УДК 004.454

РАЗРАБОТКА ДРАЙВЕРОВ В ОПЕРАЦИОННОЙ СРЕДЕ WINDOWS

Евстратов Е.К, Теплинский С.В.
Донецкий национальный технический университет
evstrat@net.dn.ua

Рассматривается структура драйверов WINDOWS, принципы и порядок работы с ними. Приводится описание функций работы с драйвером, представлена инструкция по его сборке и установке.

Введение

Написание драйвера дает возможность доступа ко всем системным ресурсам WINDOWS, так как его код может выполняться в привилегированном режиме процессора. Наибольший смысл это имеет, если необходим доступ к портам ввода-вывода для связи с каким-либо внешним устройством.

1 Структура драйвера

Простейшая программа драйвера состоит из следующих функций:

- регистрации;
- создания устройства;
- удаления устройства;
- обмена с устройством.

Также стандартной, но не всегда обязательной, является функция выгрузки драйвера. Для драйвера является обязательным регистрация по крайней мере одного устройства, которая производится в функции регистрации. В функцию регистрации передается ссылка на структуру объекта драйвера, которая должна заполниться на этом этапе [1]. В эту структуру входит массив адресов стандартных функций, здесь они и инициализируются для дальнейшего обращения. Наконец при регистрации задается имя драйвера, по которому к нему можно будет обратиться, оно может отличаться от имени его файла.

2 Работа с драйвером

Пользовательское приложение работает с зарегистрированным в драйвере устройством, видя его как файл. Приложение открывает устройство по имени драйвера и закрывает с помощью функций CreateFile и CloseHandle соответственно, вызовы этих функций формируют пакеты запроса ввода/вывода (Input/output Request Packet – IRP) и передают их в функции драйвера, отвечающие за подготовку к началу работы с устройством или завершению, соответственно [2]. Имя драйвера в функцию CreateFile нужно передавать в формате «\\.\имя», функция вернет дескриптор устройства в случае успеха, или код ошибки в случае неудачи. Далее обращение к драйверу будет производиться с помощью полученного дескриптора, для закрытия устройства также передается этот дескриптор.

Любой обмен данными с устройством производится при помощи функции DeviceIoControl, в которую передаются следующие параметры:

- дескриптор устройства ;
- код операции ;
- буфер входных данных и его размер (не обязательный параметр) ;
- буфер для получаемых данных и его размер (не обязательный);
- ссылка на счетчик, в который будет помещено реальное число байт, переданных драйвером (не обязательный) ;
- указатель на дополнительную структуру для асинхронного обмена (не обязательный).

Функция формирует из данных параметров IRP пакет, и передает его соответствующую

функцию драйвера. Код операции является 32-разрядным числом и состоит из четырех полей, которые упаковываются при помощи специального макроса CTL_CODE. Три поля несут информацию, обрабатываемую системой, а непосредственно код действия имеет длину 12 бит и должен быть в диапазоне 0x800 … 0xFFF, значения ниже 0x800 зарезервированы для WINDOWS, таким образом можно использовать до 2048 кодов [3].

3 Сборка драйвера

Для сборки драйвера необходимо установить пакет Windows Driver Developer Kit (WinDDK). Пакет будет запускаться как консольное приложение, для компиляции нужно зайти в папку, где расположен драйвер, и ввести команду «build». В папке с драйвером, для драйверов, которые написаны на «С», должны находиться файлы «makefile» и «sources» (без расширения). Первый файл имеет стандартное содержание и его можно и нужно скопировать из любого примера, включенного в пакет, во втором находится информация о входных и выходных файлах, например:

`TARGETNAME=driv` – имя выходного *.sys файла

`TARGETPATH=C:\WinDDK\7600.16385.1\MY _ TESTS` – папка для выходных файлов

`TARGETTYPE=DRIVER` – тип создаваемого объекта

`SOURCES=driv.c` – исходный файл с кодом драйвера

При отсутствии ошибок мы получим *.sys файл и сообщение «1 executable built», иначе ошибки будут указаны.

4 Установка драйвера

Существует два основных варианта установки – статический и динамический.

Статический – информация о драйвере заносится в реестр, а сам файл драйвера – в папку windows/system32/drivers, при этом его загрузка происходит на старте системы, а выгрузка – по завершению работы системы. Конкретно в реестре в папке «HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\» создается папка с тем же именем, что и у файла драйвера, а в ней создаются следующие ключи:

```
"Type"=dword:00000001
"Start"=dword:00000002
"ErrorControl"=dword:00000001
«Group»=»Extended Base»
```

Теперь драйвер будет запущен при следующем старте системы.

Динамический – драйвер устанавливается как сервис, через менеджер сервисов, в этом случае его также можно и удалить в любой момент времени.

Все вышеупомянутые этапы рассмотрены на примере написания драйвера некоторого псевдо - устройства. В функции разрабатываемого драйвера входило вычисление квадрата числа, переданного пользовательской программой, и возврат результата.

Был проведен эксперимент по созданию, установке драйвера, и проверке функционирования его обмена данными с пользовательским приложением. Функция обмена со стороны драйвера выглядит следующим образом:

```
NTSTATUS CtlDispatch(IN PDEVICE _ OBJECT pDeviceObject,IN PIRP Irp)
{
    PIO _ STACK _ LOCATION pIrpStack;
    PUSHORT pIBuffer,pOUT; // указатели типа unsigned short на вх. и вых. буфер
    USHORT val;
    pIrpStack = IoGetCurrentIrpStackLocation(Irp);
    pIBuffer = (PUSHORT)(Irp->AssociatedIrp.SystemBuffer);
    pOUT = (PUSHORT)(Irp->UserBuffer);
    switch (pIrpStack->Parameters.DeviceIoControl.IoControlCode)
    {
    case IOCTL _ TASK: // заданный нами код операции
       val = pIBuffer[0]; // берем данные из входного буфера
```

```
        pOUT[0] = val*val; // формируем ответ в выходной буфер
        Irp->IoStatus.Information = 2; // количество переданных байт
        break;
    }
    Irp->IoStatus.Status = STATUS _ SUCCESS; // код успешного завершения запроса
    IoCompleteRequest(Irp,IO _ NO _ INCREMENT); // завершение запроса
    return STATUS _ SUCCESS; // успешное завершение функции
};
```

Открытие драйвера приложением выглядит следующим образом:

```
hDrv = CreateFile("\\\\.\\MYDRIVER", GENERIC _ READ | GENERIC _ WRITE, 0, NULL,
                  OPEN _ EXISTING, FILE _ ATTRIBUTE _ NORMAL, NULL);
Обмен данными со стороны приложения выглядит следующим образом:
DeviceIoControl(hDrv, // HANDLE драйвера
IOCTL _ TASK, // код действия
ToDriver, // указатель на отправляемые данные (в драйвере – pIBuffer)
4, // размер буфера
FromDriver, // указатель на получаемые данные (в драйвере – pOUT)
4, // размер буфера
&cbRet, NULL);
```

Драйвер был протестирован в операционной системе WINDOWS XP.

Выводы

Написание драйвера является достаточно непростым делом, в меру сложности конкретной задачи. Драйвера могут использоваться как для доступа к аппаратным средствам компьютера, так и скрытым системным ресурсам. Был успешно проведен эксперимент по созданию и использованию драйвера.

Литература

[1] Рудаков П.И, Финогенов К.Г. Язык ассемблера: уроки программирования. – М.:Диалог-МИФИ, 2001. – 640 с.

[2] Солдатов В.П. Программирование драйверов WINDOWS, Изд. 2-е, перераб. и доп. – М.: ООО «Бином-Пресс», 2004 г. – 480 с: ил.

[3] Пишем первый драйвер. Часть 2. Электронный ресурс. Режим доступа: http://www.pcports.ru/articles/ddk3.php