ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

OTUS

Диспетчер ввода-вывода

Подсистема ввода-вывода

Обработка IRP запросов

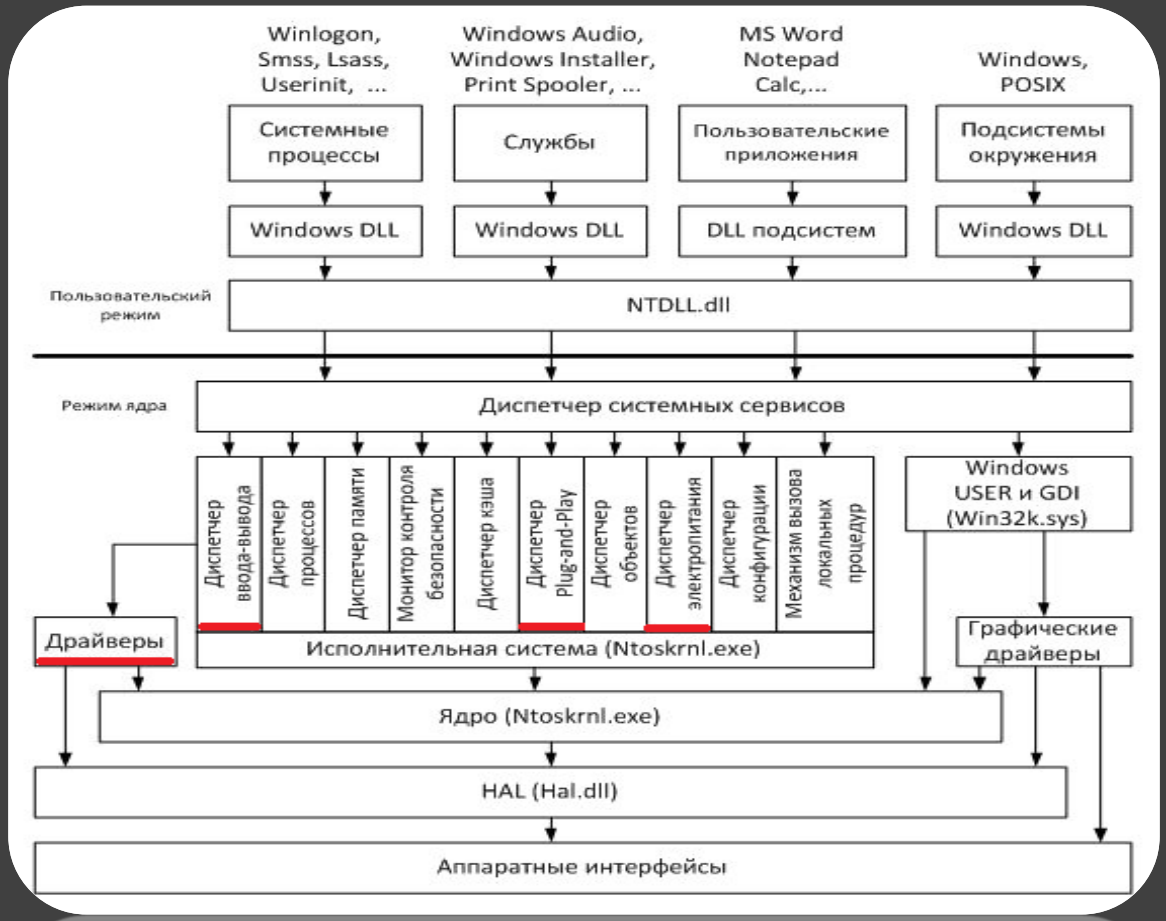
Пакулов Артур

1

Подсистема ввода-вывода

Подсистема ввода-вывода

Архитектура Windows



Подсистема ввода-вывода

ОС 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)

Управление устройствами

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) Закрытие

FILE_OBJECT

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

Управление устройствами

OTUS

Существует два понятия:

- Объект «драйвер» DRIVER_OBJECT
- Объект «устройство» DEVICE_OBJECT

DRIVER_OBJECT

OTUS

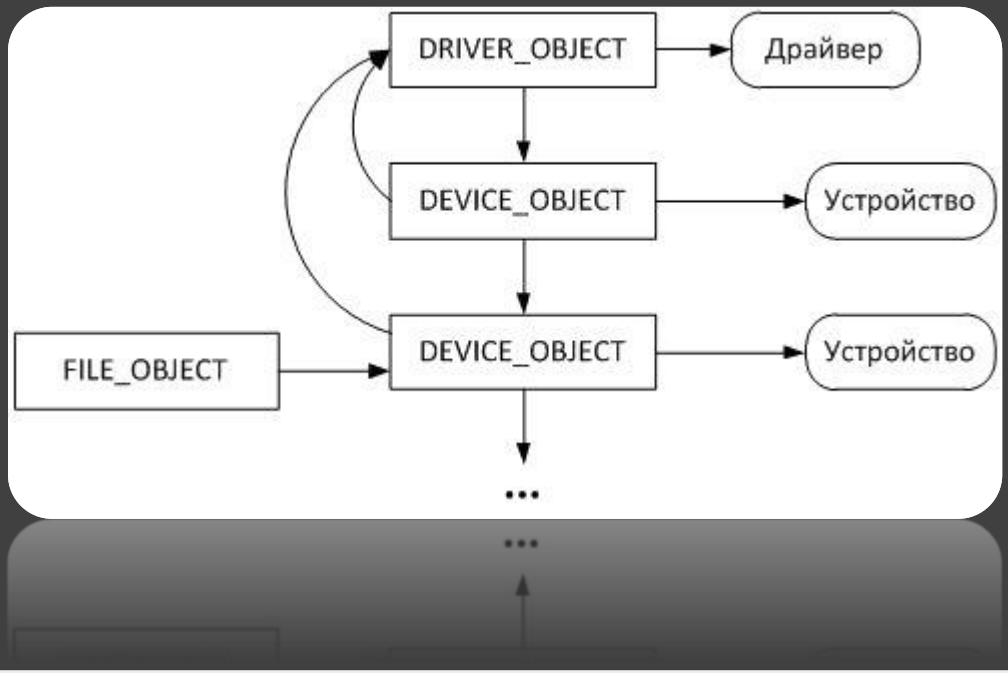
- Объект DRIVER_OBJECT создаётся при загрузке в систему нового драйвера
- Объект DRIVER_OBJECT может управлять несколькими **устройствами**, создав для каждого объект **DEVICE_OBJECT**. Чтобы **устройство** можно было найти, объекту DEVICE_OBJECT задаётся имя
- В объекте DRIVER_OBJECT находятся ***функции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

DEVICE_OBJECT

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

Итог

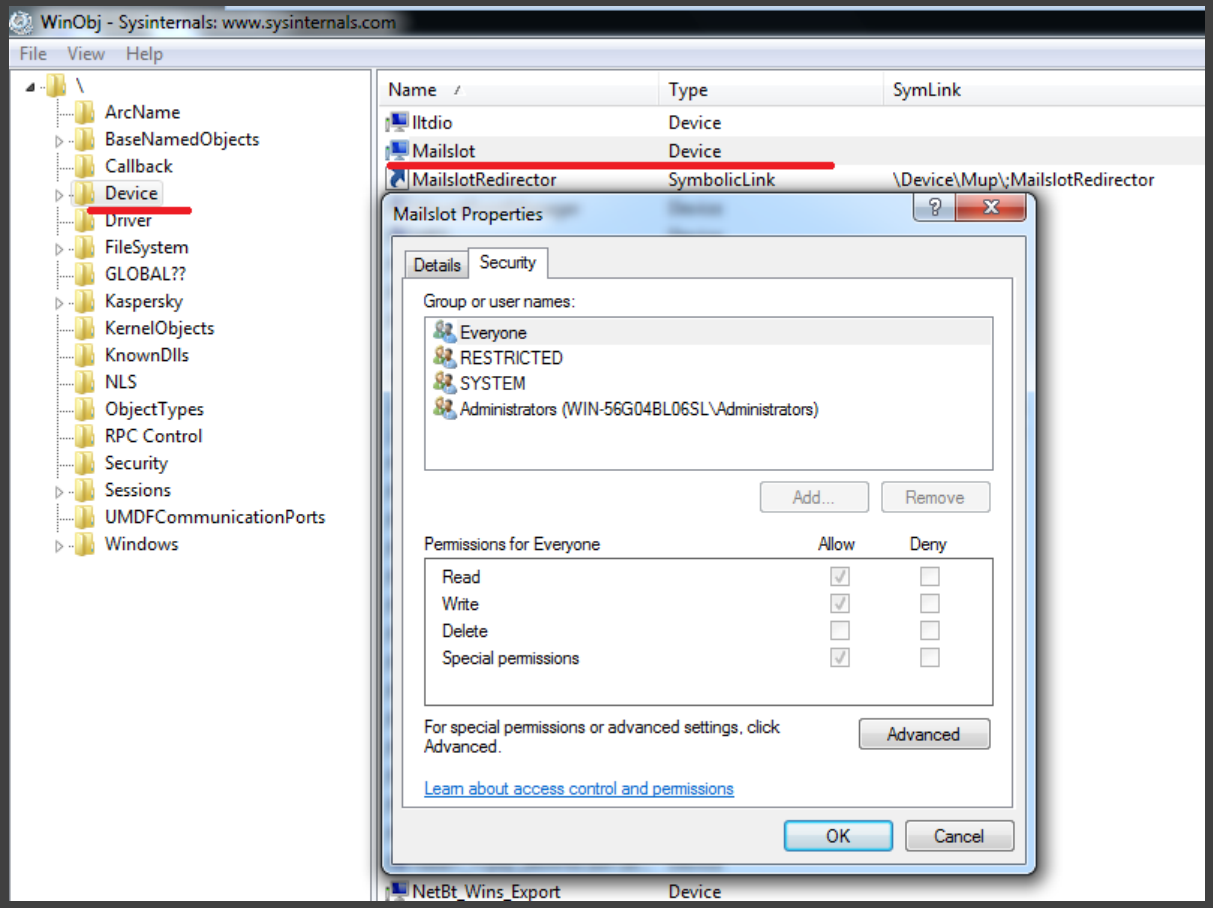
- Драйвер в системе описывается объектом типа DRIVER_OBJECT
- При открытии файла создается файловый объект типа FILE_OBJECT, который связан с объектом, представляющим конкретное устройство (DEVICE_OBJECT)



WinObj - Sysinternals

OTUS

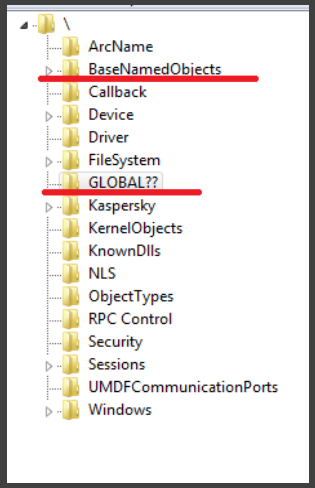
Просмотр объектов DEVICE_OBJECT



Доступ к именам объектов

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

- \BaseNamedObjects
- \GLOBAL??

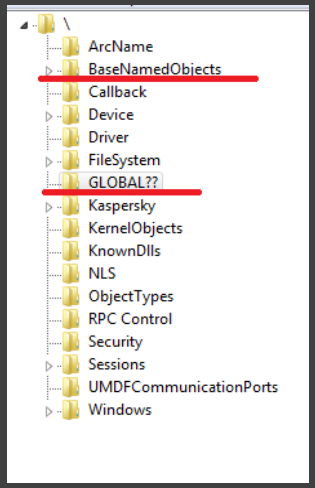


ВЫВОД

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

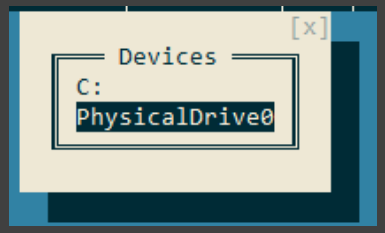
\BaseNamedObjects

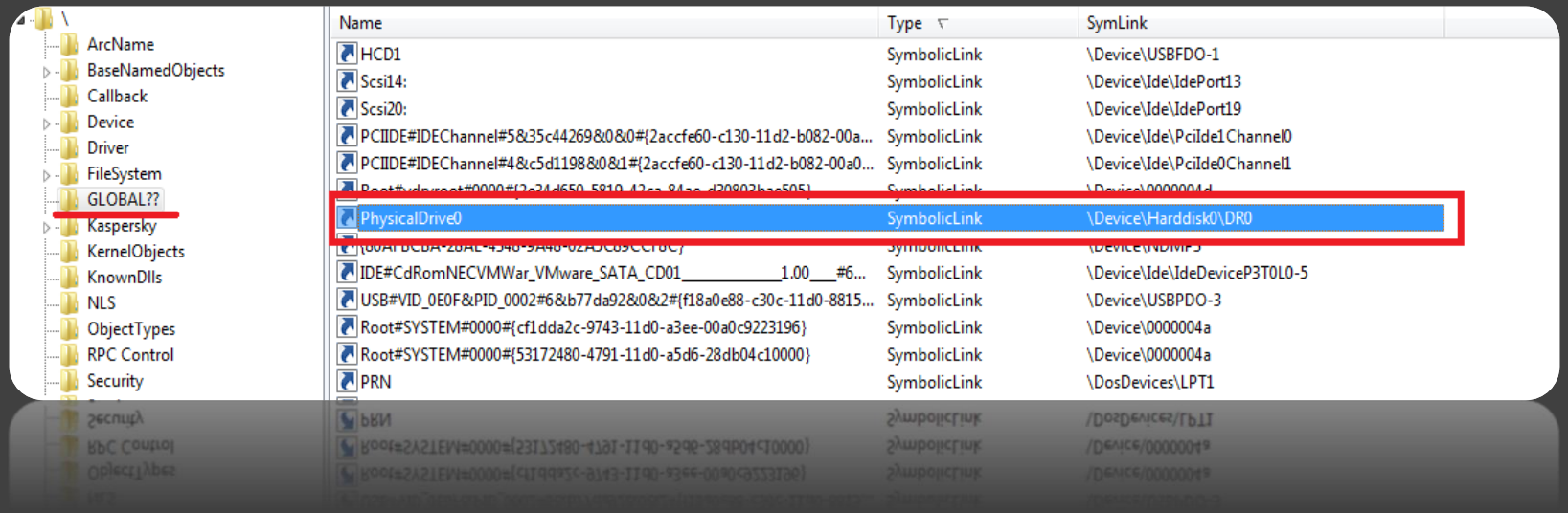
\GLOBAL??



Решение

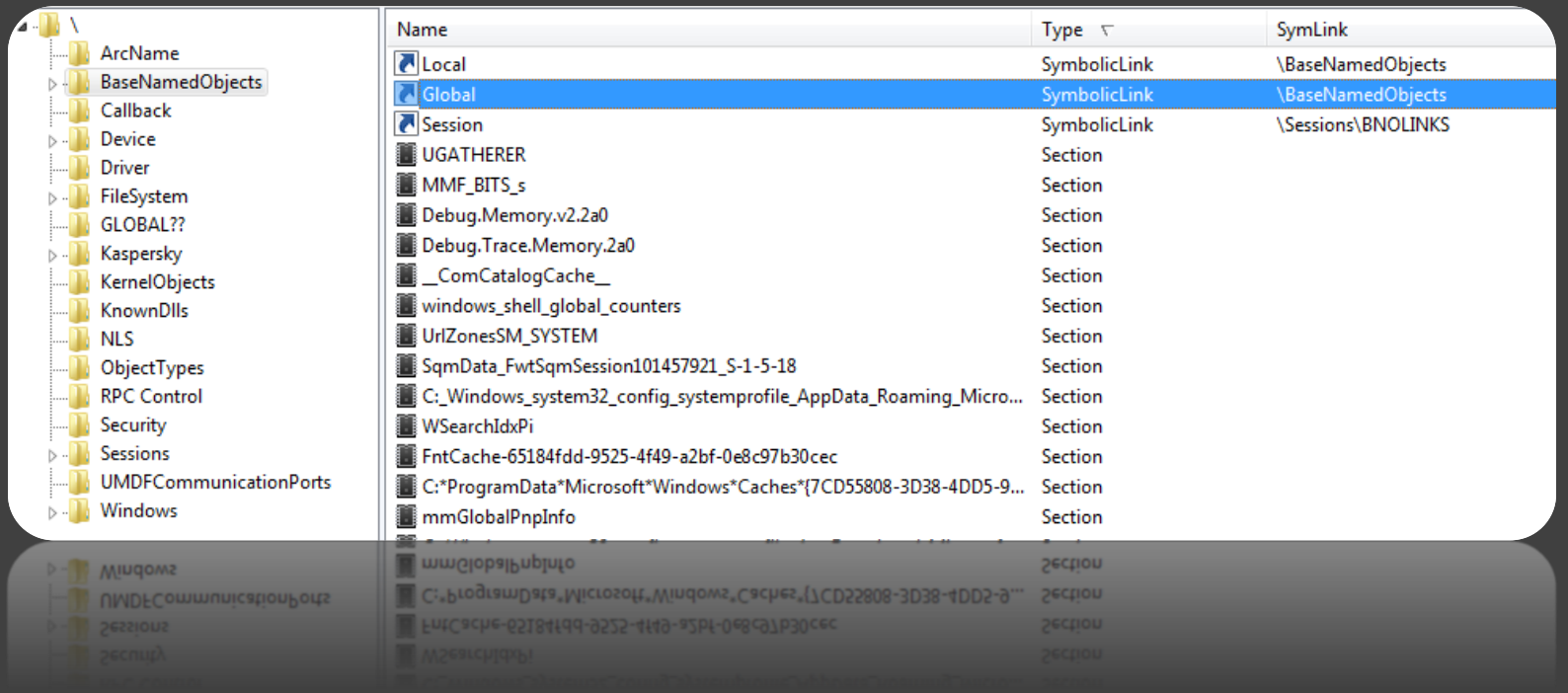
Создаём объект «**символьная ссылка**» в каталоге **\GLOBAL??**, который будет связан с нужным объектом в недоступном из User Mode каталоге \Device





BaseNamedObjects

В каталоге **BaseNamedObjects** хранятся имена всех базовых объектов ядра (мьютексы, секции, события, *символьные ссылки*). В одном и том же каталоге имена объектов должны различаться



Процедуры диспетчеризации

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;
```

Процедуры диспетчеризации

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

```
NTSTATUS DriverEntry(PDRIVER_OBJECT pDriverObject, PUNICODE_STRING pRegistryPath)
{

    NTSTATUS status;
    UNICODE_STRING  DeviceName;
    PDRIVER_DISPATCH *pDispacher;
    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);

        pDispacher = pDriverObject->MajorFunction;

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

        pDriverObject->DriverUnload = UnloadRoutine;
        pDispacher[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;
}
```

Процедуры диспетчеризации

Последовательность работы с файлами:

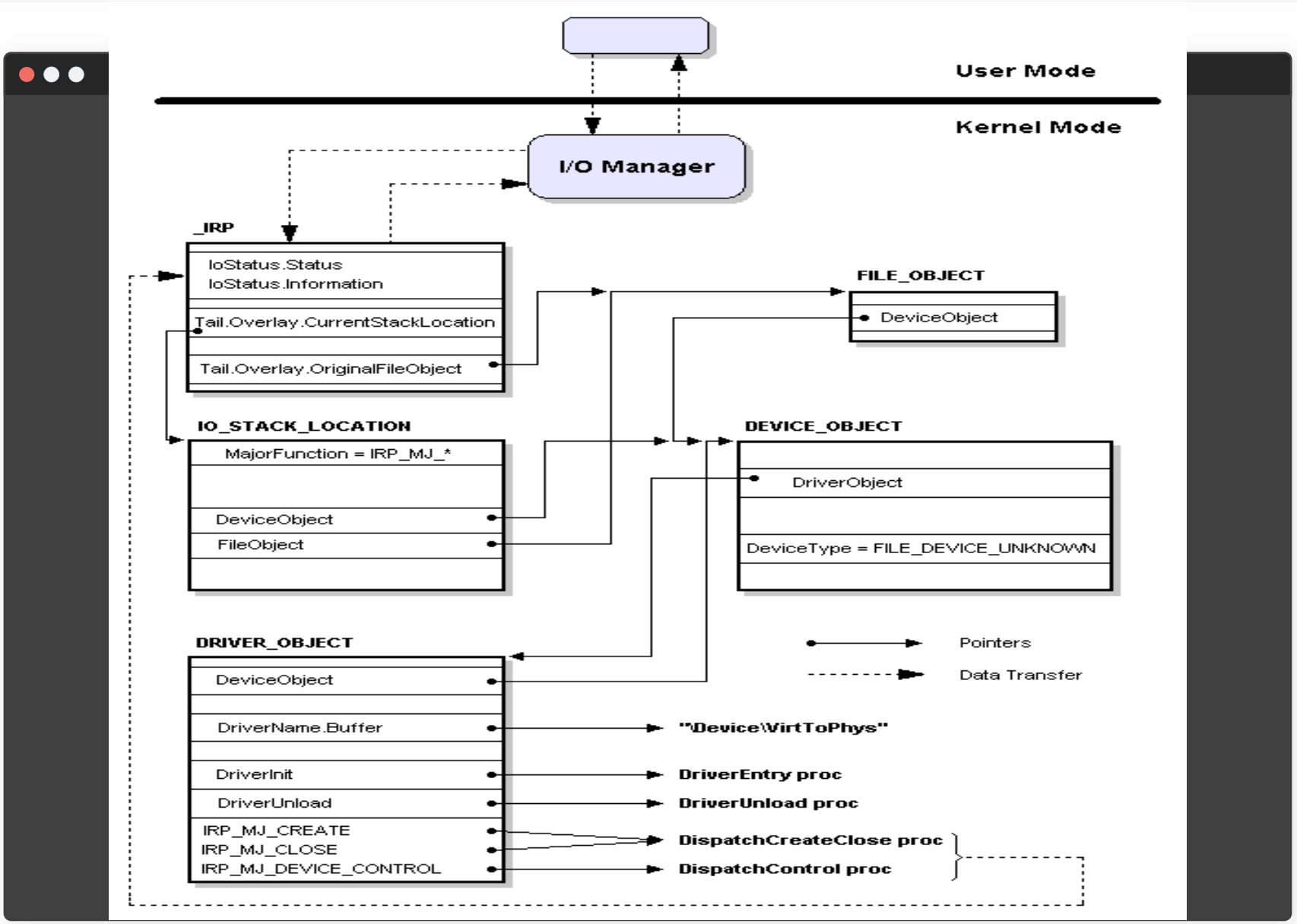
1) Открытие

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**

Взаимосвязи объектов «файл», «устройство» и «драйвер»



2

Обработка IRP запросов

Виды управления буферами

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)

Буферизованный ввод-вывод

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

Буферизованный ввод-вывод

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;
```

OTUS

Вопросы???

Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!