1

УДК 004.457

СРЕДСТВА ОТЛАДКИ РЕЖИМА ЯДРА ОПЕРАЦИОННОЙ СИСТЕМЫ WINDOWS

Баринов С.С., Шевченко О.Г.
Донецкий национальный технический университет

Приведен сравнительный анализ отладки режима пользователя и режима ядра применительно к операционной системе Microsoft Windows, выделены отличия и проблемы организации отладки последнего. На основе полученных результатов сформулированы основные требования к построению отладчиков режима ядра в случае аварийной и интерактивной отладки. Проведен анализ существующих решений на предмет соответствия требованиям. В частности, особое внимание уделено отладчику Microsoft Windows Debugger.

Отладка – это процесс определения и устранения причин ошибок в программном обеспечении (ПО). В некоторых проектах отладка занимает до 50 % общего времени разработки [1]. Отладка может быть значительно упрощена при использовании специализированных инструментов, которые постоянно совершенствуются. Основным таким инструментом является отладчик, позволяющий контролировать выполнение ПО, наблюдать за его ходом и вмешиваться в него. Средства отладки ядра преимущественно используются разработчиками драйверов [2].

Инструментарий разработки прикладного программного обеспечения предлагает программисту широкий спектр возможностей. Любая интегрированная среда разработки включает в себя и возможность отладки без необходимости использования сторонних утилит. Если же речь идет о системном программном обеспечении и разработке драйверов в частности, то в силу его специфики процесс разработки чрезвычайно затруднен и мало автоматизирован. Все фазы разработки, в числе которых и отладка,

являются раздельными. Для проведения каждой из них требуются особые условия: написание программного кода выполняется на полноценной компьютерной системе, отладка – на отладочной системе, тестирование – в зависимости от обстоятельств и т.д. Сам же отладчик режима ядра более сложен в освоении и, соответственно, менее дружественен.

В целом можно говорить о недостатке средств отладки ядра. Хотя таковые средства имеются в наличии, зачастую говорить об альтернативах не приходится. Например, отладчик Microsoft Windows Debugger имеет слишком высокий порог вхождения. Многие программисты говорят о первом негативном опыте при знакомстве с ним, а большинство его возможностей остаются незатребованными.

Исходя из структуры виртуального адресного пространства, если в приложении допущена ошибка, вследствие которой приложение выполнит запись данных в произвольное место памяти, то приложение повредит только собственную память и не повлияет на работу других приложений и операционной системы. Тогда как программный код режима ядра в состоянии повредить важные структуры данных операционной системы, что неминуемо приведет к общему сбою. Неэффективно написанный драйвер также может стать причиной серьезной деградации всей операционной системы.

Современные отладчики обеспечивают следующие базовые функции:

- отладка на уровне исходного кода;
- управление выполнением;
- просмотр и изменение памяти;
- просмотр и изменение содержимого регистров процессора;
- просмотр стека вызовов.

Для облегчения работы с дизассемблированным кодом применяются т.н. отладочные символы. Во время работы компоновщика, кроме образа исполняемого файла также может быть создан файл данных, содержащий информацию, которая не требуется при выполнении программы, но чрезвычайно полезна

при ее отладке: имена функций, глобальных переменных, описание структур. Отладочные символы доступны для всех исполняемых файлов операционной системы Windows [3].

Под управлением выполнения подразумевается способность прерывать и возобновлять выполнение программного кода по достижению заданной команды в программном коде. Если программный код исполняется в пошаговом режиме – прерывание возникает для каждой лексемы языка программирования или при выходе из подпрограммы. При свободном исполнении прерывание выполнения возникает в заранее оговоренных участках кода – местах, в которых установлены точки останова.

При прерывании кода режима ядра возникают следующая дилемма. Отладчик для взаимодействия с программистом использует интерфейс пользователя. Т.е. как минимум видимая часть отладчика выполняется в пользовательском режиме и для его построения естественно используется интерфейс прикладного программирования (Windows API), который в свою очередь опирается на модули режима ядра. Таким образом, приостановка кода режима ядра может привести к взаимной блокировке: система перестанет отвечать на запросы пользователя.

Для доступа к памяти ядра составные части отладчика также должны выполняться в режиме ядра. Это ведет к возникновению сразу двух проблем, являющиеся очевидным следствием организации памяти в защищенном режиме процессора.

Первая проблема касается трансляции виртуальных адресов памяти. Драйвера постоянно взаимодействуют с приложениями пользовательского режима, выполняя обращение к их памяти. Операционная система Windows транслирует виртуальные адреса в физические, руководствуясь понятием контекста потока. Контекст потока – структура, отражающая состояние потока и включающая, в частности, набор регистров и некоторую другую информацию. Когда управление передается другому потоку, возникает переключение контекста, при котором сохраняется информация об одном потоке и восстанавливается информации о другом. При переключении

контекста потока на поток другого процесса переключается также и каталог страниц, используемый для трансляции виртуальных адресов в физические [2].

Особенность заключается в том, что при диспетчеризации системных вызовов операционная система Windows не переключает контекст. Благодаря этому код режима ядра может использовать виртуальные адреса пользовательского режима.

Иначе дело обстоит при диспетчеризации прерываний или выполнении системных потоков. Прерывание может произойти в любой момент, поэтому нельзя предугадать, какой контекст потока будет использоваться. Системные же потоки не принадлежат какому-либо процессу и не могут транслировать виртуальные адреса пользовательского режима [4]. Отсюда следует, что в этих ситуациях нельзя обратиться к памяти пользовательского режима.

Второй проблемой является обращение к перемещаемой памяти. Большая часть информации в памяти является перемещаемой и в любой момент может быть перемещена из физической памяти на жесткий диск в страничный файл. Если обратиться к странице, которая отсутствует в физической памяти, в нормальной ситуации процессор сгенерирует прерывание Page Fault, которое будет обработано диспетчером памяти, и в результате страница будет прочитана из страничного файла и загружена в физическую память.

Описанное поведение нарушается, если программный код отладчика вынужден использовать высокий уровень запросов прерываний (interrupt request levels, IRQL). При IRQL, совпадающем с или превышающем IRQL диспетчера памяти последний не сможет загрузить отсутствующую страницу, т.к. операционная система будет блокировать прерывание Page Fault. Это приведет к краху операционной системы [5].

Отладку принято разделять на интерактивную и аварийную. При интерактивной локальной отладке отладчик выполняется в той же системе, что и объект отладки. При интерактивной удаленной отладке отладчик и объект отладки выполняется в разных системах.

При отладке кода ядра система должна контролироваться, начиная с первых этапов ее загрузки, когда сеть еще не функционирует, поэтому для связи систем применяют простые последовательные интерфейсы, такие как COM, FireWire, USB. В последнее время, благодаря тенденциям развития виртуализации ПО на разных уровнях абстракций, все чаще привлекают виртуальные машины. Гостевая ОС выступает в качестве отлаживаемой, размещенная ОС включает интерфейс пользователя отладчика.

Интерактивная отладка удобна и применима не во всех случаях. Хотя основные версии операционной системы Windows не являются системами реального времени, некоторые драйвера управляют высокоскоростными устройствами (драйвера шин, мультимедиа-устройства), следовательно, вмешательство в их работу не позволит проверить работоспособность в реальной ситуации. Основное же поле применения аварийной отладки – выявление ошибок в отправленном заказчику ПО. Взаимодействие с заказчиком строится по следующей схеме:

- возникновение сбоя;
- сохранение состояния системы в момент сбоя;
- отправка данных о состоянии разработчикам для анализа [6].

Таким образом, для аварийной отладки не требуется установка средства отладки на тестовом компьютере. В дистрибутив операционной системы Windows включены механизмы для реализации аварийной отладки. Перед перезагрузкой операционная система может сохранять информацию о своем состоянии, которую разработчик может проанализировать и выяснить причину. Такая информация, сохраненная в файл, называется дампом памяти.

Основные средства отладки режима ядра предоставляются самим производителем операционной системы Windows в рамках свободно распространяемого пакета «Debugging Tools for Windows». Средства включают графический и консольный отладчики WinDbg и KD соответственно (далее Windows Debugger). Работа этих отладчиков опирается на механизмы, предусмотренные

разработчиками операционной системы и заложенные в ее ядре.

Основным режимом для Windows Debugger является режим интерпретатора команд. Благодаря модульной структуре, наряду с поставляемыми разработчиками командами Windows Debugger поддерживает сторонние модули, называемыми расширениями. В действительности большинство встроенных команд также оформлено в виде расширений.

Windows Debugger ориентирован на удаленную интерактивную и аварийные отладки, при использовании которых раскрываются все его возможности. В тоже время полноценная локальная интерактивная отладка не поддерживается: отладчик позволяет только просматривать некоторые структуры ядра.

Существует модуль расширения для Windows Debugger под названием LiveKD, созданный Марком Руссиновичем, который в некотором смысле реализует локальную интерактивную отладку. LiveKD на ходу создает дамп памяти рабочей системы и использует его для отладки.

Пакет инструментов «Debugging Tools for Windows» регулярно обновляется и поддерживает все современные операционный системы Windows.

Отладчик ядра SoftICE, выпускавшийся компанией Compuware в пакете программ DriverStudio, традиционно выступал альтернативой пакету «Debugging Tools for Windows». Отличительной чертой SoftICE являлась реализация локальной интерактивной отладки на поддерживаемом аппаратном обеспечении. Отладчик практически полностью мог контролировать работу операционной системы.

С 3 апреля 2006 года продажа продуктов семейства «DriverStudio» была прекращена по причине «множества технических и деловых проблем, а также общего состояния рынка». Последней версией операционной системы, поддержка которой была реализована, является Windows XP Service Pack 2. Как правило, пакеты сервисных обновлений не изменяют прикладной интерфейс операционной системы, но номера системных вызовов и другая

1

недокументированная информация может претерпевать изменение. Отладчик SoftICE опирался на жестко-прописанные адреса внутренних структур данных. Как следствие – с выходом Service Pack 3 совместимость была нарушена. Очевидно, что более поздние версии операционной системы Windows также не поддерживаются.

Syser Kernel Debugger создан небольшой китайской компанией Sysersoft как замена отладчику SoftICE. Первая финальная версия была выпущена в 2007 году. Как и SoftICE, Syser Kernel Debugger способен выполнять интерактивную отладку на работающей системе. Поддерживаемыми являются только 32-разрядные редакции современных версий Windows.

На данный момент Windows Debugger является основным инструментом среди разработчиков модулей ядра. Его также использует команда разработчиков ядра операционной системы Windows.

Литература

[1] Макконнелл С. Совершенный код. Мастер-класс / Пер. с англ. – М.: Издательский дом «Русская редакция»; СПб.: Питер, 2005. – 896 стр.: ил.

[2] Руссинович М., 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.

[3] Microsoft Corporation. MSDN Library. Debug Help Library [Електронний ресурс] – Електрон. дан. – 2010. – Режим доступу: http://msdn.microsoft.com/en-us/library/ms679309(v=VS.85).aspx

[4] Солдатов В.П. Программирование драйверов Windows. Изд. 2-е, перераб. и доп. – М.: ООО “Бином-Пресс”, 2004 г. – 480 с: ил.

[5] Microsoft Corporation. MSDN Library. Windows Driver Kit [Електронний ресурс] – Електрон. дан. – 2010. – Режим доступу: http://msdn.microsoft.com/en-us/library/ff557573(v=VS.85).aspx

[6] Mario Hewardt, Daniel Pravat. Advanced Windows debugging. – Addison-Wesley, 2007 г. – 840 с.