пакет. Она позволяет, в том числе, отражать атаки повторного воспроизведения – это атаки на систему аутентификации путём записи и последующего воспроизведения ранее посланных сообщений. Для этого считываются поля *payload* и *timestamp*, вычисляется хэш-сумма полученного сложения и сравнивается с тем, что находится внутри пакета. При любом несовпадении обработка пакета прерывается, и обработчик переходит к следующему пакету. Таким образом создан защищённый протокол передачи данных для ИК-интерфейса.

IX. Заключение

Мы живем в век быстрого развития технологий. Технология Интернета вещей включает в себя использования множества современной техники. В статье представлен общий подход к анализу событий безопасности и его реализация. Использование ИК-канала, а также множество различных датчиков позволяет реализовать автоматизацию и дистанционное управление между различными системами в доме, что повышает комфорт жильцов умного дома. Протокол, разработанный в ходе данной работы, имеет довольно небольшую скорость, но при наличии более продвинутого оборудования можно увеличить скорость отправления и объем информации.

1. Himadri Nath Saha, Supratim Auddy, Avimita Chatterjee and others – «IoT Solutions for Smart Cities». Industrial Automation and Electromechanical Engineering Conference (IEMECON), 2017 8th Annual, pp. 74-80.
2. Olena Starkova, Kostiantyn Herasymenko, Yekaterina Babailova – «Remote Control Systems of Household Appliances». 2017 4th International Scientific-Practical Conference, Kharkiv, Ukraine, 2017, pp. 585-588.
3. Henrik Karlzen – «An Analysis of Security Information and Event Management Systems». 2009 Department of Computer Science and Engineering CHALMERS UNIVERSITY OF TECHNOLOGY UNIVERSITY OF GOTHENBURG Göteborg, Sweden
4. Zegzhda D. P., Lavrova D. S – «Approach to Internet of Things Detection of Security Incidents Using SIEM Technology». 2017 Peter the Great St. Petersburg Polytechnic University St. Petersburg, Russia
5. «SIEM systems: Basic concepts and glossary», <https://learn.techbeacon.com/units/siem-systems-basic-concepts-glossary>
6. Десницкий В.А., Котенко И.В., Копчак Я.М. – «Анализ событий для управления инцидентами безопасности на примере системы контроля и управления доступом». 2016 Санкт-Петербургский институт информатики и автоматизации РАН (СПИИРАН)
7. «What is infrared?», <https://www.livescience.com/50260-infrared-radiation.html>
8. «Infrared remote control technology», <http://www.epanorama.net/links/irremote.html>
9. «IR Communication», <https://learn.sparkfun.com/tutorials/ir-communication>
10. «NEC протокол инфракрасного управления», https://led-displays.ru/ir/nec_protocol.html
11. Yuxiang Feng, Yukai Weng and others, «A Replay-Attack Resistant Authentication Scheme for the Internet of Things». 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), pp.541-547.

Димитриев А.П., Лавина Т.А.

О некоторых предпосылках для автоматизированной обработки данных

*Чувашский государственный университет имени И.Н. Ульянова
(Россия, Чебоксары)*

doi:10.18411/lj-04-2018-131

idsp: 000001:lj-04-2018-131

Аннотация

Возможность реализации алгоритмов обработки данных в среде разработки Turbo Delphi основана на ряде технических предпосылок. В статье рассмотрены элементы пользовательского интерфейса для работы с файлами, функции, реализующие операции над файлами и каталогами, а также вопросы организации параллельной обработки информации. Для иллюстрации рассмотрено применение этих предпосылок в программных комплексах.

Ключевые слова: обработка данных, работа с файлами, операции над директориями, распараллеливание вычислений, Delphi

Введение

Некоторые научные работы по информационным технологиям связаны с автоматизированной обработкой данных на персональном компьютере. Этот вопрос является актуальным для магистров профиля «Методы и алгоритмы обработки данных в информационно-вычислительных системах», обучающихся в ФГБОУ ВО «ЧГУ им. И.Н. Ульянова». Кроме того, для студентов, обучающихся по направлению «Приборостроение», актуально получение навыков применения компьютерных технологий для обработки информации из различных источников [6].

Указанные выше данные могут находиться в файлах, в базе данных или поступать от каких-либо устройств, а обрабатываться они могут длительное время, что зависит от алгоритма обработки, который нередко приходится реализовывать на языках программирования.

Цель работы – выявление некоторых технических предпосылок для возможности реализации алгоритмов обработки данных. Для этого рассмотрим такие технические предпосылки, как элементы пользовательского интерфейса, операции, связанные с файлами и директориями и организацию параллельной обработки информации.

Элементы пользовательского интерфейса, связанные с файлами

Для автоматизированной обработки данных можно разрабатывать прикладные программы, для чего требуется среда разработки. В качестве примера применения среды разработки программного обеспечения рассмотрим Turbo Delphi, так как это бесплатная среда разработки, установленная в некоторых компьютерных классах Чувашского государственного университета.

В офисных программных системах часто встречается главное меню (элемент типа TMainMenu), где имеется пункт «Файл». Здесь имеется возможность выбора имени открываемого файла с помощью стандартного диалогового окна. Для этого используется компонент типа TOpenDialog и его методы и свойства Execute (открыть окно диалога) и FileName (имя выбранного файла). Для того, чтобы диалог отображал не весь список файлов, а файлы с указанным пользователем расширением, применяется свойство диалога Filter. В нем перечисляются расширения файлов и их описания.

Рисунок 1. Пример применения элемента типа TListView

Обработка данных может производиться не только для отдельных файлов, но и для совокупности файлов и каталогов с содержащимися в них файлами и подкаталогами. В качестве примера задачи, связанной с такой обработкой, можно привести шифрование [5]. Разработана программа, шифрующая группу файлов (первичная обработка) и создающая из них выходной файл. Из выходного файла эта программа извлекает данные, выполняет их расшифровку и создает исходные файлы в указанном каталоге. В примере для отображения содержимого каталога (рис. 1) использован элемент `ListView1` типа `TListView`. Для добавления строк в список применяется метод `ListView1.Items.Add`. Для внесения значений в столбцы списка применяется метод `Listitem.SubItems.Add`.

Возможно, все файлы в каталоге обрабатывать не нужно, а только некоторые. С этой целью предусмотрены флажки в левой части выводимой таблицы. Обрабатываться будут только те файлы и каталоги, около которых флажок включен. Для этого свойство `CheckBoxes` элемента `ListView1` установлено в значение `True`.

Пользователю удобнее производить визуальный поиск в списковой информации, если она расположена в алфавитном порядке. В связи с этим разработчику программ необходимо уметь так программировать, чтобы выводимые таблицы могли быть отсортированы. Например, в рассматриваемом примере предусматривается возможность сортировки выводимой таблицы.

Широко применяемое программное обеспечение, например, Проводник Windows, для сортировки содержимого таблиц часто предлагает пользователю всего лишь щелкнуть мышью по заголовку столбца, по которому должна производиться сортировка. В Turbo Delphi для этого следует программировать обработчик некоторого события.

Так, в рассматриваемом примере для этого сначала требуется в отдельном списке `Listbox1` типа `TListBox` выбрать способ сортировки, а именно, по какому из столбцов. Для идентификации выделенного столбца используется метод `Selected`, например, `Listbox1.Selected[4]` означает выбор пятого столбца как ключа сортировки (нумерация начинается с нуля). Последующий двойной щелчок по таблице приводит к нужному результату: таблица будет отсортирована по выбранному полю. С этой целью в программе применяется метод `ListView_SortItems`, выполняющий сортировку содержимого элемента `TListView` по указанному столбцу. Для сортировки в обратном порядке нужно еще раз дважды щелкнуть по списку. Это инвертирует один из параметров вышеуказанного метода, что приводит к изменению порядка сортировки.

Операции над файлами и директориями

При использовании файлов как источника данных необходимо умение применять такие операции, как чтение и запись файлов (функции `read`, `blockread`, `write` и `blockwrite`), позиционирование файла (`seek`), открытие файла (`reset`) в различных режимах (как текст, как последовательность блоков; для чтения, для чтения и записи, для добавления в конец - `append`), закрытие файла (`close`), создание файла (`rewrite`), проверка достижения конца файла (`eof`).

По аналогии, например, с программой 7-Zip File Manager, необходимо реализовать переход (навигацию) по отображаемым каталогам. Как вариант, для этого применяется кнопка «Open», нужно только предварительно выделить папку. Обработчик щелчка по этой кнопке запускает метод `ChDir`, который производит смену текущего каталога (от англ. `change directory`). Затем список очищается и выполняется подпрограмма заполнения списка содержимым каталога. Эта подпрограмма использует методы `FindFirst`, `FindNext` и `FindClose`, работающие с указываемым путем, атрибутом поиска и переменной типа `TSearchRec`, которая содержит атрибуты очередного файла. К атрибутам файла относятся его полное имя, имя MS-DOS, время и дата создания, размер и некоторые другие. Для их извлечения имеются методы с соответствующими названиями. В примере использованы такие методы и свойства: `ExtractFileName`, `Name`, `Size`, `ExtractFileExt`, `Time`, `Attr`. Свойство

Attr используется для определения, является ли запись директории файлом или каталогом. Если это каталог, в соответствующем столбце программа отображает строку «<DIR>».

Для защиты от ошибок пользователя нужно принимать некоторые меры. Например, пользователь может попытаться открыть с помощью программы неподходящий файл. Для предотвращения непредсказуемых действий с таким файлом он должен содержать заголовок, в котором указывается формат файла. В примере каждый выходной файл программы в своем начале имеет три символа «D25», после которых следует список файлов и папок в каталоге. Программа при открытии такого файла сначала проверяет, начинается ли он с этих трех символов, и если нет, никакой обработки не производится. Предпочтительнее, если пользователю при этом будет выведено сообщение о неверном формате файла. Для вывода сообщений предусмотрена функция Showmessage. Если при выводе сообщения требуется выбрать дальнейшее действие, можно вызвать диалоговое окно функцией InputBox.

В приведенном примере предполагается, что из выходного файла создаются другие файлы, как это производится, например, в архиваторах. Если таких новых файлов много, то имеет смысл помещать их в отдельную папку. Поэтому необходимо предложить пользователю выбор папки для сохранения файлов. Программа в данном примере для этого вызывает стандартный диалог Windows открытия папки, вызываемый функцией SelectDirectory, в параметрах которого указывается также исходный путь к каталогу. Для использования этой функции в секцию Uses нужно добавить модуль FileCtrl. Предпочтительнее, если пользователь в этом же диалоге сможет создавать новую папку. Так же, как и в случае первичной обработки данных, флажок «Computer» позволяет указывать папку, начиная либо с текущего каталога, либо с «Мой компьютер». Чтобы определить путь к текущей папке, используется функция GetCurrentDir, которая требует объявления SysUtils в секции Uses.

В выбранной папке для извлечения файлов уже могут находиться файлы с такими же именами, в которых может содержаться нужная информация. Для того чтобы избежать случайного уничтожения таких файлов, программа должна выдавать запрос на перезапись при каждом совпадении имен существующего и нового файлов. Для проверки существования файла используется функция FileExists.

Для того, чтобы программа совершала рекурсивную обработку подкаталогов, от программиста требуется умение программировать рекурсии, а также применения функции для смены текущего каталога. Для этого существует функция MkDir, причем если каталог уже существует, возникает ошибка. Поэтому предварительно нужно проверять существование каталога функцией DirectoryExists. Как альтернатива, возможно использование функции CreateDir.

Параллельная обработка данных

При высокой загрузке центрального процессора иногда полезно распараллелить вычисления. Например, создать в приложении поток, занимающийся вычислениями, и поток, обрабатывающий события мыши. Для этого может использоваться функция CreateThread, либо можно воспользоваться методом Create, объявив тип соответствующего программного модуля как class(TThread).

Однако, иногда требуется одновременно выполнять вычисления на разных процессорах для ускорения этого процесса. Например, можно запустить сразу два одинаковых процесса, обрабатывающие одну базу данных. Таким образом производится наполнение семантической базы данных результатами обработки текстовых файлов [2, 3, 4]. Суть обработки состоит в обновлении информации в совокупности полей файла формата DBF на основе данных, полученных в результате обработки входных текстовых файлов и таблицы русских основ.

При этом часто возникают ситуации одновременного обращения к одной записи таблицы базы данных двумя процессами, и возникает ошибка. Если это позволяет логика,

то используя конструкцию try... except можно продолжить работу приложения, не завершая его.

Если же требуется обрабатывать данные без использования базы данных, причем когда данные логически могут обрабатываться одновременно более чем одним процессом, можно передавать данные между двумя процессами. Если совершать такую передачу с помощью буфера обмена, часто возникают ошибки доступа к буферу обмена, после чего соответствующий поток досрочно завершается. Если использовать для передачи данных компоненты наподобие TCPClient и TCPServer, то для передачи коротких сообщений требуется длительное время, связанное с установлением соединения по протоколу TCP. Если использовать общий файл, изнашивается физический носитель данных. В этой ситуации предлагается применять протокол транспортного уровня UDP, используя подключаемые модули Winsock и StdCtrls. С помощью тестовой программы выявлено, что в секунду можно передавать и принимать около 1670 сообщений по протоколу UDP, тогда как по TCP – приблизительно 19,6.

Необходимо иметь в виду, что использование протокола UDP в Turbo Delphi требует громоздких конструкций в исходном тексте программ, включая такие как создание сокета, установку его опций и др. [1]. Необходимо создание прослушивающего сеть методом listen модуля в отдельном потоке, чтобы принимать и обрабатывать сообщения в бесконечном цикле.

Для иллюстрации такого подхода разработан программный комплекс, выполняющий оптимизацию генетическим алгоритмом одновременно двумя процессами. Эти процессы через заданные временные интервалы передают друг другу сведения о наилучшей своей особи в терминах генетического алгоритма [7]. Полученными от другого процесса сведениями они заменяют свой наихудший экземпляр популяции. Процессы могут быть запущены на двух разных компьютерах в локальной сети. Применены такие методы модуля Winsock, как socket (создание сокета), bind (привязывание сокета к адресу), SendTo (отправка сообщения), recv (получение сообщения), и др.

Заключение

Для возможности программирования обработки данных на носителе информации, кроме изучения технологий баз данных, требуется изучение функций для работы с файлами и каталогами, компонентов среды разработки для отображения содержимого каталога, для вывода различных диалогов и вспомогательных компонентов. Важно также овладение навыками организации межпроцессной передачи данных, в частности, использования протокола UDP.

1. Delphi Sources. Форум. URL: <http://www.delphisources.ru/forum/showthread.php?t=13625> (дата обращения 29.03.2018)
2. Дмитриев А.П. Заполнение семантической базы данных автоматизированного переводчика // Компьютерные технологии и моделирование: сб. науч. тр. Вып. 9. Чебоксары, 2013. С. 77-79.
3. Дмитриев А.П. Чувашско-русский переводчик: программная реализация // Прикладная информатика. 2011 г., № 6 (36), ноябрь-декабрь. С. 43-46.
4. Дмитриев А.П. База данных, применяемая в программе чувашско-русского перевода // Мироновские чтения сборник статей. Чебоксары, 2011. С. 45-51.
5. Дмитриев А.П. Модификация метода шифрования для сокращения потерь данных при их повреждении // Фундаментальные исследования. 2017. № 11-1. С. 58-62.
6. Зверева Е.А., Лавина Т.А. Производственная практика как условие формирования ИКТ-компетентности бакалавров направления «Приборостроение» // Вестник Чувашского государственного педагогического университета им. И.Я. Яковлева. 2017. № 4 (96). С. 125-132.
7. Каширина И. Л. Введение в эволюционное моделирование: учеб.пособие. Воронеж: ВГУ, 2007. – 40 с.