



ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно
&& видно?



Напишите в чат, если есть проблемы!

Ставьте если все хорошо

Диспетчер ввода-вывода

Подсистема ввода-вывода

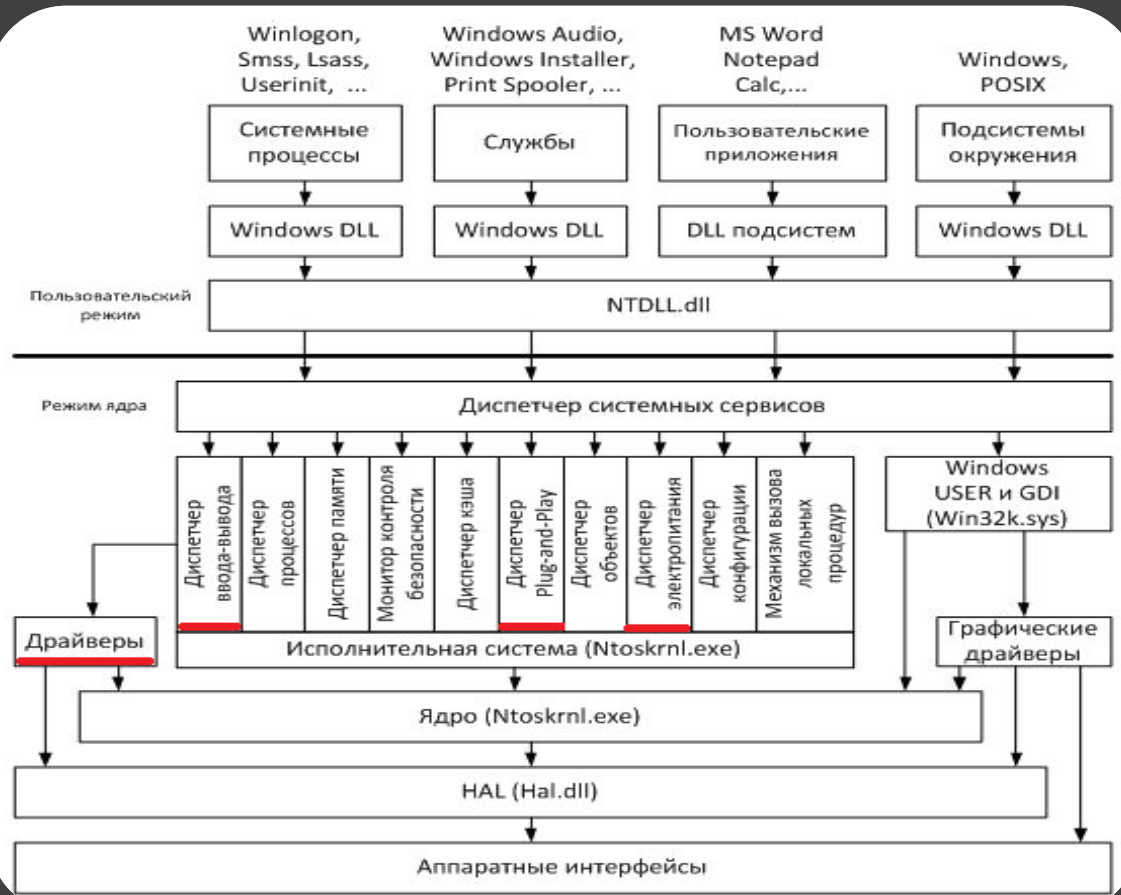
Обработка IRP запросов



1

Подсистема ввода-вывода

Архитектура Windows



OS Windows управляет устройствами с помощью подсистемы ввода-вывода, состоящей из:

- диспетчер ввода-вывода (I/O manager)
- диспетчер PnP (Plug and Play manager)
- диспетчер электропитания (power manager)
- драйверы устройств (обеспечивают функционал конкретного устройства)
- HAL (Hardware Abstraction Layer)

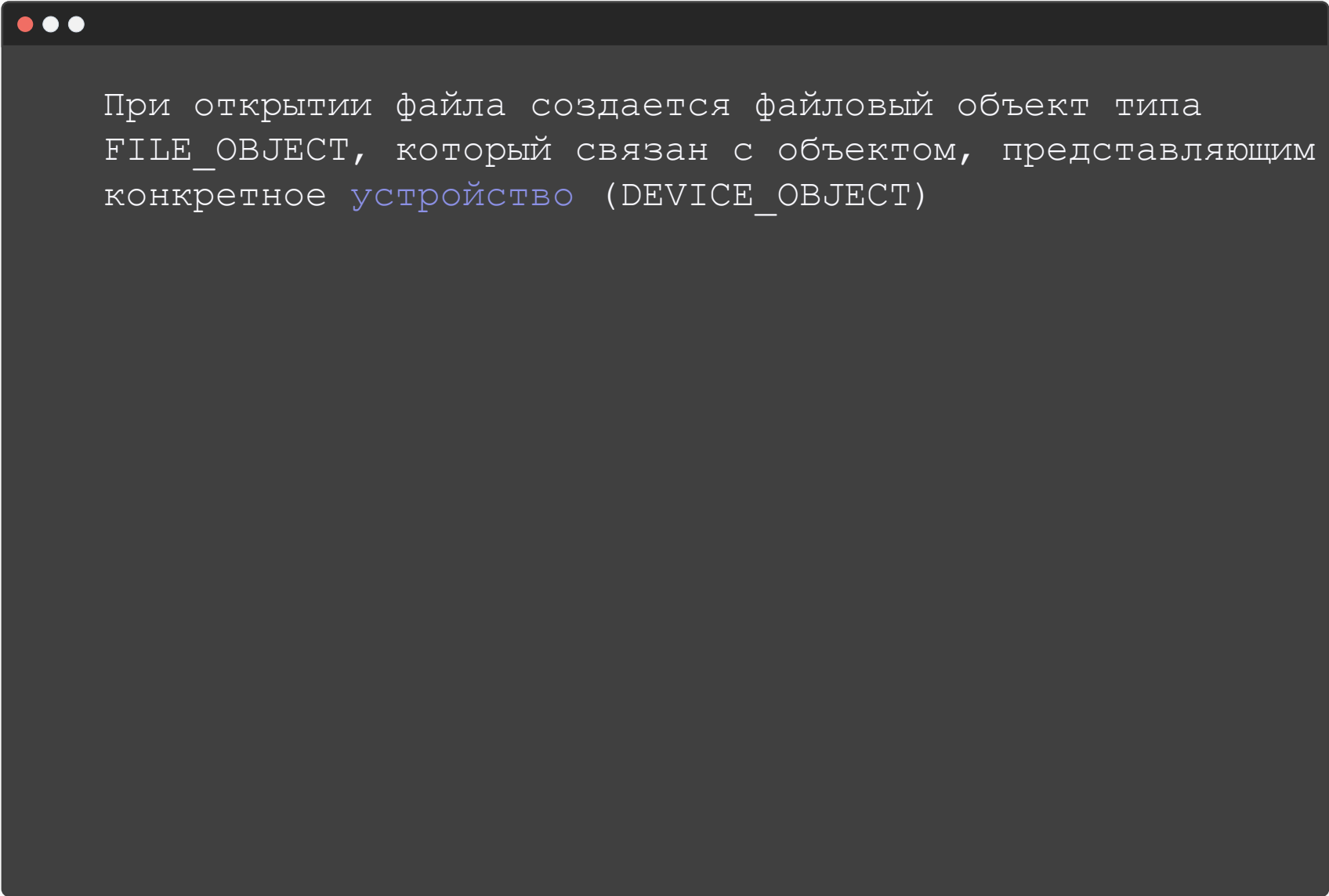
Для *User Mode* приложений, устройства представляются в виде **файлов**. В ядре, объект **файл связан с объектом устройство**

У файлов есть:

- ✓ Имя
- ✓ Операции чтения-записи

Последовательность работы с файлами:

- 1) Открытие
- 2) Чтение/запись
- 3) Заккрытие



При открытии файла создается файловый объект типа `FILE_OBJECT`, который связан с объектом, представляющим конкретное устройство (`DEVICE_OBJECT`)

Существует два понятия:

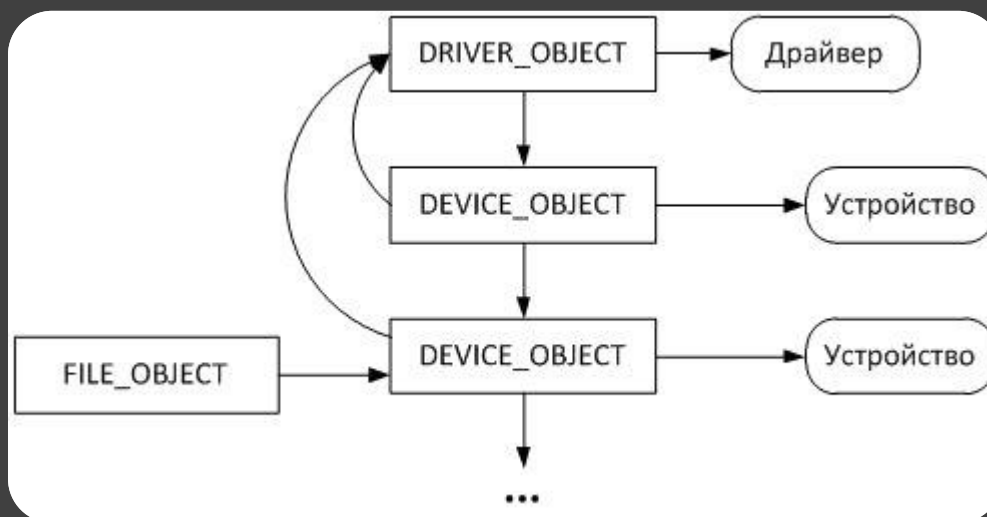
Объект «драйвер» DRIVER_OBJECT

Объект «устройство» DEVICE_OBJECT

- Объект DRIVER_OBJECT создаётся при загрузке в систему нового драйвера
- Объект DRIVER_OBJECT может управлять несколькими **устройствами**, создав для каждого объект **DEVICE_OBJECT**. Чтобы **устройство** можно было найти, объекту DEVICE_OBJECT задаётся имя
- В объекте DRIVER_OBJECT находятся **функции диспетчеризации** пакетов ввода-вывода
- Объект драйвер находится в каталоге **\Driver** в пространстве имен диспетчера объектов

- Объект `DRIVER_OBJECT` создаёт объект `DEVICE_OBJECT` и даёт ему своё имя
- По соглашению, имя устройство должно находиться в каталоге `\Device` в пространстве имен диспетчера объектов

- Драйвер в системе описывается объектом типа DRIVER_OBJECT
- При открытии файла создается файловый объект типа FILE_OBJECT, который связан с объектом, представляющим конкретное устройство (DEVICE_OBJECT)



Просмотр объектов DEVICE_OBJECT

The screenshot shows the WinObj application window with the 'Device' folder selected in the left pane. The main pane displays a table of device objects:

Name	Type	SymLink
ltdio	Device	
Mailslot	Device	
MailslotRedirector	SymbolicLink	\Device\Mup\MailslotRedirector

The 'Mailslot' object is selected, and its properties dialog is open. The 'Security' tab is active, showing the following permissions for the 'Everyone' group:

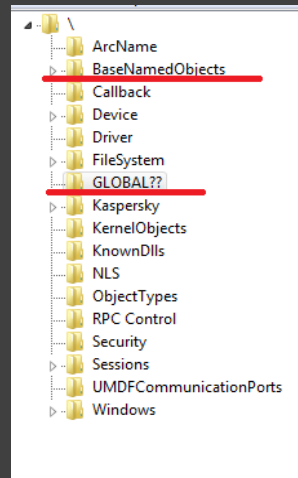
Permissions for Everyone	Allow	Deny
Read	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Delete	<input type="checkbox"/>	<input type="checkbox"/>
Special permissions	<input checked="" type="checkbox"/>	<input type="checkbox"/>

At the bottom of the dialog, there is an 'Advanced' button and a link: [Learn about access control and permissions](#). The 'OK' and 'Cancel' buttons are at the bottom right.

Все каталоги в пространстве имен диспетчера объектов невидимы для кода режима пользователя, кроме:

\BaseNamedObjects

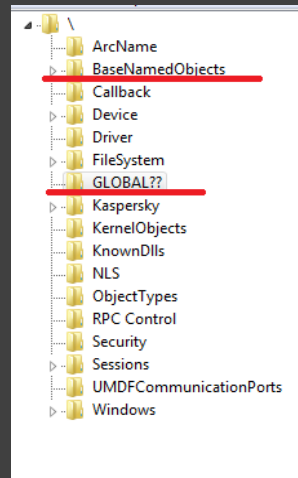
\GLOBAL??



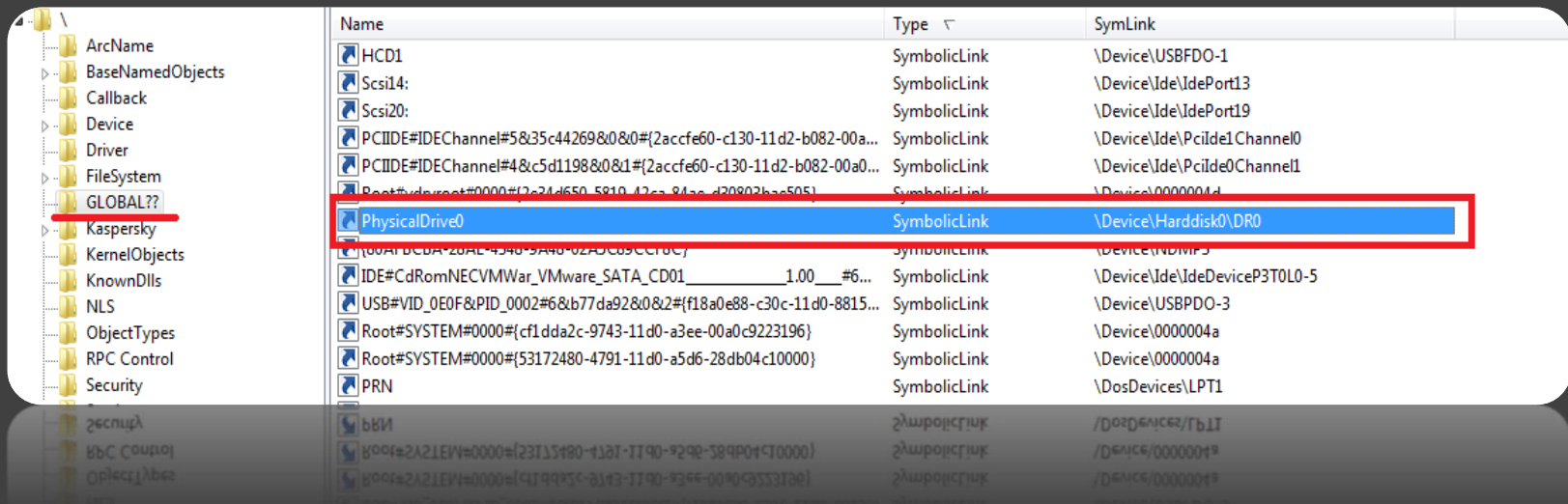
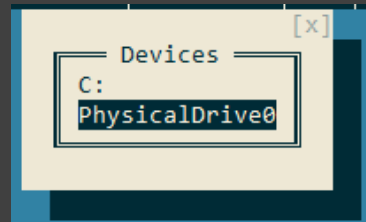
Из USER MODE не получится получить доступ к именам в каталогах \Device и \Driver

\BaseNamedObjects

\GLOBAL??



Создаём объект «**символьная ссылка**» в каталоге `\GLOBAL??`, который будет связан с нужным объектом в недоступном из User Mode каталоге `\Device`

A screenshot of a Windows Explorer window showing the file system structure on the left and a table of symbolic links on the right. The "GLOBAL??" folder is highlighted in red in the left pane. The table has columns for Name, Type, and SymLink. The row for "PhysicalDrive0" is highlighted in blue and has a red border around it.

Name	Type	SymLink
HCD1	SymbolicLink	\Device\USBPDO-1
Scsi14:	SymbolicLink	\Device\Ide\IdePort13
Scsi20:	SymbolicLink	\Device\Ide\IdePort19
PCIIDE#IDEChannel#5&35c44269&0&0#{2accfe60-c130-11d2-b082-00a...	SymbolicLink	\Device\Ide\PciIde1 Channel0
PCIIDE#IDEChannel#4&c5d1198&0&1#{2accfe60-c130-11d2-b082-00a...	SymbolicLink	\Device\Ide\PciIde0 Channel1
Root#SYSTEM#0000#{cf1dda2c-9743-11d0-a3ee-00a0c9223196}	SymbolicLink	\Device\0000004a
Root#SYSTEM#0000#{53172480-4791-11d0-a5d6-28db04c10000}	SymbolicLink	\Device\0000004a
PRN	SymbolicLink	\DosDevices\LPT1
IDE#CdRomNECVMWare_VMware_SATA_CD01_____1.00__#6...	SymbolicLink	\Device\Ide\IdeDeviceP3T0L0-5
USB#VID_0E0F&PID_0002#6&b77da92&0&2#{f18a0e88-c30c-11d0-8815...	SymbolicLink	\Device\USBPDO-3
PhysicalDrive0	SymbolicLink	\Device\Harddisk0\DR0

В каталоге `\BaseNamedObjects` хранятся имена всех базовых объектов ядра (мьютексы, секции, события, *СИМВОЛЬНЫЕ ССЫЛКИ*). В одном и том же каталоге имена объектов должны различаться

Name	Type	SymLink
Local	SymbolicLink	\BaseNamedObjects
Global	SymbolicLink	\BaseNamedObjects
Session	SymbolicLink	\Sessions\BNOLINKS
UGATHERER	Section	
MMF_BITS_s	Section	
Debug.Memory.v2.2a0	Section	
Debug.Trace.Memory.2a0	Section	
ComCatalogCache	Section	
windows_shell_global_counters	Section	
UrlZonesSM_SYSTEM	Section	
SqmData_FwtSqmSession101457921_S-1-5-18	Section	
C:\Windows_system32_config_systemprofile_AppData_Roaming_Micro...	Section	
WSearchIdxPi	Section	
FntCache-65184fdd-9525-4f49-a2bf-0e8c97b30cec	Section	
C:\ProgramData\Microsoft\Windows\Caches\{7CD55808-3D38-4DD5-9...	Section	
mmGlobalPnpInfo	Section	

В состав структуры DRIVER_OBJECT входит массив MajorFunction[28]

C++

```
typedef struct _DRIVER_OBJECT {
    CSHORT          Type;
    CSHORT          Size;
    PDEVICE_OBJECT  DeviceObject;
    ULONG           Flags;
    PVOID           DriverStart;
    ULONG           DriverSize;
    PVOID           DriverSection;
    PDRIVER_EXTENSION DriverExtension;
    UNICODE_STRING  DriverName;
    PUNICODE_STRING HardwareDatabase;
    PFAST_IO_DISPATCH FastIoDispatch;
    PDRIVER_INITIALIZE DriverInit;
    PDRIVER_STARTIO  DriverStartIo;
    PDRIVER_UNLOAD   DriverUnload;
    PDRIVER_DISPATCH MajorFunction[IRP_MJ_MAXIMUM_FUNCTION + 1];
} DRIVER_OBJECT, *PDRIVER_OBJECT;
```

Функции предназначены для обработки разных типов пакетов запроса ввода-вывода (IRP)

```
NTSTATUS DriverEntry(PDRIVER_OBJECT pDriverObject, PUNICODE_STRING pRegistryPath)
{
    NTSTATUS status;
    UNICODE_STRING DeviceName;
    PDRIVER_DISPATCH *pDispatcher;
    int i;

    RtlInitUnicodeString(&DeviceName, deviceName);
    RtlInitUnicodeString(&SymbolicLinkName, symbolicName);

    status = IoCreateDevice(pDriverObject, 0, &DeviceName, FILE_DEVICE_NULL, 0, FALSE, &deviceObject);
    if (status == STATUS_SUCCESS)
    {
        status = IoCreateSymbolicLink(&SymbolicLinkName, &DeviceName);

        pDispatcher = pDriverObject->MajorFunction;

        for (i = 0; i < IRP_MJ_MAXIMUM_FUNCTION; i++)
            pDispatcher[i] = CtlPass;

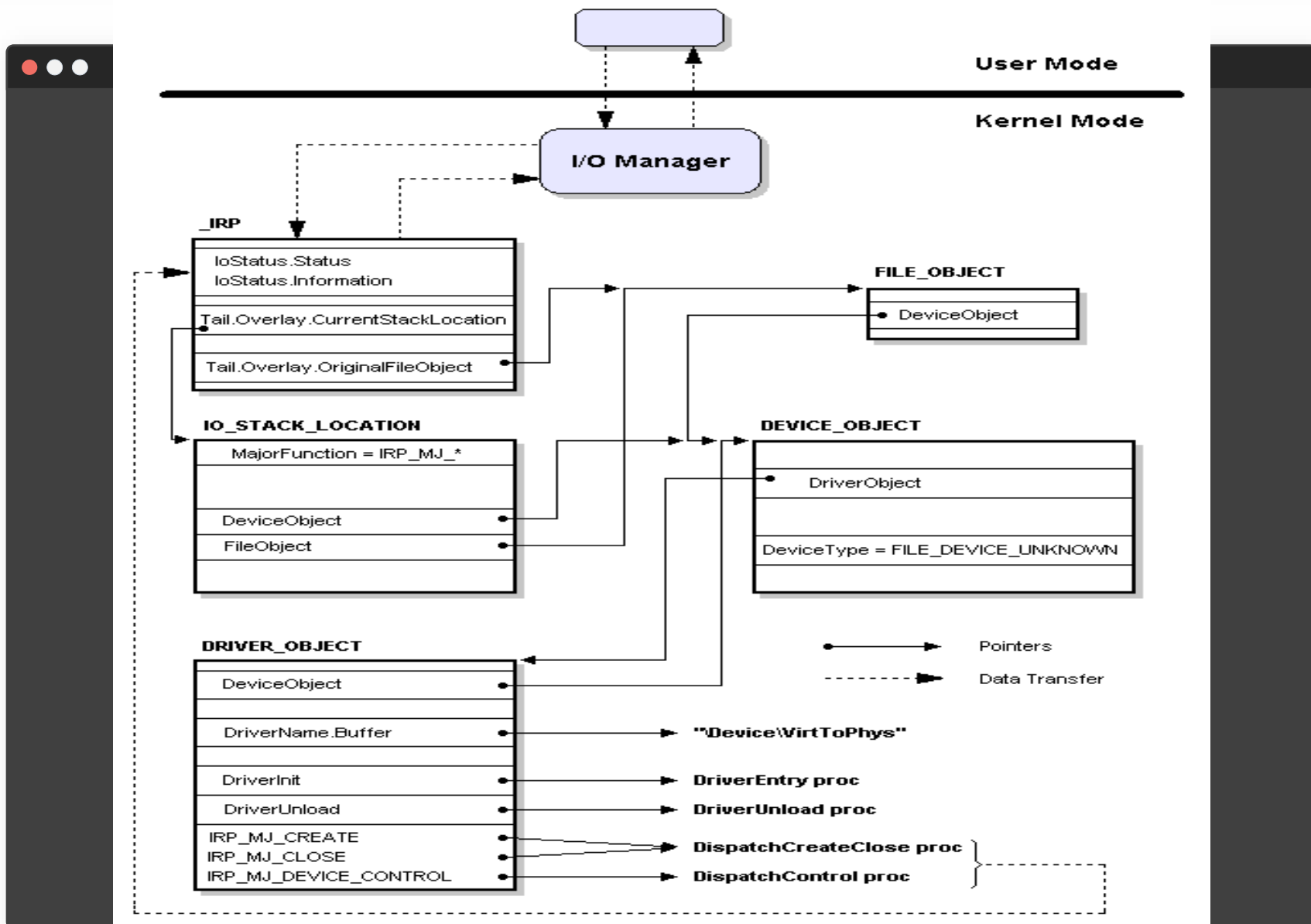
        pDriverObject->DriverUnload = UnloadRoutine;
        pDispatcher[IRP_MJ_DEVICE_CONTROL] = CtlDriverDispatch;

        if (status == STATUS_SUCCESS)
        {
            #ifdef DEBUG
                DbgPrint("driver has been loaded!\n");
            #endif
            return status;
        }
    }
    #ifdef DEBUG
        DbgPrint("driver not loaded!\n");
    #endif
    return status;
}
```

Последовательность работы с файлами:

- 1) Открытие
- 2) Чтение/запись
- 3) Закрытие

Тип запроса	Описание
IRP_MJ_CREATE	формируется диспетчером ввода-вывода при вызове кодом режима пользователя функции CreateFile
IRP_MJ_DEVICE_CONTROL	формируется диспетчером ввода-вывода при вызове кодом режима пользователя функции DeviceIoControl
IRP_MJ_CLOSE	формируется диспетчером ввода-вывода при вызове кодом режима пользователя функции CloseHandle



2

Обработка **IRP** запросов

Диспетчер ввода-вывода поддерживает три вида управления буферами:

a) буферизованный ввод-вывод (buffered I/O)

b) прямой ввод-вывод (direct I/O)

c) ввод-вывод без управления (neither I/O)

Диспетчер ввода-вывода выделяет в системном неподкачиваемом пуле памяти буфер, равный по размеру наибольшему из пользовательских буферов

Создавая IRP, при операции записи, диспетчер ввода-вывода копирует данные из пользовательского входного буфера в этот системный, а указатель на системный буфер заносит в IRP->AssociatedIrp.**SystemBuffer**, размер скопированных данных в IO_STACK_LOCATION->Parameters.DeviceIoControl.**InputBufferLength**

Завершая IRP, при операции чтения, диспетчер ввода-вывода копирует данные из системного буфера в пользовательский выходной буфер

Input:

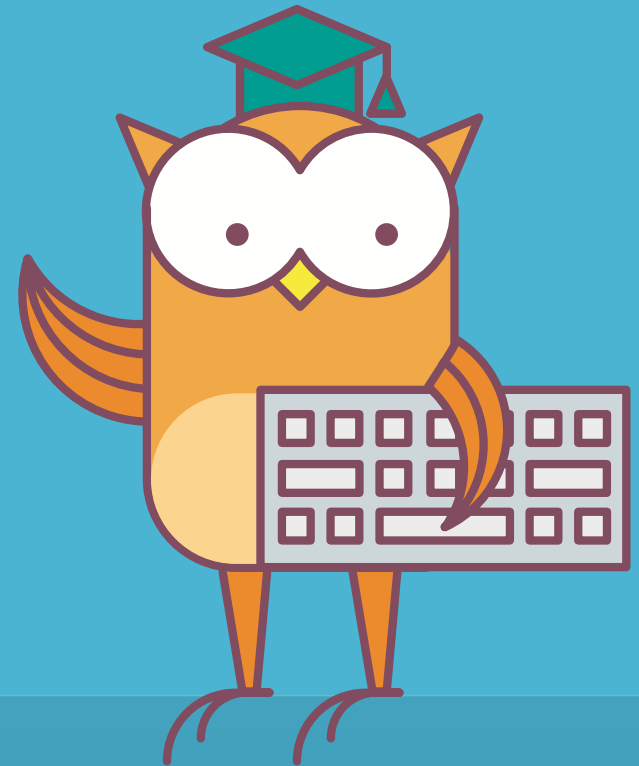
```
//for write
userInputSize = pIrpStack->Parameters.DeviceIoControl.InputBufferLength;
userInputBuf = (char *)Irp->AssociatedIrp.SystemBuffer;
//for read
bufForRead = (char *)Irp->UserBuffer;
forReadSize = pIrpStack->Parameters.DeviceIoControl.OutputBufferLength;
```

Output:

```
pIrpStack->IoStatus.Information = x;
```

O T U S

Вопросы???





Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!

