**Лабораторная работа № 22**  
**Взаимодействие прикладных программ с помощью протоколов электронной почты** **и протоколов прикладного уровня**

*Цель работы: изучение принципов работы протоколов электронной почты, разработка прикладных программ, осуществляющих взаимодействие и передачу данных друг другу на основе прикладных протоколов.*

*Для выполнения лабораторной работы требуется написать программу, которая выполняется под управлением ОС типа*Windows или UNIX и использует для взаимодействия с другими программами заданный протокол электронной почты или иной прикладной протокол сети Internet. Для разработки программы рекомендуется использовать среду Delphi версии 3.0 или старше под управлением ОС типа Windows 95/98 или Windows NT/2000.

Краткие теоретические сведения

Протокол электронной почты SMTP

Электронная почта - это средство передачи сообщений между любыми двумя ЭВМ, имеющими соединение в сети. Принцип работы электронной почты предельно прост (и в этом сродни работе обычной, "бумажной" почты) - для того, чтобы передать сообщение, достаточно только иметь текст сообщения и знать адрес получателя. Пользователь электронной почты не должен заботится о том, каким образом и с помощью каких средств должно быть передано его сообщение. Электронная почта - это принадлежность не только сети Internet, она использовалась (и используется) также в других сетях, но с появлением всемирной сети это средство получило истинно всемирный размах, сравнявшись по широте возможностей со своим "бумажным" аналогом, а в чем-то даже превосходя его.

SMTP (SimpleMailTransferProtocol - простой протокол электронной почты) принят в качестве стандартного метода передачи электронной почты в сети Internet. Этот протокол во многом подобен протоколу FTP. Название его вполне соответствует сути - с ним достаточно просто работать. В качестве транспортного протокола SMTP использует TCP, соединение устанавливается через порт с номером 25. Для обслуживания этого соединения используется специальная программа, которая именуется почтовым сервером. Для формирования сообщения и установления соединения используется почтовая программа (утилита) пользователя.

Главной целью протокола SMTP является надежная и эффективная доставка электронных почтовых сообщений. Для реализации протокола требуется только надежный канал связи. Средой дляSMTP может служить отдельная локальная сеть, система сетей или же вся всемирная сеть Internet. Если между отправителем и получателем письма имеется непосредственная связь, адрес пользователя имеет вид *имя\_пользователя*@адрес\_ЭВМ. Когда получатель находится на ЭВМ, которая не поддерживает соединение по протоколу SMTP, и передача происходит через промежуточный сервер, то адрес получателя письма может иметь иной вид, например, *имя\_пользователя%имя\_сервера*@адрес\_ЭВМ. Адреса ЭВМ в сети Internet представляют собой имя домена и преобразуются в IP-адреса согласно протоколу DNS.

Протокол SMTP базируется на следующей модели коммуникаций: в ответ на запрос пользователя почтовая программа-отправитель сообщения устанавливает двухстороннюю связь с программой-приемником (почтовым сервером). Получателем может быть оконечный или промежуточный адресат. Если необходимо, почтовый сервер может установить соединение с другим сервером и передать сообщение дальше. SMTP-команды генерируются отправителем и посылаются получателю. На каждую команду должен быть получен отклик.

Перечень команд протокола SMTP приведен в табл. 3. Это внутренние команды протокола. Если пользователь использует для работы с электронной почтой некоторую утилиту, то эти команды ему недоступны. Они представляют интерес только при программировании взаимодействия программ на основе этого протокола.

Таблица 3.

Внутренние команды протокола SMTP 

|  |  |
| --- | --- |
| Команда | Описание команды |
| DATA | Сообщение (за командой следует текст сообщения) |
| EXPN | Расширение списка получателей |
| HELO | Используется при инициализации обмена при установлении соединения |
| HELP | Запрос о помощи |
| MAIL | Адрес отправителя |
| NOOP | Никаких действий (пустая команда) |
| QUIT | Разрыв соединения |
| RCPT | Адрес получателя (таких команд может быть несколько) |
| RSET | Закончить текущую транзакцию |
| SAML | Послать письмо на терминал пользователя и в почтовый ящик |
| SEND | Послать письмо на терминал |
| SOML | Если можно, послать письмо на терминал, иначе - в почтовый ящик |
| TURN | Изменить направление передачи (отправитель и получатель меняются местами) |
| VRFY | Проверить имя пользователя |

Через систему электронной почты передаются сообщения, которые должны иметь строго определенный формат. Этот формат достаточно прост. Любое почтовое сообщение можно разделить на три части: "конверт", заголовки и собственно текст. "Конверт" используется почтовым сервером, он содержит две команды - MAIL и RCPT (то, как они передаются серверу, описано ниже). Заголовок используется почтовой программой пользователя. Он содержит девять специальных полей: Received, From, To, Date, Subject, Message-Id, X-Phone, X-Mailer, Reply-To. Назначение этих полей понятно из их названия. Каждое из этих полей содержит имя, за которым после двоеточия идет его значения. При желании, пользователь может ознакомится с содержимым всех полей заголовка письма.

Текст сообщения должен передаваться в виде 7-разрядных символов ASCII. Конец сообщения представляет собой строку, содержащую только символы точки (".") и перевода строки. Если по каким-то причинам такая строка непосредственно встречается внутри текста сообщения, то передающая сторона автоматически дублирует точку, чтобы принимающая сторона не приняла ее за конец сообщения (на принимающей стороне дублированный символ точки в начале строки заменяется на одну точку).

С 7-битной кодировкой сообщений связана проблема пересылки писем, написанных на национальных языках, в т.ч. и на русском, поскольку для русского алфавита требуется 8-битная кодировка. Большинство современных почтовых серверов поддерживают не только 7-битные, но и 8-битные кодировки, однако, для совместимости со старыми почтовыми системами может использоваться специальная схема преобразования 8-битных кодов в 7-битные (естественно, при этом увеличивается объем сообщения, поскольку один 8-битный символ заменяется на специальную последовательность 7-битных).

Более сложная проблема заключается в том, что нет единого стандарта на кодировку русского алфавита, и поэтому на сторонах отправителя и получателя могут использоваться различные кодировки. Для того чтобы сообщение, посланное отправителем в одной кодировке, было понятно адресату, который использует другую кодировку, почтовый сервер использует специальные таблицы смены кодировок. С помощью них текст сообщения, посланного в одной кодировке, преобразуется в другую кодировку. Чтобы сервер знал, какую кодировку использовать, информация об исходной кодировке включается почтовой системой отправителя в текст или в заголовок сообщения (в зависимости от используемой системы). Одно сообщения в процессе своего прохождения через почтовые сервера может претерпеть несколько перекодировок. В случае, когда у сервера нет информации об используемой кодировке или тип кодировки указан неверно (непонятен серверу), текст сообщения может исказиться.

Передача сообщения по протоколу SMTP происходит следующим образом: после установления соединения стороны обмениваются кодами аутентификации (с помощью команд HELO), затем одна из них посылает команду MAIL, в которой указан адрес отправителя и сведения о письме. Если получатель готов к приему сообщения, он посылает положительный отклик. Далее отправитель посылает одну или несколько команд RCPT, в которых идентифицирует адресатов сообщения. Если получатель может принять сообщение для указанного адресата, то он снова выдает положительный отклик на каждую команду. После этого передается команда DATA за которой следует само письмо (сколько бы не было адресатов, само письмо передается только один раз). В конце сеанса отправитель дает команду QUIT.

Конечно, маловероятно, что при приеме почтовым сервером сообщения, адресованному некоторому пользователю ЭВМ, этот пользователь установит с сервером непосредственную связь по протоколу SMTP. Обычно сервер размещает сообщение в почтовом ящике пользователя. Почтовый ящик - это некоторое промежуточное хранилище электронных сообщений, позволяющее почтовому серверу осуществить их передачу адресату не сразу, а в тот момент, когда он установит связь с сервером. Обычно сообщения хранятся в почтовом ящике не вечно - по истечении некоторого времени почтовый сервер удаляет их оттуда. Политика обслуживания почтовых ящиков регламентируется администратором системы.

Для того, чтобы получить сообщение из своего почтового ящика, почтовая программа пользователя соединяется с сервером уже не по протоколу SMTP, а по специальному почтовому протоколу получения сообщений. Такой протокол позволяет работать с почтовым ящиком: забирать сообщения, удалять сообщения, сортировать их и выполнять другие операции. Самым популярным в настоящее время протоколом такого рода является протокол POP3 (PostOfficeProtocol v.3). Он предусматривает соединение с почтовым сервером на основе транспортного протокола TCP через порт 110. Другой, более сложный, известный протокол - IMAP4.

Общая схема пересылки сообщения от отправителя к адресату с использованием прикладных протоколов электронной почты представлена на рис. 4.

Во многих версиях ОС UNIX протокол SMTP, а также другие почтовые протоколы, реализован в простейших утилитах типа sendmail или mmdf. Первая из них может служить как почтовым клиентом, так и почтовым сервером. Обычно она работает в фоновом режиме и ждет поступления сообщений от пользователя или извне. В ОС типа Windows NT ту же роль играет утилита, именуемая mdaemon (maildaemon).

Для непосредственного взаимодействия с пользователем существует простейшая программа, именуемая mail или mailx. Однако в современных ОС ею мало кто пользуется, поскольку существует большое количество прикладных программ с развитым интерфейсом, которые предоставляют пользователю возможности работы с электронной почтой. Примерами таких программ могут служить широко известные пакеты OutlookExpress (производство фирмы Microsoft) и NetscapeCommunicator (производство фирмы Netscape). Внешний интерфейс этих программных продуктов чрезвычайно прост, так что использовать их может даже неквалифицированный пользователь. Все эти программы формируют сообщения в требуемом формате и передают их утилитам ОС, работающим в фоновом режиме, для отправки. К ним же они обращаются и для приема сообщений.

Протокол и утилита обмена файлами FTP

Протокол FTP (FileTransferProtocol - Протокол передачи файлов) предназначен для передачи файлов между машинами без установления дистанционного соединения между ними (без использования Telnet). С его помощью можно передавать файлы, работать с каталогами и пользоваться электронной почтой, но он не позволяет запускать программы на удаленной ЭВМ.

Для своей работы FTP использует транспортный протокол TCP, но связь по FTP поддерживается с помощью двух соединений: по одному из них передаются команды FTP, а по другому - данные. Поэтому программная реализация FTP предполагает работу двух процессов на каждой из взаимодействующих машин. Первый из них отвечает за передачу команд и называется протокольным интерпретатором (PI - protocolinterpreter), а второй - за передачу данных и называется процессом передачи данных (DTP - datatransferprocess). Протокол TCP обеспечивает установление, разрыв соединения и надежность передачи данных. На удаленной ЭВМ (сервере) процессу передачи данных соответствует порт 20, а протокольному интерпретатору - порт 21.

Также как и протокол Telnet, протокол FTP использует для своей работы специальные внутренние команды. Эти команды могут использоваться специализированным программным обеспечением и администратором системы, но обычному пользователю они, как правило, недоступны. Пользователи для работы с протоколом FTP используют сервисные утилиты. Взаимодействие программных модулей при передаче данных по FTP показано на рис. 3.

В ОС типа UNIX для обслуживания запросов по FTP на удаленной машине запускается процесс ftpd (FTP daemon), в других ОС для этого могут использоваться другие процессы, предоставляющие аналогичный сервис. На локальной ЭВМ (клиенте) может выполняться любое пользовательское приложение, обеспечивающее соединение по FTP. В настоящее время разработано большое количество программ, обеспечивающих работу с этим протоколом, в том числе и программы с развитым графическим интерфейсом под ОС и графические оболочки типаMSWindows 3.11, MS Windows NT, UNIX (X Window). Но широко известна самая простейшая утилита, которая носит то же название, что и протокол - ftp. Она имеет тривиальный командный интерфейс, аналогичный интерфейсу утилиты telnet.

Общие принципы работы с утилитой ftp.

Также как при работе с утилитой telnet, при запуске утилиты ftp требуется указать имя или IP-адрес удаленной машины, с которой требуется установить соединение. Если этот адрес не указан, тоftp переходит в командный режим без установления соединения. Тогда для начала сеанса связи надо воспользоваться командой open. После того, как соединение установлено, необходимо пройти авторизацию доступа, для чего требуется ввести имя и пароль пользователя. В большинстве систем право доступа имеют только зарегистрированные пользователи, но иногда допускается так называемый "анонимный" вход, который будет рассмотрен далее. Как и при работе с telnet, при работе с ftp имя и права пользователя определяются удаленной, северной, а не локальной ЭВМ (в общем случае имена пользователя на сервере и клиенте могут быть различны). Для работы с каталогами, копирования и удаления файлов с удаленной машины пользователь должен располагать соответствующими правами, иначе эти команды выполняться не будут.

Следует обратить внимание, что и после установления соединения с удаленной ЭВМ, все команды пользователя ftp выполняются относительно локальной ЭВМ, в отличие от telnet. Особенно аккуратно следует выполнять команды копирования файлов, чтобы не потерять содержимое нужных файлов на машине-клиенте.

Протокол FTP позволяет передавать файлы в нескольких форматах, которые в общем случае системно зависимы. В большинстве систем (включая UNIX и Windows NT) определены два режима передачи файлов - текстовый и двоичный. Текстовый файл состоит из строк ASCII-символов, разделенных кодами перевода строки и возврата каретки, при его передаче выполняются действия по перекодировке символов. Для двоичных файлов не предусмотрено никакого определенного формата и никакой перекодировки, их передача осуществляется быстрее. Текстовый файл можно передавать как двоичный (в расчете произвести перекодировку на локальной машине или если перекодировка вовсе не требуется), но не наоборот.

Типичный сеанс работы с ftp включает следующие этапы:

* + запуск утилиты, установление соединения с удаленной ЭВМ;
  + переход в требуемый каталог на удаленной ЭВМ (сервере);
  + выбор режима передачи файлов;
  + передача данных согласно командам пользователя;
  + завершение работы, разрыв соединения.

Эти процедуры выполняются последовательно в каждом сеансе. Самые распространенные команды пользователя в утилите ftp приведены в табл. 2.

Таблица 2.

Наиболее употребительные команды утилиты ftp 

|  |  |  |
| --- | --- | --- |
| Команда | Параметры | Описание |
| ascii |  | Переключиться в режим передачи текстовых файлов |
| binary |  | Переключиться в режим передачи двоичных файлов |
| cd | имя каталога | Сменить рабочий каталог на удаленной ЭВМ |
| close |  | Закрыть соединение с удаленной ЭВМ |
| del | имя файла | Удалить файл на удаленной ЭВМ |
| dir | маска файлов | Отобразить содержимое текущего каталога сервера |
| get | имя файла | Получить файл с удаленной ЭВМ |
| hash | on/off | Включить или выключить режим вывода знака "#" для каждого переданного блока данных при передаче файлов |
| help |  | Получить подсказку |
| lcd | имя каталога | Сменить рабочий каталог на локальной ЭВМ |
| mget | маска файлов | Получить несколько файлов с удаленной ЭВМ |
| mput | маска файлов | Отправить несколько файлов на удаленную ЭВМ |
| open | адрес ЭВМ | Установить соединение с указанной ЭВМ |
| put | имя файла | Передать файл на удаленную ЭВМ |
| pwd |  | Вывести имя текущего каталога |
| quote | команда | Передать команду непосредственно FTP (для ввода команд администратора) |
| quit |  | Завершить работу с утилитой |

Более подробное описание команд утилиты ftp можно получить в предлагаемой литературе и в руководствах по работе с утилитой. Как уже было сказано выше, для различных ОС существует множество других приложений, обеспечивающих работу с протоколом FTP. Такого рода утилиты могут обладать рядом дополнительных возможностей.

Возможности работы с FTP при анонимном доступе.

Протокол FTP широко используется для обмена данными в сети Internet. Многие сервера сети поддерживают этот протокол. Поскольку количество пользователей в этой сети очень велико, то задача выделения каждому из них индивидуальных прав доступа представляется абсолютно нереальной. В то же время, большое число серверов стремится предоставить услуги по обмену данными максимально возможному количеству клиентов сети. Например, такие услуги предоставляют сервера сервисных центров фирм, занимающихся распространением свободно используемого программного обеспечения.

В таких случаях для установления соединения по протоколу FTP используется метод анонимного доступа. В этом варианте в качестве имени пользователя используется слово *anonymous*, а вместо пароля - чаще всего слово *guest* (гость). Иногда для анонимного доступа может использоваться и другие варианты пароля: например, слово *ftp* или адрес электронной почты пользователя (в последнем случае доступ будет уже не столь анонимным, хотя, конечно, никто не проверит истинность введенных пользователем данных).

При анонимном доступе пользователь имеет, как правило, очень ограниченные права. Чаще всего при анонимном доступе пользователь может только перемещаться по ограниченному перечню каталогов и получать файлы с удаленной ЭВМ (выполнять команды *get* и *mget*).

Метод анонимного доступа - это основной метод, который используют для обмена данными по протоколу FTP программы-браузеры (browser) сети Internet, например, широко распространенные программы *"InternetExplorer"*и *"NetscapeNavigator"*. Эти программы имеют возможность только получать файлы с удаленных ЭВМ, в то время, как полный доступ по протоколу FTP дает пользователю также возможность передавать файлы на удаленные ЭВМ. Имя и пароль для анонимного доступа в этих программах можно настраивать. 

Программирование взаимодействия программ на основе прикладных протоколов

Рассмотренные в этой лабораторной работе прикладные протоколы позволяют осуществлять обмен через сеть Internet различного рода данными. Это полезное свойство, безусловно, находит практическое применение.

Во многих ОС прикладные протоколы реализованы в различных утилитах (часть из которых была рассмотрена здесь и в предыдущей работе). Эти утилиты ориентированы, прежде всего, на работу с пользователями операционных систем. Однако часто возникает необходимость осуществить обмен данными непосредственно между прикладными программами, минуя вмешательство человека или при его ограниченном участии. Тогда у разработчиков программ возникает потребность в том, чтобы сами прикладные программы могли непосредственно использовать те же самые протоколы, что и описанные выше утилиты. Естественно, к этому нет никаких препятствий - прикладные протоколы сети Internet являются открытыми протоколами, их команды и принцип работы известны и регламентированы общедоступными стандартами.

Для решения такой задачи можно, конечно, прибегнуть к вызову утилит ОС из прикладной программы с помощью командной строки. Такой подход прост в реализации, но имеет два недостатка: во-первых, скорость работы прикладной программы в этом варианте будет ниже, чем при непосредственном обмене данными по протоколу; во-вторых, необходимо параллельное выполнение утилиты ОС и прикладной программы с организацией взаимодействия между ними, что не всегда возможно. Гораздо эффективнее включить обмен данными по заданному протоколу непосредственно в код разрабатываемой прикладной программы.

Можно разработать программу, которая будет обеспечивать обмен по выбранному протоколу с любой стороны - и со стороны сервера, и со стороны клиента (в конце концов, сами все утилиты ОС - это тоже прикладные программы, которые были когда-то кем-то разработаны). Важно только, чтобы такая программа соответствовала спецификации выбранного протокола. Если же это будет не так, и вновь созданная программа не будет удовлетворять требованиям протокола или будет удовлетворять им не полностью, то она не сможет установить связь и осуществить обмен данными с другими программами и утилитами, уже существующими в сети. (Возможно, конечно, разработать две совершенно новых программы - клиента и сервера - которые будут обеспечивать обмен данными по своему протоколу, недоступному другим, но тогда такая система перестанет быть открытой системой, и по сути, перестанет быть частью всемирной сети).

Поскольку в качестве транспортного протокола для всех протоколов прикладного уровня используется TCP, то чтобы реализовать обмен данными по любому выбранному протоколу необходимо использовать обмен данными на основе именно TCP. Для этого в составе многих ОС имеются специальные библиотеки программных средств, позволяющие прикладным программам пользователя осуществлять обмен данными по протоколу TCP. В ОС типа UNIX или Linux это библиотека "сокетов" (socket - ее программный код входит в ядро ОС). В языке программирования "Си", который является базовым для этих ОС, ей обычно соответствует файл заголовков "socket.h". В ОС типа Windows NT (начиная с версии 3.5) и графических средах Windows-95/98 этот протокол реализует [динамическая библиотека "WinSock.DLL"](http://vmk.ugatu.ac.ru/sock.htm) (в более ранних версиях этих ОС она не поставляется и должна устанавливаться дополнительно).

Такие библиотеки позволяют осуществлять самые простые операции: установить связь с ЭВМ, для которой известны адрес и порт связи, передать блок данных по протоколам TCP или UDP, получить блок данных, поступивший на локальную ЭВМ. Этого достаточно для того, чтобы запрограммировать любой прикладной протокол и любой обмен данными, поскольку известны все команды протоколов и используемые для обмена информацией номера портов. Однако это достаточно трудоемкий процесс, поскольку необходимо корректно запрограммировать передачу всех команд выбранного протокола и реакцию на полученные команды. Поэтому к такому методу прибегают обычно только в том случае, если нет возможности использовать стандартные прикладные протоколы и приходится организовывать обмен данными по своему, специфичному протоколу (такой подход рассматривается в следующей лабораторной работы). Для реализации стандартных прикладных протоколов уже разработано большое количество программных библиотек в различных языках программирования, поэтому нет необходимости создавать их заново (например, см. [здесь](http://pcmag.ru/archive/9906/069932.asp)).

Варианты таких библиотек в языке ObjectPascal разработанные фирмой Borland для среды программирования BorlandDelphi рассмотрены далее. Поскольку чаще всего возникает необходимость создать прикладную программу "клиентской" стороны протокола, то далее речь пойдет именно о создании программ такого рода.

Средства реализации прикладных протоколов Internet в среде разработки BorlandDelphi

В среде программирования BorlandDelphi прикладные библиотеки для реализации прикладных протоколов Internet существуют, начиная с версии 3.0 (они существуют и в более ранних версиях этой среды, но поставляются только сторонними разработчиками). Поскольку эта среда программирования основана на использовании компонентной модели, то и все прикладные протоколыInternet реализованы в ней в виде компонент.

Каждому типу протокола в среде программирования соответствует свой класс объектов, который представлен в виде компоненты в рамках среды разработки (графически он отображается в виде "иконки" при проектировании приложения). Для того, чтобы работать с выбранным протоколом, разработчик должен создать экземпляр объекта нужного класса. Это можно сделать в среде разработке при проектировании приложения или же в программном коде с помощью операторов языка программирования ObjectPascal. В первом случае среда разработки автоматически выберет и подключит к коду программы нужную библиотеку, а в последнем случае это должен не забыть сделать сам разработчик.

После того, как экземпляр объекта создан, с ним можно работать, используя свойства и методы объекта. Если необходимо использовать несколько соединений по одному и тому же протоколу, то нужно создать столько экземпляров объекта, сколько соединений предполагается использовать. Соответственно, если надо использовать несколько различных протоколов, то надо создать несколько экземпляров объектов разных классов.

В среде программирования BorlandDelphi версии 3.0 для реализации протоколов FTP, SMTP и POP3 существуют классы объектов типа TFTP, TSMTP и TPOP3 соответственно (обычно они находятся на страничке "Internet" в палитре компонент). В среде программирования BorlandDelphi версии 3.0 для этой же цели служат компоненты TNMFTP, TNMPOP3 и TNMSMTP (на страничке "FastNet" палитры компонент). К сожалению, названия свойств и методов (процедур и функций классов) у этих компонент несколько различаются, но суть и принципы работы с ними от этого не меняются.

Как уже было сказано выше, для работы с прикладным протоколом необходимо создать экземпляр класса, реализующего такую работу. Следующим действием, как правило, становится установка необходимых параметров (свойств) для созданного класса - примерами таких параметров служат, прежде всего, адрес и порт ЭВМ.с которой будет устанавливаться соединение, затем, имя и пароль пользователя и т.д. Конкретный состав параметров зависит от типа протокола и того, для какой цели его планируется использовать. Разработчики компонент предусмотрели для каждой из них широкий набор свойств, соответствующих тем или иным параметрам протокола, но они также позаботились и о том, чтобы пользователь не создавал большой объем кода для задания всех свойств - большинство из них имеют значения по умолчанию, которые, в принципе, не всегда надо изменять. Поэтому разработчик должен позаботится об установке именно тех свойств, которые ему нужны (то, откуда он будет брать значения этих свойств, определяет уже сам создатель программы в соответствии с целью своей работы и выбранным интерфейсом).

После того, как все необходимые параметры заданы, можно инициировать обмен данными. Для этого нужно вызвать некоторый метод протокола и передать ему нужные аргументы. Тогда объект (экземпляр класса) установит соединение, а также автоматически транслирует значения всех установленных свойств в последовательность команд протокола. Когда все команды будут выполнены, прикладная программа может завершить обмен, или же, изменив какие-то данные, инициировать его снова. В процессе обмена могут возникать события, связанные, например, с разрывом соединения, ошибкой передачи данных, отсутствием прав пользователя и другими ситуациями. По многим из этих ситуаций у объектов предусмотрена активизация связанных с ними событий - функций и процедур языка программирования, в которых разработчик может предусмотреть все необходимые действия по реакции прикладной программы на событие.

По завершении обмена данными прикладная программа должна закрыть соединение и удалить экземпляр класса. Если этот экземпляр создавался в среде разработке автоматически, то она сама позаботится о том, чтобы породить программный код, необходимы для его удаления; если же пользователь создавал его "вручную" (непосредственно операторами языка), то он должен не забыть и удалить объект.

Порядок выполнения работы

* 1. Получить вариант задания у преподавателя.
  2. Разработать прикладную программу в соответствии с заданием.
  3. Подготовить и защитить отчет.
  4. Написать и отладить программу на ЭВМ.
  5. Сдать работающую программу преподавателю.

Требования к оформлению отчета

Кроме общих требований, которые можно посмотреть на http://vmk.ugatu.ac.ru/treb.htm, отчет должен содержать следующую информацию:

* + Описание алгоритма взаимодействия прикладной программы с внешним миром на основе заданного протокола.
  + Перечень используемых классов, их свойств и функций ОС или среды разработки для организации протокола взаимодействия.

Основные контрольные вопросы

1. Что такое электронная почта, каким целям она служит ? Каковы преимущества электронной почты перед другими видами коммуникаций ? Расскажите, какие существуют варианты и протоколы организации электронной почты.
2. Как организована передача сообщений в электронной почте, какие поля содержит заголовок сообщения ?
3. Что такое адрес электронной почты ? Из каких полей состоит адрес ?
4. Что такое протокол SMTP ? Как организована передача данных по этому протоколу ?
5. Расскажите об основных командах протокола SMTP.
6. Как организована передача сообщения по протоколу SMTP ?
7. Почему могут возникать проблемы с передачей сообщений на русском языке через электронную почту ? Расскажите о причинах этих проблем.
8. Что такое "почтовый ящик", для чего он используется? Что такое POP3-протокол ?
9. Какие действия позволяет выполнить сетевая утилита FTP ?
10. Расскажите об основных командах утилиты FTP.
11. Как осуществляется соединение и взаимодействие процессов по протоколу FTP ?
12. Какие возможности протокола FTP используются программами - "браузерами" ?
13. Что такое FTP-сервер ? Для чего используется анонимное соединение с сервером ?
14. Какие существуют средства для реализации протоколов Internet в прикладных программах ?
15. Что такое библиотека "сокетов" ? Каким целям она служит ?
16. Как реализовано программирование прикладных протоколов Internet в среде BorlandDelphi ?
17. Расскажите об общих принципах программирования работы с прикладными протоколами.

Варианты заданий

* 1. Необходимо разработать программу, которая позволяет организовать прием выбранного файла с удаленной ЭВМ по протоколу FTP. Адрес ЭВМ, имя и пароль пользователя, имя и местоположения файла должны настраиваться при использовании программы. Порт обмена данными можно считать фиксированным.
  2. Необходимо разработать программу, которая позволяет организовать передачу выбранного файла на удаленную ЭВМ по протоколу FTP. Адрес ЭВМ, имя и пароль пользователя, имя и местоположения файла должны настраиваться при использовании программы. Порт обмена данными можно считать фиксированным.
  3. Необходимо разработать программу, которая позволяет организовать передачу выбранного файла на удаленную ЭВМ по протоколу FTP. Адрес ЭВМ, имя и местоположения файла должны настраиваться при использовании программы. Для регистрации на удаленной ЭВМ требуется использовать анонимный доступ. Порт обмена данными можно считать фиксированным.
  4. Необходимо разработать программу, которая позволяет организовать прием и отображение сообщения электронной почты из почтового ящика пользователя с удаленной ЭВМ по протоколу POP3. Адрес ЭВМ, имя и пароль пользователя должны настраиваться при использовании программы. Порт обмена данными можно считать фиксированным. Допускается использование только символов кодировки ASCII.
  5. Необходимо разработать программу, которая позволяет организовать передачу сообщения электронной почты на удаленную ЭВМ по протоколу SMTP. Адрес ЭВМ, имя и пароль пользователя должны настраиваться при использовании программы. Порт обмена данными можно считать фиксированным. Допускается использование только символов кодировкиASCII.
  6. Необходимо разработать программу, которая позволяет организовать передачу файла на удаленную ЭВМ по протоколу SMTP. Адрес ЭВМ, имя и пароль пользователя должны настраиваться при использовании программы. Порт обмена данными можно считать фиксированным. Допускается использование только символов кодировки ASCII.