



H2HC III Edition



<http://nerv.5p.org.uk/>

Playing with DDK



Objetivo: prover uma introdução aos drivers para windows tal como uma visão geral sobre *hooks* disponíveis. E se divertir!

Tópicos

- **Evolução do Windows**
 - user x kernel mode, kernel monolítico x microkernel, etc.
- **Linking & Loading**
 - PE Format (IAT, EAT), Linking (.dll & .sys), Loading (subsystems), etc.
- **WDM: Conceitos**
 - I/O Request Packets (IRP), Interrupt Request Level (IRQL), Device Objects, Device Stacks, etc.
- **WDM: Exemplo**
 - Hello World!
- **Aplicações Práticas**
 - *Keylogger, File Hiding, Process Hiding, Obfuscation...*

Evolução Windows

DOS, Win 3.x, Win 9x, Win NT

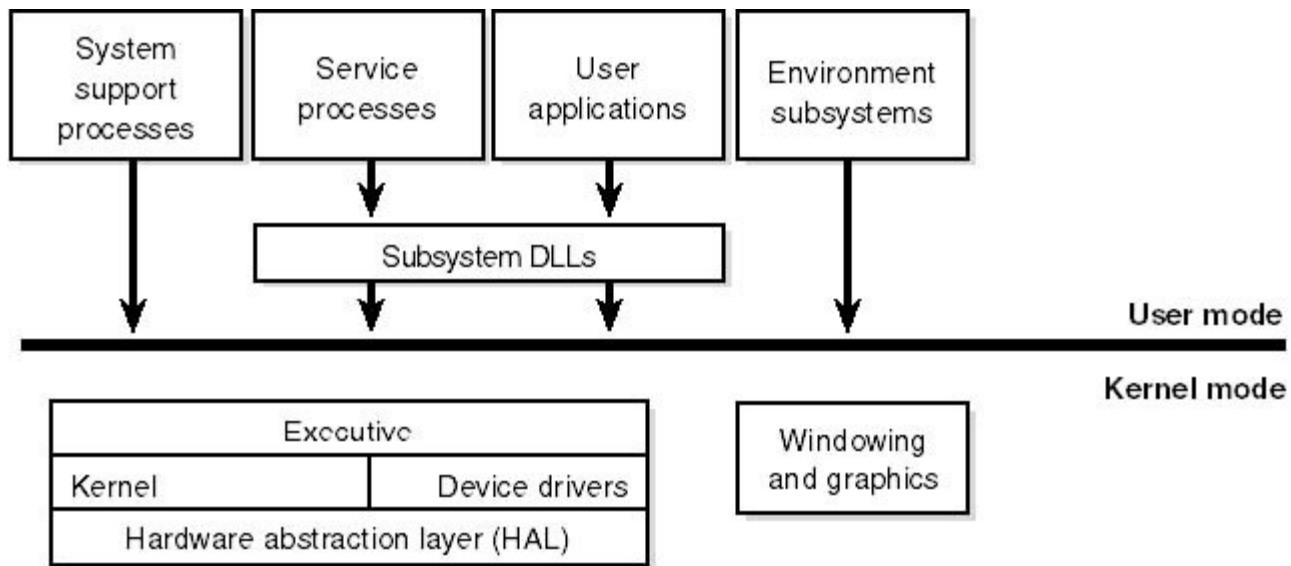
Modelo Atual

User Mode: cada processo tem seu espaço privado de endereçamento de memória.

Kernel Mode: *kernel* e *device drivers* compartilham um espaço virtual único de endereçamento.

Windows não é microkernel (vide Match), possui kernel monolítico.

User x Kernel



Evolução do Drivers

VxD: até o windows 98, sem mecanismos para isolamento do espaço de endereçamento.

WDM: a partir do windows 98, **compatibilidade do binário** para os diversos windows.

Linking & Loading

Por dentro do sistema.

Portable Executable (PE)

- ❑ Objetivo: portabilidade binária entre todas as versões do Windows...
- ❑ Baseado no COFF, que vem VAX/VMS.
- ❑ A diferença binária entre DLL e EXE é mínima.
- ❑ Arquivos .OCX e .CPL são DLLs.

.SYS vs .DLL

Em Conceito, os drivers (.sys) são bastante similares as bibliotecas dinâmicas (.dll) do windows.

.SYS

link com **ntoskrnl.exe** & **HAL.DLL**

load como subsystem **WINDOWS**

.DLL

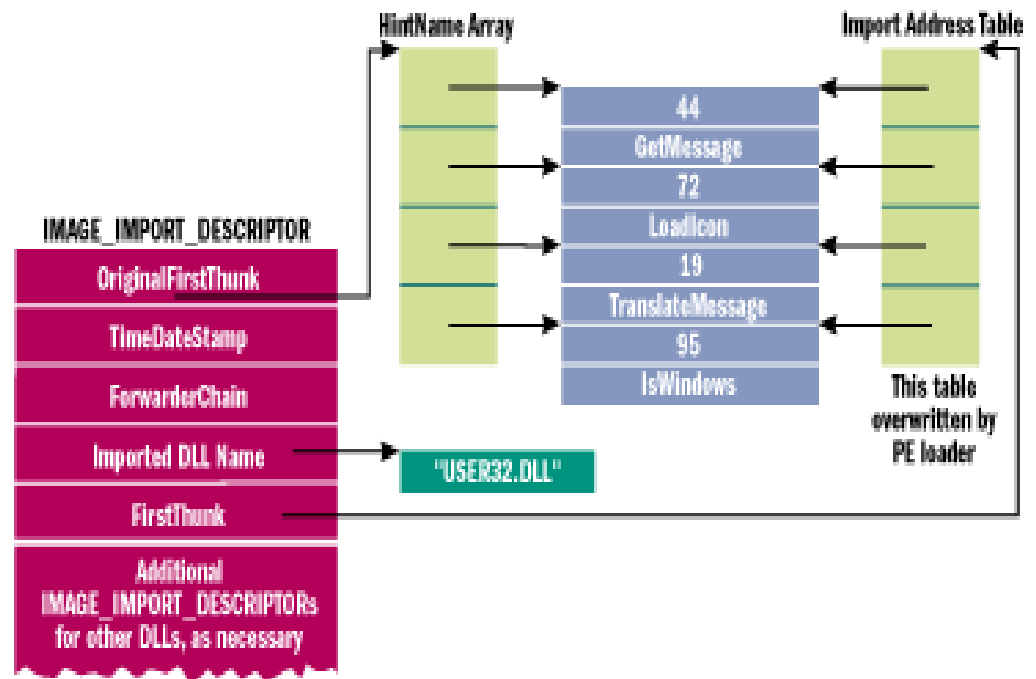
link com **kernel32.dll** & **ntdll.dll**

load como subsystem **NATIVE**

PE: Import Section

A estrutura `IMAGE_IMPORT_DESCRIPTOR` aponta para a *Import Address Tables (IAT)* e para *Import Tables Name (INT)*.

A **IAT** é sobrescrita pelo loader com os endereços das funções.



PE: Export Section

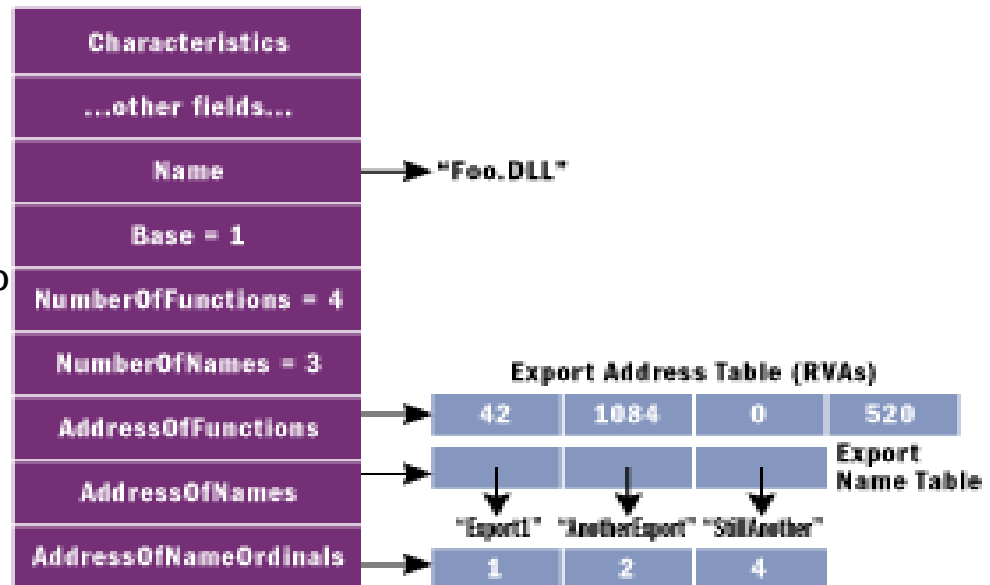
Abaixo detalhes da estrutura **IMAGE_EXPORT_DIRECTORY**, que lista funções e variáveis a serem exportadas.

Export Forwarding

(...)

HeapAlloc = NTDLL.RtlAllocHeap

(...)



user32.dll

PEView - C:\WINDOWS\system32\user32.dll

Description
Magic
Major Linker Version
Minor Linker Version
Size of Code
Size of Initialized Data
Size of Uninitialized Data
Address of Entry Point
Base of Code
Base of Data
Image Base
Section Alignment
File Alignment
Major O/S Version
Minor O/S Version
Major Image Version
Minor Image Version
Major Subsystem Version
Minor Subsystem Version
Win32 Version Value
Size of Image
Size of Headers
Checksum
Subsystem IMAGE_SUBSYSTEM_WINDOWS_GUI
DLL Characteristics
Size of Stack Reserve
Size of Stack Commit
Size of Heap Reserve
Size of Heap Commit
Loader Flags

Viewing IMAGE_OPTIONAL_HEADER

PEView - D:\WINDDK\Hello\objfre_wnet_x86\i386\hello.sys

Description	Value
Magic	IMAGE_NT_OPTIONAL_HDR32_MAGIC
Major Linker Version	
Minor Linker Version	
Size of Code	
Size of Initialized Data	
Size of Uninitialized Data	
Address of Entry Point	
Base of Code	
Base of Data	
Image Base	
Section Alignment	
File Alignment	
Major O/S Version	
Minor O/S Version	
Major Image Version	
Minor Image Version	
Major Subsystem Version	
Minor Subsystem Version	
Win32 Version Value	
Size of Image	
Size of Headers	
Checksum	
Subsystem IMAGE_SUBSYSTEM_NATIVE	
DLL Characteristics	
0400	IMAGE_DLLCHARACTERISTICS_NO_SEH
Size of Stack Reserve	
Size of Stack Commit	
Size of Heap Reserve	
Size of Heap Commit	

Viewing IMAGE_OPTIONAL_HEADER

hello.sys

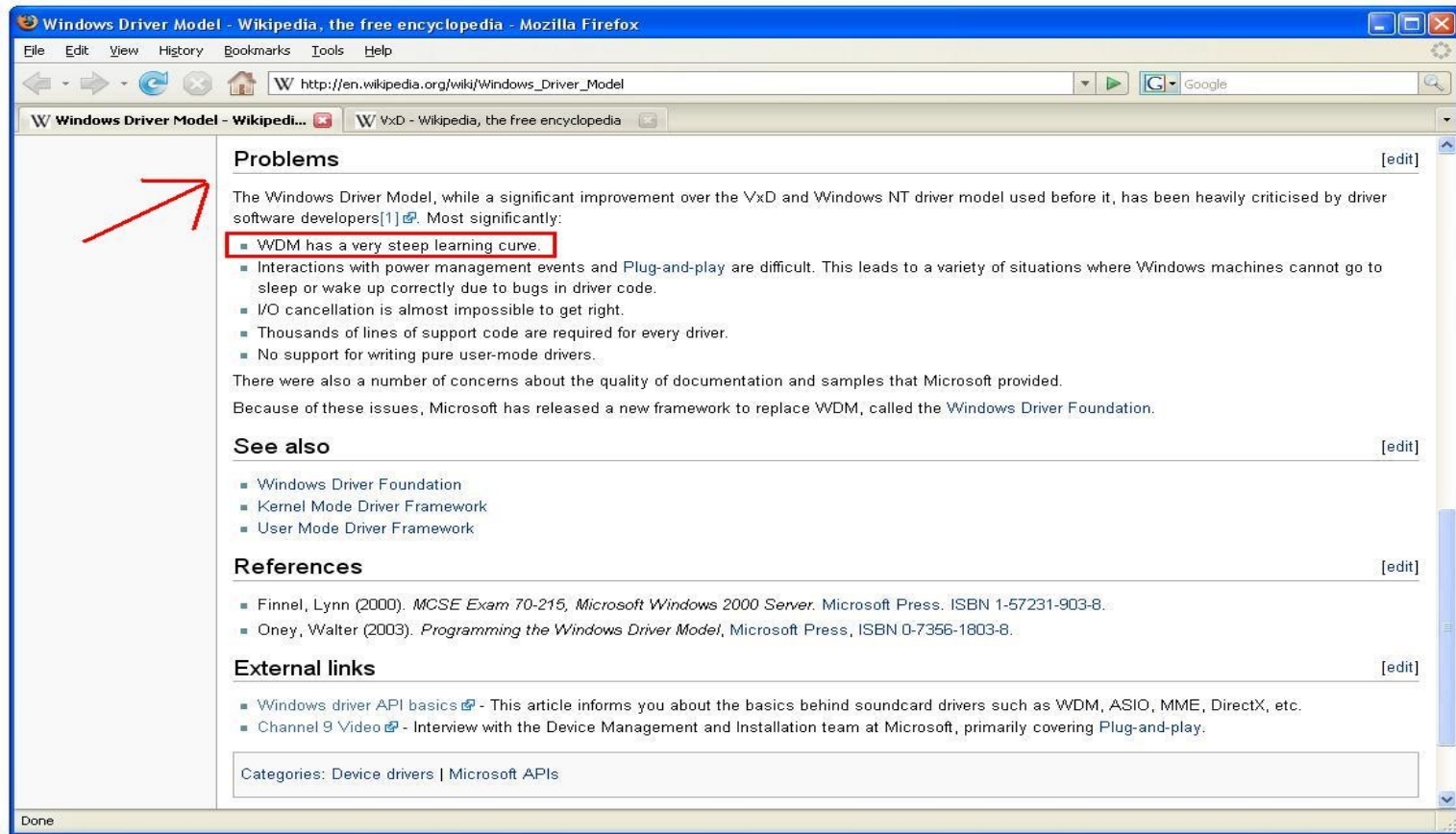
User Level Hooking...

- Via Registro
 - Desvantagens: Unload só com reboot, apenas aplicações gráficas.
- System Hooks com ***SetWindowsHookEx()***
 - Desvantagem: Complexidade e queda de performance.
- Injetar DLL ***CreateRemoteThread()*** e ***LoadLibrary()***
 - Desvantagens: Apenas para Win NT, exige certos privilégios.
- Browser Helper Objects(BHO) add-ins
 - Apenas para o Internet Explorer (IE).
- MicroSoft Office add-ins
 - Apenas para Office.

WDM: Conceitos

Por onde iniciar.

“steep learning curve”



Windows Driver Model - Wikipedia, the free encyclopedia - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://en.wikipedia.org/wiki/Windows_Driver_Model

W Windows Driver Model - Wikipedi... W VxD - Wikipedia, the free encyclopedia

Problems [edit]

The Windows Driver Model, while a significant improvement over the VxD and Windows NT driver model used before it, has been heavily criticised by driver software developers[1]. Most significantly:

- WDM has a very steep learning curve.
- Interactions with power management events and Plug-and-play are difficult. This leads to a variety of situations where Windows machines cannot go to sleep or wake up correctly due to bugs in driver code.
- I/O cancellation is almost impossible to get right.
- Thousands of lines of support code are required for every driver.
- No support for writing pure user-mode drivers.

There were also a number of concerns about the quality of documentation and samples that Microsoft provided.

Because of these issues, Microsoft has released a new framework to replace WDM, called the Windows Driver Foundation.

See also [edit]

- Windows Driver Foundation
- Kernel Mode Driver Framework
- User Mode Driver Framework

References [edit]

- Finnel, Lynn (2000). *MCSE Exam 70-215, Microsoft Windows 2000 Server*. Microsoft Press. ISBN 1-57231-903-8.
- Oney, Walter (2003). *Programming the Windows Driver Model*, Microsoft Press, ISBN 0-7356-1803-8.

External links [edit]

- Windows driver API basics - This article informs you about the basics behind soundcard drivers such as WDM, ASIO, MME, DirectX, etc.
- Channel 9 Video - Interview with the Device Management and Installation team at Microsoft, primarily covering Plug-and-play.

Categories: Device drivers | Microsoft APIs

Done

“things to avoid”

The screenshot shows a Mozilla Firefox browser window displaying a Microsoft support article. The browser's address bar shows the URL: <http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q186775&>. The page title is "INFO: Tips for Windows NT Driver Developers -- Things to Avoid".

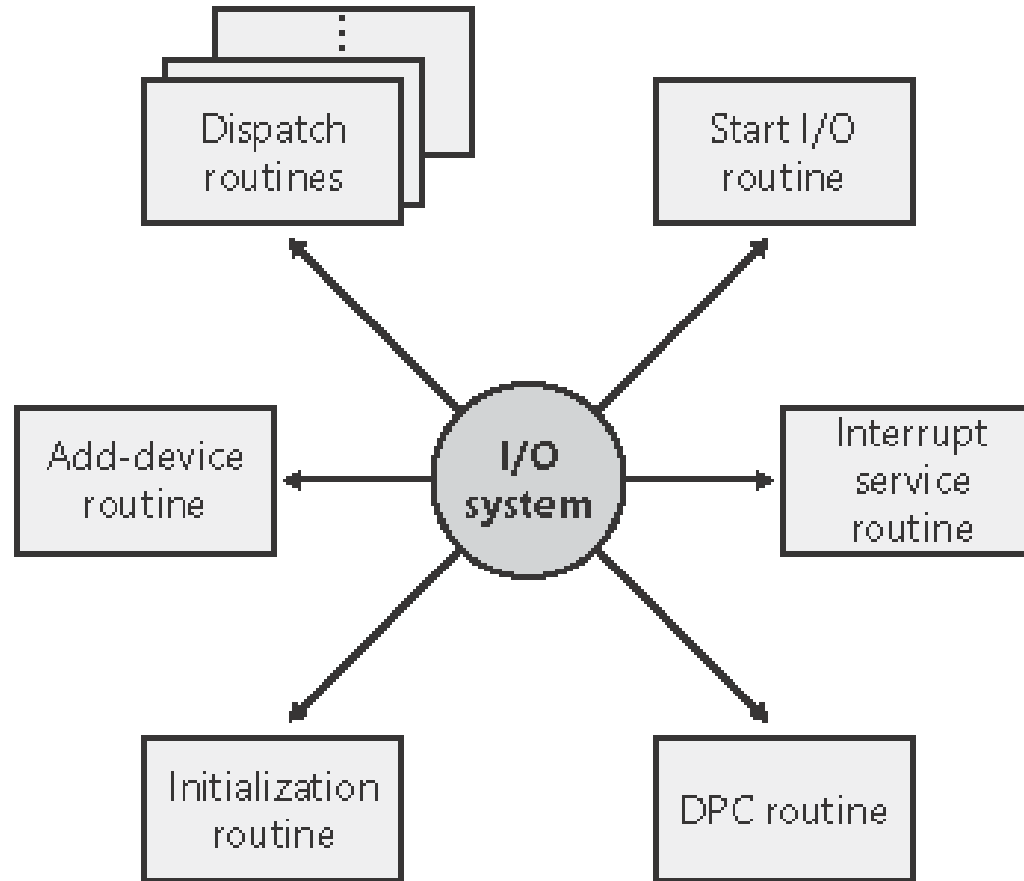
The article content includes:

- INFO: Tips for Windows NT Driver Developers -- Things to Avoid** (highlighted with a red box and an arrow)
- View products that this article applies to.**
- This article was previously published under Q186775.
- SUMMARY**
Following are some tips for creating Windows NT device drivers. The tips presented apply to all technologies. You can also use this as a checklist for troubleshooting driver problems.
- You need to have a basic knowledge of Windows NT architecture and some device driver development experience to use the information presented below effectively. For more information on device driver development, please see the Windows NT device driver kit (DDK), which is available through MSDN Professional membership.
- [Back to the top](#)
- MORE INFORMATION**
Following is a list of things that developers should avoid when working with Windows NT device drivers:
- 1. Never return STATUS_PENDING from a dispatch routine without marking the I/O request packet (IRP) pending (IoMarkIrpPending).
- 2. Never call KeSynchronizeExecution from an interrupt service routine (ISR). It will deadlock your system.
- 3. Never set DeviceObject->Flags to both DO_BUFFERED_IO and DO_DIRECT_IO. It can confuse the system and eventually lead to fatal error. Also, never set METHOD_BUFFERED, METHOD_NEITHER, METHOD_IN_DIRECT or METHOD_OUT_DIRECT in DeviceObject->Flags, because these values are only used in defining IOCTLs.
- 4. Never allocate dispatcher objects from a paged pool. If you do, it will cause occasional system bugchecks.
- Done

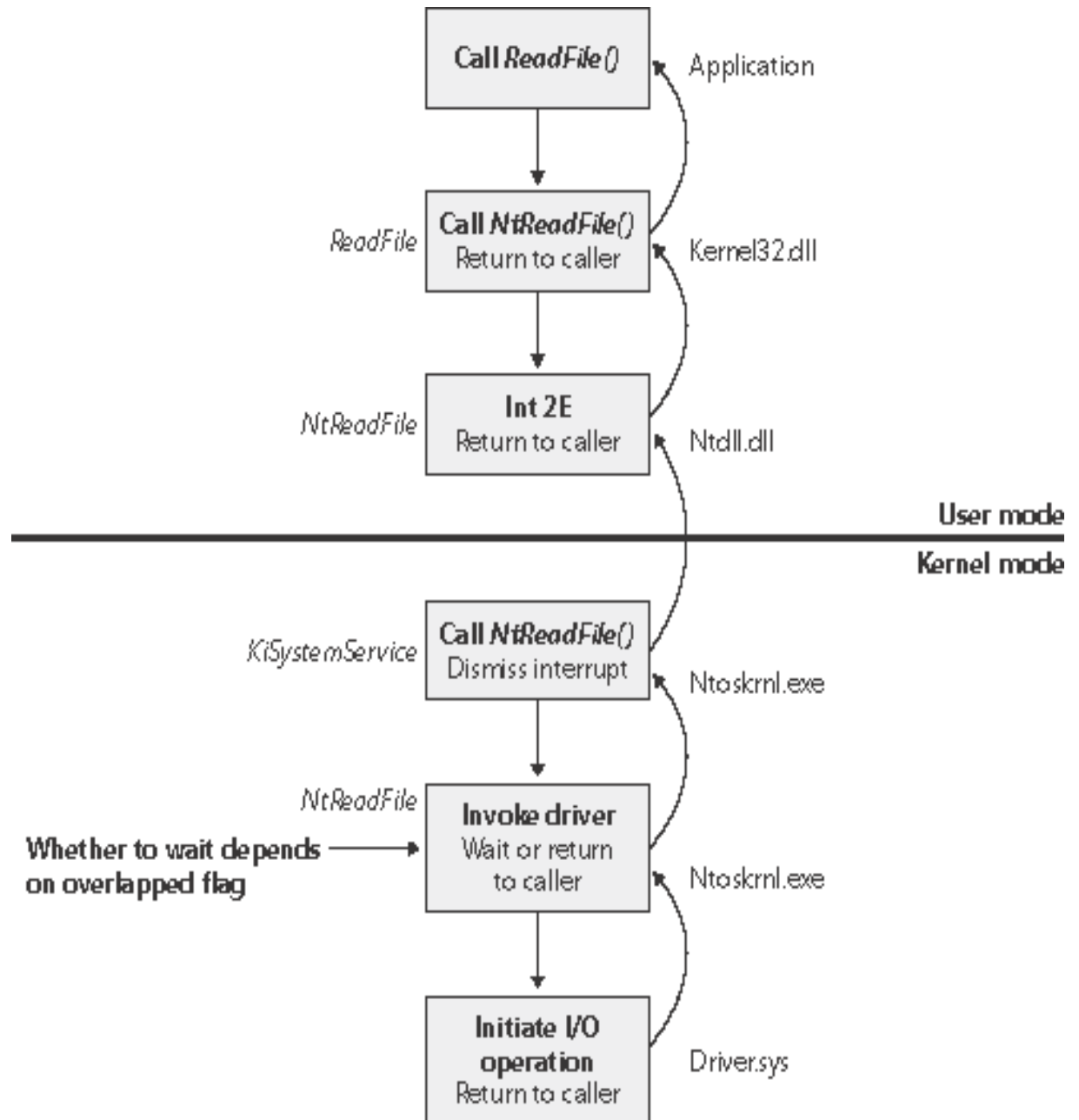
On the left side of the browser window, there is a sidebar with a list of items. Item 58 is circled in red, and a red arrow points to it from the main article content. The sidebar also includes sections for "REFERENCES", "APPLIES TO", and "Keywords".

At the bottom of the browser window, there is a footer with the text: "© 2006 Microsoft Corporation. All rights reserved. [Terms of Use](#) | [Trademarks](#) | [Privacy Statement](#)".

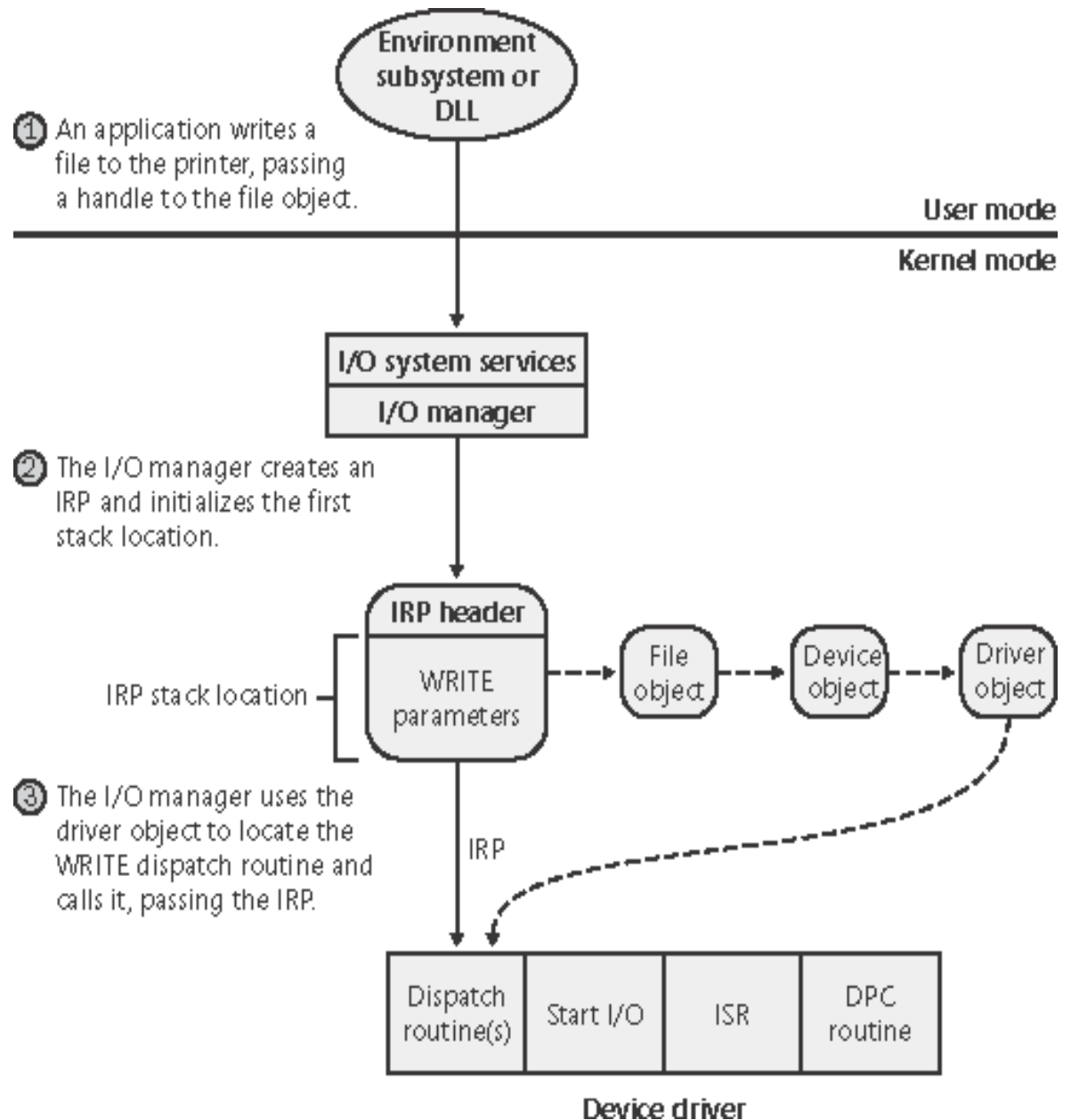
Visão Geral



Flow Control



Data Structure



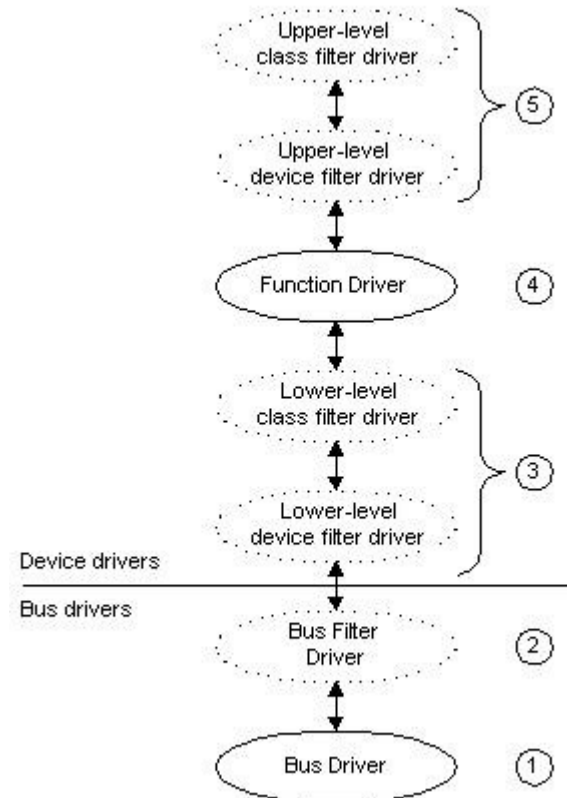
Windows Driver Model (**WDM**).

Requisitos para se encaixar no modelo...

- ❑ Possuir o header wdm.h, não o ntddk.h
- ❑ Ser um **bus**, **function** ou **filter** driver
- ❑ Criar **Device Objects** e atachar no **Device Stack**
- ❑ Suportar:
 - *Plug-and-Play*
 - *Power Management*
 - *Windows Management Instrumentation (WMI)*

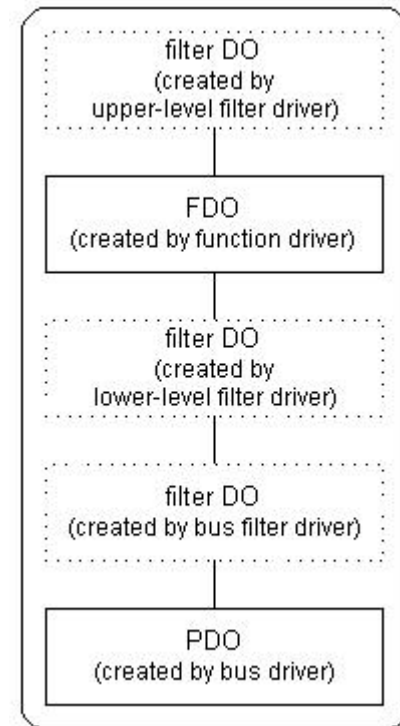
WDM: Driver Layers

- Bus Driver (1)
- **Bus Filter Driver** (2)
- **Lower-Level Filter** (3)
- Function Driver (4)
- **Upper-Level Filter** (5)



WDM: Device Object

- **PDO**, *Physical Device Object*
- **FDO**, *Functional Device Object*
- **Filter DO**, *Filter Device Object*
 - *Upper-Level (filtro de entrada)*
 - *Class Filter Driver*
 - *Device Filter Driver*
 - *Lower-Level (filtro de saída)*
 - *Class Filter Driver*
 - *Device Filter Driver*



IRQL e IRP

Interrupt Request Level (IRQL)

- Defini o atual *Level* e mascara interrupcoes definidas em um nivel mais baixo de IRQ.

Interrupt Request Packet (IRP)

- Estrutura de dados que e repassa por cada componente registrado no do atravez do device stack.

WDM: Device Stack

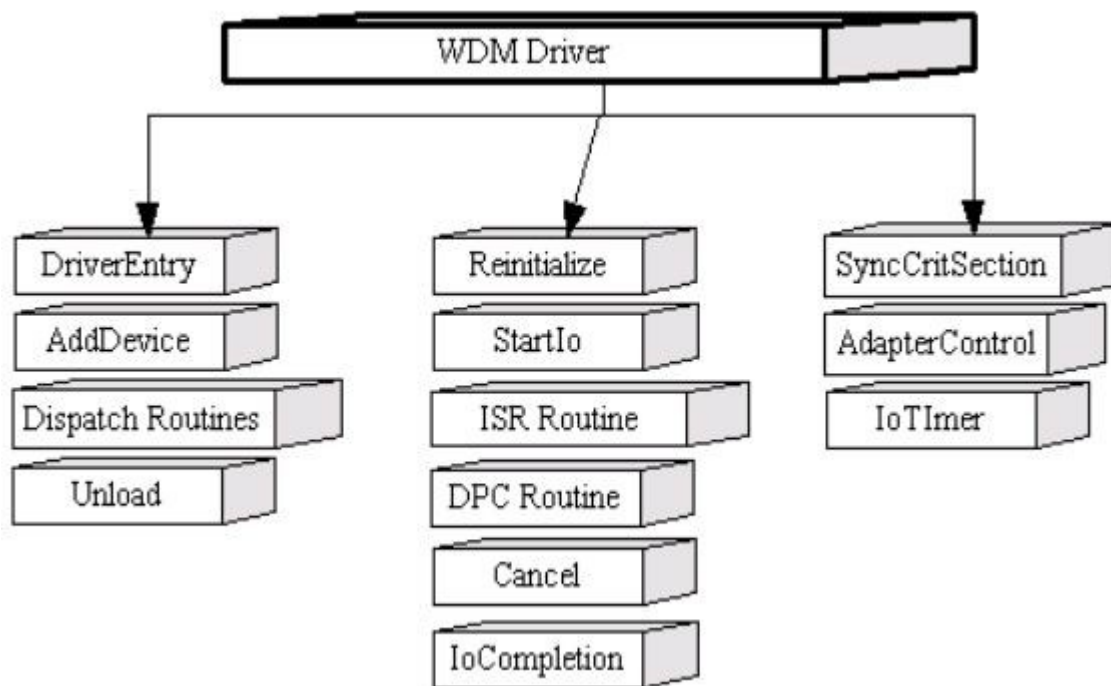
Driver adicionam-se ao Driver Stack que trata I/O referentes a um dado device criando um `DEVICE_OBJECT` (**IoCreateDevice**) e adicionando este ao device stack (**IoAttachDeviceToDeviceStack**).

`IoAttachDeviceToDeviceStack` determina o atual topo do stack e adiciona o novo objeto ao topo do device stack...

WDK: Exemplos

E como funciona?

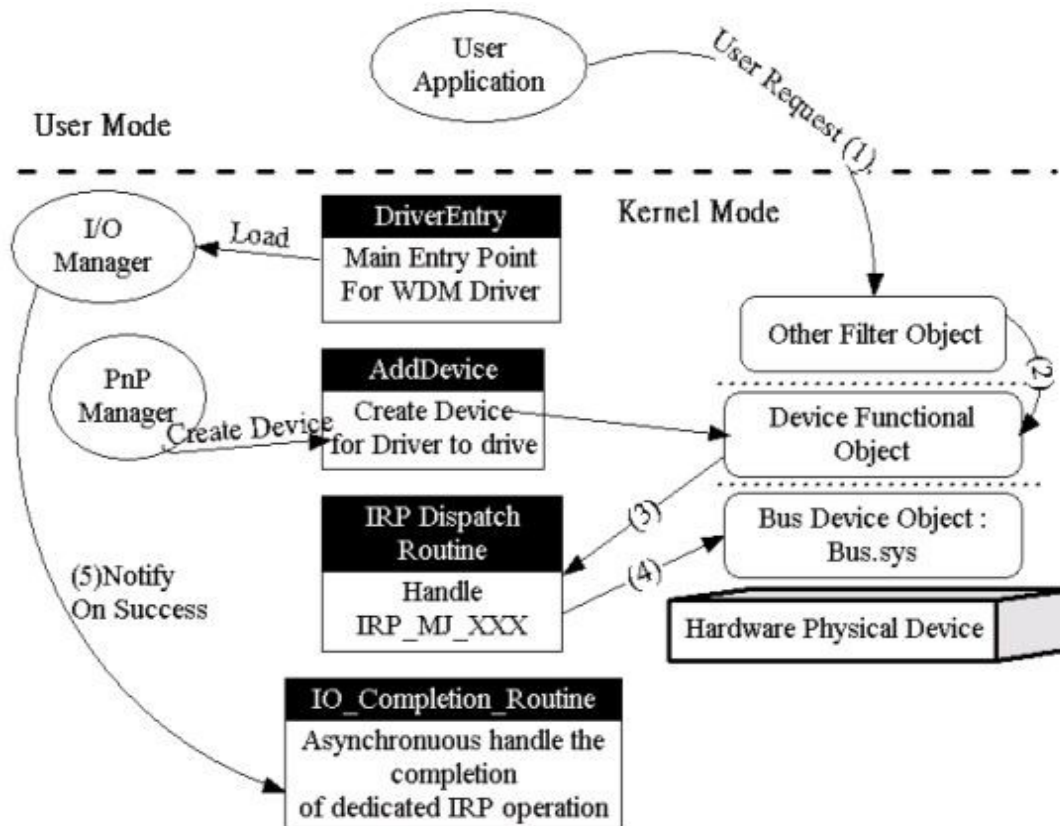
WDM: Funções Importantes



WDM: Funções Requeridas

- **DriverEntry (1)**, Inicializa driver e seu respectivo driver object.
- **AddDevice (2)**, Inicializa device e criar o device object do mesmo.
- **Dispatch Routines (3)**, Recebe e processa (Input and Output Request Packets (**IRPs**)).
- **Unload (4)**, Libera recursos utilizados pelo driver.

Workflow das Funções Requeridas



1: hello_world (DriverEntry)

```
#include <ntddk.h>
```

```
NTSTATUS
```

```
DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath) {  
    DbgPrint("Driver loading!\n");  
    return STATUS_SUCCESS;  
}
```

Compila e roda!

2: hello_world (AddDevice)

(...)

```
NTSTATUS DriverAddDevice(PDRIVER_OBJECT Driver) {  
    (...)  
    status = IoCreateDevice(Driver, sizeof(DEVICE_EXTENSION),  
        NULL, FILE_DEVICE_KEYBOARD, 0, FALSE, &device);  
    (...)  
    return STATUS_SUCCESS;  
}
```

NTSTATUS

```
DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath) {  
    DriverObject->DriverExtension->AddDevice = DriverAddDevice;  
    (...)  
}
```


3: hello_world (Dispatch...)

```
#include <ntddk.h>
```

```
(...)
```

```
NTSTATUS
```

```
DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath) {
```

```
    (...)
```

```
    Driver->MajorFunction[IRP_MJ_CLOSE]           = Example_Close;
```

```
    Driver->MajorFunction[IRP_MJ_CREATE]          = Example_Create;
```

```
    Driver->MajorFunction[IRP_MJ_READ]            = Example_Read;
```

```
    (...)
```

```
}
```

4: hello_world (Unloading)

```
#include <ntddk.h>
```

```
void
```

```
DriverUnload(PDRIVER_OBJECT pDriverObject) {
```

```
    DbgPrint("Driver unloading!\n");
```

```
}
```

```
(...)
```

```
NTSTATUS DriverEntry(PDRIVER_OBJECT Driver, PUNICODE_STRING RegPath) {
```

```
    DriverObject->DriverUnload = DriverUnload;
```

```
    (...)
```

```
    return STATUS_SUCCESS;
```

```
}
```

Aplicações Práticas

O que realmente interessa.

Motivação

Utilizar filtros, ferramentas legítimas, para coletar e transformar dados.

Exemplos: Gravar Teclado

- Vetores de ataques:
 - I8042ptr.sys (port driver)
 - kbdclass.sys (class driver)
- Solução:
 - *Upper Level **Class** Filter* em kbdclass.sys
 - *Upper Level **Device** Filter* em I8042ptr.sys

Exemplos: Esconder Arquivos

- Ataques
 - Interceptar IRP_MJ_DIRECTORY_CONTROL em um File System Driver e não retornar match para dado um nome.

Exemplos: Network Sniffers

Os mesmo drivers, utilizado para firewalls podem ser usados em sniffers?

- **Filter-Hook Driver**
 - “A filter-hook driver is kernel-mode driver that is used to filter network packets. Only a single filter-hook driver can be installed.”
- **TDI Driver**
 - “Transport Driver Interface allows you to write a new transport driver independent of the card.”
- **NDIS Driver**
 - “The Network Driver Interface Specification (NDIS) library abstracts the network hardware from network drivers.”

Referências

Caso deseje se aprofundar, procure por...

- Design A Kernel Key Log by Clandestiny
- Toby Opferman's documents about DDK
- And lots of papers and samples of...
 - codeproject.com
 - osronline.com
 - sysinternals.com

Perguntas??

Obrigado!!

**"Security is a process,
not a product."**

- Bruce Schneier