах языка стандартной обобщенной разметки (Standard Generalized Markup Language - SGML). Язык SGML представляет собой метод создания структурированных документов, а также языков для их разметки.

В языке SGML каждый документ имеет три части:

1. Декларации языка SGML, привязывающие к определенным значениям параметры обработки, а также имена синтаксиса. Например, декларация SGML в описании типа документа HTML объявляет, что строка, с которой начинается метка, - это </ , а максимальная длина имени составляет 40 символов.
2. Пролог, состоящий из одной или нескольких деклараций о типе документа. Они определяют типы элементов, взаимосвязи между элементами и их атрибуты, а также условные обозначения, которые могут быть задействованы при разметке. Декларация HTML DTD, например, указывает, что элемент HEAD содержит по крайней мере один элемент TITLE.
3. Данные, которые состоят из разметки документа и собственно информации.

Мы используем термин HTML для обозначения как типа документа, так и языка разметки для кодировки документов данного типа.

Все документы типа HTML придерживаются единых деклараций языка SGML и пролога. Следовательно, реализации программы WorldWide Web в общем случае лишь передают и сохраняют ту часть документа HTML, которая содержит данные. Чтобы создать для обработки на анализаторе SGML объект с документом, необходимо поставить текст HTML DTD перед имеющимися данными.

И наоборот, для реализации анализатора языка HTML необходимо лишь воссоздать те части анализатора SGML, которые необходимы для разбора данных, появляющихся вслед за разбором деклараций HTML DTD.

Структурированный текст. Данные в формате HTML похожи на текстовой файл, за исключением того, что некоторые из символов интерпретируются как разметка. Разметка придает документу некую структуру.

Данные представляют собой иерархию элементов. Каждый элемент имеет имя, атрибуты и несет некую информацию. Большинство элементов представлены в документе в виде начальной метки, указывающей имя и атрибуты. Далее следует собственно содержание элемента. И наконец, заканчивает все это конечная метка.

Гораздо более простой и удобный, чем SGML, язык HTML позволяет определять оформление элементов документа и имеет некий ограниченный набор инструкций - тэгов, при помощи которых осуществляется процесс разметки. Инструкции HTML, в первую очередь, предназначены для управления процессом вывода содержимого документа на экране программы-клиента и определяют этим самым способ представления документа, но не его структуру. В качестве элемента гипертекстовой базы данных, описываемой HTML, используется текстовый файл, который может легко передаваться по сети с использованием протокола HTTP. Эта особенность, а также то, что HTML является открытым стандартом и огромное количество пользователей имеет возможность применять возможности этого языка для оформления своих документов, безусловно, повлияли на рост популярности HTML и сделали его сегодня главным механизмом представления информации в Web.

Однако современные приложения нуждаются не только в языке представления данных на экране клиента, но и в механизме, позволяющем определять структуру документа, описывать содержащиеся в нем элементы. HTML обладает несложным набором команд и вполне успешно справляется с задачей описания текстовой информации

и отображением ее на экране программы просмотра- броузера. Однако сами отображаемые данные никак не связаны с теми тэгами, которые используются для форматирования, поэтому у программ-анализаторов нет возможности использовать тэги HTML для поиска нужных нам фрагментов документа. Т.е. встретив, например, такое описание

rose,

программа просмотра будет знать, каким цветом отобразить текст, содержащийся внутри тэгов и, вероятно, отобразит его правильно, но ей абсолютно безразлично, в каком месте документа встретился этот тэг, в какие другие тэги заключен текущий фрагмент, существуют ли вложенные в него фрагменты, правильно ли построены отношения между объектами. Такое "безразличие" к структуре документа приводит к тому, что поиск или анализ информации внутри него ничем не будет отличаться от работы со сплошным, не разбитым на элементы текстовым файлом. А это, как известно, не самый эффективный способ работы с информацией.

Другим существенным недостатком HTML можно назвать ограниченность набора его тэгов. DTD- правила для HTML определяют фиксированный набор дескрипторов и поэтому у разработчика нет возможности вводить собственные, специальные тэги. Хотя время от времени появляются новые расширения языка(на сегодняшний день последней версией HTML является HTML 4.0), но долгий путь их стандартизации, сопровождаемый постоянными разногласиями между основными производителями броузеров делают практически невозможной быструю адаптацию языка, его использование для отображения специализированной информации(например, мультимедийной, математических, химических формул и т.д.).

Подводя итог всему сказанному, можно утверждать, что HTML уже сегодня не удовлетворяет в полной мере требованиям, предъявляемым современными разработчиками к языкам подобного рода. И ему на смену был предложен новый язык гипертекстовой разметки, мощный, гибкий, и, одновременно с этим, удобный язык XML. В чем же заключается его достоинства?

XML

XML (*Extensible Markup Language*[1]) - это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. Т.е. сам по себе XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Таким образом, если, например, мы считаем, что для обозначения элемента *rose* в документе необходимо использовать тэг <*flower*>;, то XML позволяет свободно использовать определяемый нами тэг и мы можем включать в документ фрагменты, подобные следующему:

<flower>rose</flower>

Набор тэгов может быть легко расширен. Если, предположим, мы хотим также указать, что описание цветка должно по смыслу идти внутри описания оранжереи, в которой он цветет, то просто задаем новые тэги и выбираем порядок их следования:

<conservatory>
<flower>rose</flower>
</conservatory>

Если мы хотим посадить туда еще несколько цветочков, то должны внести следующие изменения:

<conservatory>
<flower>rose</flower>
<flower>tulip</flower>
<flower>cactus</flower>
</conservatory>

Как видно, сам процесс создания XML документа очень прост и требует от нас лишь базовых знаний HTML и понимания тех задач, которые мы хотим выполнить, используя XML в качестве языка разметки. Таким образом, у разработчиков появляется уникальная возможность определять собственные команды, позволяющие им наиболее эффективно определять данные, содержащиеся в документе. Автор документа создает его структуру, строит необходимые связи между элементами, используя те команды, которые удовлетворяют его требованиям и добивается такого типа разметки, которое необходимо ему для выполнения операций просмотра, поиска, анализа документа.

Еще одним из очевидных достоинств XML является возможность использования его в качестве универсального языка запросов к хранилищам информации. Сегодня в глубинах W3C находится на рассмотрении рабочий вариант стандарта XML-QL(или XQL), который, возможно, в будущем составит серьезную конкуренцию SQL. Кроме того, XML-документы могут выступать в качестве уникального способа хранения данных, который включает в себя одновременно средства для разбора информации и представления ее на стороне клиента. В этой области одним из перспективных направлений является интеграция Java и XML - технологий, позволяющая использовать мощь обеих технологий при построении машинно-независимых приложений, использующих, кроме того, универсальный формат данных при обмене информации.

XML позволяет также осуществлять контроль за корректностью данных, хранящихся в документах, производить проверки иерархических соотношений внутри документа и устанавливать единый стандарт на структуру документов, содержимым которых могут быть самые различные данные. Это означает, что его можно использовать при построении сложных информационных систем, в которых очень важным является вопрос обмена информацией между различными приложениями, работающими в одной системе. Создавая структуру механизма обмена информации в самом начале работы над проектом, менеджер может избавить себя в будущем от многих проблем, связанных с несовместимостью используемых различными компонентами системы форматов данных.

Также одним из достоинств XML является то, что программы-обработчики XML-документов не сложны и уже сегодня появились и свободно распространяются всевозможные программные продукты, предназначенные для работы с XML-документами. XML поддерживается сегодня в Microsoft Internet Explorer 4/0 и в бэта-версиях IE5. Было заявлено о его поддержке в последующих версиях Netscape Communicator, СУБД Oracle, DB-2, в приложениях MS-Office . Все это дает основания предполагать, что, скорее всего, в ближайшем будущем XML станет основным языком обмена информации для информационных систем, заменив собой, тем самым, HTML. На основе XML уже сегодня созданы такие известные специализированные языки разметки, как SMIL, CDF, MathML, XSL, и список рабочих проектов новых языков, находящихся на рассмотрении W3C, постоянно пополняется.

Как выглядит XML-документ?

Если Вы знакомы с HTML, изучение XML не потребует от вас особых усилий. Хотя XML, безусловно, сильно отличается по своим возможностям и предназначению от языка гипертекстовой разметки, оба эти языка являются подмножествами SGML, и, следовательно, наследуют его базовые принципы.

Структура документа

Простейший XML- документ может выглядеть так, как это показано в Примере 1

```
<?xml version="1.0"?>
<list_of_items>
<item id="1"><first/>Первый</item>
<item id="2">Второй <sub_item>подпункт 1</sub_item></item>
<item id="3">Третий</item>
<item id="4"><last/>Последний</item>
</list_of_items>
```

Обратите внимание на то, что этот документ очень похож на обычную HTML-страницу. Также, как и в HTML, инструкции, заключенные в угловые скобки называются тэгами и служат для разметки основного текста документа. В XML существуют открывающие, закрывающие и пустые тэги (в HTML понятие пустого тэга тоже существует, но специального его обозначения не требуется).

Тело документа XML состоит из элементов разметки (markup) и непосредственно содержимого документа - данных (content). XML - тэги предназначены для определения элементов документа, их атрибутов и других конструкций языка. Более подробно о типах применяемой в документах разметки мы поговорим чуть позже.

Любой XML- документ должен всегда начинаться с инструкции <?xml?>, внутри которой также можно задавать номер версии языка, номер кодовой страницы и другие параметры, необходимые программе-анализатору в процессе разбора документа

Правила создания XML- документа

В общем случае XML- документы должны удовлетворять следующим требованиям:

- В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация
- Каждый открывающий тэг, определяющий некоторую область данных в документе обязательно должен иметь своего закрывающего "напарника", т.е., в отличие от HTML, нельзя опускать закрывающие тэги
- В XML учитывается регистр символов
- Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки
- Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов
- Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные и поэтому учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML)

Если XML- документ не нарушает приведенные правила, то он называется *формально-правильным* и все анализаторы, предназначенные для разбора XML-документов, смогут работать с ним корректно.

Однако кроме проверки на формальное соответствие грамматике языка, в документе могут присутствовать средства контроля над содержанием документа, за соблюдением правил, определяющих необходимые соотношений между элементами и формирующих структуру документа. Например, следующий текст, являясь вполне правильным XML- документом, будет абсолютно бессмысленным:

<country><title>Russia</title><city><title>Novosibirsk</country></title></city>

Для того, чтобы обеспечить проверку корректности XML- документов, необходимо использовать анализаторы, производящие такую проверку и называемые верифицирующими.

На сегодняшний день существует два способа контроля правильности XML-документа: DTD - определения(Document Type Definition) и схемы данных(Semantic Schema). Более подробно об использовании DTD и схемах мы поговорим в следующих разделах. В отличии от SGML, определение DTD- правил в XML не является необходимостью, и это обстоятельство позволяет нам создавать любые XML- документы, не ломая пока голову над весьма непростым синтаксисом DTD.

Конструкции языка

Содержимое XML- документа представляет собой набор элементов, секций CDATA, директив анализатора, комментариев, спецсимволов, текстовых данных. Рассмотрим каждый из них подробней.

Элементы данных

Элемент - это структурная единица XML- документа. Заключая слово rose в в тэги <flower> </flower> , мы определяем непустой элемент, называемый <flower>, содержимым которого является *rose*. В общем случае в качестве содержимого элементов могут выступать как просто какой-то текст, так и другие, вложенные, элементы документа, секции CDATA, инструкции по обработке, комментарии, - т.е. практически любые части XML- документа.

Любой непустой элемент должен состоять из начального, конечного тэгов и данных, между ними заключенных. Например, следующие фрагменты будут являться элементами:

<flower>rose</flower>
<city>Novosibirsk</city>
,а эти - нет:
<rose>
<flower>
rose

Набором всех элементов, содержащихся в документе, задается его структура и определяются все иерархическое соотношения. Плоская модель данных превращается с использованием элементов в сложную иерархическую систему со множеством возможных связей между элементами. Например, в следующем примере мы описываем месторасположение Новосибирских университетов (указываем, что Новосибирский Университет расположен в городе Новосибирске, который, в свою очередь, находится в России), используя для этого вложенность элементов XML :

<country id="Russia">
<cities-list>
<city>
<title>Новосибирск</title>
<state>Siberia</state>
<universities-list>
<university id="2">
<title>Новосибирский Государственный Технический Университет</title>
<noprivate/>
<address URL="www.nstu.ru"/>
<description>очень хороший институт</description>
</university>
<university id="2">
<title>Новосибирский Государственный Университет</title>
<noprivate/>
<address URL="www.nsu.ru"/>
<description>тоже не плохой</description>
</university>
</universities-list>
</city>
</cities-list>
</country>

Производя в последствии поиск в этом документе, программа клиента будет опираться на информацию, заложенную в его структуру - используя элементы документа. Т.е. если, например, требуется найти нужный университет в нужном городе, используя приведенный фрагмент документа, то необходимо будет просмотреть содержимое конкретного элемента <university>, находящегося внутри конкретного элемента <city>. Поиск при этом, естественно, будет гораздо более эффективен, чем нахождение нужной последовательности по всему документу.

В XML документе, как правило, определяется хотя бы один элемент, называемый корневым и с него программы-анализаторы начинают просмотр документа. В приведенном примере этим элементом является <country>

В некоторых случаях тэги могут изменять и уточнять семантику тех или иных фрагментов документа, по разному определяя одну и ту же информацию и тем самым предоставляя приложению-анализатору этого документа сведения о контексте использования описываемых данных. Например, прочитав фрагмент **<city>**Holliwood**</city>** мы можем догадаться, что речь в этой части документа идет о городе, а вот во фрагменте **<restaurant>**Holliwood**</restaurant>** - о забегаловке.

В случае, если элемент не имеет содержимого, т.е. нет данных, которые он должен определять, он называется пустым. Примером пустых элементов в HTML могут служить такие тэги HTML, как **
, <hr>, <img>**;. Необходимо только помнить, что начальный и конечные тэги пустого элемента как бы объединяется в один, и надо обязательно ставить косую черту перед закрывающей угловой скобкой (например, **<empty/>**;)

Комментарии

Комментариями является любая область данных, заключенная между последовательностями символов <!-- и --> Комментарии пропускаются анализатором и поэтому при разборе структуры документа в качестве значащей информации не рассматриваются.

Атрибуты

Если при определении элементов необходимо задать какие-либо параметры, уточняющие его характеристики, то имеется возможность использовать атрибуты эдлемента. Атрибут - это пара "название" = "значение", которую надо задавать при определении элемента в начальном тэге. Пример:

<color **RGB**="true">#ff08ff</color>

<color **RGB**="false">white</color>

или

<author id=0>Ivan Petrov</author>

Примером использования атрибутов в HTML является описание элемента :

Black

Специальные символы

Для того, чтобы включить в документ символ, используемый для определения каких-либо конструкций языка (например, символ угловой скобки) и не вызвать при этом ошибок в процессе разбора такого документа, нужно использовать его специальный символьный либо числовой идентификатор. Например, < , > " или $ (десятичная форма записи), &#x1a (шестнадцатеричная) и т.д. Строковые обозначения спецсиволов могут определяться в XML документе при помощи компонентов (entity), о чем мы еще поговорим немного позже.

Директивы анализатора

Инструкции, предназначенные для анализаторов языка, описываются в XML документе при помощи специальных тэгов - *<?* и *?>*;. Программа клиента использует эти инструкции для управления процессом разбора документа. Наиболее часто инструкции используются при определении типа документа (например, <? Xml version=”1.0”?>) или создании пространства имен.

CDATA

Чтобы задать область документа, которую при разборе анализатор будет рассматривать как простой текст, игнорируя любые инструкции и специальные символы, но, в отличии от комментариев, иметь возможность использовать их в приложении, необходимо использовать тэги <![CDATA] и]]>. Внутри этого блока можно помещать любую информацию, которая может понадобится программе- клиенту для выполнения каких-либо действий (в область CDATA, можно помещать, например, инструкции

JavaScript). Естественно, надо следить за тем, чтобы в области, ограниченной этими тэгами не было последовательности символов]].

XHTML

XHTML — это основанный на XML язык разметки гипертекста, максимально приближенный к текущим стандартам HTML. XHTML отличается от HTML строгостью написания кода. Если HTML позволял писать практически любые конструкции и браузер их корректно распознавал, то теперь, с появлением XHTML , это стало невозможным. Последний требует строгого соблюдения всех правил, предъявляемых W 3 C . Строгие требования к оформлению XHTML -кода позволяют избежать многих ошибок ещё на стадии написания и отладки.

XHTML — это новый язык, который пришёл на смену старому HTML . Новых версий HTML больше не будет. В итоге все браузеры, как предполагается, перейдут на XHTML (очевидно, что при этом сохранится совместимость со старым HTML , но не более того). XHTML совместим с HTML . Это означает, что даже самые старые браузеры, которые понимают HTML , будут работать и с XHTML . Для проверки правильности написания XHTML-кода существуют программы-валидаторы.

Чем же XHTML 1.0 отличается от HTML ? Существует несколько *требований* , которые разработчик обязан выполнять:

• в начале документа должен указываться один из возможных DTD (Document Type Definition):

• <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

• <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

• <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

• в теле XHTML -документа должны обязательно присутствовать следующие тэги: «< html >», «< head >», «< title >» и «< body >»;

• имена тегов и атрибутов должны записываться в нижнем регистре;

• все значения атрибутов должны заключаться в "кавычки";

• все тэги должны закрываться; если у элемента нет закрывающего тэга, следует добавлять в его конец слеш («< br />» или «< br />» — пробел для совместимости со старыми браузерами);

• необходимо соблюдать корректную вложенность тэгов («< b >< i >текст</ b ></ i >» — неверно; следует писать «< b >< i >текст</ i ></ b >»);

• запрещается использовать минимизированные атрибуты (« nowrap » следует заменить на « nowrap = " nowrap "»); полный список таких атрибутов: checked , compact , declare , defer , disabled , ismap , noresize , noshade , nowrap , multiple , readonly , selected .

• на следующие элементы налагаются ограничения по включению в них других элементов:

• a не может содержать другие элементы a ;

• form не может содержать другие элементы form;

• label не может содержать другие элементы label;

• pre не может содержать img , object , big , small , sub или sup ;

• button не может содержать элементы input, select, textarea, label, button, form, fieldset, iframe или isindex;

• специальные символы в истинном значении должны заменяться на свои эквиваленты:

• « &» на «&amp ; » ;

• «<» на «<» ;

• «>» на «&gt ;».

Кроме того, существует ряд необязательных *рекомендаций* , которые разработчик не обязан выполнять в версии XHTML 1.0, но в последующих версиях этого языка возможно рекомендации перерастут в требования. А именно:

• декларация XML -документа в самом начале кода перед DTD («<? xml version ="1.0" encoding =" windows -1251"?>»);

• наличие атрибута xmlns в элементе html ;

• следование элемента title после тега head ;

• использование атрибута id вместо name (name считается устаревшим атрибутом);

• наличие атрибута type в элементах подключаемых файлов (таблиц стилей и скриптов);

• отказ от использования атрибута target .

Приведём минимальный код *правильной* XHTML -страницы:

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html     xmlns="http://www.w3.org/1999/xhtml"     xml:lang="ru"
lang="ru">
<head>
  <title> Заголовок </title>
  <meta http-equiv="Content-Type"
  content="text/html; charset=windows-1251" />
</ head >
<body>
  Содержимое документа
</body>
</html>
```

Если вставить этот код в файл, сохранить его как « file . html » и открыть через веб-сервер, то вся информация будет получена клиентом как «text/htm l ». То есть, как и обычная HTML -страница. Фактически для браузера это будет не XHTML , а HTML -документ. У XHTML есть свой собственный MIME -тип: «application/xhtml+xml».

MIME — это специальный набор расширений, который указывает программам, как обрабатывать входящую информацию. Изначально MIME -типы были разработаны для почтовых программ, откуда и получили своё название.

Итак, XHTML -данные правильно отдавать клиенту именно в формате «application/xhtml+xml», так как все преимущества, помимо кросс-браузерности (увеличение скорости анализа кода процессором XML , сообщение об ошибках самим браузером и пр.), могут достигаться только в случае, если пользовательский агент поддерживает XHTML и ему сообщается о том, что входящие данные — XHTML -код. Единственное, что надо помнить при отправке XHTML -кода: если браузер понимает XHTML , то только тогда информацию можно отправить как «application/xhtml+xml»; если же нет, то только как «text/htm l ». Список современных клиентов, поддерживающих XHTML : MZ , Opera , NN . IE , к сожалению, пока не понимает «application/xhtml+xml». Проверять, поддерживает ли пользовательский агент нужный MIME -тип можно по исходящему от браузера заголовку «Accept», где содержатся все MIME -типы, известные клиенту. Приведём пример, как это можно сделать с помощью Perl -скрипта:

MIME (Multipurpose Internet Mail Extension) — многоцелевые расширения электронной почты .

```
#!/usr/bin/perl -w
# Выясняем, поддерживает ли браузер XHTML.
my $html = "text/html";
my $xhtml = "application/xhtml+xml";
my $type = $ENV{HTTP_ACCEPT} =~ m/\Q$xhtml\E/ ? $xhtml : $html;
```

```
# Вывод соответствующего заголовка.
print "Content-Type: $type\n\n";
# Вывод (X)HTML-документа.
print "...";
```

При отправке данных как «application/xhtml+xml», надо учитывать ещё несколько моментов, без которых возможно появление ошибок. Так как синтаксически XHTML — это XML , элементы «script» и «style» в XHTML — это #PCDATA-блоки (а не #CDATA). Содержимое таких блоков необходимо помещать в специальную секцию CDATA , иначе процессор XML преобразует специальные символы в их эквиваленты ещё до обработки браузером таблицы стилей или сценария. Следующий пример показывает, как можно это сделать:

```
<script type="text/javascript"><!--//--><![CDATA[//><!--
  ...
  //--><!]]></script>
...
<style type="text/css"><!--/*--><![CDATA[/*><!--*/
...
/*]]>*/--></style>
```

Такой синтаксис универсален. Этот код будет корректно работать и при «text/htm l » и при «application/xhtml+xml».

Хорошим и самым простым решением будет подключение внешних файлов таблиц стилей и скриптов. В XHTML это делается так же как и в HTML :

```
<!-- Подключение CSS -файла (не забудьте о закрывающем слеше).
-->
< link rel = "stylesheet" type = "text/css" href = "file.css"
title
= "" media = "screen" / >
<!-- Подключение JS -файла. -->
< script type = "text/javascript" src = "file.js" ></
script >
```

Рекомендуемая литература [10, 12, 14]

Тема 5 Языки сценариев.

Гипертекстовая информационная система состоит из множества информационных узлов, множества гипертекстовых связей, определенных на этих узлах, и инструмента манипулирования узлами и связями. Технология World Wide Web — это технология ведения гипертекстовых распределенных систем в Internet, и, следовательно, она должна соответствовать общему определению таких систем. Это означает, что все перечисленные выше компоненты гипертекстовой системы должны быть и в Web.

Web, как гипертекстовую систему, можно рассматривать с двух точек зрения. Во-первых, как совокупность отображаемых страниц, связанных гипертекстовыми переходами (ссылками — контейнер ANCHOR). Во-вторых, как множество элементарных информационных **объектов**, составляющих отображаемые страницы (текст, графика, мобильный код и т.п.). В последнем случае множество гипертекстовых переходов страницы — это такой же информационный фрагмент, как и встроенная в текст картинка.

При втором подходе гипертекстовая сеть определяется на множестве элементарных информационных объектов самими HTML-страницами, которые и играют роль гипертекстовых связей. Этот подход более продуктивен с точки зрения построения отображаемых страниц "на лету" из готовых компонентов.

При генерации страниц в Web возникает дилемма, связанная с архитектурой "клиент-сервер". Страницы можно генерировать как на стороне клиента, так и на стороне сервера. В 1995 году специалисты компании Netscape создали механизм управления страницами на клиентской стороне, разработав язык программирования **JavaScript**.
Таким образом, JavaScript — это язык управления сценариями просмотра гипертекстовых страниц Web на стороне клиента. Если быть более точным, то JavaScript — это не только язык программирования на стороне клиента. Liveware, прародитель JavaScript, является средством подстановок на стороне сервера Netscape. Однако наибольшую популярность JavaScript обеспечило программирование на стороне клиента.

Основная идея JavaScript состоит в возможности изменения значений атрибутов HTML-контейнеров и **свойств** среды отображения в процессе просмотра HTML-страницы пользователем. При этом перезагрузки страницы не происходит.
На практике это выражается в том, что можно, например, изменить цвет фона страницы или интегрированную в документ картинку, открыть новое окно или выдать предупреждение.

Название "JavaScript" является собственностью Netscape. Реализация языка, осуществленная разработчиками Microsoft, официально называется Jscript. **Версии** JScript **совместимы** (если быть совсем точным, то не до конца) с соответствующими версиями JavaScript, т.е. JavaScript является подмножеством языка JScript.
JavaScript стандартизован ECMA (European Computer Manufacturers Association – Ассоциация европейских производителей компьютеров). Соответствующие стандарты носят названия ECMA-262 и ISO-16262. Этими стандартами определяется язык ECMAScript, который примерно эквивалентен JavaScript 1.1. Отметим, что не все реализации JavaScript на сегодня полностью соответствуют стандарту ECMA. В рамках данного курса мы во всех случаях будем использовать название JavaScript.

Понятие объектной модели применительно к JavaScript

Для создания механизма управления страницами на клиентской стороне было предложено использовать объектную модель документа. Суть модели в том, что каждый HTML-контейнер — это объект, который характеризуется тройкой:

- свойства;
- **методы**;
- события

Объектную модель можно представить как способ связи между страницами и браузером. Объектная модель — это представление объектов, методов, свойств и событий, которые присутствуют и происходят в программном обеспечении браузера, в виде, удобном для работы с ними кода HTML и исходного текста сценария на странице. Мы можем с ее помощью сообщать наши пожелания браузеру и далее — посетителю страницы. Браузер выполнит наши команды и соответственно изменит страницу на экране.

Объекты с одинаковым набором свойств, методов и событий объединяются в классы однотипных объектов. Классы — это описания возможных объектов. Сами

объекты появляются только после загрузки документа браузером или как результат работы программы. Об этом нужно всегда помнить, чтобы не обратиться к объекту, которого нет.

Свойства

Многие HTML-контейнеры имеют атрибуты. Например, контейнер якоря <A ...>...</A> имеет атрибут HREF, который превращает его в гипертекстовую ссылку:

<A HREF=intuit.htm>intuit</A>

Если рассматривать контейнер якоря <A ...>...</A> как объект, то атрибут HREF будет задавать свойство объекта "якорь". Программист может изменить значение атрибута и, следовательно, свойство объекта:

document.links[0].href="intuit.htm";

Не у всех атрибутов можно изменять значения. Например, высота и ширина графической картинки определяются по первой загруженной в момент отображения страницы картинке. Все последующие картинки, которые заменяют начальную, масштабируются до нее. Справедливости ради следует заметить, что в Microsoft Internet Explorer размер картинки может меняться.

Для общности картины свойствами в JavaScript наделены объекты, которые не имеют аналогов в HTML-разметке. Например, среда исполнения, называемая объектом Navigator, или окно браузера, которое является вообще самым старшим объектом JavaScript.

Методы

В терминологии JavaScript методы объекта определяют функции изменения его свойств. Например, с объектом "документ" связаны методы open(), write(), close(). Эти методы позволяют сгенерировать или изменить содержание документа. Приведем простой пример:

```
function hello()
{ id=window.open("","example","width=400,height=150");
id.focus(); id.document.open();
id.document.write("<H1>Привет!</H1>");
id.document.write("<HR><FORM>");
id.document.write("<INPUT TYPE=button VALUE='Закрыть окно' ");
   id.document.write("onClick='window.opener.focus();window.close();'>");
id.document.close();
}
```

В этом примере метод open() открывает поток записи в документ, метод write() осуществляет эту запись, метод close() закрывает поток записи в документ. Все происходит так же, как и при записи в обычный файл. Если у окна есть поле статуса (обычно в нем отображается уровень загрузки документа), то при незакрытом потоке записи в документ в нем будет "метаться" прямоугольник продолжения записи, как это происходит при загрузке документа.

События

Кроме методов и свойств объекты характеризуются событиями. Собственно, суть программирования на JavaScript заключается в написании обработчиков этих событий. Например, с объектом типа Button (контейнер INPUT типа button — "Кнопка") может происходить событие click, т.е. пользователь может нажать на кнопку. Для этого атрибуты контейнера INPUT расширены атрибутом обработки события click — onClick. В качестве значения этого атрибута указывается программа обработки события, которую должен написать на JavaScript автор HTML-документа:

<INPUT TYPE=button VALUE="Нажать" onClick="window.alert('Пожалуйста, нажмите еще раз');">

Обработчики событий указываются в тех контейнерах, с которыми эти события связаны. Например, контейнер BODY определяет свойства всего документа, поэтому обработчик события завершения загрузки всего документа указывается в этом контейнере как значение атрибута onLoad.

Примечание. Строго говоря, каждый браузер, будь то Internet Explorer, Netscape Navigator или Opera, имеет свою объектную модель. Объектные модели разных браузеров (и даже разные версии одного) отличаются друг от друга, но имеют принципиально одинаковую структуру. Поэтому нет смысла останавливаться на каждой из них по отдельности. Мы будем рассматривать относительно подход применительно ко всем браузерам, иногда, конечно, заостряя внимание на различиях между ними.

JavaScript, объекты, методы, свойства, способ размещения кода, история, совместимость, версии.

Размещение кода на HTML-странице

Главный вопрос любого начинающего программиста: "Как оформить программу и выполнить ее?". Попробуем на него ответить как можно проще, но при этом не забывая обо всех способах применения JavaScript-кода.

Во-первых, исполняет JavaScript-код браузер. В него встроен интерпретатор JavaScript. Следовательно, выполнение программы зависит от того, когда и как этот интерпретатор получает управление. Это, в свою очередь, зависит от функционального применения кода. В общем случае можно выделить четыре способа функционального применения JavaScript:

- гипертекстовая ссылка (схема URL);
- обработчик события (handler);
- подстановка (entity) (в Microsoft Internet Explorer реализована в версиях от 5.X и выше);
- вставка (контейнер SCRIPT).

В учебниках по JavaScript описание применения JavaScript обычно начинают с контейнера SCRIPT. Но с точки зрения программирования это не совсем правильно, поскольку такой порядок не дает ответа на ключевой вопрос: как JavaScript-код получает управление? То есть каким образом вызывается и исполняется программа, написанная на JavaScript и размещенная в HTML-документе.

В зависимости от профессии автора HTML-страницы и уровня его знакомства с основами программирования возможны несколько вариантов начала освоения JavaScript. Если вы программист классического толка (C, Fortran, Pascal и т.п.), то проще всего начинать с программирования внутри тела документа, если вы привыкли программировать под Windows, то в этом случае начинайте с программирования обработчиков событий, если вы имеете только опыт HTML-разметки или давно не писали программ, то тогда лучше начать с программирования гипертекстовых переходов.

URL-схема JavaScript

Схема URL (Uniform Resource Locator) — это один из основных элементов Web-технологии. Каждый информационный ресурс в Web имеет свой уникальный URL. URL указывают в атрибуте HREF контейнера A, в атрибуте SRC контейнера IMG, в атрибуте ACTION контейнера FORM и т.п. Все URL подразделяются на схемы доступа, которые зависят от протокола доступа к ресурсу, например, для доступа к FTP-архиву применяется схема ftp, для доступа к Gopher-архиву — схема gopher, для отправки электронной почты — схема smtp. Тип схемы определяется по первому компоненту URL: http://intuit.ru/directory/page.html

В данном случае URL начинается с http — это и есть определение схемы доступа (схема http).

Основной задачей языка программирования гипертекстовой системы является программирование гипертекстовых переходов. Это означает, что при выборе той или иной гипертекстовой ссылки вызывается программа реализации гипертекстового перехода. В

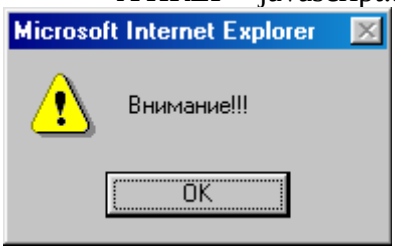
Web-технологии стандартной программой является программа загрузки страницы. JavaScript позволяет поменять стандартную программу на программу пользователя. Для того чтобы отличить стандартный переход по протоколу HTTP от перехода, программируемого на JavaScript, разработчики языка ввели новую схему URL — javascript:

<A HREF="javascript:JavaScript_код">...</A>
<IMG SRC="javascript:JavaScript_код>

В данном случае текст "JavaScript_код" обозначает программы-обработчики на JavaScript, которые вызываются при выборе гипертекстовой ссылки в первом случае и при загрузке картинки — во втором.

Например, при нажатии на гипертекстовую ссылку Внимание!!! можно получить окно предупреждения: (открыть)

<A HREF="javascript:alert('Внимание!!!');">Внимание!!!</A>



А при нажатии на кнопку типа Submit в форме можно заполнить текстовое поле этой же формы:

```
<FORM NAME=f METHOD=post
ACTION="javascript:window.document.f.i.VALUE='Нажали кнопку Click';void(0);">
<TABLE BORDER=0>
<TR>
<TD><INPUT NAME=i></TD>
<TD><INPUT TYPE=submit VALUE=Click></TD>
<TD><INPUT TYPE=reset VALUE=Reset></TD>
</TABLE>
</FORM>
```

В URL можно размещать сложные программы и вызовы функций. Следует только помнить, что схема javascript работает не во всех браузерах, а только в версиях Netscape Navigator и Internet Explorer, начиная с четвертой.

Таким образом при программировании гипертекстового перехода интерпретатор получает управление после того, как пользователь "кликнул" по гипертекстовой ссылке.

Обработчики событий

Такие программы, как обработчики событий (handler), указываются в атрибутах контейнеров, с которыми эти события связаны. Например, при нажатии на кнопку происходит событие click:

<FORM><INPUT TYPE=button VALUE="Кнопка" onClick="window.alert('intuit');"></FORM>

Подстановки

Подстановка (entity) встречается на Web-страницах довольно редко. Тем не менее это достаточно мощный инструмент генерации HTML-страницы на стороне браузера. Подстановки используются в качестве значений атрибутов HTML-контейнеров. Например, как значение по умолчанию поля формы, определяющего домашнюю страницу пользователя, будет указан URL текущей страницы:

```
<SCRIPT>
function l()
{
str = window.location.href;
```

```
return(str.length);
}
</SCRIPT>
<FORM><INPUT VALUE="&{window.location.href};" SIZE="&{l()};"></FORM>
<!-- Это комментарий
...JavaScript-код...
// -->
</SCRIPT>
<BODY>
... Тело документа ...
</BODY>
</HTML>
```

HTML-комментарии здесь вставлены для защиты от интерпретации данного фрагмента страницы HTML-парсером в старых браузерах (у высокого начальства еще встречаются). В свою очередь, конец HTML-комментария защищен от интерпретации JavaScript-интерпретатором (// в начале строки). Кроме того, в качестве значения атрибута LANGUAGE у тега начала контейнера указано значение "JavaScript". VBScript, который является альтернативой JavaScript — это скорее экзотика, чем общепринятая практика, поэтому данный атрибут можно опустить — значение "JavaScript" принимается по умолчанию.

Очевидно, что размещать в заголовке документа генерацию текста страницы бессмысленно — он не будет отображен браузером. Поэтому в заголовок помещают декларации общих переменных и функций, которые будут затем использоваться в теле документа. При этом браузер Netscape Navigator более требовательный, чем Internet Explorer. Если не разместить описание функции в заголовке, то при ее вызове в теле документа можно получить сообщение о том, что данная функция не определена.

Приведем пример размещения и использования функции:

```
<HTML>
<HEAD>
<SCRIPT>
function time_scroll()
{
d = new Date();
window.status = d.getHousrs()+":"+d.getMinutes()+":"+d.getSconds();
setTimeout('time_scroll();';500);
}
</SCRIPT>
</HEAD>
<BODY onLoad=time_scroll()>
<CENTER>
<H1>Часы в строке статуса</H1>
```

В Internet Explorer 4.0 подстановки не поддерживаются, поэтому пользоваться ими следует аккуратно. Прежде чем выдать браузеру страницу с подстановками, нужно проверить тип этого браузера.

В случае подстановки интерпретатор получает управление в момент разбора браузером (компонент парсер) HTML-документа. Как только парсер встречает конструкцию &{..} у атрибута контейнера, он передает управление интерпретатору JavaScript, который, в свою очередь, после исполнения кода это управление возвращает парсеру. Таким образом данная операция аналогична подкачке графики на HTML-страницу.

Вставка (контейнер SCRIPT — принудительный вызов интерпретатора)

Контейнер SCRIPT — это развитие подстановок до возможности генерации текста документа JavaScript-кодом. В этом смысле применение SCRIPT аналогично Server Site Includes, т.е. генерации страниц документов на стороне сервера. Однако здесь мы забежали чуть вперед. При разборе документа HTML-парсер передает управление интерпретатору после того, как встретит тег начала контейнера SCRIPT. Интерпретатор получает на исполнение весь фрагмент кода внутри контейнера SCRIPT и возвращает управление HTML-парсеру для обработки текста страницы после тега конца контейнера SCRIPT.

Контейнер SCRIPT выполняет две основные функции:

- размещение кода внутри HTML-документа;
- условная генерация HTML-разметки на стороне браузера.

Первая функция аналогична декларированию переменных и функций, которые потом можно будет использовать в качестве программ переходов, обработчиков событий и подстановок. Вторая — это подстановка результатов исполнения JavaScript-кода в момент загрузки или перезагрузки документа.

Размещение кода внутри HTML-документа

Собственно, особенного разнообразия здесь нет. Код можно разместить либо в заголовке документа, внутри контейнера HEAD, либо внутри BODY. Последний способ и его особенности будут рассмотрены в разделе "Условная генерация HTML-разметки на стороне браузера". Поэтому обратимся к заголовку документа.

Код в заголовке размещается внутри контейнера SCRIPT

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE=JavaScript>
<FORM>
<INPUT TYPE=button VALUE="Закрыть окно" onClick=window.close()>
</FORM>
</CENTER>
</BODY>
</HTML>
```

В этом примере мы декларировали функцию time_scroll() в заголовке документа, а потом вызвали ее как обработчик события load в теге начала контейнера BODY(onLoad=time_scroll()).

В качестве примера декларации переменной рассмотрим изменение статуса окна-потомка из окна-предка:

Создадим дочернее окно с помощью следующей функции, продекларировав ее, а затем и вызвав:

```
function sel()
{
id = window.open("","example","width=500,height=200,status,menu");
id.focus();
id.document.open();
id.document.write("<HTML><HEAD>");
id.document.write("<BODY>");
id.document.write("<CENTER>");
id.document.write("<H1>Change text into child window.</H1>");
id.document.write("<FORM NAME=f>");
id.document.write("<INPUT TYPE=text NAME=t SIZE=20 MAXLENGTH=20 VALUE='This
is the test'>");
id.document.write("<INPUT TYPE=button
```

```
VALUE='Close the window' onClick=window.close()></FORM>");
id.document.write("</CENTER>");
id.document.write("</BODY></HTML>");
id.document.close();
}
<INPUT TYPE=button VALUE="Изменить поле статуса в окне примера"
onClick="id.defaultStatus='Привет';id.focus();">
```

Открывая окно-потомок, мы поместили в переменную id указатель на объект окно id=window.open(). Теперь мы можем использовать ее как идентификатор объекта класса Window. Использование id.focus() в нашем случае обязательно. При нажатии на кнопку "Изменить поле статуса в окне примера" происходит передача фокуса в родительское окно. Оно может иметь размер экрана. При этом изменения будут происходить в окне-потомке, которое будет скрыто родительским окном. Для того чтобы увидеть изменения, надо передать фокус. Переменная id должна быть определена за пределами каких-либо функций, что и сделано. В этом случае она становится свойством окна. Если мы поместим ее внутри функции открытия дочернего окна, то не сможем к ней обратиться из обработчика события click.

Условная генерация HTML-разметки на стороне браузера

Всегда приятно получать с сервера страницу, подстроенную под возможности нашего браузера или, более того, под пользователя. Существует только две возможности генерации таких страниц: на стороне сервера или непосредственно у клиента. JavaScript-код исполняется на стороне клиента (на самом деле, серверы компании Netscape способны исполнять JavaScript-код и на стороне сервера, только в этом случае он носит название LiveWire-код; не путать с LiveConnect), поэтому рассмотрим только генерацию на стороне клиента.

Для генерации HTML-разметки контейнер SCRIPT размещают в теле документа. Простой пример — встраивание в страницу локального времени:

```
<BODY>
...
<SCRIPT>
d = new Date();
document.write("<BR>");
document.write("Момент загрузки страницы: "+d.getHours()+":"+d.getMinutes()
+":"+d.getSeconds());
document.write("<BR>");
</SCRIPT>
...
<BODY>
```

Рекомендуемая литература [11, 13]

Тема 6 Сценарии стороны сервера. Технология PHP.

PHP был задуман где-то 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 упрощенного 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.

Сегодня (в середине -1998г.) как PHP/FI так и PHP3 поставляется с рядом коммерческих продуктов типа C2's StrongHold web server и RedHat Linux и консервативной оценкой, основанной на экстраполяции чисел, предоставленных NetCraft было бы то, что PHP используется на 150,000 сайтах во всем мире. В перспективе, их больше чем сайтов, запущенных на Netscape's flagship Enterprise server в Интернете.

Рекомендуемая литература [6, 8, 9]

Тема 7 Технология создания приложений с применением серверов баз данных.

MySQL – это одна из самых популярных и самых распространенных СУБД (система управления базами данных) в интернете. Она не предназначена для работы с большими объемами информации, но ее применение идеально для интернет сайтов, как небольших, так и достаточно крупных.

MySQL отличатся хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей. Поддержка сервера MySQL автоматически включается в поставку PHP.

Немаловажным фактором является ее бесплатность. MySQL распространяется на условиях общей лицензии GNU (GPL, GNU Public License).

Ранее для долговременного хранения информации мы работали с файлами: помещали в них некоторое количество строчек, а затем извлекали их для последующей работы. Задача длительного хранения информации очень часто встречается в программировании Web-приложений: подсчёт посетителей в счётчике, хранение сообщений в форуме, удалённое управление содержанием информации на сайте и т.д.

Между тем, профессиональные приёмы работы с файлами очень трудоёмки: необходимо заботится о помещении в них информации, о её сортировке, извлечении, при этом не нужно забывать, что все эти действия будут происходить на сервере хост-провайдера, где с очень большой вероятностью стоит один из вариантов Unix - следовательно, нужно так же заботится о правах доступа к файлам и их размещении. При этом объём кода значительно возрастает, и совершить ошибку в программе очень просто.

Все эти проблемы решает использование базы данных. Базы данных сами заботятся о безопасности информации и её сортировке и позволяют извлекать и размещать информацию при помощи одной строчки. Код с использованием базы данных получается более компактным, и отлаживать его гораздо легче. Кроме того, не нужно забывать и о скорости - выборка информации из базы данных происходит значительно быстрее, чем из файлов.

Примечание

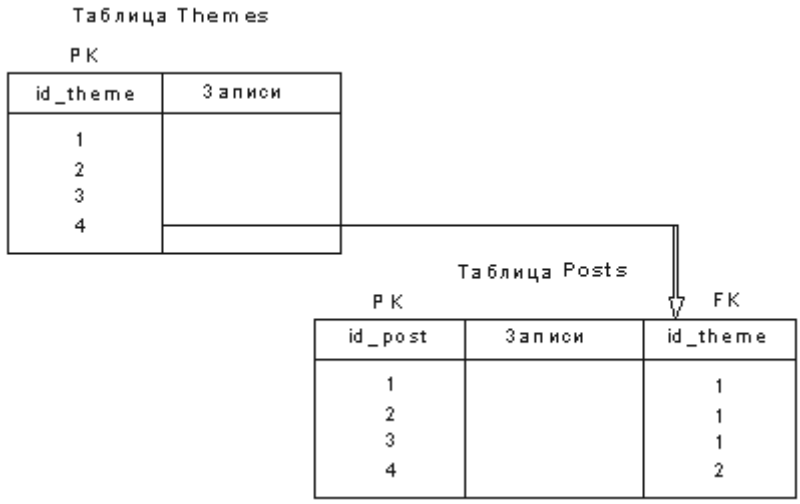
Приложение на PHP, использующее для хранения информации базу данных (в частности MySql) всегда работает быстрее приложения, построенного на файлах. Дело в том, что базы данных написаны на языке C++, и написать на PHP программу, которая работала бы с жёстким диском эффективнее базы данных - задача неразрешимая по определению, поскольку программы на PHP в принципе работают медленнее, чем программы на C++, так как PHP - интерпретатор, а C++ - компилятор.

Таким образом, основное достоинство базы данных заключается в том, что она берёт на себя всю работу с жёстким диском и делает это очень эффективно.

Первичный ключ (primary key) представляет собой один из примеров уникальных индексов и применяется для уникальной идентификации записей таблицы. Никакие из двух записей таблицы не могут иметь одинаковых значений первичного ключа. Первичный ключ обычно сокращенно обозначают как PK (primary key).

Как мы уже говорили, в реляционных базах данных практически всегда разные таблицы логически связаны друг с другом. Первичные ключи как раз используются для однозначной организации такой связи.

К примеру, в базе данных Forum таблицы themes и posts связаны между собой следующим образом:



Первичным ключом таблицы themes является id_theme, а таблицы posts - id_post. Обратите внимание, что поле id_theme присутствует и в таблице posts. Каждое значение этого поля в таблице posts является **внешним ключом** (в данном случае это внешний ключ для первичного ключа таблицы themes). Внешний ключ сокращенно обозначают как FK (foreign key).Как видно из рис.1, внешний ключ ссылается на первичный ключ таблицы themes, устанавливая однозначную логическую связь между записями таблиц themes и posts. Иначе говоря, если внешний ключ для записи (сообщения) с PK=1 в таблице posts имеет значение внешнего ключа равное 1, то это значит, что это сообщение относится к теме с PK=1 таблицы themes.

По способу задания первичных ключей различают **логические** (естественные) ключи и **суррогатные** (искусственные).

Для логического задания первичного ключа нужно выбрать в базе данных то, что естественным образом определяет запись. Примером такого ключа является номер паспорта в базе данных о паспортных данных жителей.

Если подходящих примеров для естественного задания первичного ключа не находится, пользуются суррогатным ключом. Суррогатный ключ представляет собой дополнительное поле в базе данных, предназначенное для обеспечения записей первичным ключом.

3.5 Работа с сервером MySQL

В этом разделе мы поговорим о том, как работать с клиентской программой mysql, с помощью которой можно подсоединяться к MySQL-серверу, выполнять SQL-запросы и просматривать результаты этих запросов. Текст этого раздела рассчитан на то, что на вашем компьютере уже установлена утилита mysql и существует связь с сервером MySQL.

При подключении к серверу MySQL с помощью программы mysql нужно ввести имя пользователя, и, как правило, пароль. Если сервер и клиент находятся на разных машинах, необходимо также указать имя хоста, на котором запущен сервер MySQL:

```
shell> mysql -h host -u user -p
```

После этого на экране появится запрос Enter password:, и вам нужно будет ввести свой пароль. Если соединение прошло нормально, то на экране появляется следующая информация и метка командной строки mysql>:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 459 to server version:
Type 'help' for help.
mysql>
```

Появление метки mysql> означает, что программа mysql готова к работе.

Отсоединиться от сервера можно в любой момент, набрав команду QUIT:

```
mysql> QUIT
Bye
```

Кроме этого, разорвать соединение с сервером можно также, одновременно нажав клавиши +.

После того, как вы подсоединитесь к серверу, для того, чтобы освоиться с синтаксисом команд, можно выполнить несколько простых запросов. Поскольку пока еще никакой базы данных не выбрано, приводимые ниже запросы носят общий характер.

Ниже приведена простая команда, запрашивающая у сервера информацию об его версии и текущей дате:

```
mysql> SELECT VERSION(), CURRENT_DATE;
```

Ответом MySQL на этот запрос будет следующая таблица:

```
+--------------+--------------------+
| version() | current_date |
+--------------+--------------------+
1 row in set (0.02 sec)
```

На примере выполнения этого запроса можно увидеть следующие основные особенности работы с MySQL:

- Команда, посылаемая серверу, обычно состоит из SQL-выражения, за которым следует точка с запятой. Из этого правила есть несколько исключений, когда после команды точка с запятой не ставится, к примеру, уже упомянутая команда QUIT;
- MySQL выводит результаты запроса в виде таблицы;
- После вывода таблицы с результатами запроса, mysql сообщает количество возвращаемых строк и время выполнения запроса. Это удобно, поскольку позволяет оценить как производительность сервера, так и эффективность выполняемого запроса;
- После вывода результатов запроса и времени его выполнения, mysql выводит новую строку mysql>, что означает готовность к выполнению новых команд.

Рекомендуемая литература [8, 9, 6]

ЛИТЕРАТУРА

Основная:

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Учебник. СПб: Питер, 1999.
2. Петров В.Н. Информационные системы. Учебник. - СПб: Питер, 2002.
3. Филимонов А.Ю. Протоколы Интернета. - СПб:БХВ-Петербург,2003.
4. Найк Д. Стандарты и протоколы Интернета. Пер. с англ. - М.:1999.
5. Ганеев Р.М. Проектирование интерактивных WEB-приложений. - М.: 2001.
6. Успенский И.И. Интернет как инструмент маркетинга. - СПб: БХВ-Петербург, 2000.
7. Ливингстон Д., Белью К., Браун М. Perl 5. Web - профессионалам: Пер. с англ. - К.: BHV, 2001.

Дополнительная:

8. Косентино К. PHP. Web - профессионалам: Пер. с англ. - К.: BHV, 2001.
9. Кузнецов С.Д. PHP 4.0.Руководство пользователя.- М.: Майор, 2001.
10. Леонтьев Б. Web -дизайн: Хитрости и тонкости: -М.: МиК, 2001.
11. Николенко Д.В. Практические занятия по JavaScript.СПб.:2002.
12. Гультяев А.К. Машин В.А. Уроки WEB-мастера.СПб.: 2002.
13. Рик Дарнелл. Javascript-справочник.2001.
14. Жумагалиев Б.И. Лабораторный практикум по интернет-технологиям. Учебное пособие. - Алматы: ААЭиС, 2003.