УДК 681.518

М. И. ГЛАВЧЕВ, канд. экон. наук, НТУ "ХПИ" (г. Харьков),
О. С. ПАНИЧЕВ, НТУ "ХПИ" (г. Харьков)

ФОРМИРОВАНИЕ САМООБРАБАТЫВАЮЩИХСЯ ФАЙЛОВ

У статті розглянута можливість побудови файлів, які здатні самостійно оброблювати інформацію що зберігається у них самих. Розкрито особливості формування програмного коду та вказані напрямки використання цих файлів.

The article is considered possibility of files building, which capable by itself to process information, which is saved in them themselves, open particularities of the shaping the programmed code and is specified directions of the use of these files.

Постановка проблемы. В настоящее время в связи с увеличением объемов информации, хранящейся на различных накопителях и передаваемой по системам передачи данных, особенно актуальным стал вопрос сжатия информации (архивирования). Сжатие позволяет сэкономить место на диске и снизить объемы передаваемой информации, а, следовательно, уменьшить время передачи этой информации и преодолеть некоторые физические ограничения (например, пропускную способность канала).

Одновременно с этим для повышения защищенности хранимой и передаваемой информации может применяться ее шифрование. Для беспрепятственного дешифрации такой информации необходимо знать ключ (пароль), с которым производилось шифрование, а также метод шифрования информации.

На сегодняшний день существует огромное количество методов сжатия и шифрования, которые применяются как отдельно друг от друга, так и в сочетании друг с другом. При этом алгоритм распаковки (дешифрации) данных зависит от алгоритма их сжатия (шифрования), т. е. для распаковки информации, обязательно нужно знать метод, с помощью которого происходило сжатие. Это создает определенную проблему, особенно при передаче данных, так как приемник может не иметь необходимых алгоритмов.

В качестве решения как описанной выше, так и некоторых других проблем, могут быть использованы файлы с возможностью самообработки (самообрабатывающиеся файлы), т. е. файлы, которые в своем составе содержат обрабатывающий их модуль.

Анализ литературы. Наиболее широкое применение самообрабатывающиеся файлы получили в системах сжатия данных (архиваторах, инсталляционных менеджерах и т.д.) [1, 2].

Среди основных преимуществ использования самообрабатывающихся

файлов выделяют следующие:

- возможность для пользователя использовать привычный для него интерфейс операционной системы, не касаясь интерфейса архиватора;
- возможность выбора интерфейса (диалоговый режим, командная строка);
- простота использования (в большинстве случаев необходимо лишь указать путь для распаковки данных);
- возможность отделить обрабатывающий модуль, получив обыкновенный архив, и, наоборот, присоединить к архиву модуль, получив самообрабатывающийся архив.

В MS-DOS для обеспечения корректной совместной работы дополняемого модуля (ДМ), который добавляется к основному модулю (ОМ) для организации полноценного запуска исходного файла, необходимо:

- сформировать ДМ в виде .COM- или .BIN-файла;
- поместить ДМ в конец дополняемого .EXE-файла;
- в заголовке (header) основного модуля исправить поля:
  – длину загружаемой части в 512-байтных страницах;
  – точки входа CS и IP (чтобы они указывали на ДМ);
- в ДМ обеспечить возврат на старую точку входа ОМ.

При работе с современными операционными системами обрабатывающий модуль обычно помещается в начало дополняемого файла [3, 4]. Для обработки sfx-архива, например, в WinRAR требуется пропустить sfx-модуль и найти в архиве блок-маркер [4].

Блок-маркер (MARK_HEAD) имеет следующую структуру:

HEAD_CRC (2 байта)	Всегда 0x6152
HEAD_TYPE (1 байт)	Тип заголовка: 0x72
HEAD_FLAGS (2 байта)	Всегда 0x1a21
HEAD_SIZE (2 байта)	Размер блока = 0x0007

Блок-маркер в действительности считается фиксированной последовательностью байт: 0x52 0x61 0x72 0x21 0x1a 0x07 0x00.

Целью работы является разработка новой системы создания самораспаковывающихся архивов с минимальным дополнительным обрабатывающим модулем.

Основная часть. Самообрабатывающиеся файлы рассмотрим на примере самораспаковывающихся архивов (sfx-архивы), т.е. архивов, для распаковки которых не требуется наличие программы-архиватора, с помощью которой архив был создан.

Для этого к файлу архива добавляется (обычно, в начало) обрабатывающий модуль, который содержит:

- все алгоритмы, использующиеся для распаковки архива;
- необходимые для них структуры данных;
- графический интерфейс.

Основное требование, предъявляемое ко всем методам сжатия – наибольший коэффициент сжатия (отношение размера файла до сжатия к размеру этого файла после сжатия). А при небольшом размере архива размер обрабатывающего модуля составляет значительную часть общего размера, значительным образом влияя на общий размер. Поэтому основное условие, которому должен удовлетворять обрабатывающий модуль – он должен иметь минимальный размер.

Весомую часть приложения обычно занимает графический интерфейс приложения. Поэтому для уменьшения размера модуля, графический интерфейс необходимо свести к минимуму. В рассматриваемом примере самообрабатывающихся файлов графический интерфейс отсутствует вообще и используется консольный режим работы, т.е. ввод/вывод осуществляется в командной строке. При этом размер модуля был уменьшен на 247 808 байт.

Для дальнейшего уменьшения размера модуля была создана новая библиотека *classes*, в которую из стандартной библиотеки *classes* Delphi [5, 6], подключаемой к модулю, были скопированы только те константы и классы, которые используются в модуле. Так из модуля удалено 72 192 байта кода.

На следующей схеме (рисунок) показан размер модуля на описанных выше стадиях.

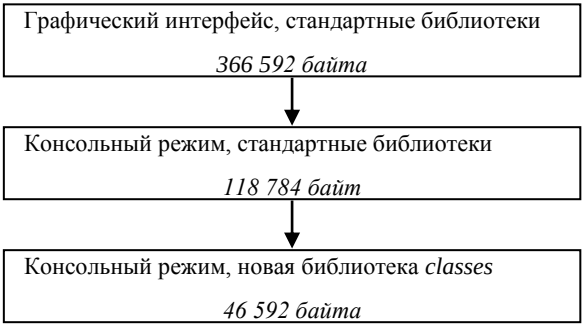

Рис.

Таким образом, имеем обрабатывающий модуль, содержащий все алгоритмы, использующиеся для распаковки архива, необходимые для них структуры данных и вспомогательные подпрограммы, работающий в консольном режиме и имеющий размер 46 592 байта.

Как было указано выше, обрабатывающий модуль помещается в начало архива, который он обрабатывает. Изначально модуль находится в директории вместе с программой-архиватором в виде отдельного файла. При создании самораспаковывающегося архива содержимое этого файла копируется в новый файл-архив, после чего к нему дописывается остальная часть архива. При запуске архива начинает работу основная функция обрабатывающего модуля, которая поочередно вызывает необходимые

подпрограммы.

Алгоритм роботы следующий. В командной строке вводится имя будущего распакованного файла. После этого из строки параметров запуска архива читается нулевой параметр, который содержит собственное имя файла архива. При помощи этого имени файл архива открывается для чтения (его необходимо открывать как файл потока, так как он запущен).

Первые N байт файла – обрабатывающий модуль, где N – размер этого модуля в байтах – определяется после компиляции модуля и вносится в его исходный текст в виде константы, после чего модуль перекомпилируется. Поэтому N читается из константы, и первые N байт модуля пропускаются.

Следующие три байта – контрольный маркер начала архива (соответствуют строке “rar”). Этот маркер читается и проверяется. Если его значение неправильное – произошла ошибка (поврежден архив, либо сам обрабатывающий модуль; последний имеет неверный размер и т.д.), выводится сообщение об ошибке, выполнение программы прерывается.

В случае, если маркер имеет верное значение, вызываются подпрограммы распаковки и начинается распаковка архива.

Выводы. Описанная в данной статье система создания самораспаковывающихся архивов позволяет сжимать файлы, добавляя к созданным архивам обрабатывающий модуль. Добавление такого модуля увеличивает размер файла на 46 592 байта, однако дает возможность распаковать архив без наличия архиватора, так как содержит все алгоритмы и структуры данных, необходимые для распаковки архива. Это решает проблему переносимости файлов, так как машине-приемнику не обязательно иметь алгоритмы распаковки или дешифрации информации.

При дальнейшем совершенствовании алгоритмов сжатия и уменьшении размеров обрабатывающего модуля область применения самообрабатывающих файлов будет расти.

Возможно также применение этой методики для создания самораскодирующихся и саморасшифровующихся файлов [7, 8], что представляет широкие возможности применения приведенного алгоритма.

Список литературы: **1.** *Ватолин Д., Ратушняк А., Смирнов М., Юкин В.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с. **2.** *Кричевский Р.Е.* Сжатие и поиск информации. – М.: Радио и связь, 1989. – 168 с. **3.** Programming manual for Winzip. – Документ http://winzip.com. **4.** Техническое описание WinRAR v3.2. – Документ http://www.rarlab.com. **5.** *Гофман В.* Delphi 6 в подлиннике. – К.: BHV, 2001. – 1152 с. **6.** *Карпов В.Н.*. Delphi: специальный справочник. – Санкт-Петербург: Изд. Питер, 2001. – 688 с. **7.** *Домашев А.В., Попов В.О., Правиков Д.И., Прокофьев И.В., Щербаков А.Ю.* Программирование алгоритмов защиты информации. – М.: Нолидж, 2000. – 288 с. **8.** *Scheneier B.* Applied Criptography. – John Wiley & Sons, 1996. – 758 p.

Поступила в редакцию 02.10.2004