УДК 681.3.06

*В.А. Лавров*

# ИСПОЛЬЗОВАНИЕ MIRROR ДРАЙВЕРА ДЛЯ ЗАПИСИ ВСЕХ ИЗМЕНЕНИЙ ЭКРАНА КОМПЬЮТЕРА В ФАЙЛ ВИДЕОФОРМАТА FBR

Системы для записи всех изменений экрана в файл используются во множестве задач. Такие задачи можно разбить на три класса – создание демонстрационных видеороликов, тестирование программных продуктов и служба поддержки. К различным классам предъявляются разные требования. Но существует, по крайней мере, одно требование, общее для всех классов – программа записи не должна снижать общую производительность компьютера. В статье предлагается использовать mirror видеодрайвер для быстрого сохранения изменений при минимизации общего объема сохраняемой информации. Предложенный способ реализован в коммерческом продукте BB FlashBack, продающемся на рынке через Интернет.

Существует, по крайней мере, два широко распространенных способа захвата видеоинформации с экрана компьютера. Оба способа основаны на периодическом захвате содержимого всего экрана с последующей компрессией и сохранением в файл. Первый способ использует стандартную графическую подсистему Windows GDI, а второй способ использует функции мультимедиа библиотеки DirectX.

Эти механизмы обладают двумя существенными недостатками. Изменения экрана между двумя захватами теряются, а так как передача данных из памяти видеокарты в системную память компьютера в несколько раз медленнее, чем передача данных из системной памяти в системную память, происходит существенное уменьшение общей производительности компьютера во время записи.

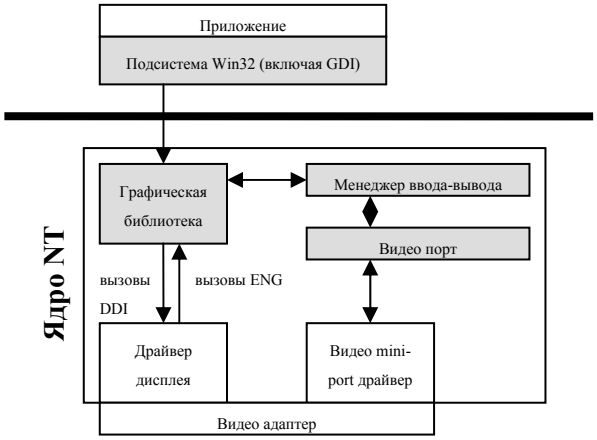
Существует ряд улучшений вышеописанных способов, основанных на особенностях ОС Windows. Например, в программе WinVNC [5] используются специальные хуки для определения областей экрана, которые подверглись изменению. Это позволяет существенно уменьшить количество данных, пересылаемых из памяти видеокарты в память компьютера. Но, к сожалению, данный способ не дает 100% гарантии на обнаружение всех изменений экрана.

Для детектирования изменений экрана можно воспользоваться специальным свойством всех ОС Windows, начиная с версии 2000, которое дублирует все вызовы графической подсистемы на mirror видеодрайвер.

Автором был написан собственный mirror драйвер, работающий в нулевом кольце защиты ОС, а также записывающее Win32 приложение, которое использует mirror драйвер для захвата изменений экрана.

## СТРУКТУРА ВИДЕОДРАЙВЕРА

Следующая схема показывает все необходимые компоненты, требуемые для отображения информации под управлением Windows 2000 и выше [3, 4, 6].




Компоненты, выделенные серым цветом, поставляются вместе с операционной системой и они не зависят от типа видеооборудования и от видеодрайверов.

Компоненты белого цвета поставляются сторонними производителями.

Эта схема работает следующим образом. Приложение, работающее в User Mode, пытается вывести на экран некоторую информацию. Все графические функции в User Mode выполняются библиотекой *GDI32.DLL* подсистемы Win32. Библиотека *GDI32.DLL* не может работать с видеодрайвером напрямую, так как она (как и приложение) работает в третьем кольце защиты. Поэтому для обработки запроса происходит переход в нулевое кольцо, в котором управление передается в графическую библиотеку GDI в Kernel Mode.

Библиотека Kernel Mode GDI имеет информацию о том, какие из всего набора DDI (Device Driver Interface) функций поддерживаются драйвером дисплея на аппаратном уровне. Если затребованная функция поддерживается, то библиотека GDI вызывает соответствующую DDI функцию в драйвере дисплея. Если требуемая функция не поддерживается, то тогда вызывается программная эмуляция функции. Библиотека Kernel Mode GDI может программно выполнить любую DDI-функцию, она имеет весь необходимый для этого код.

Если драйвер дисплея поддерживает DDI-функцию не полностью, то он может вызывать любую из ENG функций, входящих в состав Kernel Mode GDI для программной эмуляции той ее части, которая не реализована на аппаратном уровне.

Например, приложение вызывает WIN API функцию *LineTo*. Управление передается в библиотеку GDI32, которая так же, как и приложение, работает в User Mode. Библиотека User Mode GDI использует программное прерывание INT или команду SYSENTER для передачи управления в библиотеку Kernel Mode GDI. Kernel Mode GDI просматривает таблицу поддерживаемых драйвером дисплея DDI-функций. Если в этой таблице имеется точка входа для DDI-функции *DrvLineTo*, то библиотека GDI вызывает эту функцию в драйвере дисплея. Если драйвер дисплея по каким-то причинам не может выполнить эту функцию, то он может воспользоваться ее программной эмуляцией, имеющейся в библиотеке Kernel Mode GDI – *EngLineTo*. После того, как линия нарисована, управление передается обратно в библиотеку Kernel Mode GDI. Эта библиотека осуществляет обратный переход в третье кольцо защиты и передает управление User Mode GDI, которая возвращает управление приложению, вызывавшему API-функцию *LineTo*.

## MIRROR ДРАЙВЕР

Mirror драйвер это видеодрайвер для виртуального устройства, который отображает графические операции

одного и более дополнительных физических устройств. Он должен быть написан как любой другой видеодрайвер, однако часть этого драйвера – miniport – содержит только минимальный набор функций, который требуется для реального видеодрайвера.

Любой видео miniport драйвер для семейства Windows NT является Kernel Mode драйвером. Любая графическая карта должна иметь два драйвера – видео miniport драйвер и драйвер дисплея. Видео miniport драйвер работает в связке с драйвером видеопорта, который поставляется с операционной системой и является динамически загружаемым Kernel Mode драйвером. Для работы с оборудованием и системой miniport драйвер может вызывать только функции драйвера видеопорта *VideoPortXxx*. Начиная с Windows 2000 все miniport драйверы должны поддерживать технологию Plug and Play.

GDI поддерживает функцию виртуального рабочего стола и обеспечивает возможность реплицирования части этого стола на mirror драйвер. GDI реализовывает виртуальный рабочий стол как слой верхнего уровня над драйвером дисплея. Все графические операции начинаются на этом виртуальном рабочем столе. GDI отсекает и преобразовывает графические операции на конкретный физический дисплей, который существует на виртуальном рабочем столе.

Mirror драйвер может указать любой регион отсечения на виртуальном рабочем столе, включая даже такой, который разделяется между физическими устройствами вывода. GDI посылает для mirror драйвера все графические операции, которые пересекаются с регионом отсечения. Mirror драйвер может установить регион отсечения целиком покрывающий только одно физическое устройство, что гарантируют высокую эффективность отображения всех графический операций этого устройства на mirror драйвер.

Таким образом, mirror драйвер должен состоять как минимум из двух драйверов – это видео miniport драйвер и драйвер дисплея.

Как правило, функций, которые могут быть использованы этим драйверами, недостаточно для написания полезного драйвера. Поэтому к этим двум драйверам можно добавить еще один Kernel Mode драйвер, который будет выполнять все дополнительные функции, связанные с ядром операционной системы.

## РАЗРАБОТАННЫЙ MIRROR ДРАЙВЕР BB CAPTURE DRIVER

BB Capture Driver является разновидностью mirror драйвера и используется для захвата графической информации. Для написания драйвера был использован компилятор Visual C++ и набор библиотек Windows 2000 DDK (Driver Development Kit). Драйвер BB Capture Driver состоит из трех компонент:

– видео miniport драйвер bbcap.sys;

– драйвер дисплея bbcap.dll;

– драйвер для связи с ядром ОС bbchlp.dll.

Любой miniport драйвер должен включать несколько обязательных функций. Самая главная функция, которая является точкой входа, имеет следующее определение:

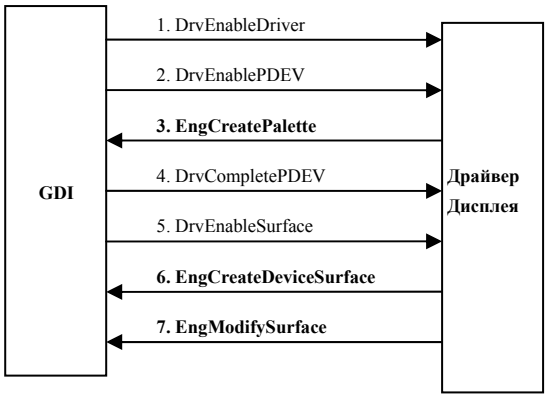
```
ULONG DriverEntry ( PVOID Context1,
PVOID Context2)
```

Это единственная функция драйвера, которая должна быть экспортирована. Все остальные функции регистрируются в специальной структуре в теле функции DriverEntry.

Драйвер дисплея bbcap.dll выполняет самую главную функцию – он перехватывает все графические операции, делает предварительную обработку данных и передает результат на дальнейшую обработку Win32 приложению.

Видео miniport драйвер и драйвер дисплея инициализируются сразу же после загрузки ядра NT и подсистемы Win32. Система загружает все те драйверы, которые указаны в реестре и затем определяет, какую из пар [miniport драйвер, драйвер дисплея] использовать. В ходе этого процесса GDI открывает все необходимые драйвера дисплея в соответствии с менеджером Windows.

Базовую процедуру инициализации драйвера дисплея можно представить в виде следующей диаграммы:




Любой драйвер дисплея экспортирует только одну функцию:

```
BOOL  DrvEnableDriver(
            IN ULONG  iEngineVersion,
            IN ULONG  cj,
            OUT DRVENABLEDATA  *pded
);
```

Все остальные функции драйвера указываются в специальной таблице поддерживаемых функций.

Когда GDI пытается создать первый контекст устройства (DC – Device Context) для видеооборудования, то GDI вызывает функцию DrvEnableDriver в драйвере дисплея. В теле этой функции драйвер заполняет структуру DRVENABLEDATA, в которой указываются все те DDI-функции, которые поддерживаются драйвером, и точки входов в эти функции.

Затем GDI вызывает функцию DrvEnablePDEV для получения описания физических характеристик устройства. В эту функцию передается структура DEVMODEW, идентифицирующая режим, который хочет выставить GDI. Если GDI запросила режим, который не поддерживается miniport драйвером или драйвером дисплея, то функция DrvEnablePDEV должна вернуть ошибку.

Функция DrvEnablePDEV инициализирует структуру PDEV. Надо сказать, что для одного и того же устройства DrvEnablePDEV может быть вызвана несколько раз, но только одна из всего множества созданных структур PDEV является активной в конкретный мо-

мент времени. Поэтому драйвер дисплея не должен использовать глобальных переменных. Все свои данные драйвер должен выделять динамически, а указатели на выделенные области хранить в структуре PDEV.

Как только инициализация прошла успешно, то GDI вызывает функцию DrvCompletePDEV. При помощи этой функции GDI передает драйверу сгенерированный дескриптор физического устройства.

На последней стадии инициализации создается поверхность для видео оборудования посредством вызова функции DrvEnableSurface. Создание поверхности разрешает вывод графической информации на физическое устройство.

После успешной инициализации драйвера дисплея GDI может вызывать любую из DDI-функций, поддерживаемых драйвером.

Так как целью драйвера дисплея bbcap является перехват всех графических операций, то в него была заложена поддержка всех возможных DDI-вызовов.

Драйвер утилит bbchlp.dll является Kernel Mode драйвером и содержит набор функций, которые используются драйвером дисплея для связи я ядром ОС.

Так как драйвер bbchlp содержит только те функции, которые необходимы для драйвера дисплея bbcap, то bbchlp не содержит точки входа, характерной для всех драйверов.

Драйвер bbchlp экспортирует следующие функции:

| Функция | Значение |
|---|---|
| init | Выделяет блок памяти в системной области из невыгружаемого на диск пула. В нем хранится структура HelperData с данными, необходимыми для передачи параметров от драйвера дисплея драйверу bbchlp и в обратном направлении. |
| done | Освобождается ранее выделенный блок памяти. |
| start | Инициализируются поля структуры HelperData. |
| stop | Очищаются поля структуры HelperData. |
| usercall-back | Производит передачу управления в приложение Win32 User Mode. |
| createand-mapmem-ory | Выделяет память из невыгружаемого на диск пула указанного размера и отображает ее на частное адресное пространство текущего процесса. Такой блок памяти доступен для чтения-записи как драйверу, так и приложению Win32 User Mode. |
| unmapand-destroy-memory | Уничтожает блок памяти, который был выделен функцией createandmapmemory. |
| querytime-stamp | Возвращает текущее значение высокоточного таймера. |
| waitevent2 | Ждет наступления события 2. |
| setevent1 | Переводит событие 1 в сигнальное состояние. |

Для использования функций утилит драйвер дисплея сначала загружает драйвер bbchlp в системную память при помощи GDI Kernel Mode функции:

```
HANDLE EngLoadImage(
          IN LPWSTR  pwszDriver
);
```

Для получения точек входов необходимых функций используется другая GDI-функция:

```
PVOID EngFindImageProcAddress(
          IN HANDLE  hModule,
          IN LPSTR  lpProcName
);
```

## WIN32 ПРИЛОЖЕНИЕ USER MODE

Win32 приложение BB FlashBack содержит весь необходимый пользовательский интерфейс, заложенный в систему. Все элементы интерфейса доступны через систему меню в системной области панели задач на рабочем столе.

Интерфейс состоит из трех частей: простой, продвинутой и профессиональной. Простой интерфейс предназначен для обычного рядового пользователя, который может начать запись, не настраивая никаких дополнительных параметров. Пользователь может начать запись, выбрав пункт Record из меню системной области.

Продвинутый интерфейс пользователя обеспечивает ряд настроек, которыми регулируются параметры записи. Например, имеется возможность переключать разрешение экрана перед началом записи и возвращать исходное разрешение после ее окончания.

Дополнительные настройки, рассчитанные на профессионального пользователя, включают в себя управление уровнем компрессии результирующего файла, типом компрессии, а также включение или отключение поддержки системы процессорных команд MMX/SSE. Эти команды используются в драйвере для оптимизации алгоритмов предварительной обработки изображений. Если пользователь заметил какие-нибудь отклонения при работе с драйвером дисплея bbcap, то он может попробовать отключить поддержку MMX/SSE и устранить обнаруженную проблему.

Взаимодействие между Win32-приложением и драйвером дисплея происходит при помощи специальной GDI-функции:

```
int DrawEscape(
      HDC hdc,// handle to DC
      int nEscape,// escape function
      int cbInput, // size of structure
      for input
      LPCSTR lpszInData    // structure
      for input
);
```

Данная функция обеспечивает выполнение специальных функций видеоадаптером, которые не поддерживаются библиотекой GDI. Библиотека Kernel Mode GDI вызывает функцию DrvDrawEscape в драйвере дисплея, если она поддерживается драйвером.

```
ULONG DrvDrawEscape(
      IN SURFOBJ  *pso,
      IN ULONG  iEsc,
      IN CLIPOBJ  *pco,
```

```
    IN RECTL  *prcl,
    IN ULONG  cjIn,
    IN PVOID  *pvIn
);
```

Таким образом, драйвер дисплея bbcap, работающий в нулевом кольце защиты, может получать команды и параметры из Win32-приложения, работающего в третьем кольце защиты, оставаясь в контексте того процесса, который вызывал функцию DrawEscape.

Драйвер поддерживает следующий набор команд:

| Команда | Значение |
|---|---|
| START | По команде START драйвер производит все необходимые действия, связанные с началом записи. Выделяется память, создаются события, инициализируются структуры, необходимые для записи. |
| PAUSE | Команда PAUSE переводит драйвер в режим ожидания. После получения этой команды драйвер перестает обрабатывать графическую информацию и не передает никаких данных в win32 приложение. |
| STOP | По команде STOP драйвер останавливает все действия по захвату и обработке графической информации, освобождает память и уничтожает все созданные события. |
| VERSION | Команда VERSION запрашивает текущую версию драйвера. Эта команда позволяет записывающей win32 программе корректно обрабатывать данные, поступающие от драйвера. |
| STORE-SCREEN | Если win32 приложению необходим весь экран в виде одно графического события формата FBR, то приложение может послать драйверу команду STORESCREEN для копирования всего экрана целиком. |
| STORE-BACK-SCREEN | По команде STOREBACKSCREEN драйвер сохраняет дополнительный буфер экрана в виде одного графического события формата FBR. |
| STOREUSEREVENT | Иногда приложению win32 необходимо встроить свое графическое событие в непрерывную цепочку событий, поступающих от драйвера. Это событие должно иметь время наступления, вписывающееся в контекст всей цепочки событий и того места, куда оно вставляется. Win32 приложение формирует событие в виде структуры и передает ее драйверу, используя команду STOREUSEREVENT. Драйвер изменяет только одно поле этой структуры – время наступления этого события. После чего структура встраивается в порядке очереди в цепочку событий и при наступлении подходящего момента передается обратно win32 приложению для записи в выходной файл. |

После начала записи драйвер производит захват графической информации. Он производит предварительную обработку данных и передает результат на дальнейшую обработку в Win32-приложение. Непосредственный обмен данными идет через специальную область памяти, которая доступна на чтение и запись и драйверу и Win32-приложению.

Win32-приложение производит обработку данных, поступивших от драйвера. Так как драйвер bbcap перехватывает все графические операции, то имеется возможность определять те из них, которые являются векторными и сохранять не двоичный результат, а описание векторной операции, что существенно уменьшает размер выходных данных.

Драйвер bbcap в состоянии определить рисование точек, линий, полигонов, а также операции сдвига изображений.

Результат обработки сохраняется в выходной файл формата FBR [1].

## ЗАКЛЮЧЕНИЕ

Написан специальный mirror драйвер bbcap и win32 приложение, использующее этот драйвер для захвата содержимого экрана. Созданная система оказывает минимальное влияние на общую производительность системы при минимизации объема сохраняемых данных.

Программа и драйвер используются в коммерческом пакете BB FlashBack, демонстрационная версия которого может быть закачена с сайта http://www.bbconsult.co.uk/.

ЛИТЕРАТУРА


1. *Лавров В.А.* Векторно-растровый формат хранения видеоизображений // Обработка данных и управление в сложных системах. Вып. 6. Томск: Изд-во Том. ун-та, 2004. С. 106–117.
2. *Лавров В.А.* Тестирование программных продуктов с использованием технологии FlashBack // Материалы VIII Всеросс. научн.-практ. конф. «Научное творчество молодежи». Томск: Изд-во Том. ун-та, 2004. 4.1. С. 46–47.
3. *Microsoft Corporation.* MSDN Library [Электронный ресурс]. 2004. Режим доступа: http://msdn.microsoft.com/, свободный.
4. *Microsoft Corporation.* Windows Driver Development Kit [Электронный ресурс]. 2004. Режим доступа: http://www.microsoft.com/whdc /devtools/ddk/default.mspx, платный.
5. *AT&T.* Virtual Network Computing. [Электронный ресурс]. 2004. Режим доступа: http://www.uk.research.att.com/vnc, свободный.
6. *David A. Solomon, Mark E. Russinovich.* Inside Microsoft Windows 2000. Third Edition // Microsoft Press.



Статья представлена кафедрой теоретических основ информатики факультета информатики Томского государственного университета, поступила в научную редакцию «Информатика» 30 апреля 2004 г.