УДК 004.457

РАЗРАБОТКА УПРАВЛЯЕМОЙ СБОРКИ ВЗАИМОДЕЙСТВИЯ ДЛЯ ОТЛАДОЧНОГО ЯДРА ОПЕРАЦИОННОЙ СИСТЕМЫ MICROSOFT WINDOWS

Баринов С.С., Шевченко О.Г.
Донецкий национальный технический университет
кафедра «Компьютерная инженерия»
E-mail: barynov@gmail.com

Аннотация
Баринов С.С., Шевченко О.Г. Разработка управляемой сборки взаимодействия для отладочного ядра операционной системы Microsoft Windows. *Изучен отладочный стек отладчика Windows Debugger и определены низкоуровневые интерфейсы для доступа к отладочному ядру, на которые он опирается. Написана сборка взаимодействия, позволяющая использовать отладочное ядро из управляемого кода и построить собственный отладчик ядра. Для тестирования сборки взаимодействия создана демонстрационная программа.*

Основная часть:

Работа отладчика Windows Debugger опирается на динамически подключаемую библиотеку DbgEng.dll, представляющую собой отладочное ядро (debugging engine) операционной системы и дающую возможность манипулировать отлаживаемой системой. В свою очередь, отладочное ядро использует вспомогательную отладочную библиотеку DbgHelp.dll. Обе библиотеки включены в состав современных версий операционной системы Windows.

Взаимодействие с отладочным ядром осуществляется посредством COM-технологии (Component Object Model). Модель компонентных объектов — это объектно-ориентированная, основанная на интерфейсах, программная архитектура. Объект в COM после создания предоставляет свою функциональность вызвавшему процессу, а после использования — уничтожается.

Пакет «Debugging Tools for Windows» содержит набор статических библиотек для связывания исполняемого файла с функциями библиотеки отладочного ядра, а также ряд заголовочных файлов, описывающих COM-интерфейсы и другую необходимую для использования отладочного ядра информацию. Заголовочные файлы и статические библиотеки присутствуют для архитектур процессоров IA-32, AMD64 и IA-64. Выступая в роли COM-сервера, ядро описывает ряд интерфейсов, таких как IDebugClient, IDebugControl, IDebugRegisters, и функцию DebugCreate, выполняющую роль точки входа: с помощью нее можно, указав глобально уникальный идентификатор интерфейса (GUID), получить экземпляр базового класса. Этот экземпляр в дальнейшем дает возможность запросить остальные экземпляры.

Технология COM позволяет не только создавать объекты COM на произвольном языке программирования, но и сами объекты могут быть доступны из любого совместимого с технологией COM языка. Эта, безусловно, положительная характеристика архитектуры COM имеет и негативный оттенок — невозможно учесть особенности и возможности каждого совместимого с технологией COM языка. Поэтому для удобного использования COM-объектов из конкретного языка программирования возникает необходимость в создании набора высокоуровневых классов. Эти классы призваны полностью инкапсулировать работу с COM-объектами и в полной мере реализовать возможности данного языка, такие как: использование исключений для обработки возникающих ошибок (обработка ошибок при работе непосредственно с технологией COM приводит к усложнению кода), подключение

обработчиков событий, управление памятью.
В случае с COM-объектами отладочного ядра целью является построение собственного отладчика, который должен упростить процесс отладки режима ядра, снизив порог вхождения, который по-прежнему высок для отладчика Windows Debugger. Для этого разрабатываемый отладчик должен иметь современный дружественный графический интерфейс пользователя. Для операционной системы Microsoft Windows на данный момент таким требованиям к построению интерфейса более всего отвечает графическая подсистема Windows Presentation Foundation (WPF), входящая в Microsoft .NET Framework начиная с версии 3.0.
Программы, написанные с использованием инфраструктуры .NET Framework, содержат т.н. управляемый код. Этот термин введен компанией Microsoft для именования байт-кода, интерпретируемого общеязыковой исполняющей средой (Common Language Runtime, CLR), входящей в инфраструктуру .NET Framework. В противовес этому, классический программный код, представляющий собой набор инструкций для конкретной архитектуры процессоров и не требующий наличия CLR, называется неуправляемым. Программы используют набор классов из библиотеки базовых классов (Framework Class Library, FCL) или сторонние библиотеки управляемого кода, называемые сборками. Таким образом, для работы с отладочной средой необходима соответствующая сборка, инкапсулирующая работу с COM-объектами в соответствии с описанным шаблоном и называемая сборкой взаимодействия.
Компания Microsoft не поставляет сборку взаимодействия с отладочным ядром Windows. В открытом доступе существует сторонний проект такой сборки под названием mdbglib, однако он находится в зачаточном состоянии и не развивается. Кроме того, проект ориентирован на отладку приложений пользовательского режима.
CLR не позволяет напрямую вызывать неуправляемый код, однако включает в себя специальный механизм взаимодействия под названием Platform Invocation Services (обычно упоминаемый как P/Invoke). Этот механизм осуществляет маршалинг данных — преобразование и передачу данных между управляемой и неуправляемой памятью. COM-интерфейс описывается на управляемом языке программирования (в данном случае C#) и снабжается дополнительными атрибутами, которые помогают механизму взаимодействия правильно передавать данные. При вызове метода COM-интерфейса механизм взаимодействия на основе правил по умолчанию и дополнительных атрибутов устанавливает соответствие между управляемыми и неуправляемыми типами данных и в зависимости от ситуации может: копировать типы, передаваемые по значению, в стек, помещать данные в неуправляемую кучу и по окончанию выполнения метода копировать результат в управляемую кучу, закреплять данные в управляемой куче и передавать ссылку на них через стек.
Для автоматической генерации описания интерфейсов COM-объектов на управляемом языке и снабжения этого описание необходимыми дополнительными атрибутами существует утилита tlbimp. Для работы утилите требуется описание интерфейсов COM-объектов на специальном языке IDL. Проблема заключается в том, что для библиотеки DbgEng.dll компания Microsoft не предоставила описание типов на языке IDL. Поэтому интерфейсы были описаны вручную, хотя этот путь трудоемок и предрасполагает к ошибкам.
Все описанные интерфейсы являются внутренними для сборки и не видимы за ее пределами. Вместо этого экземпляры COM-объектов, реализующих эти интерфейсы, являются закрытыми членами высокоуровневых классов и создаются внутри них. Это абстрагирует программиста от вопросов COM-взаимодействия и позволяет сосредоточиться на использовании функциональных возможностей сборки. В общем случае, при описании высокоуровневых классов использовалась следующая методика:

- методы GetX и SetX заменялись на одно свойство, доступное для чтения и записи;

- методы, имеющие различные имена из-за невозможности перегрузки, приводились к одному имени;
- структуры, содержащие служебную информацию, такую как размер строки или количество элементов массива, заменялись упрощенными версиями, поскольку для управляемого кода эта информация является избыточной;
- при необходимости многократного вызова метода с индексом в качестве параметра для запроса списка элементов применялись коллекции;
- целые числа без знака по возможности заменялись числами со знаком для соответствия со стандартной системой типов (Common Type System), что дает возможность использовать сборку в управляемых языках, не поддерживающих числа без знаков, таких как VB.NET;
- интерфейсы обратного вызова, предназначавшиеся для уведомления программы об изменениях в отладочном ядре, реализовывались закрытыми подклассами, которые вызывали соответствующие обработчики языковых событий;
- имена методов приводились в соответствие с правилами именования.

Набор COM-интерфейсов представляет собой функционально мощную, но громоздкую в использовании систему. Для построения простейшей программы, которая должна инициализировать отладочное ядро, подключаться к отлаживаемой системе, получать уведомления о возникающих событиях и контролировать выполнение, требуется самостоятельно создать несколько объектов и подготовить их. Чтобы предоставить интерфейс более простого уровня, который объединяет другие сущности, однако не скрывает их полностью, применен шаблон проектирования «фасад». Класс-фасад дает возможность получить базовый функционал простым способом, однако в случае необходимости настроить работу отладочного ядра под свои нужды.

Выводы

На данный момент в сборке взаимодействия реализованы возможности:

- подключение к удаленной системе;
- приостановка и возобновление работы удаленной системы по требованию;
- получение уведомлений о событиях (изменение состояния регистров, памяти, возникновении исключений и прерываний и др.);
- просмотр и изменение всех регистров процессора (включая системные регистры).

Созданная сборка упрощает взаимодействие с отладочным ядром и является основой для построения отладчика ядра с богатым графическим интерфейсом. Помимо главного своего предназначения, сборка может быть использована в скриптах оболочки Windows PowerShell, что позволит автоматизировать некоторые операции отладки.

Литература

1. Microsoft Corporation. MSDN Library. Debugger Engine and Extension APIs [Електронний ресурс] — Електрон. дан. — 2010. — Режим доступу: http://msdn.microsoft.com/en-us/library/ff540525(v=vs.85).aspx

2. Juval Lowy. COM & .NET Component Services. — O'Reilly Media, 2001 г. — 384 с: ил.

3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 386 с.: ил. (Серия «Библиотека программиста»)

4. Руссинович М. и Соломон Д. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP и Windows 2000. Мастер класс. / Пер. с англ. — 4-е изд. — М.: Издательско-торговый дом «Русская редакция»; СПб.: Питер; 2005. — 992 стр.: ил.