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

Кафедра электронных приборов

ГЛОБАЛЬНЫЕ КОМПЬЮТЕРНЫЕ СЕТИ

Методические указания к лабораторным работам
для студентов направления «Электроника и микроэлектроника»
(специальность «Электронные приборы и устройства»)

2012

Шандаров Евгений Станиславович

Глобальные компьютерные сети = Глобальные компьютерные сети: Методические указания к лабораторным работам для студентов направления «Электроника и микроэлектроника» (специальность «Электронные приборы и устройства») / Е.С. Шандаров; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Томский государственный университет систем управления и радиоэлектроники, Кафедра электронных приборов. - Томск : ТУСУР, 2012. - 25 с.

Данный курс лабораторных работ посвящен дисциплине «Глобальные компьютерные сети» и включает в себя описание 6 лабораторных работ.

Лабораторные работы по курсу проводятся с использованием программного обеспечения операционной системы Linux, бесплатно распространяемого пакета OpenOffice.org и программного продукта PHP, также бесплатно-распространяемого.

В рамках данного курса студенты изучают различные аспекты сетевого взаимодействия компьютеров, работу сетевых протоколов прикладного уровня.

Предназначено для студентов очной и заочной форм, обучающихся по направлению «Электроника и микроэлектроника» (специальность «Электронные приборы и устройства») по курсу «Глобальные компьютерные сети»

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

Кафедра электронных приборов

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

ГЛОБАЛЬНЫЕ КОМПЬЮТЕРНЫЕ СЕТИ

Методические указания к лабораторным работам
для студентов направления «Электроника и микроэлектроника»
(специальность «Электронные приборы и устройства»

Разработчик

_____Е.С. Шандаров
_____2012 г

2012

СОДЕРЖАНИЕ

Введение

Данный курс лабораторных работ посвящен дисциплине «Глобальные компьютерные сети» и включает в себя описание 6 лабораторных работ.

Лабораторные работы по курсу проводятся с использованием программного обеспечения операционной системы Linux, бесплатно распространяемого пакета OpenOffice.org и программного продукта PHP, также бесплатно-распространяемого.

В рамках данного курса студенты изучают различные аспекты сетевого взаимодействия компьютеров, работу сетевых протоколов прикладного уровня.

Лабораторная работа №1. Исследование протокола HTTP

1.1 Введение

HTTP (HyperText Transfer Protocol - протокол передачи гипертекста) был разработан как основа World Wide Web.

Работа по протоколу HTTP происходит следующим образом: программа-клиент устанавливает TCP-соединение с сервером (стандартный номер порта-80) и выдает ему HTTP-запрос. Сервер обрабатывает этот запрос и выдает HTTP-ответ клиенту.

1.2 Теоретическая часть

1.2.1 Структура HTTP-запроса

HTTP-запрос состоит из заголовка запроса и тела запроса, разделенных пустой строкой. Тело запроса может отсутствовать.

Заголовок запроса состоит из главной (первой) строки запроса и последующих строк, уточняющих запрос в главной строке. Последующие строки также могут отсутствовать.

Запрос в главной строке состоит из трех частей, разделенных пробелами:

Метод (иначе говоря, команда HTTP):

– GET - запрос документа. Наиболее часто употребляемый метод;

– HEAD - запрос заголовка документа. Отличается от GET тем, что выдается только заголовок запроса с информацией о документе. Сам документ не выдается;.

– POST - этот метод применяется для передачи данных CGI-скриптам. Сами данные следуют в последующих строках запроса в виде параметров;

– PUT - разместить документ на сервере.

Ресурс - это путь к определенному файлу на сервере, который клиент

хочет получить (или разместить - для метода PUT). Если ресурс - просто какой-либо файл для считывания, сервер должен по этому запросу выдать его в теле ответа. Если же это путь к какому-либо CGI-скрипту, то сервер запускает скрипт и возвращает результат его выполнения.

Версия протокола - версия протокола HTTP, с которой работает клиентская программа.

Таким образом, простейший HTTP-запрос может выглядеть следующим образом:

GET / HTTP/1.0

Здесь запрашивается корневой файл из корневой директории web-сервера.

Строки после главной строки запроса имеют следующий формат:

Параметр: значение

Таким образом задаются параметры запроса. Это является необязательным, все строки после главной строки запроса могут отсутствовать; в этом случае сервер принимает их значение по умолчанию или по результатам предыдущего запроса (при работе в режиме Keep-Alive).

1.2.2 Наиболее употребительные параметры HTTP-запроса

Connection (соединение)- может принимать значения Keep-Alive и close. Keep-Alive ("оставить в живых") означает, что после выдачи данного документа соединение с сервером не разрывается, и можно выдавать еще запросы. Большинство браузеров работают именно в режиме Keep-Alive, так как он позволяет за одно соединение с сервером "скачать" html-страницу и рисунки к ней. Будучи однажды установленным, режим Keep-Alive сохраняется до первой ошибки или до явного указания в очередном запросе Connection: close.

close ("закрыть") - соединение закрывается после ответа на данный запрос.

User-Agent - значением является "кодовое обозначение" браузера, например:

Mozilla/4.0 (compatible; MSIE 5.0; Windows 95; DigExt)

Accept - список поддерживаемых браузером типов содержимого в порядке их предпочтения данным браузером, например:

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, */*

Referer - URL, с которого перешли на этот ресурс.

Host - имя хоста, с которого запрашивается ресурс. Необходимо, если на сервере имеется несколько виртуальных серверов под одним IP-адресом. В этом случае имя виртуального сервера определяется по этому полю.

Accept-Language - поддерживаемый язык. Имеет значение для сервера, который может выдавать один и тот же документ в разных языковых версиях.

1.2.3 Формат HTTP-ответа

Формат ответа очень похож на формат запроса: он также имеет заголовок и тело, разделенное пустой строкой.

Заголовок также состоит из основной строки и строк параметров, но формат основной строки отличается от таковой в заголовке запроса.

Основная строка запроса состоит из 3-х полей, разделенных пробелами:

– **версия протокола** - аналогичен соответствующему параметру запроса;.

– **код ошибки** - кодовое обозначение "успешности" выполнения запроса. Код 200 означает "все нормально" (OK);

– **словесное описание ошибки** - "расшифровка" предыдущего кода. Например для 200 это OK, для 500 - Internal Server Error.

1.2.4 Наиболее употребительные параметры http-ответа

Connection - аналогичен соответствующему параметру запроса. Если сервер не поддерживает Keep-Alive (есть и такие), то значение Connection в ответе всегда close.

Content-Type ("тип содержимого") - содержит обозначение типа содержимого ответа. В зависимости от значения Content-Type браузер воспринимает ответ как HTML-страницу, картинку gif или jpeg, как файл, который надо сохранить на диске, или как что-либо еще и предпринимает соответствующие действия. Значение Content-Type для браузера аналогично значению расширения файла для такой системы как Windows.

Некоторые типы содержимого:

– text/html - текст в формате HTML (веб-страница);

– text/plain - простой текст (аналогичен "блокнотовскому");

– image/jpeg - картинка в формате JPEG;

– image/gif - то же, в формате GIF;

– application/octet-stream - поток "октетов" (т.е. просто байт) для записи на диск.

Content-Length ("длина содержимого") - длина содержимого ответа в байтах.

Last-Modified ("Модифицирован в последний раз") - дата последнего изменения документа.

1.3. Экспериментальная часть

1.3.1 Цель работы

Изучить основы работы с протоколом HTTP.

1.3.2 Задание на лабораторную работу

С помощью программы TELNET осуществить взаимодействие по протоколу HTTP с несколькими web-ресурсами, находящимися на разном "расстоянии" от нас: в локальной сети ТУСУР, в городской томской сети, в российском сегменте Интернет, в "мировом" интернете. В форме запроса клиента применить следующие опции:

- запрос обычного html документа;
- запрос изображения с сервера;
- запрос с передачей параметров по методу GET (например запрос на поисковый сервер);
- запрос с передачей параметров по методу POST (например запрос на авторизацию).

1.3.3 Методические указания по выполнению работы

Для связи с сервером по протоколу HTTP с помощью программы TELNET вам, возможно пригодится следующая конструкция:

telnet www.tusur.ru 80

По окончании работы необходимо подготовить отчет. В отчете привести тексты запросов клиента и ответов сервера.

1.3.4 Содержание отчета

Отчет по проделанной работе готовится в текстовом редакторе OpenOffice.org Write и предоставляется для проверки в электронном виде в формате электронных документов PDF.

Отчет должен состоять из следующих частей:

- введение;
- постановка задачи;
- основная часть;
- заключение;
- приложение.

Лабораторная работа №2. Исследование технологии CGI

2.1 Введение

Common Gateway Interface - средство расширения возможностей технологии World Wide Web

Спецификация CGI была разработана в Центре Суперкомпьютерных Приложений Университета штата Иллинойс (NCSA). Работы над ней велись параллельно с Mosaic. С точки зрения общей архитектуры программного обеспечения World Wide Web, CGI определила все дальнейшее развитие системных средств.

До появления этой спецификации все новые возможности реализовывались в виде модулей, включенных в библиотеку общих кодов ЦЕРН. Разработчики серверов должны были использовать эти коды для реализации программ или заменять их своими собственными аналогами. Это означало, что после компиляции сервера добавить в него новые возможности будет невозможно. CGI в корне изменила эту практику.

Главное назначение Common Gateway Interface - обеспечение единообразного потока данных между сервером и прикладной программой, которая запускается из-под сервера. CGI определяет протокол обмена данными между сервером и программой. Для тех, кто знаком с протоколом HTTP, может показаться, что CGI - это просто подмножество этого протокола. Однако это не так. Во-первых, CGI определяет порядок взаимодействия сервера с прикладной программой, в котором сервер выступает инициирующей стороной, во-вторых, CGI определяет механизм реального обмена данными и управляющими командами в этом взаимодействии, что не определено в HTTP. Естественно, что такие понятия, как метод доступа, переменные заголовка, MIME, типы данных, заимствованы из HTTP и делают спецификацию прозрачной для тех, кто знаком с самим протоколом.

2.2 Теоретическая часть

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

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

- Шлюз - это CGI-скрипт, который используется для обмена данными с другими информационными ресурсами Internet или приложениями-демонами.

Обычная CGI-программа запускается сервером HTTP для выполнения некоторой работы, возвращает результаты серверу и завершает свое выполнение.

Шлюз выполняется точно также, только, фактически, он инициирует взаимодействие в качестве клиента с третьей программой. Если эта третья программа является сервисом Internet, например, сервер Gopher, то шлюз становится клиентом Gopher, который посылает запрос по порту Gopher, а после получения ответа пересылает его серверу HTTP.

Механизмы обмена данными

Собственно спецификация CGI описывает четыре набора механизмов обмена данными:

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

Переменные окружения

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

К общим переменным окружения относятся:
ERVER_SOFTWARE - определяет имя и версию сервера.
SERVER_NAME - определяет доменное имя сервера.
GATEWAY_INTERFACE - определяет версию интерфейса.

К запрос-ориентированным относятся:
SERVER_PROTOCOL - протокол сервера. Вообще говоря, CGI разрабатывалась не только для применения в World Wide Web с протоколом HTTP, но и для других протоколов также, но широкое применение получила только в WWW;

SERVER_PORT - определяет порт TCP, по которому осуществляется взаимодействие. По умолчанию для работы по HTTP используется 80 порт, но он может быть и переназначен при конфигурировании сервера;

REQUEST_METHOD - определяет метод доступа к информационному ресурсу. Это важнейшая переменная в CGI. Разные методы доступа используют различные механизмы передачи данных. Данная переменная может принимать значения GET, POST, HEAD и т. п.;

PATH_INFO - передает программе путь, часть спецификации URL, в

том виде, в котором она указана в клиентом. Реально это означает, что передается путь (адрес скрипта) в виде, указанном в HTML-документе;

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

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

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

"/cgi-bin/test", а в PATH_TRANSLATED -

"/usr/local/etc/httpd/cgi-bin/test".

SCRIPT_NAME - определяет адрес скрипта так, как он указан клиентом.

Если не указаны параметры, то значение этой переменной будут совпадать с PATH_INFO, но если переменные указаны, то все, что следует за знаком "?" будет отброшено.

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

SCRIPT+NAME --------> "/cgi-bin/search"

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

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

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

Следующий набор переменных связан с идентификацией пользователя и его машины:

REMOTE_HOST - доменный адрес машины, с которой осуществляется запрос.

REMOTE_ADDR - IP-адрес запрашивающей машины.

AUTH_TYPE - тип идентификации пользователя. Используется в случае если скрипт защищен от несанкционированного использования.

REMOTE_USER - используется для идентификации пользователя.

REMOTE_IDENT - данная переменная порождается сервером, если он поддерживает идентификацию пользователя по протоколу RFC-931. Рекомендовано использование этой переменной для первоначального использования скрипта.

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

CONTENT_TYPE - определяет MIME-тип данных, передаваемых скрипту. Используя эту переменную можно одним скриптом обрабатывать различные форматы данных.

CONTENT_LENGTH - определяет размер данных в байтах, которые передаются скрипту. Данная переменная чрезвычайно важна при обмене данными по методу POST, т.к. нет другого способа определить размер данных, которые надо прочитать со стандартного ввода.

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

Опции командной строки

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

необходимо передавать много данных, то лучше сразу выбрать метод POST, т.е. передачу данных через стандартный ввод.

Формат стандартного ввода

Стандартный ввод используется при передаче данных в скрипт по методу POST. Объем передаваемых данных задается переменной окружения CONTENT_LENGTH, а тип данных – переменной CONTENT_TYPE.

Если из HTML-формы надо передать запрос типа: a=b&b=c,

то

CONTENT_LENGTH=7,

CONTENT_TYPE=application/x-www-form-urlencoded, а первым символом в стандартном вводе будет символ "a". Следует всегда помнить, что конец файла сервером в скрипт не передается, а поэтому завершать чтение следует по числу прочитанных символов.

Формат стандартного вывода

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

Content-type, Location, Status.

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

Location используется для переадресации. Иногда переадресация помогает преодолеть ограничения сервера или клиента на обработку встроенной графики или серверной предобработки. В этом случае скрипт создает файл на диске и указывает его адрес в Location. Сервер, таким образом, передает реально существующий файл.

2.3 Экспериментальная часть

2.3.1 Цель работы

Познакомиться с технологией CGI и языком PHP. Написать простейшие CGI-скрипты.

2.3.2 Задание на лабораторную работу

На языке PHP написать скрипт, который выводит на странице имена и значения переменных окружения, текущее время и дату на сервере.

2.3.3 Методические указания по выполнению работы

Написать скрипт.

По итогам выполнения работы подготовить отчет. В отчете привести исходный код скрипта и результат его работы.

2.3.4 Содержание отчета

Отчет по проделанной работе готовится в текстовом редакторе OpenOffice.org Write и предоставляется для проверки в электронном виде в формате электронных документов PDF.

Отчет должен состоять из следующих частей:

- введение;
- постановка задачи;
- основная часть;
- заключение;
- приложение.

Лабораторная работа №3. Изучение механизма Cookies
3.1

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

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

Сами по себе cookies не могут делать ничего, это только лишь некоторая информация. Однако, сервер может реагировать на содержащуюся в cookies информацию. Например, в случае авторизованного доступа к чему либо через WWW, в cookies сохраняется login и password в течение сессии, что позволяет не вводить их при запросе каждого запаролированного документа. Другой пример: cookies могут использоваться для построения персонализированных страниц.

Полное описание поля Set-Cookie HTTP заголовка:

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

Минимальное описание поля Set-Cookie HTTP заголовка:

Set-Cookie: NAME=VALUE;

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

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

domain=DOMAIN_NAME - домен, для которого значение cookie действительно. Например, domain=cit-forum.com. В этом случае значение cookie будет действительно и для сервера cit-forum.com, и для www.cit-forum.com. Если этот атрибут опущен, то по умолчанию используется доменное имя сервера, с которого было выставлено значение cookie.

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

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

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

3.3 Экспериментальная часть

3.3.1 Цель лабораторной работы

Познакомиться с механизмом использования Cookies

3.3.2 Задание на лабораторную работу

Создать систему авторизации пользователя с помощью механизма Cookies.

3.3.3 Методические указания по выполнению работы

Создать скрипт с формой авторизации пользователя. Обеспечить доступ к внутренней странице только авторизованным пользователям.

По окончании работы подготовить отчет с исходными текстами скриптов и внешним видом страниц сайта.

3.3.4 Содержание отчета

Отчет должен состоять из следующих частей:

- введение;
- постановка задачи;
- основная часть;
- заключение;
- приложение.

Лабораторная работа №4. Исследование протокола SMTP

4.1 Введение

Основная задача протокола SMTP (Simple Mail Transfer Protocol) заключается в том, чтобы обеспечивать передачу электронных сообщений (почту). Для работы через протокол SMTP клиент создаёт TCP соединение с сервером через порт 25. Затем клиент и SMTP сервер обмениваются информацией пока соединение не будет закрыто или прервано. Основной процедурой в SMTP является передача почты (Mail Procedure). Далее идут процедуры форвардинга почты (Mail Forwarding), проверка имён почтового ящика и вывод списков почтовых групп. Самой первой процедурой является открытие канала передачи, а последней - его закрытие.

Команды SMTP указывают серверу, какую операцию хочет произвести клиент. Команды состоят из ключевых слов, за которыми следует один или более параметров. Ключевое слово состот из 4-х символов и разделено от аргумента одним или несколькими пробелами. Каждая командная строка заканчивается символами CRLF. Вот синтаксис всех команд протокола SMTP (SP - пробел):

HELO <SP> <domain> <CRLF>
MAIL <SP> FROM:<reverse-path> <CRLF>
RCPT <SP> TO:<forward-path> <CRLF>
DATA <CRLF>
RSET <CRLF>
SEND <SP> FROM:<reverse-path> <CRLF>
SOML <SP> FROM:<reverse-path> <CRLF>
SAML <SP> FROM:<reverse-path> <CRLF>
VRFY <SP> <string> <CRLF>
EXPN <SP> <string> <CRLF>
HELP <SP> <string> <CRLF>
NOOP <CRLF>
QUIT <CRLF>

Обычный ответ SMTP сервера состоит из номера ответа, за которым через пробел следует дополнительный текст. Номер ответа служит индикатором состояния сервера.

Отправка почты

Первым делом подключаемся к SMTP серверу через порт 25. Теперь надо передать серверу команду HELLO и наш IP адрес:

C: HELLO 88.204.75.135
S: 250 ms.tusur.ru is ready

При отправке почты передаём некоторые нужные данные (отправитель, получатель и само письмо):

C: MAIL FROM:<shandarov> 'указываем отправителя

S: 250 OK

C: RCPT TO:<shandarov@mail.ru> 'указываем получателя
S: 250 OK
указываем серверу, что будем передавать содержание письма (заголовок и тело письма)
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
передачу письма необходимо завершить символами CRLF.CRLF
S: 250 OK

C: From: Shandarov <shandarov@mail.ru>
C: To: support <support@mail.ru>
C: Subject: Hello
между заголовком письма и его текстом не одна пара CRLF, а две.
C: Hello Drol!
C: You will be die on next week!
заканчиваем передачу символами CRLF.CRLF
S: 250 OK
Теперь завершаем работу, отправляем команду QUIT:
S: QUIT
C: 221 ms.tusur.ru is closing transmission channel

Другие команды

– SEND - используется вместо команды MAIL и указыает, что почта должна быть доставлена на терминал пользователя.

– SOML, SAML - комбинации команд SEND или MAIL, SEND и MAIL соответственно.

– RSET - указвает серверу прервать выполнение текущего процесса. Все сохранённые данные (отправитель, получатель и др) удаляются. Сервер должен отправить положительный ответ.

– VRFY - просит сервер проверить, является ли переданный аргумент именем пользователя. В случае успеха сервер возвращает полное имя пользователя.

– EXPN - просит сервер подтвердить, что переданный аргумент - это список почтовой группы, и если так, то сервер выводит членов этой группы.

– HELP - запрашивает у сервера полезную помощь о переданной в качестве аргумента команде.

– NOOP - на вызов этой команды сервер должен положительно ответить. NOOP ничего не делает и никак не влияет на указанные до этого данные.

4.3 Экспериментальная часть

4.3.1 Цель лабораторной работы

Познакомиться с протоколом SMTP

4.3.2 Задание на лабораторную работу

Освоить основные команды протокола SMTP, написать web-приложение для отправки почты.

4.3.3 Методические указания по выполнению работы

С помощью программы TELNET отправить электронное письмо.

Создать скрипт с формой подготовки и отправки электронного письма.

По окончании работы подготовить отчет с исходными текстами скриптов и внешним видом страниц сайта.

4.3.4 Содержание отчета

Отчет должен состоять из следующих частей:

- введение;
- постановка задачи;
- основная часть;
- заключение;
- приложение.

Лабораторная работа №5. Исследованиие протокола FTP

5.1 Теоретическая часть

FTP (англ. File Transfer Protocol — протокол передачи файлов) — протокол, предназначенный для передачи файлов в компьютерных сетях. FTP позволяет подключаться к серверам FTP, просматривать содержимое каталогов и загружать файлы с сервера или на сервер.

FTP является одним из старейших прикладных протоколов, появившимся задолго до HTTP, в 1971 году. Он и сегодня широко используется для распространения ПО и доступа к удалённым хостам.

Протокол FTP относится к протоколам прикладного уровня и для передачи данных использует транспортный протокол TCP. Команды и данные, в отличие от большинства других протоколов, передаются по разным портам. Исходящий порт 20, открываемый на стороне сервера, используется для передачи данных, порт 21 для передачи команд. Порт для приема данных клиентом определяется в диалоге согласования. В случае, если передача файла была прервана по каким-либо причинам, протокол предусматривает средства для докачки файла, что бывает очень удобно при передаче больших файлов.

5.2. Экпериментальная часть

5.2.1 Цель лабораторной работы

Познакомиться с протоколом FTP.

5.2.2 Задание на лабораторную работу

Освоить основные команды протокола FTP, загрузить файл с FTP-сервера, отправить файл на FTP-сервер.

5.2.3 Методические указания по выполнению работы

Изучить команды протокола FTP с помощью системы man.

С помощью программы FTP подключиться к ftp-серверу ed.tusur.ru.

Загрузить с сервера на клиентский компьютер файл текстовый и файл бинарный.

Отправить на сервер небольшой файл.

Протестировать функцию докачки при обрыве соединения

По окончании работы подготовить отчет со скриншотами, списком и описанием команд протокола.

5.2.4 Содержание отчета

Отчет по проделанной работе готовится в текстовом редакторе OpenOffice.org Write и предоставляется для проверки в электронном виде в формате электронных документов PDF.

Отчет должен состоять из следующих частей:

- введение;
- постановка задачи;
- основная часть;
- заключение;
- приложение.

Лабораторная работа №6. Знакомство с MIME-типами

6.1 Экспериментальная часть
6.1.1 Цель лабораторной работы

Познакомиться с MIME-типами.

6.1.2 Задание на лабораторную работу

Создать web-приложение с помощью которого представить данные в различных MIME-типах.

6.1.3 Методические указания по выполнению работы

Изучить набор стандартных MIME-типов.
На сервере в своей папке подготовить следующий набор файлов:
- текстовый файл;
- html-документ;
- xml-документ;
- изображение в формате JPEG;
- файл PDF.

Создать web-приложение с помощью которого можно указанные файлы представить в следующих MIME-типах:
- text/plain;
- text/html;
- text/xml;
- image/jpeg;
- application/pdf;
- application/octet-stream.

По окончании работы подготовить отчет со скриншотами, исходным текстом программ и собственными выводами по работе.

6.3.4 Содержание отчета

Отчет должен состоять из следующих частей:
- введение;
- постановка задачи;
- основная часть;
- заключение;
- приложение.

Приложение А

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

Кафедра электронных приборов

дисциплина «Глобальные компьютерные сети»

ОТЧЕТ
по лабораторной работе
«_____»

Выполнил
Студент гр. _____
_____И.О. Фамилия
_____ 2012 г

Проверил преподаватель
_____ И.О. Фамилия
_____2012 г

2012

Учебное пособие

Шандаров Е.С.

Глобальные компьютерные сети
Методические указания по лабораторным лаботам

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