Томский государственный университет

Факультет информатики

А.В. Скворцов, Т.Н. Поддубная

Обмен информацией по сети

(методические указания к

лабораторной работе № 10)

Томск – 1999

Указания РАССМОТРЕНЫ и УТВЕРЖДЕНЫ методической комиссией факультета информатики.

Протокол №____ от «____»_____1999 г.

Председатель методической комиссии факультета информатики,

профессор _____ Поддубный В.В.

Утверждаю. Зав. кафедрой прикладной информатики, доцент

_____ Сущенко С.П. «____» _____ 1999 г.

Методические указания посвящены программной среде Borland Delphi компании Inprise Corporation – одному из мощнейших современных средств разработки приложений для Windows.

Указания разработаны для студентов межфакультетской специализации факультета информатики Томского государственного университета. Содержат подробное описание лабораторной работы: цель, задачи, описание работы и используемых компонентов Delphi. В конце работы приведены тексты программ.

Резенцент – канд. техн. наук, доцент **Ю.Л. Костюк.**

© Скворцов А.В., Поддубная Т.Н., 1999

© Оформление и верстка: Скворцов А.В., 1999

В наш век информационных технологий все большую роль играют компьютерные сети как средство быстрой и точной передачи информации. В настоящее время почти на любом компьютере имеются средства доступа к локальной сети и к Интернету.

Для того чтобы разные компьютеры могли свободно понимать друг друга, передача информации ведется в соответствии с некоторыми специальными протоколами. В Интернете используется семейство протоколов TCP/IP. На основе таких протоколов работает механизм так называемых сокетов (socket, англ. – розетка), позволяющий обмениваться произвольной информацией между программами, запущенными на одном или разных компьютерах в сети.

В семействе протоколов TCP/IP каждый компьютер в сети имеет некоторый адрес, состоящий из 4 чисел (от 0 до 255), разделенных точкой, например 193.124.83.15.

На каждом компьютере в сети могут быть установлены специальные программы, которые предоставляют какой-то сервис, например файловые сервера позволяют обращаться к файлам, Web-серверы – открывать страницы Internet и т.д. Каждый сервис имеет свой уникальный на компьютере номер от 0 до 65535, который называется «портом». Программа, предоставляющая какой-либо сервис, при этом называется сервером.

В данной работе необходимо создать программу для общения между двумя пользователями по протоколу TCP/IP с помощью механизма сокетов.

Цель работы

Изучение принципов обмена информацией между приложениями по сети.

Задачи работы

1. Краткое знакомство с компонентами для обмена информацией между компьютерами по локальной сети или по Интернету.
2. Создание программы для обмена текстовыми сообщениями между пользователями на разных компьютерах.

Описание работы

Перед началом работы необходимо, как обычно, создать новый проект и сохранить его в отдельном каталоге Lab10 в соответствии с рекомендациями, приведенными в предыдущих лабораторных работах.

Для работы с механизмом сокетов в Delphi на закладке «Internet» имеются два компонента `TServerSocket` и `TClientSocket`. Первый из них предназначен для создания программы-сервера, предоставляющей «сервис» (в данном случае возможность приема и обработки сообщений) с заданным номером порта, к которому могут подсоединяться и обращаться одновременно множество других программ в режиме клиентов, для чего уже используются компоненты второго типа.

На одном компьютере может одновременно работать множество различных сервисов, но все они различаются номерами обслуживаемых портов. Некоторые номера портов зарезервированы для специального использования в системе, например, 80 – для обращения к HTTP-серверам (для просмотр страниц в Интернете), 20 и 21 – для обмена файлами с серверами в Интернете по протоколу FTP и т.д.

Для того чтобы сервис начал работать, в компоненте `TServerSocket` достаточно указать номер обслуживаемого порта и создать обработчики некоторых событий, чтобы он смог реагировать на поступающие команды.

Для начала работы клиентской стороны в компоненте `TClientSocket` необходимо указать имя или точный IP-адрес компьютера-сервера с нужным нам сервисом, номер порта сервиса и создать обработчики нескольких событий. После этого нужно дать команду для установления соединения. Если оно будет успешно установлено, можно начинать обмениваться данными между программой-клиентом и программой-сервером.

В табл. 25-28 приведено краткое описание основных свойств и событий объектов типов `TServerSocket` и `TClientSocket`.

Наше приложение будет содержать в себе компоненты типов как `TServerSocket`, так и `TClientSocket`. При запуске программы необходимо будет включить серверную часть, так чтобы она готова была принимать поступающие от других аналогичных программ сообщения. В то же время определим команду соединения с другой такой же программой, в которой мы должны закрыть серверный компонент и использовать клиентский для обращения к серверу. В нашей программе оба компонента одновременно работать не будут, а только один. Например, после запуска нашей программы на двух различных компьютерах и подключения одной из них к другой на одной из них будет работать серверный компонент, а на другой клиентский.

Для установления связи между нашими программами необходимо выбрать какой-то неиспользуемый стандартно номер порта. Пусть это будет номер 1414.

Таблица 25. Основные свойства объектов типа TServerSocket

Свойство	Тип	Комментарий
Active	Boolean	Работает ли сервер
Port	Integer	Номер порта, обслуживаемого сервером на компьютере

Таблица 26. Основные события объектов типа TServerSocket

Свойство	Комментарий
OnAccept	Вызывается после того, как клиент допущен для соединения с сервером
OnClientConnected	Вызывается после установления клиентом связи с сервером. Информацию об имени и адресе клиента можно получить с помощью свойств Socket.RemoteHost и Socket.RemoteAddress, где Socket – параметр, передаваемый в данное событие
OnClientDisconnected	Вызывается после закрытия каким-нибудь клиентом соединения с сервером
OnClientError	Возникает при возникновении ошибок связи. Некоторые из ошибок связи не передаются сюда, но при этом возникают исключения. Если их не отлавливать централизованно с помощью события OnException компонента типа TApplicationEvents, то они будут выдаваться пользователю в окне сообщения
OnClientRead	Вызывается при поступлении информации от клиентов. Получить информацию в виде текстовой строки можно с помощью вызова функции Socket.ReceiveText, где Socket – параметр, передаваемый в данное событие

Наше приложение должно содержать строку редактирования для ввода посылаемых сообщений и большое мемо-поле для сохранения всех отправленных, принимаемых и служебных сообщений.

Для отправки сообщений необходимо определить соответствующую команду отправки, автоматически определяющей, в каком режиме сейчас работает программа (серверном или клиентском). Для приема сообщений необходимо написать обработчики событий OnClientRead и OnRead соот-

ветственно для компонентов TServerSocket и TClientSocket. В этих обработчиках необходимо полученный с помощью вызова функции Socket.ReceiveText текст вывести в мемо-поле.

Таблица 27. Основные свойства объектов типа TClientSocket.

Свойство	Тип	Комментарий
Active	Boolean	Установлено ли соединение с компонентом-сервером на этом же или другом компьютере
Address	String	Полный IP-адрес сервера. Если указать это свойство, а не Host, подключение к серверу будет выполняться быстрее, т.к. не нужно искать по имени адрес
Host	String	Имя компьютера-сервера в локальной сети, его UNC-имя (например, mx.inf.tsu.ru) или полный IP-адрес (например, 193.124.83.15). Если свойства Host и Address не указаны, то считается, что сервис нужно искать на этом же компьютере, что и клиентская программа

Таблица 28. Основные события объектов типа TClientSocket

Свойство	Комментарий
OnConnect	Вызывается после установления связи с сервером
OnDisconnect	Вызывается после закрытия соединения с сервером
OnError	Возникает при возникновении ошибок связи
OnRead	Вызывается при поступлении информации от клиентов

В заключение остановимся на компоненте типа TApplicationEvents, находящемся на закладке палитры компонентов «Additional». Этот компонент не имеет никаких существенных свойств, но предоставляет целый ряд событий, имеющих смысл на уровне всего приложения, например события активизации приложения в Windows (OnActivate) и потери фокуса (OnDeactivate), минимизации (OnMinimize) и восстановления размеров (OnRestore); события, возникающие при возникновении ошибок (OnException), при необходимости выдачи справки по программе (OnHelp), при появлении свободного процессорного времени для выполнения каких-либо фоновых задач (OnIdle) и т.д.

В нашей программе определим обработчик события `OnException`, вызываемый при возникновении исключения при работе программы, в котором текст ошибки мы будем выводить в мемо-поле.

Внешний вид приложения и состав его меню на этапе разработки приведены на рис. 38-39.

На рис. 40 приведен внешний вид запущенного приложения после того, как окно было несколько растянуто. В листинге 21 приведен текст файла модуля.

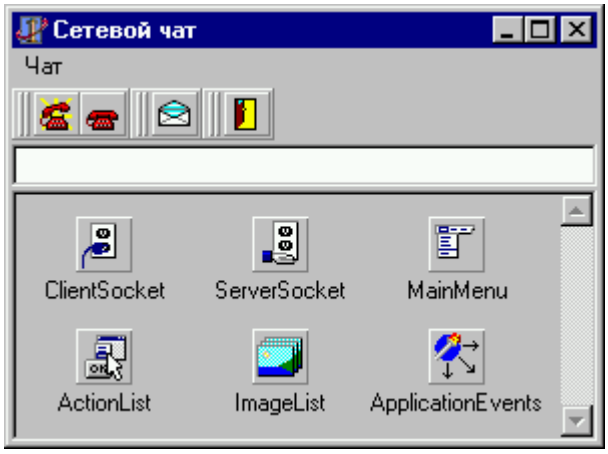

Рис. 38. Внешний вид разрабатываемого приложения

Листинг 21. Текст главного модуля ChatMain.pas

```
unit ChatMain;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls, Buttons, ScktComp, ExtCtrls, ComCtrls, ActnList, ImgList,
  ToolWin, AppEvnts;

type
  TChatForm = class(TForm)
    MainMenu: TMainMenu;
    Memo: TMemo;
    ServerSocket: TServerSocket;
    ClientSocket: TClientSocket;
    ActionList: TActionList;
    ImageList: TImageList;
    Connect: TAction;
    Disconnect: TAction;
    Exit: TAction;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    Send: TAction;
    N7: TMenuItem;
    Send1: TMenuItem;
    ControlBar1: TControlBar;
    ToolBar1: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolBar2: TToolBar;
    ToolButton7: TToolButton;
    ToolBar3: TToolBar;
    ToolButton8: TToolButton;
    PanelMessage: TPanel;
    EditMessage: TEdit;
```

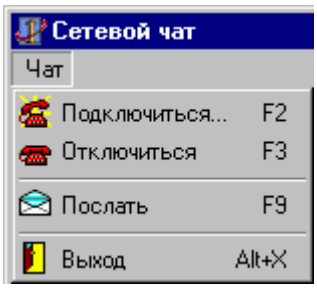

Рис. 39. Меню чата

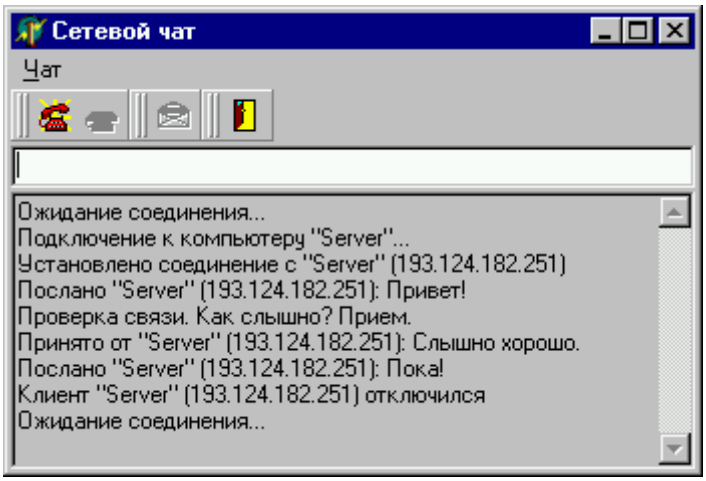

Рис. 40. Внешний вид приложения

```
    ApplicationEvents: TApplicationEvents;
    procedure FormCreate(Sender: TObject);
    procedure ClientSocketConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketRead(Sender: TObject; Socket: TCustomWinSocket);
    procedure ServerSocketClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketAccept(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketError(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
    procedure ConnectExecute(Sender: TObject);
    procedure DisconnectExecute(Sender: TObject);
    procedure ExitExecute(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure SendExecute(Sender: TObject);
    procedure EditMessageKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure SendUpdate(Sender: TObject);
    procedure DisconnectUpdate(Sender: TObject);
    procedure ServerSocketClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ApplicationEventsException(Sender: TObject; E: Exception);
  protected
    procedure ServerMode;
  end;

var
  ChatForm: TChatForm;

implementation

{$R *.DFM}

procedure TChatForm.FormCreate(Sender: TObject);
begin // При запуске программы перейти сразу в режим сервера
  ServerMode;
end;

procedure TChatForm.FormDestroy(Sender: TObject);
begin // Разорвать все соединения при выходе из программы
  ClientSocket.Close;
  ServerSocket.Close;
end;

procedure TChatForm.ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin // В режиме клиента установлена связь с сервером
  Memo.Lines.Add(Format('Установлено соединение с "%s" (%s)',[Socket.RemoteHost,
    Socket.RemoteAddress]));
end;

procedure TChatForm.ClientSocketDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin // В режиме клиента разорвана связь с сервером
  Memo.Lines.Add(Format('Клиент "%s" (%s) отключился',[Socket.RemoteHost,
    Socket.RemoteAddress]));
  ServerMode;
end;
```

```
procedure TChatForm.ClientSocketRead(Sender: TObject;
  Socket: TCustomWinSocket);
begin // В режиме клиента получено сообщение
  Memo.Lines.Add(Format('Принято от "%s" (%s): %s', [Socket.RemoteHost,
    Socket.RemoteAddress, Socket.ReceiveText]));
end;

procedure TChatForm.ClientSocketError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin // В режиме клиента возникла ошибка
  Memo.Lines.Add(Format('Ошибка связи с "%s" ("%s", %s)', [ClientSocket.Host,
    Socket.RemoteHost, Socket.RemoteAddress]));
  ErrorCode:=0; // Запретить выдачу сообщения об ошибке
  if not ClientSocket.Active then
    ServerMode;
end;

procedure TChatForm.ServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin // В режиме сервера подключился клиент
  Memo.Lines.Add(Format('Установлено соединение с "%s" (%s)', [Socket.RemoteHost,
    Socket.RemoteAddress]));
end;

procedure TChatForm.ServerSocketClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin // В режиме сервера отключился клиент
  Memo.Lines.Add(Format('Клиент "%s" (%s) отключился',[Socket.RemoteHost,
    Socket.RemoteAddress]));
end;

procedure TChatForm.ServerSocketClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
begin // В режиме сервера получено сообщение
  Memo.Lines.Add(Format('Принято от "%s" (%s): %s', [Socket.RemoteHost,
    Socket.RemoteAddress, Socket.ReceiveText]));
end;

procedure TChatForm.ConnectExecute(Sender: TObject);
const ServerName: string = '';
begin // Выполнить подключение к серверу в режиме клиента
  if InputQuery('Подключение к компьютеру', 'Адрес или имя:', ServerName) then
    if ServerName<>'' then
      with ClientSocket do
      begin
        Memo.Lines.Add(Format('Подключение к компьютеру "%s"...',[ServerName]));
        ServerSocket.Close; // Закрыть серверный сокет
        Close;              // Закрыть клиентский сокет
        Host:=ServerName;   // Указать имя хоста или адрес сервера в клиентском сокете
        Open; // Открыть клиентский сокет и подсоединиться к серверу
      end;
end;

procedure TChatForm.ServerMode;
var i: integer;
begin // Установить режим сервера
  ClientSocket.Close; // Клиентский сокет закрыть
  for i:=ServerSocket.Socket.ActiveConnections-1 downto 0 do
    ServerSocket.Socket.Connections[0].Close; // Отключить подключенных к серверу клиентов
```

```
  ServerSocket.Open; // Открыть серверный сокет
  Memo.Lines.Add('Ожидание соединения...');
end;

procedure TChatForm.DisconnectUpdate(Sender: TObject);
begin // Команда отключения от сервера доступна только если соединение установлено
  Disconnect.Enabled:=ClientSocket.Active or (ServerSocket.Socket.ActiveConnections>0);
end;

procedure TChatForm.DisconnectExecute(Sender: TObject);
begin // Отключиться от сервера и самому перейти в режим сервера
  ServerMode;
end;

procedure TChatForm.SendUpdate(Sender: TObject);
begin // Разрешить посылку сообщений только если установлено соединение
  Send.Enabled:=ClientSocket.Active or
    ServerSocket.Active and (ServerSocket.Socket.ActiveConnections>0);
end;

procedure TChatForm.SendExecute(Sender: TObject);
  procedure Send(S: TCustomWinSocket; Msg: string);
  begin
    S.SendText(Msg);
    Memo.Lines.Add(Format('Послано "%s" (%s): %s', [S.RemoteHost, S.RemoteAddress, Msg]));
  end;
var i: integer;
begin // Выполнить посылку сообщения
  if ClientSocket.Active then // Режим клиента
    Send(ClientSocket.Socket, EditMessage.Text)
  else // Режим сервера
    for i:=0 to ServerSocket.Socket.ActiveConnections-1 do
      Send(ServerSocket.Socket.Connections[i], EditMessage.Text);
  EditMessage.SelectAll;
end;

procedure TChatForm.EditMessageKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin // Клавиша "Return" пусть тоже выполняет посылку сообщения
  if Key=VK_RETURN then
    Send.Execute;
end;

procedure TChatForm.ExitExecute(Sender: TObject);
begin // Выход из программы
  Close;
end;

procedure TChatForm.ApplicationEventsException(Sender: TObject; E: Exception);
begin // Любая необработанная ошибка в программе приходит сюда
  Memo.Lines.Add('Ошибка: '+E.Message);
end;

end.
```

Вопросы и задания для самостоятельной работы

1. Что означают понятия «адрес», «сервис» и «порт» с точки зрения семейства протоколов TCP/IP? И как они задаются?
2. Как устанавливается связь между программами в Delphi с помощью механизма сокетов?
3. Для обмена информацией между компьютерами через механизм сокетов какой базовый протокол связи должен работать между ними?
4. Для чего используется компонент типа `TApplicationEvents`?

Учебное издание

Алексей Владимирович Скворцов
Поддубная Тамара Николаевна

Обмен информацией по сети

(методические указания к
лабораторной работе № 10)

Томский государственный университет. Томск, 1999. – 12 с.