2013-11-1

Project II Brief View

Agenda

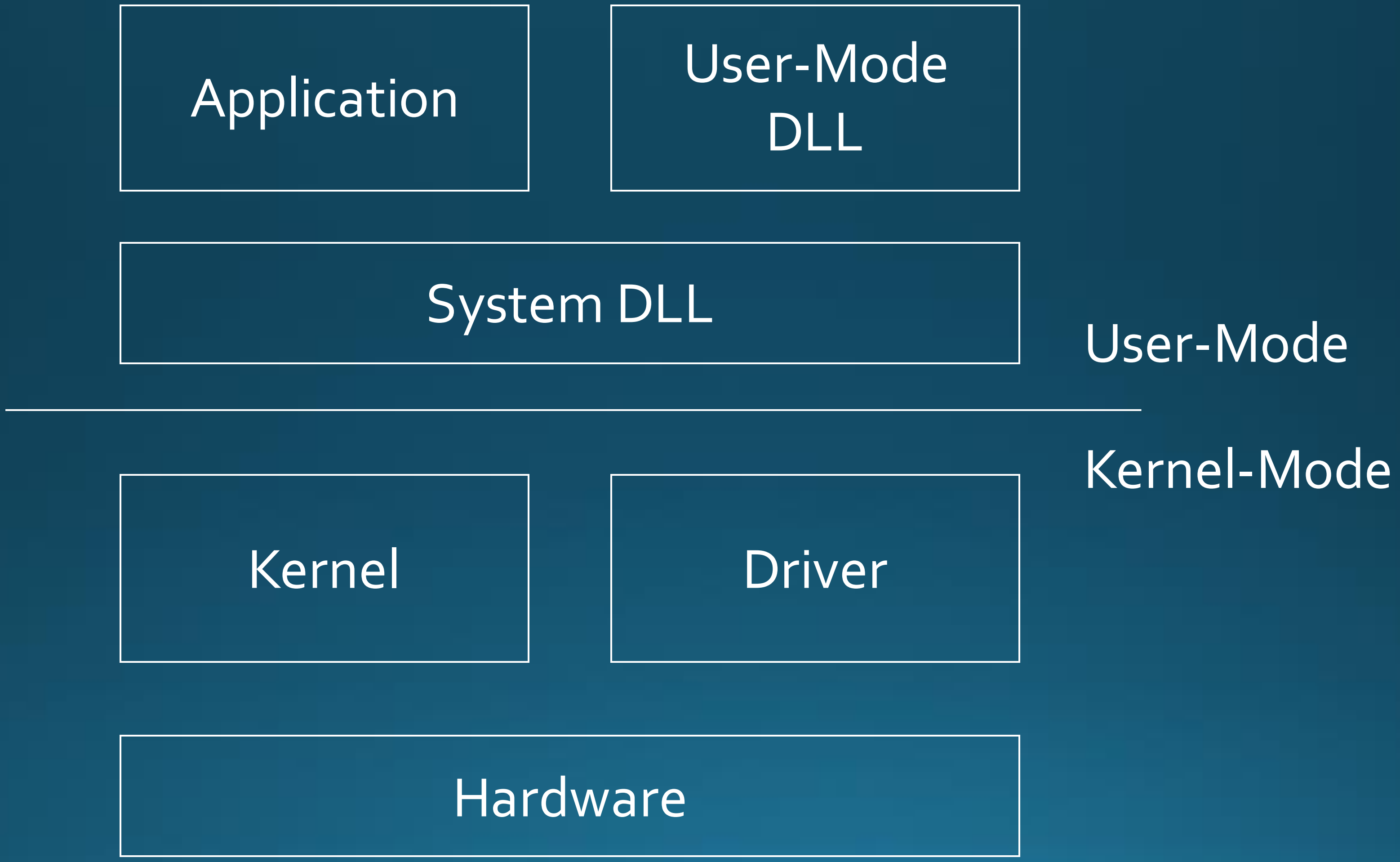
- Purpose of Project II
- Background Knowledge
- Data Structure of Windows Driver
- Basic Procedure of Writing a Driver
- Important Tools
- Demo
- Last Word

Purpose of Project II

- Further Understand user-mode & kernel-mode
- Get familiar with Windows Driver Development (also useful for driver development on other system)
- Get familiar with Windows Kernel

Background Knowledge

Windows OS


Application
User-Mode DLL
System DLL
User-Mode
Kernel-Mode
Kernel
Driver
Hardware

From Application to Driver

- Be careful about the tricks of C/C++
- No *Run Time Function* in Windows Driver ??
    - Example: C -> malloc; C++ -> new operator

- In one word: Program in kernel mode CANNOT call procedures in user mode.

- ** Demo

From Application to Driver

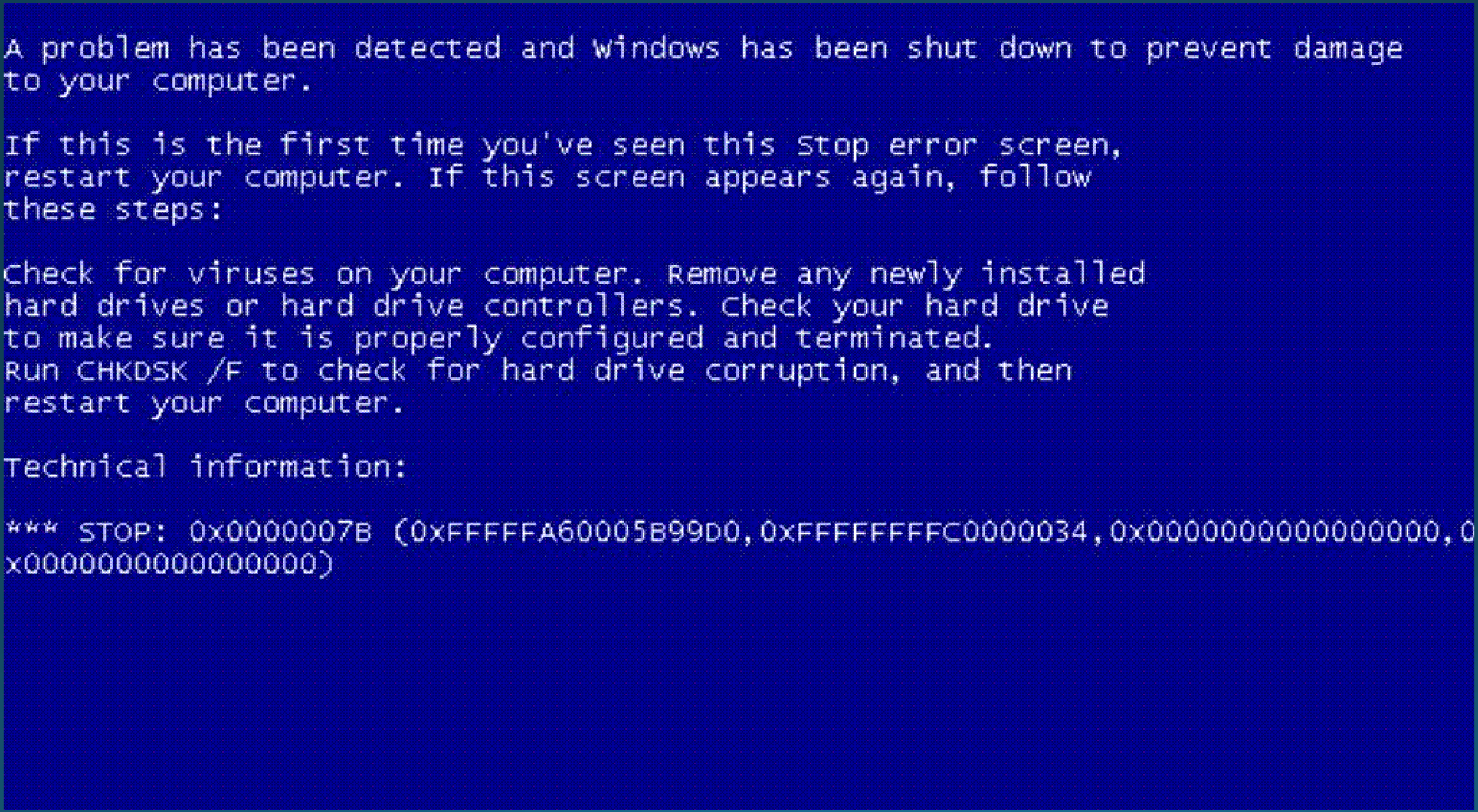
- Kernel Mode Runtime Functions
  - Form: RtlXXXX (Runtime Library XXXX)
  - Example:

```
VOID WINAPI RtlInitUnicodeString(
  _Inout_   PUNICODE_STRING DestinationString,
  _In_opt_  PCWSTR SourceString
);
```

- Some Runtime Functions do not rely on Win32 API
  - Example: strcpy

- Best Practice: Better to use DDK version Runtime Functions

Blue Screen



```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check for viruses on your computer. Remove any newly installed
hard drives or hard drive controllers. Check your hard drive
to make sure it is properly configured and terminated.
Run CHKDSK /F to check for hard drive corruption, and then
restart your computer.

Technical information:

*** STOP: 0x0000007B (0xFFFFFA60005B99D0,0xFFFFFFFFC0000034,0x0000000000000000,0
x0000000000000000)
```

Data Structure of Windows Driver

Data Structure of Windows Driver

- Driver Object (DRIVER_OBJECT)
- Device Object (DEVICE_OBJECT)
- Device Extension

Driver Object (DRIVER_OBJECT)

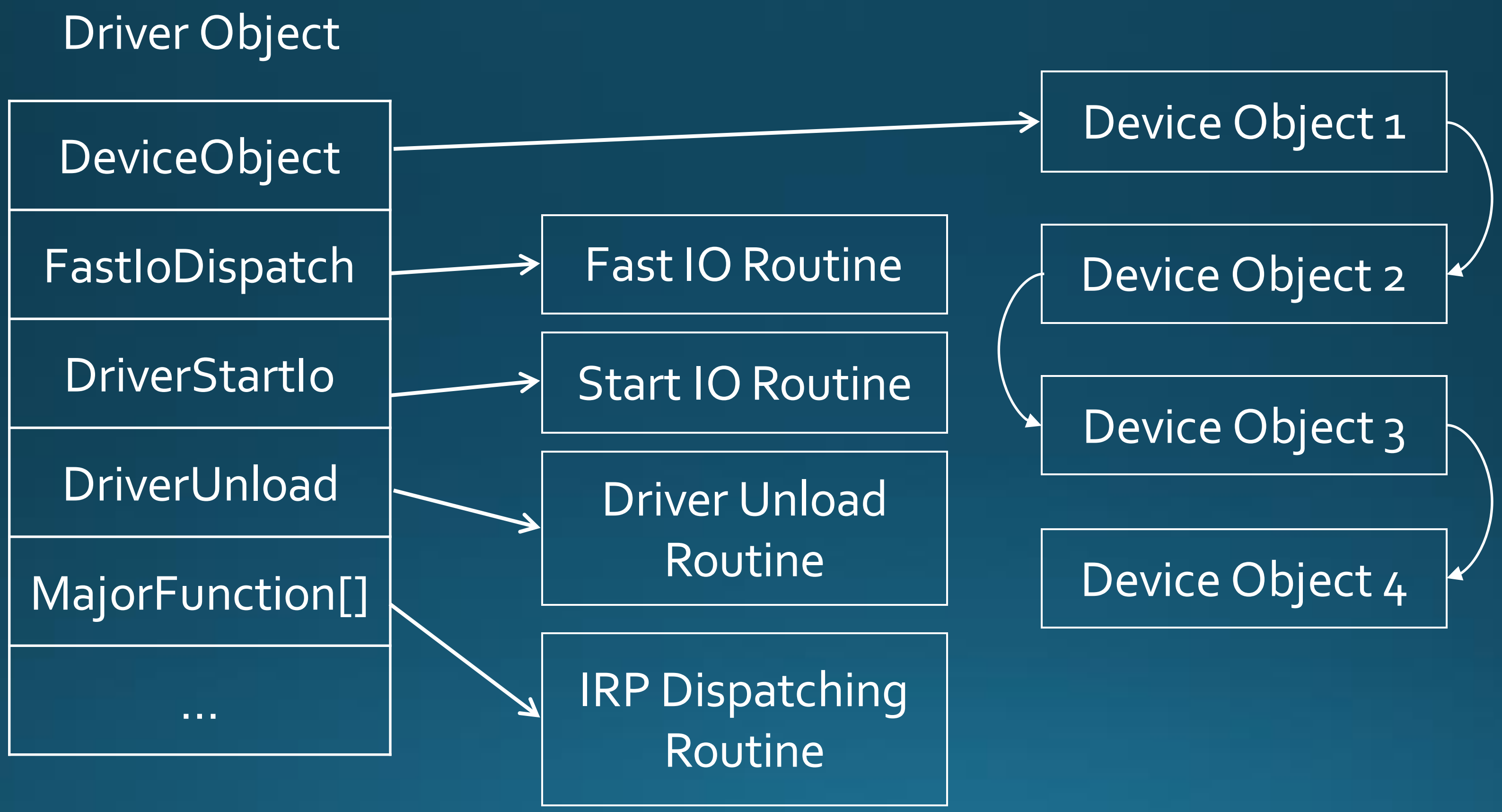
- Used to Describe a Driver

- For a certain driver, Kernel loads ONE & ONLY ONE instance, which is created by Object Manager.

- Loaded according to Registry
  OSR Loader: you need to register a driver first, then start a service for that driver.
  ** Demo

```
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;
typedef struct _DRIVER_OBJECT *PDRIVER_OBJECT;
```

Driver Object Layout



Device Object (DEVICE_OBJECT)

- DEVICE_OBJECT represents a logical, virtual, or physical device for which a driver handles I/O requests.
- One Driver can create one or more device object.
- Self Linked List (*NextDevice)

```
typedef struct DECLSPEC_ALIGN(MEMORY_ALLOCATION_ALIGNMENT)
            _DEVICE_OBJECT {
    CSHORT Type;
    USHORT Size;
    LONG ReferenceCount;
    struct _DRIVER_OBJECT *DriverObject;
    struct _DEVICE_OBJECT *NextDevice;
    struct _DEVICE_OBJECT *AttachedDevice;
    struct _IRP *CurrentIrp;
    PIO_TIMER Timer;
    ULONG Flags;                              // See above:  DO_...
    ULONG Characteristics;                        // See ntioapi:  FILE_...
    __volatile PVPB Vpb;
    PVOID DeviceExtension;
…
    USHORT SectorSize;
    USHORT Spare1;
    struct _DEVOBJ_EXTENSION  *DeviceObjectExtension;
    PVOID  Reserved;
} DEVICE_OBJECT;

typedef struct _DEVICE_OBJECT *PDEVICE_OBJECT;
```

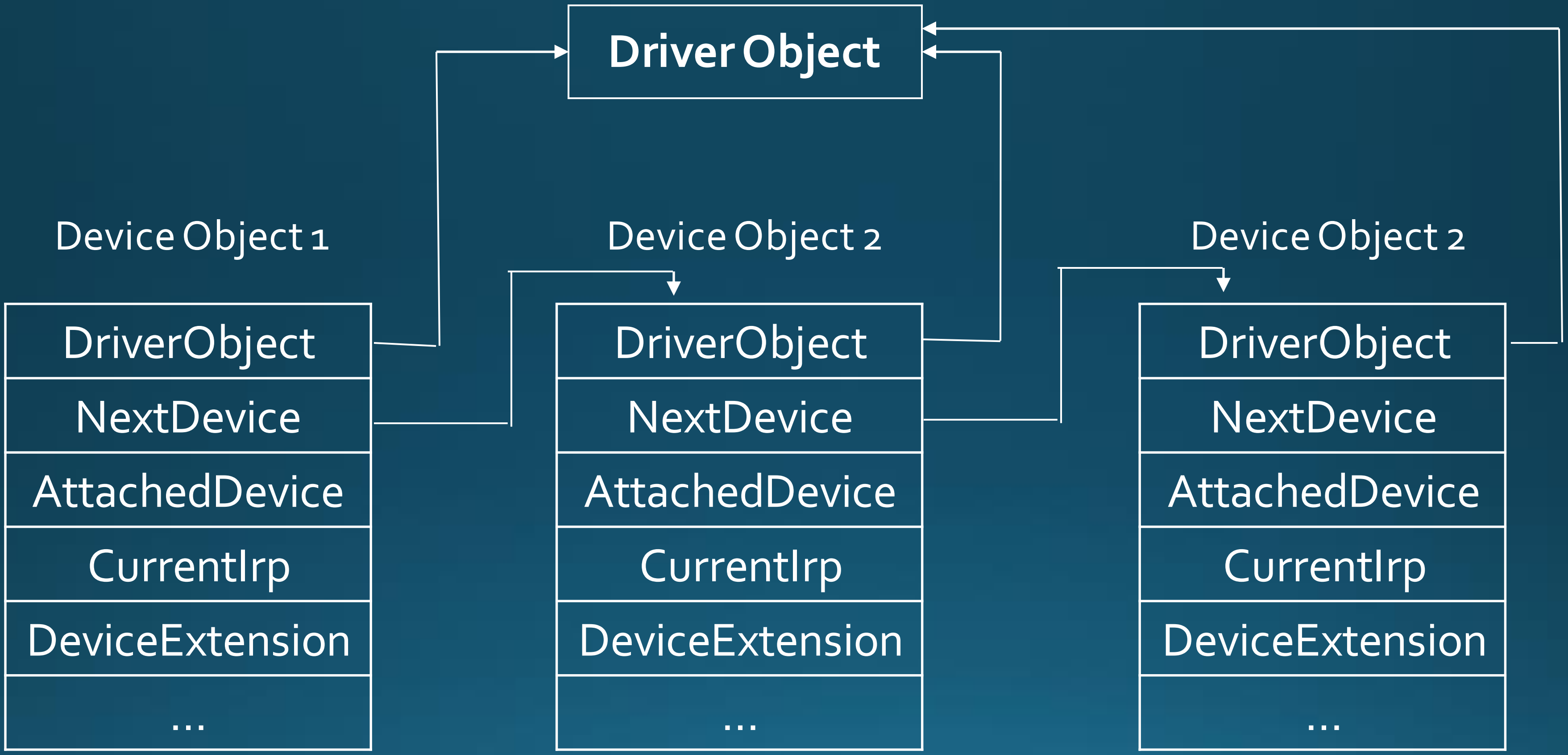
Device Extension (*DEVICE_EXTENSION)

- Device Object records GENERAL INFO.
- Device Extension records SPECIAL INFO.

- Defined by developer for specific use.
- Varies from driver to driver

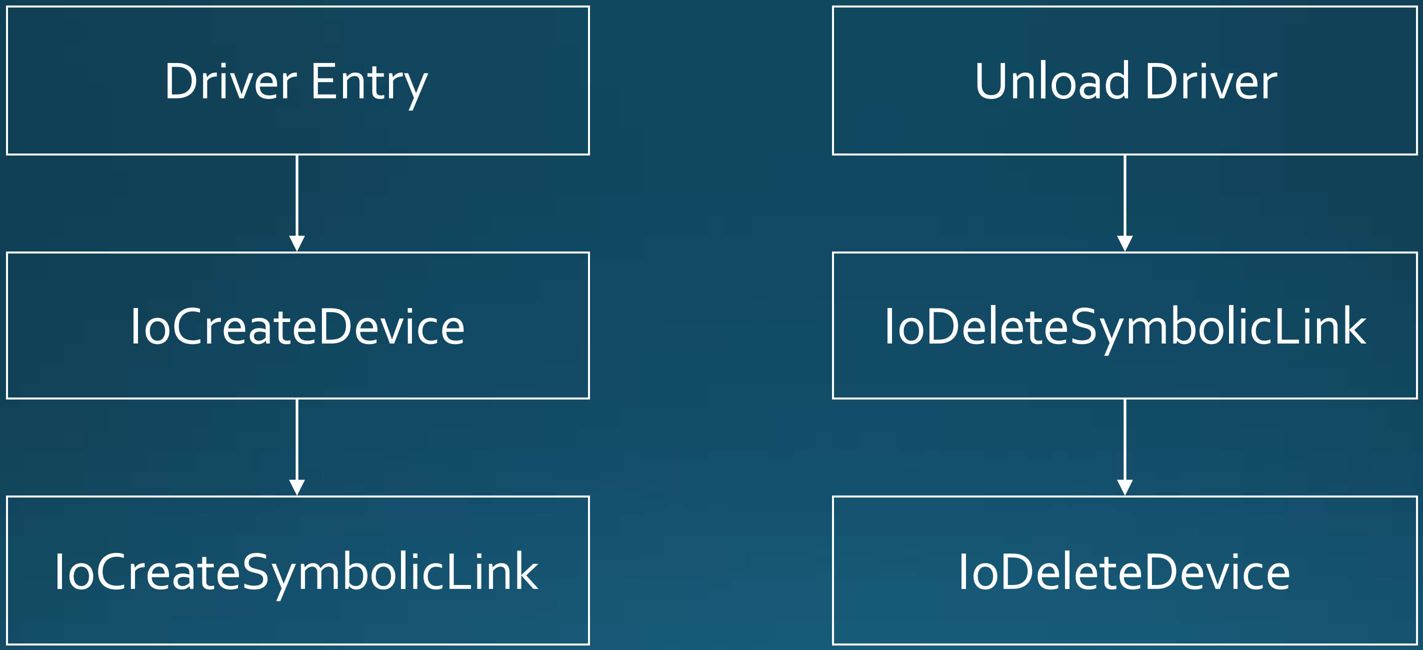
- Tips: Global Variables -> Concurrent Problem for Driver
  ** You can keep them in device extension, because device extension is private for each device.

Device Object Achitecture



Basic Procedure of Writing a Driver

Simple Work Flow of a Driver



Entry of a Driver

- Application
  `int main(int argc, char ** argv)`
  main function is the entry of common application

- Driver
  ```
  NTSTATUS DriverEntry(IN PDRIVER_OBJECT pDriverObject,
                       IN PUNICODE_STRING pRegistryPath)
  ```
  DriverEntry is the entry function for a windows driver.

```
typedef struct _UNICODE_STRING {
    USHORT Length;
    USHORT MaximumLength;
    PWCH   Buffer;
} UNICODE_STRING;
typedef UNICODE_STRING *PUNICODE_STRING;
```

Entry of a Driver (DriverEntry)

- Called by ***System Process***
- Initialize the driver program
- Be careful:
  if you use c++, please use <extern “C”> to modify DriverEntry.

- Additional Knowledge:
  _DriverEntry@8
  vs.
  ?DriverEntry@@YGJPAU_DRIVER_OBJECT@@PAU_UNICODE_STRING@@@Z

Required by Windows Driver
(C symbolic convention)
** Demo

IoCreateDevice Routine

```
NTSTATUS IoCreateDevice(
  _In_        PDRIVER_OBJECT DriverObject,
  _In_        ULONG DeviceExtensionSize,
  _In_opt_    PUNICODE_STRING DeviceName,
  _In_        DEVICE_TYPE DeviceType,
  _In_        ULONG DeviceCharacteristics,
  _In_        BOOLEAN Exclusive,
  _Out_       PDEVICE_OBJECT *DeviceObject
);
```

IoCreateSymbolicLink Routine

```
NTSTATUS IoCreateSymbolicLink(
  _In_  PUNICODE_STRING SymbolicLinkName,
  _In_  PUNICODE_STRING DeviceName
);
```

Important Tools

- WinDbg.exe
- WinObj.exe
- Debug View (used to monitor KdPrint)
- Windows Driver Kit (WDK) http://msdn.microsoft.com/en-us/windows/hardware/hh852365.aspx
- Visual Studio 2012 (with WDK8)
- WDK 8 redistributable components (a must for WDK8)
- Virtual Machine (VMWare, etc.)
- OSR Loader
- …

Demo

- Trace *new* operator
- Symbolic Link
- Hello Driver
- Driver Load & Unload (via OSR Loader & Debug Viewer)
  @see registry
  @see winobj
- Watch device-symbolLink mapping.

Last Word

Project II is just a start of Windows Driver Development