делить среди всего набора используемых системных функций группы функций, общие для различных семейств операционных систем. Примерами таких групп функций являются:

а) функции управления операционной системой;
б) функции файлового ввода-вывода;
в) функции графической подсистемы;
г) функции ввода пользовательской информации;
д) функции управления оперативной памятью;
е) функции по управлению вычислительным процессом.

- *Ресурс*. Данный компонент характеризует аппаратные составляющие компьютера. Выделяется стандартный набор ресурсов, характерный для всех классов компьютеров различных аппаратных платформ. К этим ресурсам относятся:

а) процессор (может быть установлено несколько процессоров);
б) оперативная память;
в) жесткий диск (может быть установлено несколько жестких дисков);
г) устройство ввода информации (клавиатура);
д) видеоадаптер (может быть установлено несколько видеоадаптеров);
е) сетевой адаптер (может быть установлено несколько сетевых адаптеров).

- *Объекты базы данных*. Данные компоненты характеризуют логическую и программную реализацию распределенной информационной базы данных. Примерами таких объектов могут служить: таблицы, виды, хранимые процедуры, определенные пользователем типы данных, правила.

- *Записи баз данных*. Данные компоненты представляют собой строки таблиц базы данных и состоят из набора полей и содержат информацию о каком-либо объекте.

- *Поля данных*. Являются элементарными единицами хранения информации. Поля данных могут хранить информацию различного типа: строки, числа.

Предложенный метод декомпозиции позволяет решить задачи составления имитационной модели работы вычислительных процессов в узле ЛВС и рационального выбора организации узла ЛВС.

Литература.

1. Агеенко И.В. Метод и средства автоматизации исследования вычислительного процесса информационных систем в локальных вычислительных сетях Дисс. на соиск. уч. ст. к.т.н., - Гомель, 1999г, - 257с.
2. Воруев А.В. Мониторинг и адаптация вычислительного процесса в узлах локальной вычислительной сети с использованием моделей рабочей нагрузки: Дис ... канд.техн.наук: 05.13.13., 05.13.18 - Гомель, 2001. - 291с
3. Демиденко О.М., Максимей И.В. Имитационное моделирование взаимодействия процессов в вычислительных системах. – Мн.: Беларуская навука, 2000. - 230с
4. Демиденко О.М., Максимей И.В. Проектное моделирование вычислительного процесса в локальных вычислительных сетях. – Мн.: Беларуская навука, 2001. - 252

ИСПОЛЬЗОВАНИЕ СОКЕТОВ WINDOWS ДЛЯ СЕТЕВОГО ПРОГРАММИРОВАНИЯ В СРЕДЕ РАЗРАБОТКИ DELPHI 5

Д.В. Ломаник
Научный руководитель - Н.А. Переверзева
Гродненский государственный университет имени Янки Купалы

Сокеты - один из наиболее удобных и простых способов для взаимодействия программ, они были разработаны в Калифорнийском университете как интерфейс прикладного программирования для сетевых приложений TCP/IP. Сокеты -- это набор функций Windows API, которые помогают установить связь между программами, запущенными на одном или разных компьютерах. При взаимодействии программ всегда различают клиентскую и серверную части. Клиент

обращается с запросом к серверу, а сервер удовлетворяет его запрос, и именно сокеты обеспечивают их связь.

Серверный сокет работает в режиме прослушивания сети (ждет запросов от клиента). Клиент же обращается к конкретному хосту, указывая его имя, которое может быть либо цифровым (4 байта разделенных точками) либо символьным. Для определения сервиса на серверном хосте используются номера портов (целые числа от 0 до 65535).

По методу работы сокеты делятся на три вида:

1) Клиентские сокеты. Они инициализируют соединение с сервером и должны для этого указать IP адрес сервера и номер порта интересующего сервиса.

2) Слушающие сокеты. Они устанавливают связь между клиентским и серверным сокетами. В Delphi 5 слушающий сокет встроен в компонент TServerSocket.

3) Серверные сокеты обслуживают запросы клиентских сокетов выполняя его требования. При этом клиентский сокет получает описание серверного сокета.

Рассмотрим компоненты TClientSocket и TServerSocket со страницы Internet библиотеки компонентов Delphi 5. Компонент TServerSocket выполняет следующие задачи:

- слушает указанный ему в свойстве Port порт;
- выполняет соединение с клиентским сокетом;
- получает информацию о соединении,
- обменивается данными с клиентским сокетом;
- закрывает соединение.

Клиентский сокет:

- определяет нужный сервер,
- устанавливает с ним связь;
- получает информацию о соединении,
- производит чтение и запись данных с сервера и на сервер,
- закрывает соединение.

Для открытия соединения необходимо в клиентском сокете указать в свойстве Host имя сервера или в свойстве Address его IP-адрес и задать порт в свойстве Port.

Далее необходимо свойству Active присвоить значение True или вызвать

метод Open. Прервать соединение со стороны клиента можно вызовом метода Close

Рассмотрим основные свойства и методы компонента TServerSocket, которые во многом аналогичны свойствам TClientSocket:

Port: задает номер порта или имя сервиса, который слушает компонент. Для того чтобы соединить клиентский и серверный сокеты необходимо им указать одинаковый порт.

ServerType: Задает тип сокета: блокирующий (StThreadBlocking) или не блокирующий (StNonBlocking).

Active: присвоение этому свойству значения True или вызов метода Open открывает слушающее соединение.

Важнейшим свойством компонентов TClientSocket и TServerSocket является свойство Socket. Именно с помощью его методов и осуществляется обмен данными в сокетном соединении.

Если сокеты не блокирующие, то при получении запроса на чтение или запись генерируются события OnRead или OnWrite у клиентского сокета и события OnClientRead или OnClientWrite у серверного сокета. Для прочтения данных необходимо в обработчике событий OnRead и OnClientRead использовать методы свойства Socket ReceiveText или ReceiveBuf первый из которых возвращает строку, а второй читает из сокета буфер. Узнать размер передаваемого буфера можно с помощью метода ReceiveLength. Для записи можно использовать один из трёх методов свойства Socket: SendBuf SendStream или SendText. Например, оператор:

```
Str:= Socket.ReceiveText;
```

в обработчике события OnRead клиентского или OnClientRead серверного сокета присвоит строковой переменной Str прочитанный с серверного сокета текст, а оператор

```
ClientSocket1.Socket.SendText('hi');
```

пошлет строку 'hi' серверному сокету. Одним из важнейших свойств серверного сокета является свойство Connections. Это массив, в котором находятся объекты, обслуживающие активные соединения. Количество таких соединений можно узнать из свойства ActiveConnections. Как

сервер так и клиент передают данные одним блоком только в том случае, если его длина не превышает 8192 байта, иначе данные разбиваются на несколько блоков. Кроме того строка, передаваемая методом SendText не должна содержать символы, коды которых меньше 32. При достаточно быстрой отправке нескольких блоков данных друг за другом, эти блоки соединяются и отправляются как один блок.

Автором была разработана программа LSX Controller, которая предназначена для управления удаленным компьютером по локальной сети. Она работает под операционными системами Windows 98 или Windows Me и предназначена для того, чтобы администраторы сетей под этими ОС смогли следить за действиями пользователей и при необходимости вмешиваться в их работу.

LSX Controller состоит из двух частей: агентской и управляющей. Агентская часть должна быть запущена на компьютере, которым необходимо управлять. Она прописывает себя в файл Win.ini или реестр и автоматически запускается при загрузке Windows. С помощью функций RegisterServiceProcess из kernel32.dll и ShowWindow агентская часть делается невидимой без специальных программ.

Управляющая часть программы сканирует введенный пользователем диапазон IP-адресов в поисках запущенных агентских частей и налаживает с ними связь. Далее пользователь управляющей части может:

- видеть экран удаленного компьютера;
- отслеживать и управлять передвижениями мыши и нажатиями на нем клавиш;
- выполнять команды;
- выключать или перегружать его;
- блокировать клавиатуру и мышь;
- в программу встроен чат администратора с пользователем, функция захвата экрана и рисования по нему.

При разработке программы использована информация с сайта www.doit.com.

ИСПОЛЬЗОВАНИЕ ИЗБЫТОЧНОСТИ БИТОВЫХ МАТРИЦ МУЛЬТИМЕДИЙНЫХ ФАЙЛОВ ДЛЯ СОКРЫТИЯ ДАННЫХ

А. Н. Волкович
Научный руководитель – Л.В. Рудикова
Гродненский государственный университет имени Янки Купалы

Проблема сокрытия информации имеет глубокие исторические корни. Еще в Античной Греции были разработаны различные методы защиты важной информации: скитала, квадрат Полибия, первые «сдвиговые» шифры. Позже, возникли шифры «простой замены» и «перестановочные». В XVIII веке появился шифр «по книге», который можно рассматривать как развитие шифра Цезаря. Все представленные методы защиты информации обладали достаточной стойкостью, но, при определенных усилиях, все-таки могли быть расшифрованы за конечный промежуток времени. Поэтому возникла необходимость не шифрования, а сокрытия наличия информации.

Данный подход получил название «стеганография» (от греч.: «steganos» – секрет, тайна, «graphy» – запись) и означает «тайнопись». Первые упоминания стеганографических методов встречаются уже в Древней Греции, где тексты писались на дощечках, покрытых воском. Хорошо известны также различные способы сокрытия информации между строк обычного письма, от применения молока до использования сложных химических реакций с последующей обработкой при чтении. Другие методы стеганографии включают использование микрофотоснимков, незначительные различия в написании рукописных символов, маленькие проколы определенных напечатанных символов и т.д., скрывающие истинный смысл тайного сообщения в открытой переписке.

Компьютерные технологии способствовали развитию и совершенствованию стеганографии. В области защиты информации появилась цифровая или компьютерная стеганография. Одновременно развиваются и новые методы, предназначенные для обеспечения безопасности и сохранности передачи данных по каналам телекоммуникаций, использования их в различных целях. Данные методы позволяют скрывать сообщения в файлах (контейнерах), учитывая естест-