Unit 6: Device Management

6.2. The Windows I/O System Components

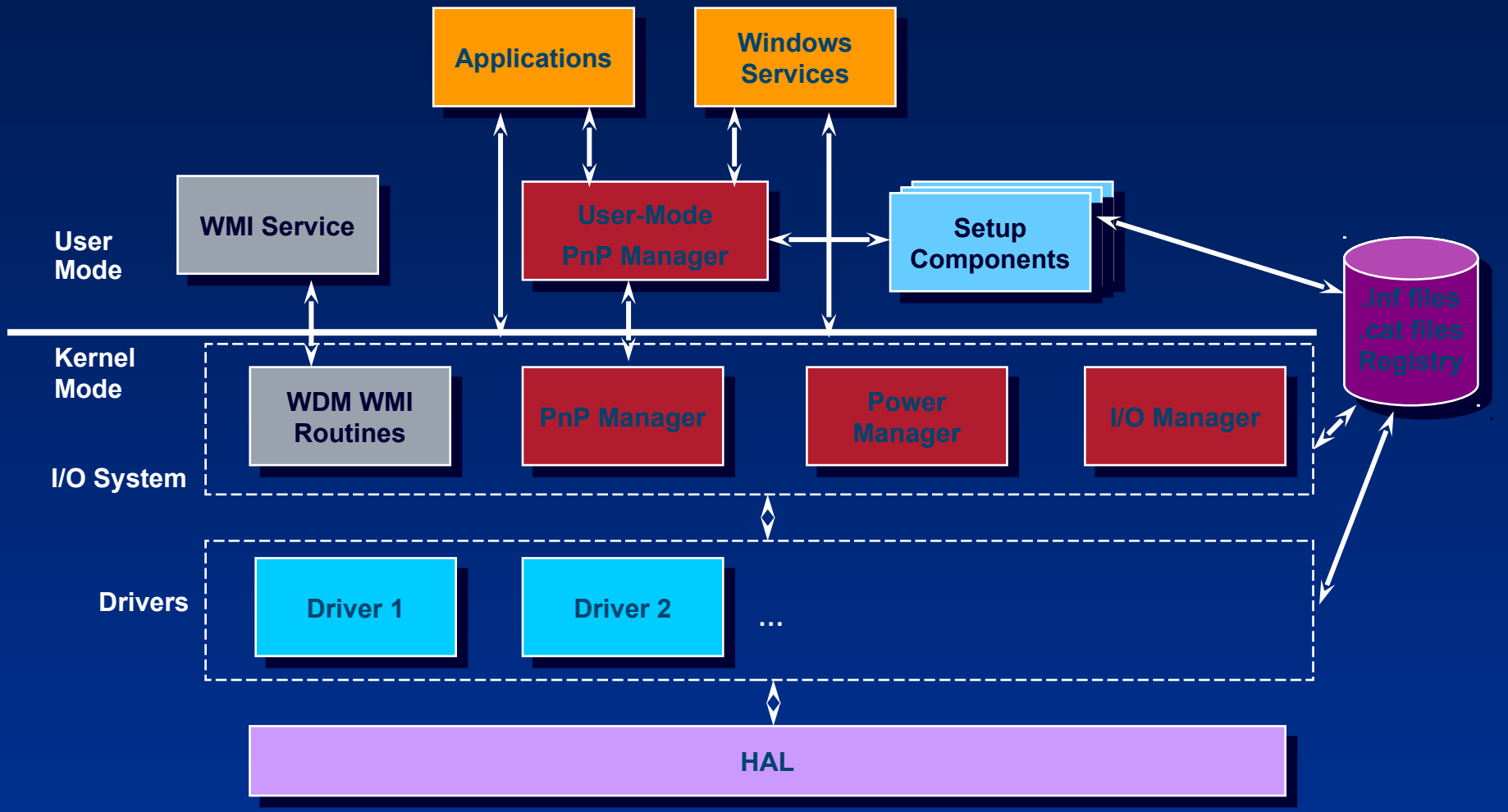
Roadmap for Section 6.2

- I/O System Components
- Functions of the I/O Manager
- Control flow for an I/O operation
- Plug and Play (PnP) and Power Manager
- Driver Verifier
- Windows Driver Model (WDM) classification

I/O System Design Goals

- Fast I/O processing on single / multiprocessor systems
- Protection for shareable resources
  - Using Windows security mechanisms
- Meet requirements dictated by different subsystems
- Provide common services for device drivers
  - Ease device driver development
  - Allow drivers to be written in high-level language
- Dynamic addition/removal of device drivers
- Support multiple file systems (FAT, CDFS, UDF, NTFS)
- Provide mapped file I/O capabilities
- Windows Management Instrumentation support and diagnosability
  - Drivers can be managed through WMI applications and scripts

I/O System Components



I/O System Components

- The I/O manager
  - Connects applications and system components to virtual, logical, and physical devices
  - Windows APIs: ReadFile, WriteFile, CreateFile, CloseFile, DeviceIoControl
  - Defines the infrastructure that supports device drivers
- A device driver typically provides an I/O interface for a particular type of device
  - Device drivers receive commands routed to them by the I/O manager that are directed at devices they manage, and they inform the I/O manager when those commands complete
  - Device drivers often use the I/O manager to forward I/O commands to other device drivers that share in the implementation of a device's interface or control.
  - Several types:
    - “ordinary”, file system, network, bus drivers, etc.
    - More information in I/O subsystem section

I/O Manager

- Framework for delivery of I/O request packets (IRPs)
- IRPs control processing of all I/O operations
  (exception: fast I/O does not use IRPs)
- I/O manager:
  - creates an IRP for each I/O operation;
  - passes IRP to correct drivers;
  - deletes IRP when I/O operation is complete
- Driver:
  - Receives IRP
  - Performs operations specified by IRP
  - Passes IRP back to I/O manager or to another driver (via I/O manager) for further processing

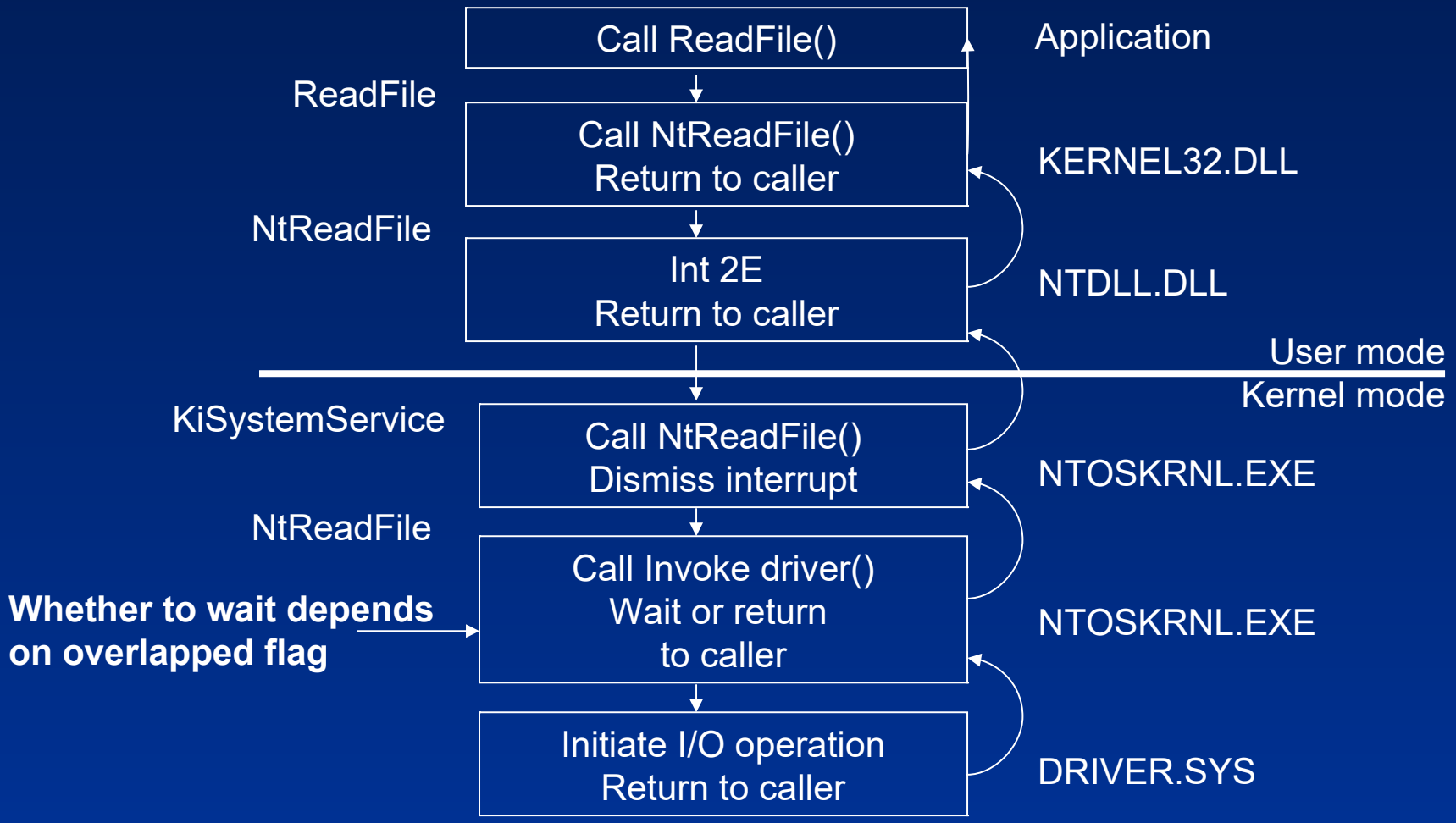
I/O Manager (contd.)

- Supplies common code for different drivers:
  - Drivers become simpler, more compact
- I/O manager:
  - Allows driver to call other drivers
  - Manages buffers for I/O requests
  - Provides time-out support for drivers
  - Records which installable file systems are loaded
  - Provides flexible I/O services to environment subsystems (Windows/POSIX asynchronous I/O)
- Layered processing of I/O requests possible:
  - Drivers can call each other (via I/O manager)

I/O Functions

- **Advanced features beyond open, close, read, write:**
- **Asynchronous I/O:**
  - May improve throughput/performance:
    continue program execution while I/O is in progress
  - Must specify FILE_FLAG_OVERLAPPED on CreateFile()
  - Programmer is responsible for synchronization of I/O requests
- **Internally, all I/O is performed asynchronously**
  - I/O system returns to caller only if file was opened for asynchronous I/O
  - For synchronous I/O, wait is done in kernel mode depending on overlapped flag in file object
- **Status of pending I/O can be tested:**
  - via Windows-API function: *HasOverlappedIoCompleted()*
  - when using I/O completion ports: *GetQueuedCompletionStatus()*

Control flow for an I/O operation



Advanced I/O Functions

- **Fast I/O**
  - Bypass generation of IRPs
  - Go directly to file system driver or cache manager to complete I/O
- **Mapped File I/O and File Caching**
  - Available through Windows-API CreateFileMapping() / MapViewOfFile()
  - Used by OS for file caching and image activation
  - Used by file systems via cache manager to improve performance
- **Scatter/Gather I/O**
  - Windows-API functions ReadFileScatter() / WriteFileScatter()
  - Read/write multiple buffers with a single system call
  - File must be opened for non-cached, asynchronous I/O;
    buffers must be page-aligned

HAL

- The hardware abstraction layer (HAL) insulates drivers from the specifics of the processor and interrupt controller by providing APIs that hide differences between platforms
  - in essence, the HAL is the bus driver for all the devices on the computer's motherboard that aren't controlled by other drivers
  - By programming to the HAL drivers are source-level compatible across CPU architectures

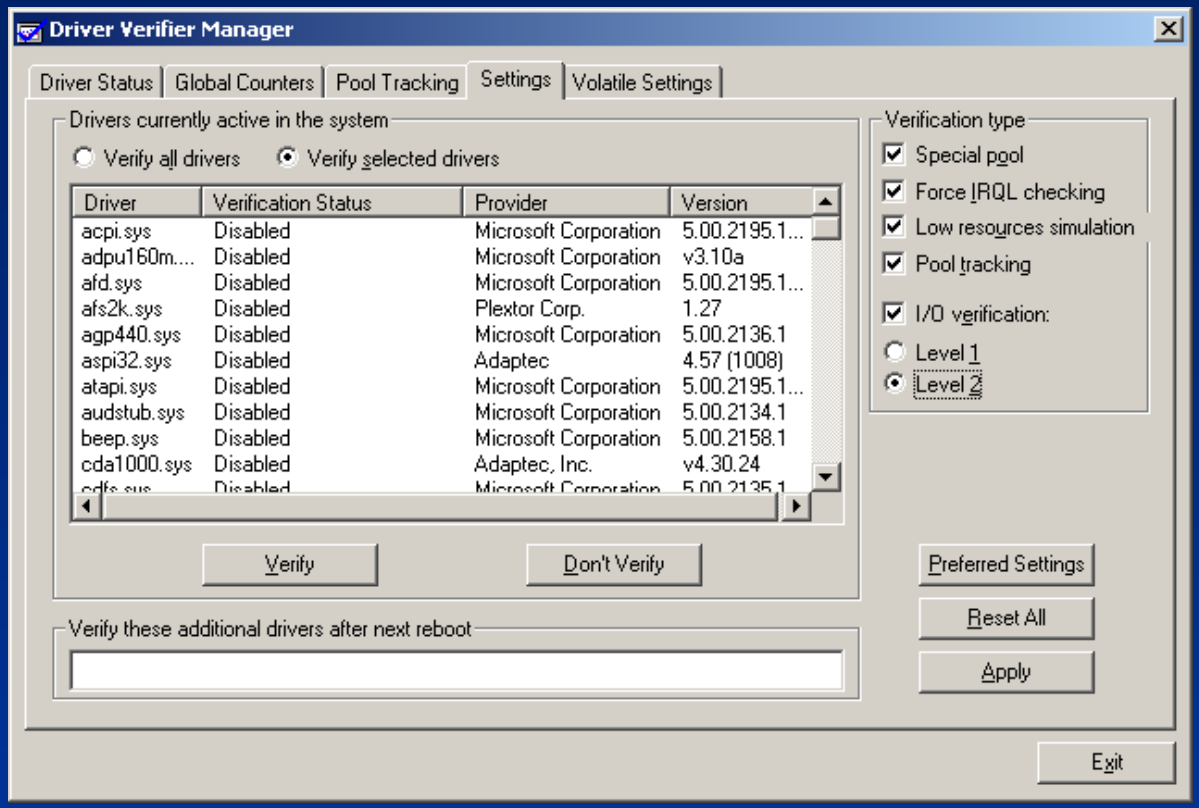
PnP and Power

- The PnP manager
  - Handles driver loading and starting
  - Performs resource arbitration
  - It relies on the I/O Manager to load drivers and send them PnP-related commands
- The power manager controls the power state of the system
  - It relies on the I/O Manager to ask drivers if they can change power state and to inform them when they should

The Driver Verifier

- Driver Verifier is a tool introduced in Windows 2000 that helps developers test their drivers and systems administrators identify faulty drivers
  - Must be run from \windows\system32\verifier.exe (no shortcut)

This is the Windows 2000 GUI to driver verifier

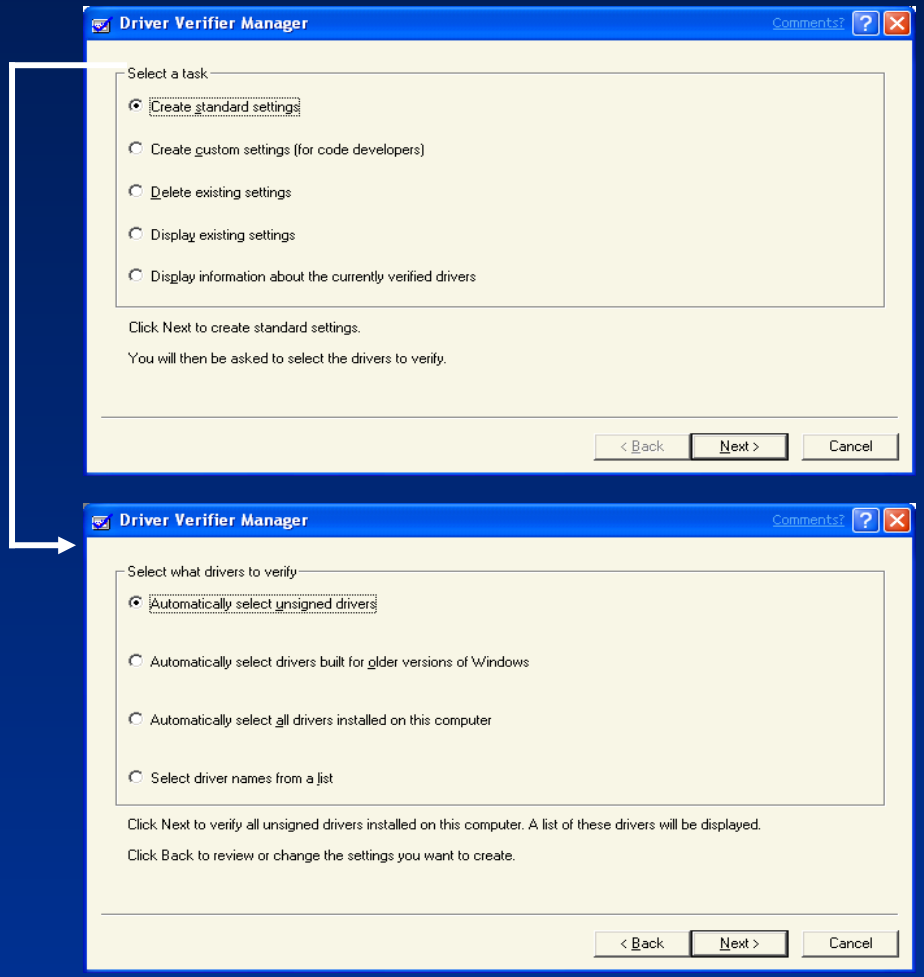


Verification Options

- Special Pool
  - The memory returned for driver memory allocations is bounded with invalid regions to catch buffer overrun and underrun
  - To be described in Crash Analysis section
- Force IRQL checking
  - Detects drivers that access paged memory when the system is in a state that can't tolerate page faults
- Low Resource Simulation
  - Randomly fails driver memory allocations
- Pool Tracking
  - Associates memory with the driver that allocated it to help identify leaks
- I/O verification
  - Ensures that I/O commands are properly formatted and handled

Driver Verifier – XP/Server 2003 Enhancements

- Simpler wizard-style UI
  - Default is verify unsigned drivers
- Four new verification options in XP:
  - DMA verification – detects improper use of DMA buffers, adapters, and map registers
  - Deadlock detection – detects lock hierarchy violations with spinlocks, mutexes, fast mutexes
  - SCSI verification - monitors the interaction between a SCSI miniport driver and the port driver
  - Enhanced I/O Verification tests drivers' support for power management, WMI, and filters
- One new in Server 2003:
  - Disk integrity checking - monitors a hard disk and detects whether the disk is preserving its data correctly



(this is the Windows XP/2003 GUI)

Kernel-Mode Drivers

- *Windows kernel-mode drivers*
  - PnP Drivers: Integrate with the power manager and PnP manager
    - Mass storage devices
    - Input devices
  - Non PnP Drivers: Don't have to integrate with the PnP manager
    - Protocol stacks
    - Network adapters
    - Virtual devices (Filemon, Regmon)
- *File system drivers* accept I/O requests to files and satisfy the requests by issuing their own, more explicit requests to mass storage or network device drivers
  - Interact closely with Memory Manager and Cache Manager

User-Mode Drivers

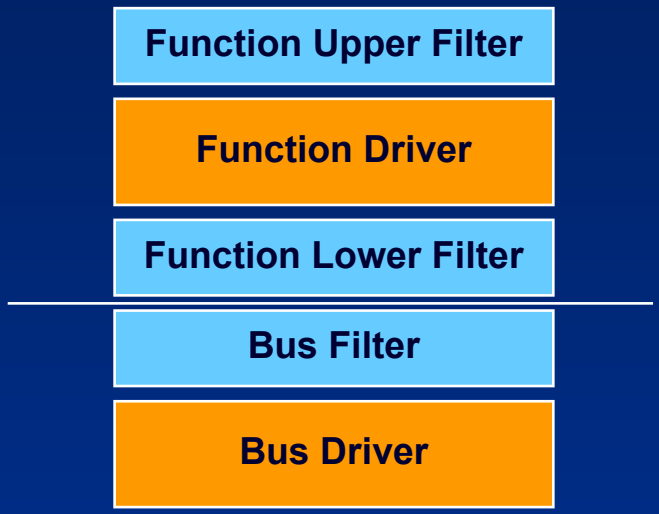
- *Virtual device drivers* (VDDs) are used to emulate 16-bit MS-DOS applications.
  - User-mode can't access hardware directly and thus must go through a real kernel-mode device driver.
  - They trap what an MS-DOS application thinks are references to I/O ports and translates them into native Windows I/O functions
- Windows subsystem *printer drivers* translate device-independent graphics requests to printer-specific commands.
  - Commands are forwarded to a kernel-mode port driver such as the parallel port driver (Parport.sys) or the universal serial bus (USB) printer port driver (Usbprint.sys)

WDM Driver Classification

- Windows Driver Model
  - Unified architecture for drivers
  - Originally intended to be Win9x/NT cross platform
  - Most PnP Drivers are WDM drivers
- There are three types of WDM drivers:
  - *Bus drivers* manage a logical or physical bus e.g. PCMCIA, PCI, …
  - *Function drivers* manage a particular type of device. Bus drivers present devices to function drivers via the PnP manager.
  - *Filter drivers* logically layer above or below function drivers, augmenting or changing the behavior of a device or another driver.

WDM Driver Classification

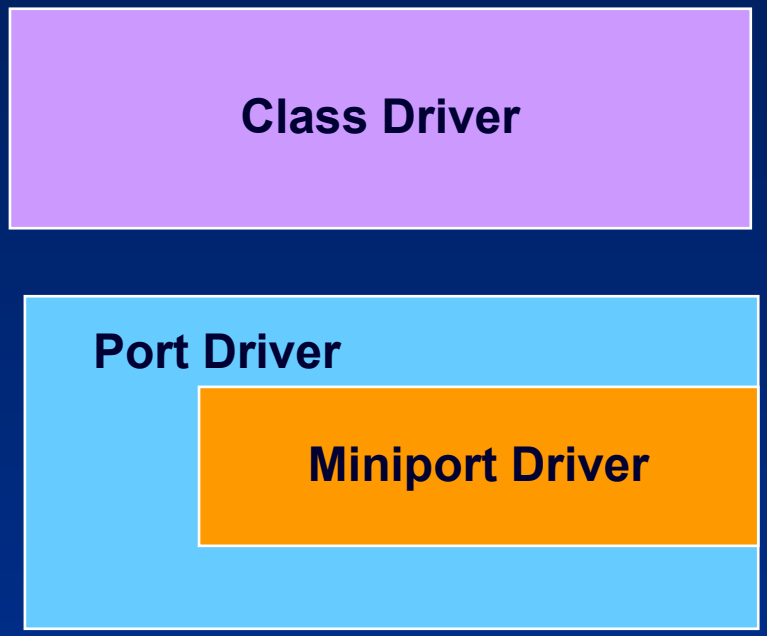
- In WDM, no one driver is responsible for controlling all aspects of a particular device.



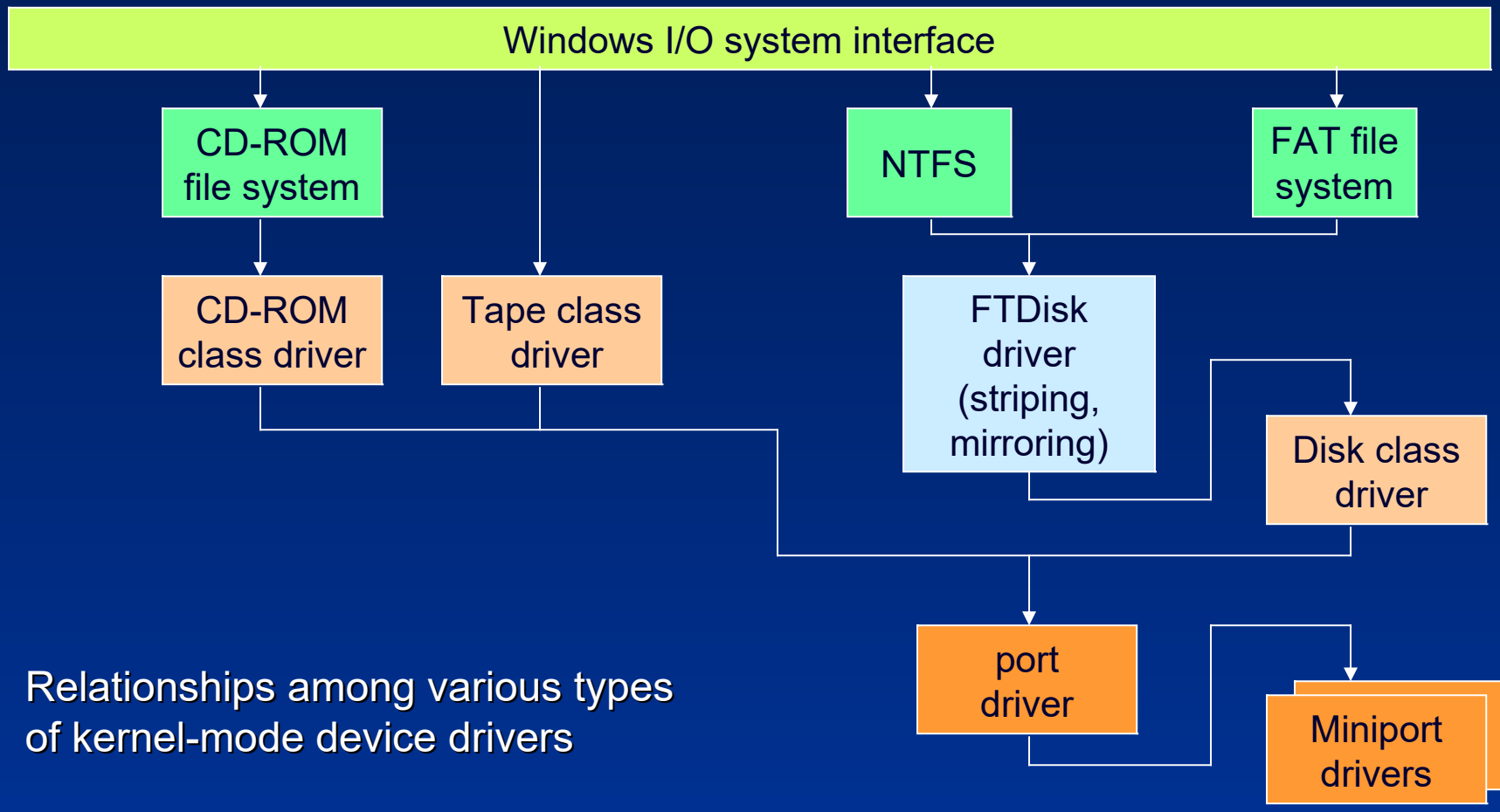
Class/Port/Miniport Driver Classification

- Hardware support might be split between different modules that implement support for different levels of abstraction
  - Microsoft typically provides the drivers for the higher levels of abstraction
  - Hardware vendors provide the lowest level, which understands a particular device
- The conventional division is three levels:
  - *Class drivers* implement the I/O processing for a particular class of devices, such as disk, tape, or CD-ROM.
  - *Port drivers* implement the processing of an I/O request specific to a type of I/O port, such as SCSI, and are also implemented as kernel-mode libraries of functions rather than actual device drivers.
  - *Miniport drivers* map a generic I/O request to a type of port into an adapter type, such as a specific SCSI adapter. Miniport drivers are actual device drivers that import the functions supplied by a port driver.

Class/Port/Miniport Driver Classification

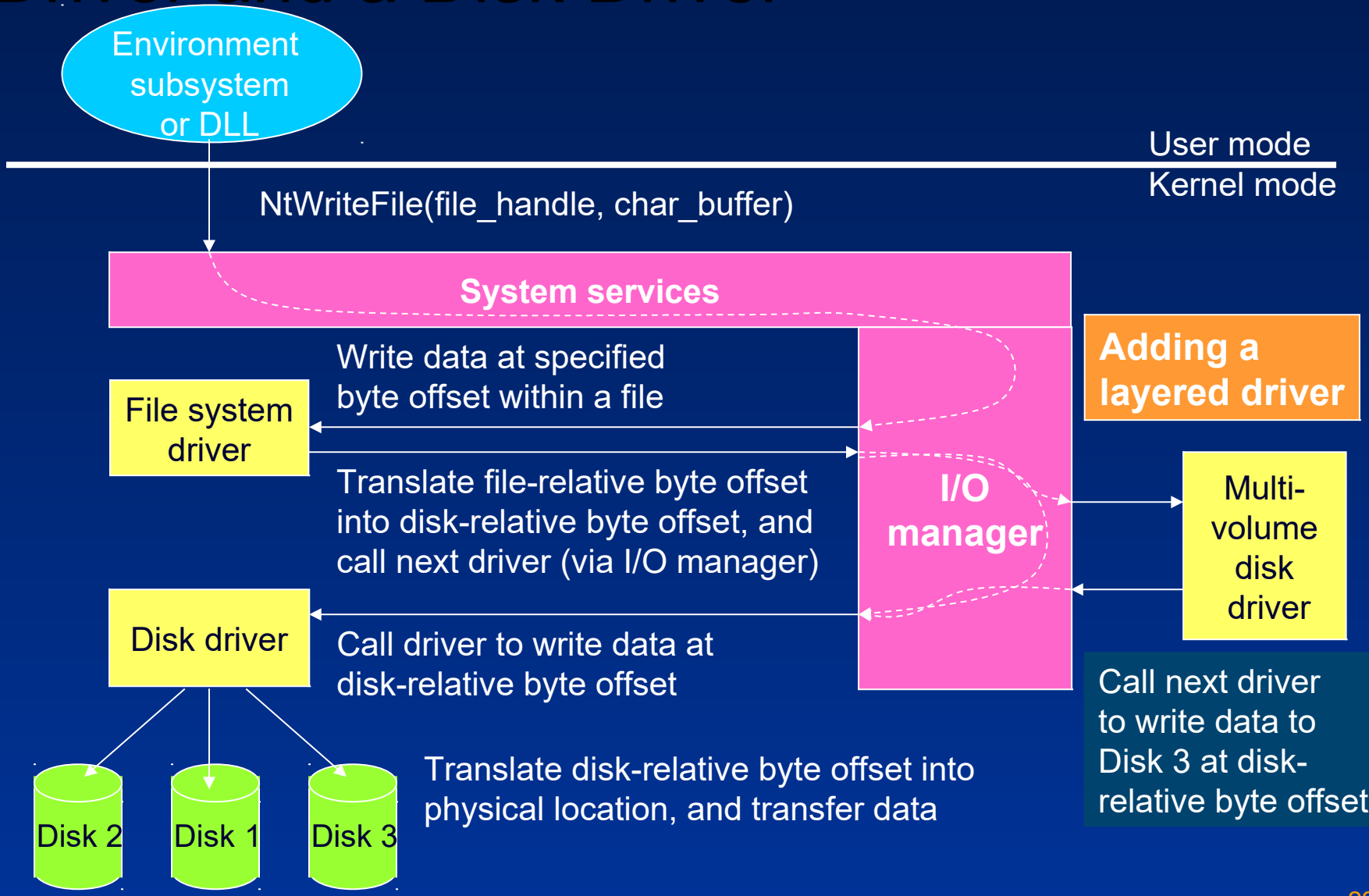


Layered Driver Structure

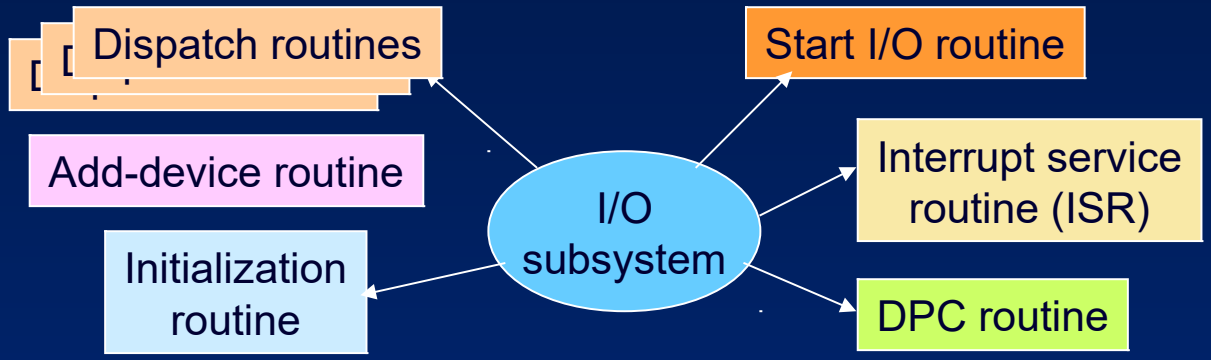


Relationships among various types of kernel-mode device drivers

Dynamically Layering a File System Driver and a Disk Driver



Internal Structure of a Driver



- I/O manager executes initialization routine when loading a driver
- PnP manager calls add-device routine on device detection
- Dispatch routines: open(), close(), read(), write()
- Start I/O routine initiates transfer from/to a device
- ISR runs in response to interrupt; schedules DPC
- DPC routine performs actual work of handling interrupt; starts next queued I/O operation on device

Other components of device drivers

- **Completion routines**
  - A layered driver may have completion routines that will notify it when a lower-level driver finishes processing an IRP (**I**/O **R**equest **P**acket)
- **Cancel I/O routine**
- **Unload routine**
  - Releases system resources
- **System shutdown notification routine**
- **Error-logging routines**
  - Notify I/O manager to write record to error log file (e.g., bad disk block)
- **Windows Driver Model (WDM)**
  - Plug & Play support
  - Source compatible between Win9x/NT

Further Reading

- Pavel Yosifovich, Alex Ionescu, et al., “Windows Internals”, 7th Edition, Microsoft Press, 2017.
  - Chapter 6 – I/O system (from pp. 669)
    - I/O system components (from pp. 669)
    - Typical I/O processing (from pp. 673)
    - Types of device drivers (from pp. 682)
    - Driver objects and device objects (from pp. 694)