**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

***Д.В. Еленев***

# РАБОТА С СЕРВИСАМИ СЕТИ ИНТЕРНЕТ

**САМАРА 2010**

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

*Д.В. Еленев*

# РАБОТА С СЕРВИСАМИ СЕТИ ИНТЕРНЕТ

*Утверждено Редакционно-издательским советом университета*
*в качестве учебного пособия*

С А М А Р А
Издательство СГАУ
2010

УДК СГАУ: 004(075)
ББК 32.97
Е504

Рецензенты:
проректор по информатизации СГАУ
доктор технических наук, профессор В. С. К у з ь м и ч ё в;
начальник информационно-вычислительного центра СамГТУ
кандидат технических наук, доцент Л. А. Л ь н о г р а д с к и й

***Еленев Д.В.***

Е504 **Работа с сервисами сети Интернет**: учеб. пособие / *Д.В. Еленев*. – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2010. – 52 с.: ил.

**ISBN 978-5-7883-0797-8**

Рассматриваются сервисы WWW, FTP и электронной почты сети Internet, описываются компоненты рассматриваемых сервисов и схемы их взаимодействия, приводятся задания на лабораторные работы, связанные с решением типовых задач при работе с данными сервисами, реализацией протоколов HTTP, FTP, SMTP, POP3 как со стороны клиента, так и со стороны сервера.

Предназначено для студентов старших курсов очной и заочной форм обучения, обучающихся по специальности «Автоматизированные системы обработки информации и управления».

УДК СГАУ: 004(075)
ББК 32.97

**ISBN 978-5-7883-0797-8** 

**СОДЕРЖАНИЕ**

## СПИСОК ПРИНЯТЫХ СОКРАЩЕНИЙ

CGI – Common Gateway Interface
FTP – File Transfer Protocol
HTTP – Hypertext Transfer Protocol
MIME – Multipurpose Internet Mail Extensions
POP3 – Post Office Protocol version 3
RFC – Request For Comments
SMTP – Simple Mail Transfer Protocol
TCP – Transmission Control Protocol
URI – Uniform Resource Identifier
URL – Uniform Resource Locator

# ВВЕДЕНИЕ

В настоящее время в сети Internet наибольшее распространение получили такие сервисы, как WWW (World Wide Web – всемирная паутина), электронная почта (e-mail) и служба передачи файлов при помощи протокола FTP.

Перечисленные сервисы имеют различные предпосылки и историю своего возникновения. Например, относительно молодой, но динамично развивающийся сервис WWW появился в 1990 г., в то время как годом появления протокола FTP можно считать 1971, когда на заре существования вычислительных сетей был предложен механизм передачи файлов, а перечень спецификаций, связанных с этим протоколом, содержит более 40 пунктов. Электронная почта также является сервисом, без которого сейчас трудно представить сеть Internet; это хорошо видно из того, что большое количество крупных компаний конкурируют между собой, предоставляя услуги бесплатной электронной почты на основе web-интерфейсов и с возможностью доступа к ней по протоколам SMTP и POP3.

Данное пособие может быть условно разделено на две части. В главах 1-4 приводятся описания сервисов и принципы их работы, а в пятой главе в форме лабораторных работ приведены типовые задачи, возникающие при работе с протоколами HTTP, FTP, SMTP и POP3. При этом для сервисов WWW и FTP предлагаются задания на разработку как клиентских, так и серверных приложений.

Приведенные в пособии задания используются с незначительными вариациями в качестве лабораторных работ для студентов очной формы обучения и контрольных работ для студентов заочной формы обучения специальности «Автоматизированные системы обработки информации и управления» по курсам «Сети ЭВМ и телекоммуникации» и «Сетевые технологии» в Самарском государственном аэрокосмическом университете имени академика С.П.Королева, начиная с 2005 года.

# 1 ОБЩИЕ ПРИНЦИПЫ РАБОТЫ СЕРВИСОВ СЕТИ INTERNET

## 1.1 Механизм «клиент-сервер»

Механизм «клиент-сервер» реализует доступ к информационным ресурсам, ввод и выполнение команд за счет использования двух взаимосвязанных программ. Первая принимает команды пользователя, называется «клиент» и использует вычислительные ресурсы пользователя. Вторая программа запускается на другом компьютере, который располагает информационными ресурсами и называется «сервер». Программа-сервер принимает заказ от своего удалённого клиента, обрабатывает его и отправляет обратно требуемую информацию с помощью соответствующего протокола передачи данных.

Таким образом, предоставлением услуг управляют программы, которые состоят, в общем случае, из двух компонент — клиента и сервера. Серверная и клиентская компоненты могут быть размещены и на одном компьютере. В большинстве случаев на одном компьютере, предоставляющем свои ресурсы пользователям, устанавливают не одну, а несколько программ-серверов. Для этого необходимо отличать отдельные службы приложений при помощи различных точек входа — портов.

Каждой программе-серверу присваивается определённый номер порта (то есть идентификатор сетевого процесса), по которой к этой программе-серверу обращается соответствующий клиент.

Поскольку сеть Internet обеспечивает связь со множеством клиентских компьютеров, для наиболее распространенных служб установлены стандартные номера портов для использования всеми пользователями. Например, служба Telnet связана с портом 23, служба передачи файлов FTP — с портами 20 и 21, служба WWW – с портом 80.

Для определения сетевых координат точки доступа к информационному ресурсу и протокола, по которому производится взаимодействие, используется универсальный идентификатор ресурсов.

## 1.2 Универсальный локатор ресурсов URL

Синтаксис универсального локатора ресурсов URL (Uniform Resource Locator) определен стандартом [4]. Общий вид URL следующий:

```
<scheme>:<scheme-specific-part>
```

URL состоит из имени используемой схемы `<scheme>`, за которым следует двоеточие, и строки `<scheme-specific-part>`, интерпретация которой зависит от конкретной используемой схемы.

Имя схемы состоит из последовательности символов, среди которых могут присутствовать символы нижнего регистра латинского алфавита (от «a» до «z»), цифры, точка, знаки «+» и «–». Для обеспечения совместимости программы, обрабатывающие URL, должны преобразовывать символы верхнего регистра в нижний регистр. В табл. 1.1 приведен перечень наиболее часто используемых схем универсального локатора ресурсов.

Т а б л и ц а 1.1. ***Основные схемы URL***

| | |
|---|---|
| ftp | Протокол передачи файлов FTP |
| http | Протокол передачи гипертекста HTTP |
| gopher | Протокол Gopher |
| mailto | Адрес электронной почты |
| news | Новости USENET |
| nntp | Новости USENET news с использованием доступа NNTP |
| telnet | Ссылка на интерактивное соединение |
| wais | Сервис Wide Area Information Servers |
| file | Имя файла |

Строка <scheme-specific-part> в общем случае имеет следующий синтаксис:

```
//<user>:<password>@<host>:<port>/<url-path>
```

Отдельные поля этой строки в зависимости от используемой схемы могут быть пропущены и в общем случае имеют следующие значения:

user – имя пользователя. Некоторые схемы (например, ftp) допускают указание имени пользователя;

password – пароль, указание которого не обязательно. Если пароль указывается, то он должен быть отделен от имени пользователя двоеточием. Если в URL присутствует имя пользователя (и, возможно, пароль), то к нему (к ним) должен быть добавлен символ "@". Если в имени пользователя или пароле встречаются символы ":", "@" или "/", то они должны быть закодированы;

host – полное доменное имя (fully qualified domain name, FQDN) находящегося в сети компьютера или его IP-адрес, записанный в виде четырех десятичных чисел, разделенных между собой точками;

port – номер порта, по которому должно производиться соединение. Большинство схем подразумевают наличие порта по умолчанию. Другой номер порта может быть указан в виде десятичного числа, отделенного от хоста двоеточием. Если номер порта не указывается, то двоеточие также не употребляется;

url-path – оставшаяся часть URL состоит из специфичных для конкретной схемы данных и обозначается как url-path (путь URL). Путь URL описывает каким образом может быть осуществлен доступ к данному ресурсу. При этом символ "/" между хостом (или номером порта) и путем URL не считается частью пути URL. Синтаксис пути URL зависит от используемой схемы.

## 1.3 Универсальный идентификатор ресурсов URI

Термин URL (универсальный локатор ресурсов) является понятием, входящим в понятие универсального идентификатора ресурсов (URI), который идентифицирует ресурсы путем представления их основного механизма доступа (т.е. их расположения в сети) вместо идентификации за счет его имени или каких-либо других атрибутов ресурса.

Следующие примеры, заимствованные из спецификации [5], показывают наиболее употребляемые варианты использования URI:

- `ftp://ftp.is.co.za/rfc/rfc1808.txt` – схема ftp для сервисов, работающих по протоколу передачи файлов FTP;
- `gopher://spinaltap.micro.umn.edu/00/Weather/California/Los%20Angeles` – схема gopher для сервисов, работающих по протоколам Gopher и Gopher+;
- `http://www.math.uio.no/faq/compression-faq/part1.html` – схема http для сервисов, использующих протокол передачи гипертекста HTTP;
- `mailto:mduerst@ifi.unizh.ch` – схема mailto для адресов электронной почты;
- `news:comp.infosystems.www.servers.unix` – схема news для новостных групп и статей USENET;
- `telnet://melvyl.ucop.edu/` – схема telnet для интерактивных сервисов, использующих протокол TELNET.

# 2 СИСТЕМА WORLD WIDE WEB

## 2.1 Базовая информация о системе WWW

Система WWW (World Wide Web, всемирная паутина) является богатейшей по своим возможностям информационной системой, основанной на механизме клиент/сервер.

Термин «web» (паутина) характеризует структура связей между отдельными файлами. В случае традиционной древовидной структуры имеется корневой каталог, от которого отходят различные дочерние подкаталоги, как, например, в файловой системе операционных систем семейства DOS. Находясь в одной из ветвей такой системы, можно переходить либо в подкаталоги низшего уровня, либо в родительский каталог. Но в древовидной структуре невозможно сразу перейти из одного каталога в другой, если эти каталоги не имеют общего корня, т.е. не связаны непосредственно. В отличие от такой схемы структура Web создаёт связи между отдельными файлами (называемыми документами). Такая структура связей напоминает паутину.

Таким образом, основная идея системы WWW заключается в том, что пользователь имеет возможность просматривать документы в удобном ему порядке, а не последовательно (постранично). Достигнуто это за счет создания специального механизма связи различных страниц текста при помощи гипертекстовых ссылок. Развитие идеи гипертекста привело к возможности построения связей не только между текстовыми документами, но и к включению в них графической, аудио-, видео- и любого другого рода информации, что существенно расширило возможности системы WWW.

Технология WWW в своей работе широко использует:

- универсальный способ адресации ресурсов в сети URL;
- протокол передачи гипертекстовой информации HTTP, описание которого приводится в разделе 2.2;

– язык гипертекстовой разметки документов HTML (Hypertext Markup Language);

– универсальный интерфейс шлюзов – CGI (Common Gateway Interface).

Протокол HTTP используется для передачи данных между программой-клиентом и web-сервером. Web-сервер ожидает запрос от клиента, при поступлении обрабатывает его и возвращает результат обработки. Гипертекстовые документы формируются с использованием языка гипертекстовой разметки документов HTML.

HTML базируется на стандартном языке разметки печатных документов, определяет форму представления информации (разметка) и структуру связей документов за счет использования гипертекстовых ссылок.

Для просмотра web-ресурсов разработано множество программ-клиентов, называемых браузерами (browser, обозреватели). Первым из них был текстовый браузер lynx, реализованный в операционной системе UNIX. С развитием и расширением возможностей системы WWW были разработаны различные браузеры с графическим интерфейсом, первым из которых стал браузер Mosaic, взятый за основу при создании широко распространенного в настоящее время браузера Microsoft Internet Explorer.

Web-сервер — это программа, принимающая запросы от клиентов и отправляющая ответы на них в соответствии с протоколом HTTP. В качестве ответа может быть HTML-документ, или результат выполнения внешней программы. Наиболее распространенным в настоящее время web-сервером является Apache.

Для расширения возможностей системы WWW за счёт использования внешнего программного обеспечения широко используется универсальный интерфейс шлюзов CGI. Интерфейс CGI обеспечивает взаимодействие web-сервера с внешними программами через стандарт-

ные потоки ввода/вывода, что значительно упрощает создание серверных web-приложений.

## 2.2 Протокол передачи гипертекста HTTP

Протокол HTTP (Hypertext Transfer Protocol) является протоколом прикладного уровня и используется, начиная с 1990 г., для передачи информации во всемирной паутине (World-Wide Web). Первая версия этого протокола, названная HTTP/0.9, являлась простым протоколом для передачи данных по сети Интернет. В версии HTTP/1.0 была добавлена возможность передачи данных с использованием формата, аналогичного MIME [10-12], т.е. с добавлением метаданных о передаваемой по данному протоколу информации и семантики запросов клиента и ответов сервера. Однако в версии HTTP/1.0 не рассматривались вопросы иерархических прокси-серверов, кэширования, необходимости постоянных соединений и виртуальных хостов, что потребовало введения следующей версии протокола, определенной в спецификации RFC 2068 [15] и дополненной спецификацией RFC 2616 [16].

Согласно спецификациям протокола HTTP все HTTP-транзакции имеют единый формат; запросы клиента и ответы сервера состоят из трех частей: строки запроса (ответа), заголовка запроса (ответа) и тела документа. Транзакция инициируется клиентом путем установления TCP-соединения с сервером по используемому сервером номеру порта. В сети Интернет за протоколом HTTP зарезервирован 80 порт, хотя использоваться может и любой другой. После установления соединения клиент отправляет серверу строку запроса документа, состоящую из метода запроса, адреса запрашиваемого документа (URI – Uniform Resource Identifier) и версии протокола HTTP. Например, строка запроса

```
GET /index.html HTTP/1.0
```

означает, что производится запрос по методу `GET` документа `index.html`, расположенного в корневом каталоге сервера, с использованием протокола HTTP версии 1.0. Признаком конца строки в протоколе HTTP считается последовательность CRLF (CR – Carriage Return, возврат каретки; LF – Linefeed, возврат строки; ASCII-коды 13 и 10 соответственно).

Далее клиент может отправить серверу необязательный заголовок запроса, в котором сообщается некоторая информация о приемлемых форматах документов, своей конфигурации, а также некоторая служебная информация. Например, для корректной отправки клиенту документов при использовании сервером возможностей виртуального хостинга требуется указать серверу адрес сайта в символьном виде, как-то vip.ssau.ru. В противном случае клиент может получить документ, относящийся к другому размещенному на данном сервере сайту. Все параметры заголовка отправляются построчно, в каждой строке содержатся имя параметра и его значение.

```
Host: vip.ssau.ru
Accept: image/gif, image/jpeg, image/pjpeg, */*
```

Признаком окончания заголовка является первое вхождение пустой строки.

После отправки заголовка запроса клиент может также передать серверу некоторые дополнительные данные. Эти данные главным образом используются CGI-скриптами, обрабатывающими запросы по методу `POST`, широко применяемому для редактирования размещенной на сайте информации.

Ответ сервера на запрошенную клиентом информацию выглядит следующим образом. Первая часть ответа – строка ответа – включает в себя версию протокола HTTP, код ответа сервера и его описание. Код ответа сервера является трехзначным десятичным числом, обозначаю-

щим результат обработки запроса сервером, а описание – мнемонической расшифровкой кода, обработка которого на стороне клиента не требуется. Заголовок ответа

```
HTTP/1.0 200 OK
```

означает, что сервером для ответа используется версия 1.0 протокола HTTP, запрос клиента согласно коду 200 был успешно обработан, а запрошенные данные будут переданы непосредственно после заголовка. Список кодов ответа сервера с их расшифровками и описаниями приведен ниже.

После строки ответа сервер передает клиенту заголовок ответа, содержащий информацию о сервере и запрошенном клиентом документе. Например, заголовок ответа сервера

```
Date: Sun, 05 Sep 2010 10:08:22 GMT
Server: Apache/2.2.8 (Unix) PHP/5.2.6
Last-Modified: Fri, 03 Sep 2010 09:38:09 GMT
Accept-Ranges: bytes
Content-Length: 25600
Connection: close
Content-Type: application/msword
```

сообщает клиенту текущую дату сервера, информацию о программном обеспечении сервера, дату последнего изменения документа, его размер в байтах и тип документа – документ MS Word.

Важной строкой в заголовке ответа сервера является указание типа передаваемых данных `Content-type`. На основании этой информации клиент должен определить, каким образом ему следует обработать переданные данные: вывести текст в окно браузера, отобразить графическую информацию и т.п. Практически при успешной обработке за-

проса клиента и последующей отправке данных наличие этой строки в заголовке является обязательным.

Как и при запросе клиента признаком окончания заголовка ответа сервера является пустая строка.

Если запрос клиента обработан успешно, то следом за заголовком отправляются запрошенные данные (тело документа), которые могут быть результатом выполнения CGI-скрипта или копией имеющегося на сервере файла. В случае, если при обработке запроса возникла ошибка, то клиенту передаются данные, содержащие изложение причин, по которым запрос не мог быть обработан успешно.

В протоколе HTTP версии 0.9 соединение закрывается сразу же после передачи данных. В версии 1.0 по умолчанию после передачи данных сервер закрывает соединение (завершая тем самым HTTP-транзакцию), но позволяет сохранить его, если в заголовке запроса указана строка

```
Connection: Keep-alive
```

В версии 1.1 по умолчанию соединение не закрывается сервером и клиент сразу же после получения данных может отправлять другие запросы. Такая схема позволяет экономить трафик и загрузку сервера и клиента, так как не требуются дополнительные расходы на многократную установку соединений при запросе всех файлов, связанных с содержимым одной www-страницы.

Каждая строка запроса клиента начинается с указания метода, по которому производится запрос документа. Метод сообщает серверу способ, которым должен быть обработан запрос. Название методов чувствительно к регистру, поэтому методы `GET` и `Get`, вообще говоря, различаются.

В табл. 2.1 приведены методы, определенные спецификацией протокола HTTP версии 1.1, и их краткие описания. Основными исполь-

зуемыми методами являются GET, HEAD и POST, в то время как остальные имеют меньшую распространенность.

Т а б л и ц а 2.1. ***Методы HTTP-запросов***

| Метод | Краткое описание |
|---|---|
| GET | Инициирует получение документа, расположенного по указанному в строке запроса адресу |
| HEAD | Аналогичен методу GET, но клиенту передается только заголовок ответа |
| POST | Метод используется для отправки клиентом данных на сервер. Передаваемые данные находятся в теле запроса |
| OPTIONS | Запрашивает информации о коммуникационных параметрах сервера. Позволяет клиенту определить параметры и/или требования, связанные с документом или совместимостью сервера, без инициации запроса данного документа или связанных с ним действий |
| PUT | Запрашивает размещение документа, отправляемого в теле запроса, по указанному в строке запроса адресу |
| DELETE | Запрашивает удаление документа, находящегося на сервере по указанному в строке запроса адресу.<br>Запросы по этому методу не кэшируются |
| TRACE | Метод позволяет клиенту увидеть без изменений информацию, полученную другой стороной. Используется для отладки.<br>Ответы на запросы по методу TRACE не должны кэшироваться |
| CONNECT | Метод зарезервирован спецификацией протокола HTTP версии 1.1 для использования прокси-серверами |

Метод GET инициирует получение документа, расположенного по указанному в строке запроса адресу. Этот метод является самым распространенным методом для получения данных с сервера. При запросе

по методу GET клиент оправляет строку и заголовок запроса, в то время как тело запроса всегда остается пустым, а сервер отвечает строкой ответа, заголовком ответа и помещенными в тело ответа запрошенными данными. Если данные не могут быть переданы, то в теле ответа передается дополнительная информация, содержащая пояснение о причине появления ошибки.

Метод `GET` может быть использован для передачи CGI-скриптам данных пользователя. Данные в этом случае передаются в строке запроса, разделенные амперсандом (&), после адреса запрашиваемого ресурса (CGI-скрипта), отделенные от адреса знаком вопроса. Например,

```
GET /cgi-bin/books.pl?author=yelenev&bookid=1
 HTTP/1.0
```

Однако при передаче серверу данных с использованием метода GET следует учитывать, что существует ограничение на максимальную длину переданных данных, практически не позволяющее производить передачу достаточно больших блоков данных на сервер (например, загрузку файлов). В случае подобной необходимости лучше использовать метод `POST`.

Метод `HEAD` идентичен методу `GET` за исключением того, что в теле ответа сервер ничего не отправляет клиенту. Метаданные, содержащиеся в заголовке ответа, полученного по методу `HEAD`, должны быть идентичны метаданным, полученным с использованием запроса по методу `GET`. Метод используется, когда клиент хочет получить только данные о документе, но не сам документ, например, при кэшировании запросов полезна информация о времени последнего изменения документа; данные о размере документа позволяют оценить время, необходимое для получения документа и т.п.

Метод POST используется для отправки клиентом данных на сервер. Данные, передаваемые на сервер, должны находиться в теле запроса. Данные направляются сервером для обработки в специальную программу (например, CGI-скрипт). Результат обработки запроса по методу POST не должен сказываться на ресурсе, расположенном по указанному в строке запроса адресу. Метод POST разработан для обеспечения следующих функций:

- комментирования существующих ресурсов;
- отправки сообщений на Интернет-форумы, новостные группы, списки рассылки и т.п.;
- передачи блоков данных, таких как результаты отправки форм, процессам обработки данных;
- удаленного внесения изменений в базы данных.

Коды ответов сервера делятся на 5 групп, приведенных в табл. 2.2.

Т а б л и ц а 2.2. ***Диапазоны кодов ответов сервера***

| **Диапазон** | **Расшифровка диапазона в спецификации протокола** | **Описание** |
|---|---|---|
| 100÷199 | Informational | Информационный |
| 200÷299 | Successful | Запрос клиента успешно принят и обработан |
| 300÷399 | Redirection | Запрос клиента переадресован |
| 400÷499 | Client Error | Ошибка клиента |
| 500÷599 | Server Error | Ошибка сервера |

В спецификации протокола HTTP в каждом диапазоне определены лишь несколько кодов, хотя для каждого конкретного HTTP-сервера при необходимости могут определяться собственные коды. При получении кода, который он не может распознать, клиент интерпретирует его в соответствии с принадлежностью кода диапазону. Коды, находящиеся в первых трех диапазонах, как правило, обрабатываются боль-

шинством браузеров без извещения пользователя, в то время как некоторые коды ошибок из диапазонов четвертого и пятого диапазонов отображаются непосредственно пользователю (например, 404 или 500).

В табл. 2.3–2.7 сведены коды ответа сервера, сгруппированные по перечисленным в табл. 2.2 диапазонам, и их краткие описания.

Таблица 2.3. ***Информационные коды ответа сервера***

| Код | Расшифровка кода | Описание |
|---|---|---|
| 100 | Continue | Код информирует клиента, что начальная часть запроса принята и пока не была отвергнута сервером, а клиент может продолжить передачу запроса |
| 101 | Switching Protocols | Сервер выполняет запрос клиента на переключение протоколов в соответствии с указанием, переданным в поле `Upgrade` заголовка HTTP-запроса |

Таблица 2.4. ***Коды ответа сервера при успешной обработке запроса***

| Код | Расшифровка кода | Описание |
|---|---|---|
| 200 | OK | Запрос успешно обработан, данные отправляются с учетом метода, которым был произведен запрос |
| 201 | Created | Код используется при создании нового URI. В заголовке ответа сервером выдается поле `Location`, содержащее адрес размещения данных |
| 202 | Accepted | Запрос принят к обработке, но обработка не завершена. Запрос может быть отклонен, когда процесс его обработки будет запущен |
| 203 | Non-Authoritative Information | Метаданные заголовка ответа не являются доступными с исходного сервера, но получены из локальной копии или с сервера третьей стороны |

Окончание табл. 2.4

| Код | Расшифровка кода | Описание |
|---|---|---|
| 204 | No Content | Сервер обработал запрос, но необходимость отправлять тело ответа отсутствует. При получении этого кода браузер не должен обновлять документ. Код используется в первую очередь для обработки действий, выполненных над активными изображениями |
| 205 | Reset Content | Клиент должен очистить форму (документ), вызвавшую отправку запроса. Данный код используется для CGI-скриптов, требующих ввод данных |
| 206 | Partial Content | Сервер отправляет часть документа, запрошенного клиентом, в соответствии с указанным клиентом диапазоном байт запрошенных данных. Используется для возобновления скачивания документов |

Т а б л и ц а 2.5. ***Коды ответа сервера, связанные с перенаправлением запроса***

| Код | Расшифровка кода | Описание |
|---|---|---|
| 300 | Multiple Choices | Запрошенный ресурс соответствует нескольким документам. Ответ сервера (за исключением запросов по методу `HEAD`) должен включать перечень ресурсов, из которых может быть выбран наиболее устраивающий. В то же время спецификация HTTP/1.1 не определяет стандартов для автоматического определения ресурса |

Окончание табл. 2.5

| Код | Расшифровка кода | Описание |
|---|---|---|
| 301 | Moved Permanently | Запрошенный документ соответствует новому постоянному URI, который должен быть указать в поле `Location` заголовка ответа сервера. При последующих запросах документа следует использовать указанный URI |
| 302 | Found | Запрошенный документ временно находится по указанному в поле Location заголовка ответа сервера адресу |
| 303 | See Other | Запрошенный документ может быть найден по URI, указанному в поле Location заголовка ответа сервера адресу и должен быть получен с использованием метода GET |
| 304 | Not Modified | Документ не изменялся со времени, указанного в поле `If-Modified-Since` заголовка запроса клиента. Ответ сервера с таким кодом не содержит тела документа, клиент должен использовать хранящуюся у него локальную копию |
| 305 | Use Proxy | Доступ к запрошенному документу должен быть произведен через прокси-сервер, адрес которого указывается в поле `Location` заголовка ответа сервера |
| 306 | (Unused) | Код использовался в предыдущих версиях протокола, но не используется с момента выхода спецификации HTTP/1.1. Код является зарезервированным |
| 307 | Temporary Redirect | Запрошенный документ временно находится по указанному в поле Location заголовка ответа сервера адресу |

Т а б л и ц а 2.6. ***Коды ответа сервера, связанные с ошибкой на стороне клиента***

| Код | Расшифровка кода | Описание |
|---|---|---|
| 400 | Bad Request | Запрос не понят сервером из-за ошибок в синтаксисе |
| 401 | Unauthorized | Для обработки запроса требуется аутентификация пользователя на сервере [17] |
| 402 | Payment Required | Зарезервирован для будущего использования |
| 403 | Forbidden | Доступ запрещен. Наиболее частые причины ответа с таким кодом – ошибки при авторизации на сервере и запрос листинга каталога, в то время как он запрещен конфигурацией сервера |
| 404 | Not Found | Документ по запрошенному URI не найден. Причины отсутствия документа не указываются. В случае, если серверу известно, что документ по этому адресу не будет находиться и в дальнейшем, серверу следует использовать код 410 (Gone) |
| 405 | Method Not Allowed | Использование метода, примененного для запроса документа, не допускается. В поле `Allow` заголовка ответа сервера указывается перечень допустимых методов |
| 406 | Not Acceptable | Документ существует по запрошенному URI, но его формат не соответствует списку поддерживаемых клиентом (на основании заголовка запроса, отправленного клиентом) |

Продолжение табл. 2.6

| Код | Расшифровка кода | Описание |
|---|---|---|
| 407 | Proxy Authentication Required | Этот код похож на код 401 (Unauthorized), но показывает, что клиент должен произвести аутентификацию на прокси-сервере |
| 408 | Request Timeout | Клиентом не был передан запрос в течение установленного сервером времени. Запрос может быть повторен клиентом позже |
| 409 | Conflict | Запрос не может быть обработан из-за конфликта, связанного с текущим состоянием документа. Применение кода возможно только в тех случаях, когда клиент может разрешить конфликт и повторить запрос. Тело ответа должно содержать достаточно информации для того, чтобы пользователь мог распознать причины конфликта. Наиболее вероятно появление подобных конфликтов при использовании метода `PUT` |
| 410 | Gone | Запрошенный ресурс не существует и навсегда удален с сервера |
| 411 | Length Required | Сервер отказывает в обработке запроса без указания клиентом поля `Content-Length` в заголовке запроса |
| 412 | Precondition Failed | Условие, заданное в одном или нескольких полях заголовка запроса, дает значение «ложь» |
| 413 | Request Entity Too Large | Тело запроса слишком велико и поэтому запрос не обрабатывается сервером |
| 414 | Request-URI Too Long | Сервер отказывается обрабатывать запрос, так как URI запрошенного документа имеет чрезмерную длину |

Окончание табл. 2.6

| Код | Расшифровка кода | Описание |
|---|---|---|
| 415 | Unsupported Media Type | Сервер отказывается обрабатывать запрос, так как тело запроса имеет не поддерживаемый сервером тип данных |
| 416 | Requested Range Not Satisfiable | Сервер не обрабатывает запрос, так как запрошенный диапазон байта является неприемлемым |
| 417 | Expectation Failed | Значение, переданное в поле `Expect` запроса, не может быть удовлетворено сервером |

Т а б л и ц а 2.7. ***Коды ответа, связанные с ошибкой на стороне сервера***

| Код | Расшифровка кода | Описание |
|---|---|---|
| 500 | Internal Server Error | При обработке запроса на сервере произошла ошибка. Наиболее часто встречающиеся причины – ошибка в CGI-скриптах или конфигурации самого сервера |
| 501 | Not Implemented | Сервером не поддерживаются функции, необходимые для выполнения запроса |
| 502 | Bad Gateway | Сервер, работающий как шлюз или прокси, получил недопустимый ответ от другого сервера во время попытки обработки запроса |
| 503 | Service Unavailable | Сервер не может обработать запрос из-за временной перегрузки или технических работ на сервере. Если серверу известно, когда будет возобновлен доступ, сервер может выдать в заголовке поле Retry-After, в противном случае клиент должен обрабатывать ответ так же, как при получении кода ответа 500 |

Окончание табл. 2.7

| Код | Расшифровка кода | Описание |
|---|---|---|
| 504 | Gateway Timeout | Сервер, работающий как шлюз или прокси, не получил необходимого для обработки запроса ответа от другого сервера |
| 505 | HTTP Version Not Supported | Сервер не поддерживает версию протокола HTTP, указанную в запросе. В теле ответа сервера должна присутствовать информация о причинах и доступных версиях |

## 2.3 Язык гипертекстовой разметки HTML

Согласно спецификациям языка HTML (Hypertext Markup Language, язык гипертекстовой разметки) [13], в упрощенном виде структура HTML-документа выглядит следующим образом:

```
<HTML>
<HEAD>
<TITLE>Название документа</TITLE>
Теги заголовка документа
</HEAD>
<BODY>
Тело документа
</BODY>
</HTML>
```

Основной текст HTML-документа расположен внутри тегов <BODY>…</BODY>, определяющих тело документа. Под тегом подразумевается управляющая конструкция, заключенная в угольные скобки. Для создания ссылок на другие расположенные в сети WWW документы используются конструкции вида:

```
<a href=”адрес_документа” class=link target=_blank>текст ссылки</a>
```

Стандартами языка HTML для описания тегов допускается использование символов как верхнего, так и нижнего регистров.

В зависимости от местонахождения документа, в котором встретилась ссылка, требуемый документ может (в соответствии с записью ссылки) быть расположен по различным адресам.

Ссылка, записанная в виде URL, например,

`http://www.ssau.ru/struct/faculties/`,

является абсолютным адресом в сети WWW и всегда прямо показывает на требуемый документ. При формировании HTTP-запроса может потребоваться отделение абсолютного пути на www-сайте от адреса сервера. Абсолютный путь на www-сайте в приведенном примере будет

`/struct/faculties/`.

*Замечание.* Так как физическое расположение файлов на web-сервере (понимая под web-сервером хост с запущенной на нем программой-сервером) практически всегда зависит от типов используемых операционной и файловой систем, то абсолютный путь к документу на www-сервере (например,

`/home/somesite/htdocs/contacts/more/index.html`)

отличается от абсолютного пути на www-сайте (в приведенном примере может быть `/contacts/more/index.html`).

Ссылка, записанная в виде абсолютного пути на www-сайте, начинается со слеша и показывает путь к документу, начиная от корня сайта. То есть ссылка `/contacts/more/index.html` указывает на документ index.html, находящийся в каталоге more, который в свою очередь находится в каталоге `contacts`.

Другие виды записи адреса документа показывают путь относительно документа, в котором встретилась данная ссылка, и при дальнейших запросах для получения корректного адреса документа должны быть дополнены соответствующими данными. Например, ссылка `../info/index.html` указывает, что требуемый документ находится в подкаталоге `info`, который, в свою очередь, расположен в родительском каталоге для документа, в котором найдена эта ссылка.

Аналогично тегом `IMG` реализуется вставка в документ графической информации:

```
<img src=”/images/logo.gif” width=400 height=100>
```

# 3 СЕРВИС ПЕРЕДАЧИ ФАЙЛОВ FTP

Протокол передачи файлов FTP (File Transfer Protocol), обеспечивающий обмен файлами между удаленными пользователями, является одним из старейших протоколов, работающих в сети Интернет. Впервые он был описан в спецификации [19] в 1971 г., а окончательная на данный момент времени редакция датирована 1985 годом. Всего было выпущено более 40 спецификаций, относящихся к этому протоколу [18-60].

FTP не только позволяет производить передачу файлов, но и обеспечивает разграничение прав доступа пользователей к ресурсам. Протоколом предусматривается также анонимный доступ.

На рис. 3.1 показана модель использования FTP.

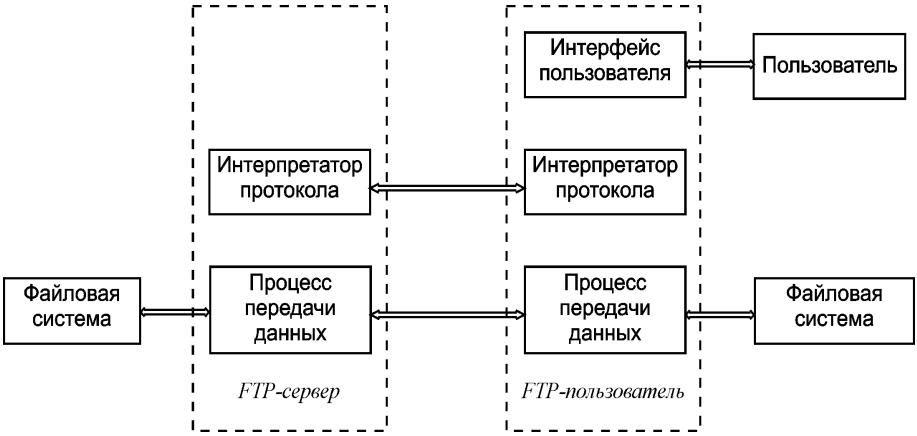


Рис. 3.1. Модель использования FTP

Сначала интерпретатор протокола создает управляющее соединение, использующее протокол Telnet. FTP-команды, инициированные пользователем, передаются на сервер по управляющему соединению.

Дальнейшие шаги пользователя включают в себя идентификацию (указание имени пользователя и пароля последовательно командами user и pass), выбор каталога, выбор режима обмена (командой TYPE), выполнение команд управления данными (команды get, mget, dir, mdel, put и mput) и завершение сеанса работы (команды quit или close).

Ниже приведен пример установления соединения (анонимным пользователем), запроса поддерживаемых сервером команд и закрытия соединения.

```
220 ProFTPD 1.3.0a Server [89.186.244.16]
USER anonymous
331 Anonymous login ok, send your complete email address as
your password.
PASS anonymous@my.mail
230-Welcome to Samara State Aerospace Univercity FTP Server
230 Anonymous access granted, restrictions apply.
HELP
214-The following commands are recognized (* =>'s unimple-
mented):
 CWD     XCWD    CDUP    XCUP    SMNT*   QUIT    PORT    PASV
 EPRT    EPSV    ALLO*   RNFR    RNTO    DELE    MDTM    RMD
 XRMD    MKD     XMKD    PWD     XPWD    SIZE    SYST    HELP
 NOOP    FEAT    OPTS    AUTH    CCC*    CONF*   ENC*    MIC*
 PBSZ    PROT    TYPE    STRU    MODE    RETR    STOR    STOU
 APPE    REST    ABOR    USER    PASS    ACCT*   REIN*   LIST
214 Direct comments to root@web.ssau.ru
QUIT
221 Goodbye.
```

Особенность FTP заключается в том, что для передачи данных (файлы или листинг каталога) используется дополнительное соединение. Для выбора номера порта, по которому будет производиться это соединение, предназначена команда PORT.

Перечень основных FTP-команд и их краткие описания приведены в табл. 3.1. Команда отделяется от аргумента пробелом. Все FTP-команды завершаются последовательностью <CRLF>. В квадратных скобках указан необязательный аргумент.

Т а б л и ц а 3.1. ***Основные FTP-команды***

| **Команда** | **Описание команды** |
|---|---|
| ABOR | Прерывание исполнения предыдущей FTP-команды и связанного с ней обмена |
| APPE <имя> | Добавить передаваемые данные к файлу |
| CDUP | Переход в родительский каталог |
| CWD <имя> | Изменить рабочий каталог |
| DELE <имя> | Удалить файл |
| HELP [<строка>] | Выдать перечень поддерживаемых команд или описание команды |
| LIST [<имя>] | Передать список файлов и каталогов. Передача осуществляется через дополнительное соединение |
| MKD <имя> | Создать каталог |
| NLST [<имя>] | Передать список файлов и каталогов в кратком виде. Передача осуществляется через дополнительное соединение |
| NOOP | Пустая операция |
| PASS <пароль> | Пароль пользователя |
| PASV | Перейти в пассивный режим передачи данных. Сервер вернет адрес и порт, к которому необходимо подключиться, чтобы получить данные |
| PORT <h1,h2,h3,h4,p1,p2> | Перейти в активный режим передачи данных. Для передачи данных сервер сам подключается к клиенту. Здесь h1, h2, h3, h4 – байты IP-адреса клиента, p1, p2 – старший и младший байты номера порта |

Окончание табл. 3.1

| Команда | Описание команды |
|---|---|
| PWD | Возвращает имя текущего каталога |
| QUIT | Завершение FTP-сессии |
| REIN | Завершение FTP-сессии и открытие новой |
| REST <маркер> | Возобновление обмена, начиная с места, указанного маркером |
| RETR <имя> | Запрос файла. Перед RETR необходима команда PASV или PORT |
| RMD <имя> | Удалить каталог |
| RNFR <имя> | Переименования файла (Rename From) |
| RNTO <имя> | Новое имя файла при переименовании (Rename To) |
| STOR <имя> | Передача файла на сервер. Перед STOR необходима команда PASV или PORT |
| SYST | Тип системы сервера |
| TYPE <тип> | Тип передаваемого файла. TYPE I для бинарного файла, TYPE A для текстового |
| USER <имя> | Имя пользователя |

Спецификация протокола определяет также минимальный набор команд и параметров, который должны поддерживать все FTP-сервера:

- TYPE – ASCII Non-print
- MODE – Stream
- STRUCTURE – File, Record
- COMMANDS – USER, QUIT, PORT, TYPE, MODE, STRU.

  Для стандартных значений –

  команды RETR, STOR, NOOP.

Стандартные значения параметров передачи данных:

- TYPE – ASCII Non-print
- MODE – Stream
- STRU – File

Список кодов возможных ответов FTP-сервера с их расшифровками приведен в табл. 3.2.

Та б л и ц а 3.2. ***Коды ответов FTP-сервера***

| **Код** | **Комментарий** |
|---|---|
| 110 | Комментарий |
| 120 | Сервис будет доступен через nnn минут |
| 125 | Соединение для передачи уже открыто, начинается обмен данными |
| 150 | Статус файла правилен, готовится открытие соединения |
| 200 | Команда корректна |
| 202 | Команда не поддерживается |
| 211 | Системный статус или отклик на справочный запрос |
| 212 | Состояние каталога |
| 213 | Состояние файла |
| 214 | Справочное сообщение |
| 220 | Сервис готов к подключению новых пользователей |
| 221 | Благополучное завершение по команде quit |
| 225 | Канал сформирован, но информационный обмен отсутствует |
| 226 | Закрытие канала, обмен завершен успешно |
| 227 | Переход в пассивный режим (h1,h2,h3,h4,p1,p2). |
| 230 | Пользователь идентифицирован, продолжайте |
| 250 | Запрошенное действие над файлом успешно завершено |
| 257 | Путь «PATHNAME» создан |
| 331 | Имя пользователя корректно, нужен пароль |
| 332 | Необходима аутентификация |
| 350 | Запрошенное действие над файлом требует больше информации |
| 421 | Сервис недоступен, управляющее соединение закрывается |
| 425 | Невозможно открытие соединения для передачи данных |
| 426 | Канал закрыт, обмен прерван |
| 450 | Запрошенное действие с файлом не выполнено, файл недоступен (например, занят) |

Окончание табл. 3.2

| Код | Комментарий |
|---|---|
| 451 | Запрошенное действие прервано: локальная ошибка |
| 452 | Запрошенное действие не выполнено. Недостаточно места |
| 500 | Синтаксическая ошибка, команда не распознана (в том числе когда команда слишком длинна) |
| 501 | Синтаксическая ошибка в параметрах или аргументах |
| 502 | Команда не используется |
| 503 | Неверная последовательность команд |
| 504 | Команда не применима для такого параметра |
| 530 | Вход в систему не выполнен |
| 532 | Необходима аутентификация для запоминания файла |
| 550 | Запрошенное действие не выполнено, файл недоступен или не найден |
| 551 | Запрошенное действие прервано. Неизвестный тип страницы |
| 552 | Запрошенное действие прервано, недостаточно памяти |
| 553 | Запрошенное действие не выполнено. Недопустимое имя файла |

# 4 ЭЛЕКТРОННАЯ ПОЧТА

## 4.1 Электронная почта в сети Internet. Система адресов

Электронная почта является одной из важнейших служб сети Интернет. Принятая система адресов электронной почты основана на доменном адресе почтового сервера. Как и любой адрес, адрес электронной почты состоит из двух частей: «кому» (имя пользователя) и «куда» (адрес сервера электронной почты). В стандартной записи адреса электронной почты эти поля принято разделять символом @, например, ssau@ssau.ru.

Для полноценной работы с электронной почтой пользователю необходимо знать не только адрес своей электронной почты, но и адреса серверов исходящей электронной почты (обычно отправляемой почтовыми клиентами с использование протокола SMTP – Simple Mail Transfer Protocol) и сервера входящей почты (как правило, работающей по протоколу POP3 – Post Office Protocol версии 3).

## 4.2 Протокол отправки электронной почты SMTP

Целью протокола SMTP является обеспечение надежной и эффективной доставки электронной почты.

Протокол SMTP не зависит от конкретных подсистем передачи и требует для работы лишь канал с гарантированной и упорядоченной доставкой потока данных.

Почтовое сообщение состоит только из трёх частей: конверта, заголовка и тела сообщения.

Конверт используется только программами доставки и не виден пользователю.

Заголовок всегда находится перед телом сообщения и отделен от него пустой строкой. Заголовок сообщения состоит из полей, которые,

в свою очередь, состоят из разделенных двоеточием имени и значения поля. В заголовке всегда есть следующие поля, которые представлены в табл. 4.1.

Т а б л и ц а 4.1. ***Поля заголовка***

| **Имя поля** | **Содержание поля** |
|---|---|
| FROM | Адрес отправителя сообщения |
| TO | Адрес получателя сообщения |
| CC | Адреса получателей копий сообщения |
| DATE | Время и дата отправки сообщения |
| MESSAGE ID | Идентификатор, используемый программами электронной почты |
| SUBJECT | Тема (краткое описание) сообщения |

В заголовке могут присутствовать и другие, необязательные поля.

Важным свойством протокола SMTP является возможность транспортировки почты через множество сетей, которые обычно называют «почтовыми трансляторами SMTP». Сети состоят из обоюдно доступных по протоколу TCP хостов публичной сети Internet, обоюдно доступных по TCP хостов приватных сетей TCP/IP, находящихся за межсетевыми экранами, или хостов некоторых иных локальных и распределенных сред, использующих на транспортном уровне протоколы, отличные от TCP. Используя протокол SMTP, процесс может передавать почту другому процессу в той же сети или в некоторых других сетях через трансляторы или шлюзы, доступные из обеих сетей. Таким путем почтовые сообщения можно передавать через множество промежуточных трансляторов (называемых «relay») или шлюзов на пути между отправителем и конечным адресатом. Для определения следующего промежуточного получателя на пути к адресату используется механизм Mail exchanger (MX) системы доменных имен.

## 4.3 Протокол получения электронной почты POP3

Протокол POP3 предписывает следующий порядок обмена информацией между клиентом и почтовым сервером:

В начале работы пользователя производится его авторизация путем проверки соответствия имени и пароля пользователя. Запросы клиента и ответы сервера с результатами обработки запросов передаются в текстовом виде.

После успешной авторизации клиент начинает работу с содержимым почтового ящика. Протокол POP3 позволяет получить сводную информацию о сообщениях – количество сообщений, их размер и список, а также выбрать определенное сообщение. После прочтения сообщений пользователь может пометить подлежащие удалению.

При получении команды завершения сеанса получения сервер удаляет помеченные сообщения и завершает сеанс работы с клиентом.

Команды протокола POP3 состоят из ключевых слов, за которыми может следовать один или более аргументов. Все команды заканчиваются парой <CRLF>.

Ответы в POP3 состоят из индикатора состояния и ключевого слова, за которым может следовать дополнительная информация. Ответ заканчивается парой <CRLF>. Протоколом POP3 предусмотрено два индикатора состояния: «+OK» – положительный и «-ERR» – отрицательный.

Если ответ на команду состоит из нескольких строк, то строки ответа разделяются последовательностью <CRLF>, а окончание ответа обозначается точкой (ASCII-символ с кодом 46) и последовательностью <CRLF>.

В минимальной реализации POP3-сервер должен поддерживать команды клиента, приведенные с их расшифровками в табл. 4.2.

Т а б л и ц а 4.2. ***Команды протокола POP3***

| | |
|---|---|
| USER | Задание имени пользователя |
| PASS | Задание пароля пользователя |
| QUIT | Завершение TCP-соединения |
| STAT | Запрос кол-ва сообщений в почтовом ящике и его размера |
| LIST [идентификатор] | Запрос идентификаторов почтовых сообщений и их размеров |
| RETR | Запрос сообщения с указанным номером |
| DELE | Отметить сообщение для удаления |
| NOOP | Фиктивное действие |
| LAST | Максимальный номер сообщения из тех, к которым обращался клиент |
| RSET | Отмена удаления сообщения, отмеченного DELE |

Пример начала сессии протокола POP3:

```
Client:      USER ssau
Server:      +OK User accepted
Client:      PASS ssaupassword
Server:      +OK Pass accepted
```

С целью повышения безопасности почтовых ящиков протокол POP3 поддерживает передачу серверу пароля в зашифрованном виде. В этом случае для авторизации используется команда APOP, имеющая два аргумента: имя почтового пользователя и дайджест – вычисленная с использованием алгоритма MD5 хэш-сумма пароля и полученной при установлении соединения от сервера временной метки.

# 5 ЗАДАНИЯ

При выполнении перечисленных ниже работ допускается использование любых сред разработки и языков программирования. Для первой и второй работ целесообразна установка на локальном компьютере соответственно web- и ftp-сервера, что позволит существенно сократить время, потребное для выполнения программы во время её тестирования. При этом необходимо помнить, что в операционных системах семейства MS Windows символы верхнего и нижнего регистра в именах файлов считаются идентичными, что в некоторых случаях является отклонением от спецификаций и может стать причиной получения некорректного результата при тестировании программ.

Первая и вторая работы связаны с реализацией web- и ftp-серверов, третья и четвертая – клиента электронной почты (отправка и получение почты соответственно), пятая и шестая работы представляют собой реализацию web- и ftp-серверов соответственно.

Единственным ограничением, накладываемым на способы и методы выполнения заданий, является запрет на использование стандартных, а также реализованных сторонними разработчиками компонент и функций, исполняющих протоколы прикладного уровня. Преобразование адреса хоста, записанного в символьном виде, в IP-адрес обычно производится автоматически при открытии сокета и поэтому может считаться не входящим в упомянутое ограничение.

## 5.1 Исследование web-сайта

Цель работы: освоение принципов работы протокола прикладного уровня HTTP, приобретение навыков формирования и обработки запросов информации по протоколу HTTP.

## Задание

Создать сетевое клиентское приложение, реализующее обращения к www-серверу по протоколу HTTP и производящее обработку полученных данных. Приложение должно выполнять функции, изложенные в задании, и обладать дружественным к пользователю интерфейсом.

Для формирования запросов и получения ответов сервера запрещается использование стандартных либо реализованных сторонними разработчиками компонент и функций, исполняющих протоколы прикладного уровня.

Адрес www-сервера и номер порта (по умолчанию 80) вводятся пользователем. Адрес может быть записан как в виде IP-адреса, так и в символьной форме (например, 89.186.244.16 или www.ssau.ru).

Приложение должно включать контроль ошибок при вводе и обработке запросов.

В табл. 5.1 приведены варианты заданий.

Ниже приведен код программы на PHP, запрашивающей информацию о заглавной странице сервера www.ssau.ru и выводящей её пользователю.

```
<?php
$content = getpage("www.ssau.ru","/");
$contents = explode("\n",$content);
foreach ($contents as $cc) {
   $cr = ereg_replace("<","&lt;",$cc);
   echo $cr."<br>\n";
}
function getpage($host,$path) {

        $nn="\r\n"; $data="";
        $request = "GET $path HTTP/1.0".$nn.
```

```
        "Referer: $host".$nn.
        "Content-Type: application/x-www-form-
urlencoded".$nn.
        "Host: $host".$nn.$nn;
        flush();
        $fp = fsockopen("$host", 80, &$errno,
&$errstr, 30);
        if(!$fp) {echo "$errstr ($errno)<br>\n";
exit;}
        fputs($fp,$request);
        while(!feof($fp)) {
                $data.= fgets($fp,4096);
        }
        fclose($fp);
   return $data;
}
?>
```

Т а б л и ц а 5.1. ***Варианты заданий***

| **Вариант** | **Задание** |
|---|---|
| 1 | Составить и вывести дерево каталогов www-сервера. Вывести информацию о сервере (кодировку страниц, версию программного обеспечения) |
| 2 | Составить и вывести список всех рисунков, используемых страницами www-сервера. Вывод разделить на две части: рисунки, расположенные на сервере, и рисунки, расположенные на других серверах |
| 3 | Определить и вывести суммарный размер всех страниц www-сервера и их количество. Вывести адреса страниц, имеющих наибольший и наименьший размеры |

Окончание табл. 5.1

| Вариант | Задание |
|---|---|
| 4 | Определить и вывести количество и суммарный размер всех файлов сервера, не являющихся HTML-документами и рисунками. Вывести их список |
| 5 | Составить и вывести список файлов (в том числе HTML-страниц), недоступных по ссылкам со страниц сервера («битые» ссылки) и расположенных на нем же |
| 6 | Составить и вывести список серверов, на которые ссылаются страницы исходного сервера |

## 5.2 Исследование структуры FTP-сервера

Цель работы: приобретение навыков работы с протоколом передачи файлов FTP и написания серверных приложений.

### Задание

Разработать клиентское приложение, позволяющее пользователю получать следующую информацию о FTP-сервере:

- суммарный размер файлов, размещенных на сервере;
- суммарный размер размещенных на сервере файлов, сгруппированных по их типам (тип файлов определяется по его расширению);
- структуру каталогов FTP-сервера.

Адрес FTP-сервера, имя пользователя и пароль вводятся пользователем.

### Литература

Для выполнения данного задания необходимо изучить спецификацию протокола FTP, краткое описание которого приведено в главе 3.

RFC 959 – File Transfer Protocol – http://www.faqs.org/rfcs/rfc959.html

## 5.3 Отправка электронной почты

Цель работы: приобретение навыков работы с простым протоколом отправки электронной почты SMTP.

### Задание

Разработать программу (почтовый клиент), позволяющую пользователю отправлять электронную почту по протоколу SMTP, используя существующие в сети Internet почтовые сервера. Программа должна поддерживать возможность аутентификации пользователя на сервере при отправке электронной почты в случае, если она требуется на выбранном пользователем сервере исходящей электронной почты. Программа должна поддерживать возможность отправки одного и того же письма нескольким пользователям и приложение к письму как минимум одного файла.

Адреса получателей, тема и текст письма, месторасположение прикладываемого к письму файла и адрес почтового сервера указываются пользователем.

Приложение должно включать контроль ошибок при вводе и обработке запросов.

### Литература

Для выполнения лабораторной работы требуется изучение спецификаций протокола отправки электронной почты SMTP, аутентификации при использовании протокола SMTP и форматов MIME.

## 5.4 Получение электронной почты

Цель работы: приобретение навыков работы с протоколом электронной почты POP3.

### Задание

Расширить возможности программы (почтового клиента), полученной в лабораторной работе № 3, доставкой пользователю электронной почты по протоколу POP3, используя существующие в сети Internet почтовые сервера. Программа должна поддерживать возможность получения приложенных к письму файлов и сохранение их на персональном компьютере пользователя.

Адрес почтового сервера, имя пользователя, пароль и номер порта указываются пользователем.

Приложение должно включать контроль ошибок при вводе и обработке запросов.

### Литература

RFC 1939 – Post Office Protocol – Version 3
RFC 1521. MIME – Multipurpose Internet Mail Extensions

## 5.5 Реализация WWW-сервера

Цель работы: приобретение навыков работы с протоколом передачи гипертекста HTTP.

### Задание

Разработать программу (web-сервер), обрабатывающую запросы пользователей по протоколу HTTP и находящуюся в памяти компьютера резидентно. Разрабатываемый web-сервер должен поддерживать следующие возможности:

- обработка запросов по методам GET и HEAD;
- ограничение доступных пользователю файлов указываемым в конфигурации web-сервера каталогом;
- отправка пользователю, помимо самих данных, типа передаваемых данных в формате MIME в зависимости от расширения запрошенного файла (как минимум text/html, text/plain, image/gif, image/jpeg и тип данных по умолчанию);
- поддержка keep-alive соединений (при условии запроса соединений такого рода пользователем, с ограничениями максимального числа запросов по соединениям такого вида за один сеанс и максимального времени ожидания последующего запроса);
- возврат длины передаваемых данных в заголовке HTTP-ответа сервера.

При выполнении задания необходимо использовать пассивное открытие сокета.

## 5.6 Реализация FTP-сервера

Цель работы: приобретение навыков работы с протоколом передачи файлов FTP и написания серверных приложений.

## Задание

Разработать серверное приложение (FTP-сервер), позволяющее пользователю производить обмен информацией с сервером по протоколу FTP. FTP-сервер должен поддерживать как минимум следующие возможности:

- разграничение прав доступа пользователей по их домашним каталогам;
- использование пользователя по умолчанию (анонимного пользователя);
- выполнение команд смены текущего каталога;
- выполнение команды листинга текущего каталога;
- передача файлов в обоих направлениях (скачивание и загрузка на сервер);
- ограничение максимальной скорости передачи данных клиенту.

Для настроек FTP-сервера должен использоваться отдельный конфигурационный файл произвольного формата.

## Литература

Для выполнения данного задания необходимо изучить спецификацию протокола FTP, краткое описание которого приведено в главе 3 настоящего пособия.

RFC 959 – File Transfer Protocol – http://www.faqs.org/rfcs/rfc959.html

## СПИСОК ЛИТЕРАТУРЫ

### Учебники и учебные пособия


1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы. – СПб.: Питер, 2010. – 944 с.
2. Камышников В.В. Основы сетевой архитектуры Internet: Учеб. пособие. – Самара: Изд-во «Самарский университет», 2001. – 107 с.
3. Нанс Б. Компьютерные сети: Пер. с англ. — М.: БИНОМ, 1996. – 400 с.


### Спецификации


4. Berners-Lee T., Masinter L., McCahill M. Uniform Resource Locators (URL), RFC 1738, 1994.
5. Standards Track RFC 2396 URI Generic Syntax / T. Berners-Lee [et al.], 1998.
6. Neuman B., Augart S. The Prospero Protocol, USC/Information Sciences Institute, 1993.
7. US-ASCII. Coded Character Set – 7-Bit American Standard Code for Information Interchange. Standard ANSI X3.4-1986, ANSI, 1986.
8. Postel J. Simple Mail Transfer Protocol, STD 10, RFC 821, 1982.
9. Myers J., Rose M. Post Office Protocol – Version 3 RFC 1939, 1996.

10. Borenstein N., Freed N. MIME – Multipurpose Internet Mail Extensions, RFC 1521, 1993.
11. Freed N., Borenstein N. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC 2045, 1996.
12. Moore K. MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text, RFC 2047, 1996.
13. Berners-Lee T., Connolly D. Hypertext Markup Language – 2.0, RFC 1866, 1995.
14. RFC 2109 – Протокол HTTP/0.9.
15. Hypertext Transfer Protocol -- HTTP/1.1, RFC 2068 / R. Fielding, J. Gettys, J. Mogul [et al.], 1997.
16. Hypertext Transfer Protocol – HTTP/1, RFC 2616 / R. Fielding, J. Gettys, J. Mogul [et al.], 1999.
17. HTTP Authentication: Basic and Digest Access Authentication, RFC 2617 / J. Franks, P. Hallam-Baker, J. Hostetler [et al.], 1999.
18. Postel J., Reynolds J. File Transfer Protocol, STD 9, RFC 959, 1985.
19. Bhushan Abhay. A File Transfer Protocol, RFC 114 (NIC 5823), MIT-Project MAC, 1971.
20. Harslem Eric, Heafner John. Comments on RFC 114 (A File Transfer Protocol), RFC 141 (NIC 6726), RAND, 1971.
21. The File Transfer Protocol, RFC 172 (NIC 6794), MIT-Project MAC / Bhushan Abhay [et al.], 1971.

22. Braden Bob. Comments on DTP and FTP Proposals, RFC 238 (NIC 7663), UCLA/CCN, 1971.
23. The File Transfer Protocol, RFC 265 (NIC 7813), MIT-Project MAC / Bhushan Abhay [et al.], 1971.
24. McKenzie Alex. A Suggested Addition to File Transfer Protocol, RFC 281 (NIC 8163), BBN, 1971.
25. Bhushan Abhay. The Use of "Set Data Type" Transaction in File Transfer Protocol, RFC 294 (NIC 8304), MIT-Project MAC, 1972.
26. Bhushan Abhay. The File Transfer Protocol, RFC 354 (NIC 10596), MIT-Project MAC, 1972.
27. Bhushan Abhay. Comments on the File Transfer Protocol (RFC 354), RFC 385 (NIC 11357), MIT-Project MAC, 1972.
28. Hicks Greg. User FTP Documentation, RFC 412 (NIC 12404), Utah, 1972.
29. Bhushan Abhay. File Transfer Protocol (FTP) Status and Further Comments, RFC 414 (NIC 12406), MIT-Project MAC, 1972.
30. Braden Bob. Comments on File Transfer Protocol, RFC 430 (NIC 13299), UCLA/CCN, 1973.
31. Thomas Bob, Clements Bob. FTP Server-Server Interaction, RFC 438 (NIC 13770), BBN, 1973.
32. Braden Bob. Print Files in FTP, RFC 448 (NIC 13299), UCLA/CCN, 1973.
33. McKenzie Alex. File Transfer Protocol, RFC 454 (NIC 14333), BBN, 1973.

34. Bressler Bob, Thomas Bob. Mail Retrieval via FTP, RFC 458 (NIC 14378), BBN-NET and BBN-TENEX, 1973.
35. Neigus Nancy. File Transfer Protocol, RFC 542 (NIC 17759), BBN, 1973.
36. Krilanovich Mark, Gregg George. Comments on the File Transfer Protocol, RFC 607 (NIC 21255), UCSB, 1974.
37. Pogran Ken, Neigus Nancy. Response to RFC 607 – Comments on the File Transfer Protocol, RFC 614 (NIC 21530), BBN, 1974.
38. Comments on the File Transfer Protocol, RFC 624 (NIC 22054), UCSB, Ames Research Center, SRI-ARC / Mark Krilanovich, George Gregg, Wayne Hathaway [et al.], 1974.
39. Bhushan Abhay. FTP Comments and Response to RFC 430, RFC 463 (NIC 14573), MIT-DMCG, 1973.
40. Braden Bob. FTP Data Compression, RFC 468 (NIC 14742), UCLA/CCN, 1973.
41. Bhushan Abhay. FTP and Network Mail System, RFC 475 (NIC 14919), MIT-DMCG, 1973.
42. Bressler Bob, Thomas Bob. FTP Server-Server Interaction – II, RFC 478 (NIC 14947), BBN-NET and BBN-TENEX, 1973.
43. White Jim. Use of FTP by the NIC Journal, RFC 479 (NIC 14948), SRI-ARC, 1973.
44. White Jim. Host-Dependent FTP Parameters, RFC 480 (NIC 14949), SRI-ARC, 1973.
45. Padlipsky Mike. An FTP Command-Naming Problem, RFC 506 (NIC 16157), MIT-Multics, 1973.

46. Day John. Memo to FTP Group (Proposal for File Access Protocol), RFC 520 (NIC 16819), Illinois, 1973.
47. Merryman Robert. The UCSD-CC Server-FTP Facility, RFC 532 (NIC 17451), UCSD-CC, 1973.
48. Braden Bob. TENEX FTP Problem, RFC 571 (NIC 18974), UCLA/CCN, 1973.
49. McKenzie Alex, Postel Jon. Telnet and FTP Implementation – Schedule Change, RFC 593 (NIC 20615), BBN and MITRE, 1973.
50. Sussman Julie. FTP Error Code Usage for More Reliable Mail Service, RFC 630 (NIC 30237), BBN, 1974.
51. Postel Jon. Revised FTP Reply Codes, RFC 640 (NIC 30843), UCLA/NMC, 1974.
52. Harvey Brian. Leaving Well Enough Alone, RFC 686 (NIC 32481), SU-AI, 1975.
53. Harvey Brian. One More Try on the FTP, RFC 691 (NIC 32700), SU-AI, 1975.
54. Lieb J. CWD Command of FTP, RFC 697 (NIC 32963), 1975.
55. Harrenstien Ken. FTP Extension: XSEN, RFC 737 (NIC 42217), SRI-KL, 1977.
56. Harrenstien Ken. FTP Extension: XRSQ/XRCP, RFC 743 (NIC 42758), SRI-KL, 1977.
57. Lebling P. David. Survey of FTP Mail and MLFL, RFC 751, MIT, 1978.
58. Postel Jon. File Transfer Protocol Specification, RFC 765, ISI, 1980.

59. Mankins David, Franklin Dan, Owen Buzz. Directory Oriented FTP Commands, RFC 776, BBN, 1980.
60. Padlipsky Michael. FTP Unique-Named Store Command, RFC 949, MITRE, 1985.

Учебное издание

***Еленев Дмитрий Валерьевич***

**РАБОТА С СЕРВИСАМИ СЕТИ ИНТЕРНЕТ**

*Учебное пособие*

Редактор Н. С. К у п р и я н о в а
Доверстка Т. Е. П о л о в н е в а

Подписано в печать 20.10.2010. Формат 60x84 1/16
Бумага офсетная. Печать офсетная.
Печ. л. 3,25.
Тираж 100 экз. Заказ .

Самарский государственный
аэрокосмический университет,
443086, Самара, Московское шоссе, 34.

---

Издательство Самарского государственного
аэрокосмического университета
443086, Самара, Московское шоссе, 34.