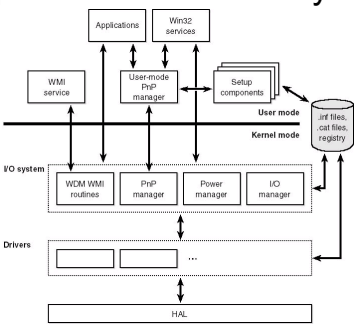
Windows 2000 - The I/O Structure

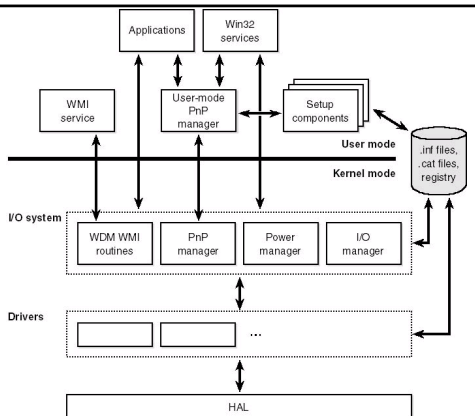
„Ausgewählte Betriebssysteme“
Institut Betriebssysteme
Fakultät Informatik

Outline

- Components of I/O System
- Plug‘n‘Play Management
- Power Management
- I/O Data Structures
  - File Object
  - Driver Object
  - Device Object
  - IRPs / DPCs

Components of the I/O System





Overview

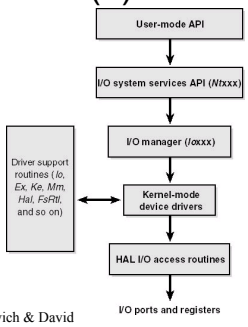
- I/O Manager:
  - Connects applications and system components to devices
  - Defines infrastructure
- Device Driver:
  - Provides I/O interface for particular device
- PnP Manager:
  - Guide allocation of HW resources
  - Detect and respond to arrival and removal of devices
- Power Manager:
  - Guides system and device drivers through power-state transitions

Overview (2)

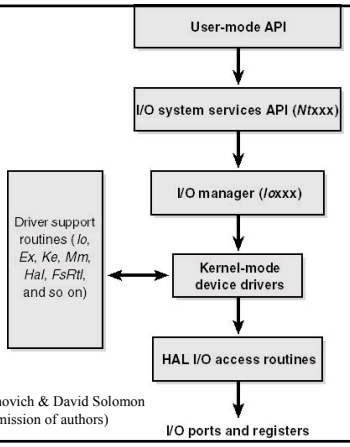
- Windows Management Instrumentation (WMI):
  - Support routines, which allow drivers to act as Windows Driver Model (WDM) WMI providers
- Registry:
  - Stores description of basic hardware devices
- INF files:
  - Driver installation files
  - Link hardware device and driver
- HAL:
  - Provides API to hide platform differences
  - Is bus driver for driver-less devices on motherboard

Overview (3)

- I/O is performed on virtual files
- All data is regarded as stream of bytes
- Typical flow of I/O request (see picture)



picture © Mark Russinovich & David Solomon (used with permission of authors)



I/O Manager

- I/O system is packet-driven (I/O request packet – IRP)
- Creates IRP and passes reference to driver and disposes it after completion
- Driver receives IRP, performs specified action and passes IRP back
- Supplies common code (e.g. Call another driver, time-out support, ...)
- Uniform, modular interface of drivers allows I/O manager to call driver without special knowledge about driver

Device Drivers

- Must conform to implementation guidelines
- Kernel mode device drivers
  - File system drivers (direct file I/O requests to mass storage)
  - Windows 2000 drivers (mass storage, protocol stacks, ...)
  - Legacy drivers (NT driver w/o power mngt., PnP)
  - Win32 subsystem display drivers (translate device independent graphic into device dependant)
  - WDM drivers (adhere to WDM = W2K drivers + WMI)
- User mode device drivers
  - Virtual device drivers: emulate 16-bit MS-DOS applications
  - Win32 subsystem printer drivers

WDM drivers

- Source code compatible between Windows 2000, Windows 98, Windows Me, Windows XP
- Bus drivers:
  - Manage logical or physical bus
  - Responsible for detecting devices and reporting to PnP manager
  - Manages power settings of bus
- Function drivers:
  - Exports operational interface of device to OS
- Filter driver:
  - Layer above or below function drivers

Functional separation

- Class driver:
  - implement I/O processing for specific class of devices (disk, tape, ...)
- Port driver:
  - implement I/O processing for specific port type (SCSI, ...)
  - Mostly implemented as kernel-mode library
- Miniport driver:
  - Map generic I/O request to type of port into adapter type

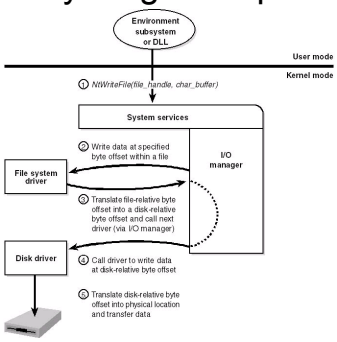
Layered Device Drivers

- “normal” layered drivers connect to lower and higher driver
- transparent drivers slip in between lower-layer driver and its clients: has to mimic behavior of lower driver
- virtual or logical device layers: expose virtual or logical device objects (e.g. named pipe)

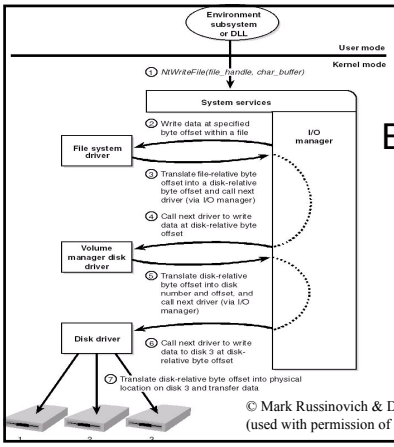
Layered Device Drivers (2)

- Call lower layer driver:
  - retrieve stack location for next lower driver
  - set up stack location (function code, parameters)
  - associate completion routine with IRP to be informed when lower driver completes
  - send IRP to lower driver (is asynchronous call)
  - mark IRP pending
- Complete request
- Create new IRPs to pass to lower drivers

Layering example



Example (2)



Driver Object

- Unique object for each loaded driver
- Consist of set of routines
- Routines process various stages of an I/O request e.g. load, unload, start I/O, etc.
- Linked list of device object, serviced by driver
- I/O Manager uses “back pointer” to find driver for object
- Driver removes device objects during unload

Key Device Driver Routines

- Initialization routine (when loading a driver)
- Add-device routine (PnP)
- Dispatch routines (process IRPs)
- Start I/O routine (initiate data transfer via I/O manager)
- Interrupt service routine (ISR – executed when interrupt occurs, queues DPC)
- Interrupt-servicing DPC routine (DPC: deferred procedure call = interrupt handling at lower IRQL)

Key Device Driver Routines (2)

- I/O completion routines (used to notify stacked drivers of I/O completion)
- Cancel I/O routine (cancel IRP processing)
- Unload routine (release system resources)
- System shutdown notification routine (called before system shuts down)
- Error-logging routine (write errors to error-log)

Outline

- Components of I/O System
- Plug‘n‘Play Management
- Power Management
- I/O Data Structures
  - File Object
  - Driver Object
  - Device Object
  - IRPs / DPCs

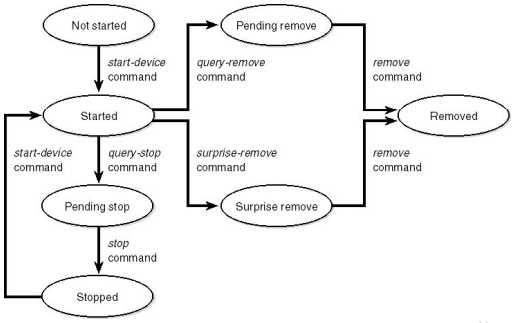
Plug and Play Manager

- Automatically recognizes installed devices (enumerates devices during boot process, detect addition or removal)
- Hardware resource allocation and reallocation for devices (interrupts, DMA addresses, etc.)
- Load appropriate drivers:
  - Determine driver and instruct I/O Manager to load
  - If none found install driver via user-mode PnP manager → “Found new Hardware”

P‘n‘P Manager (2)

- Implements mechanisms for applications and drivers to detect configuration changes (notification)
- Support level depends on attached devices and installed drivers

Device PnP state transitions



P‘n‘P state transitions

- Pending remove:
  - Finish pending I/O requests
  - Do not accept further I/O requests
  - Power down device
- Surprise remove:
  - Immediately stop running I/O requests
  - Remove pending I/O requests

Outline

- Components of I/O System
- Plug‘n‘Play Management
- Power Management
- I/O Data Structures
  - File Object
  - Driver Object
  - Device Object
  - IRPs / DPCs

Power Management

- Requires hardware that complies with Advanced Configuration and Power Interface (ACPI) specification
- 6 system power states
- Power manager request driver to move to other power state
- If driver is busy it rejects request → system stays at current power level

Power Management (2)

- Multiple driver objects associated with one device:
  - Only one driver designated as device power-policy owner
  - Decides device’s power state based on system power state
  - Asks power manager to inform other drivers
  - Thus power manager can control number of power commands in system (e.g. number of devices powering up)

ACPI standard and Win2K

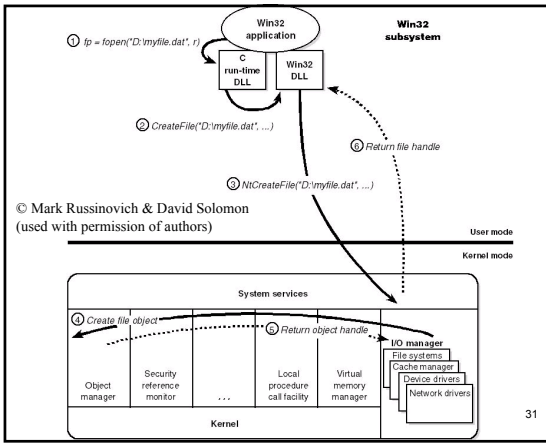
- Defines various power levels for:
  - System: S0 (fully on) – S5 (fully off)
  - Devices: D0 (fully on) – D3 (fully off)
- D1 and D2 are free to be defined by device
- Driver reports supported power level to PnP manager at load time
- Bus driver provides mapping from system power states to device power states

Outline

- Components of I/O System
- Plug‘n‘Play Management
- Power Management
- I/O Data Structures
  - File Object
  - Driver Object
  - Device Object
  - IRPs / DPCs

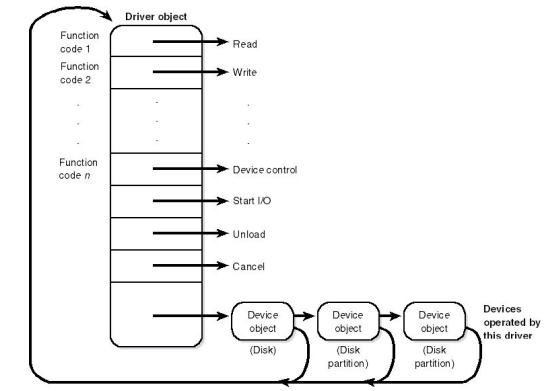
File objects

- Kernel-mode construct for handles to files or devices
- Provide memory-based representation of resources which can be read or written
- Protected by security descriptor (including ACL)
- Contains data unique to object handle (byte offset, ...)
- Every time a new file handle is opened a new file object is created
- File object is unique to process, except:
  - Child inherits from parent
  - Duplicate handle to another process



Device and Driver Objects

- Driver object:
  - Represents an individual driver
  - I/P manager obtains address of dispatch routines from it
- Device object:
  - Represents physical or logical device and its characteristics (e.g. buffer alignment and location of device queue for IRPs)
- Driver creates device object when PnP manager informs it
- Driver may export a name for device to allow applications to open it
- One driver can have multiple device objects
- Device objects have back-link to driver

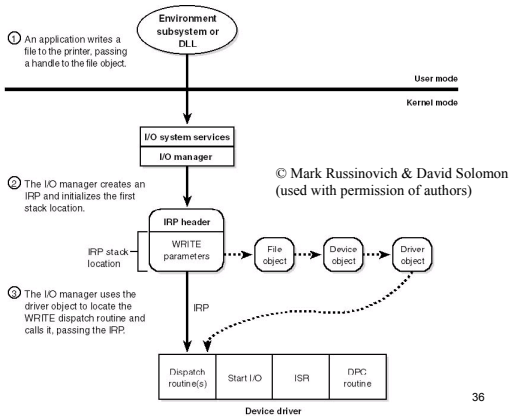


IRP

- Header contains (fixed part):
  - Type and size of request
  - Synchronous or asynchronous request
  - Pointer to buffer for buffered I/O
  - State information

IRP (2)

- Stack location contains (one or more):
  - Function code and parameters (identifies driver‘s dispatch routine)
  - Pointer to caller‘s file object
  - Used to find “answer point” for layered drivers (I/O completion routine)
  - Number of stack locations determined by number of driver layers



IRP Buffer Management

- I/O Manager perform three types of I/O buffer management:
  - Buffered I/O: I/O manager allocates memory from non-paged pool and copies data from/to user's buffer
  - Direct I/O: user's buffer is locked by I/O manager (DMA)
  - Neither I/O: I/O manager does not do any buffer management (driver might do it)
- Driver registers type of buffer management in device object

IRP Buffer Management (2)

- Drivers usually use buffered I/O if smaller than one page (4KB)
  copy operation = overhead of memory lock
- File systems driver usually use Neither I/O, because no buffer management needed for copy from/to file system cache

I/O Completion Ports

- Introduced for servers with multiple parallel threads
- Threads wait for I/O packets to arrive at port
- System controls number of currently running threads (should be the same as number of processors)
- Can be regarded as thread pool

Driver Loading/Initialization

- Enumeration-based loading: PnP manager dynamically loads drivers for devices that bus driver reports
- Explicit loading: is guided by HKLM\SYSTEM\CurrentControlSet\Services
- Registry key „Start" value:
  - 0 – at boot time (loaded by System)
  - 1 – after initialization (loaded by I/O manager)
  - 2 – auto-start (after System started)
  - 3 – demand-start (started when first called)

Types of I/O

- Synchronous I/O:
  - Application waits for I/O to complete
  - I/O manager mimics synchronous behavior to application
- Asynchronous I/O:
  - Allows application to continue
  - APC is queued after completion
  - Has to synchronize with completion of I/O using a synchronization object

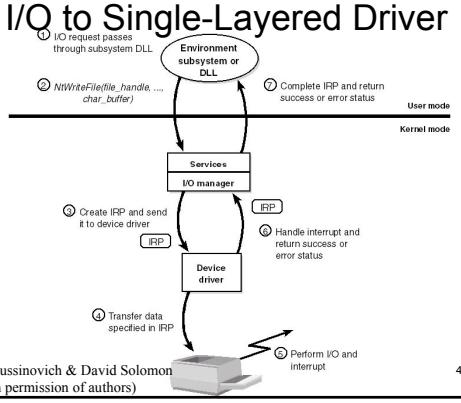
Types of I/O (2)

- Fast I/O:
  - Special mechanism, that bypasses IRP generation
  - Entry points have to be registered in driver object
  - Used to signal completed I/O request
  - E.g. used by File System to check if file is in file cache

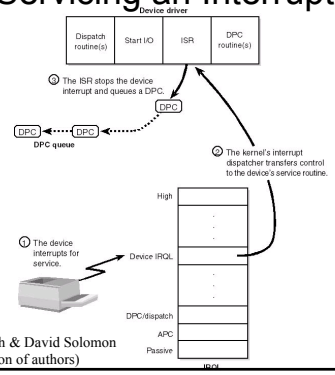
Types of I/O (3)

- Mapped File I/O and File Caching
  - File (or parts of it) loaded into memory, and paging mechanism does I/O
  - Mapped Files used by Cache Manager
- Scatter/Gather I/O
  - Allow single read or write from multiple buffers in memory to a contiguous area on disk
  - Buffers have to be page aligned
  - I/O must be aligned on device sector boundary

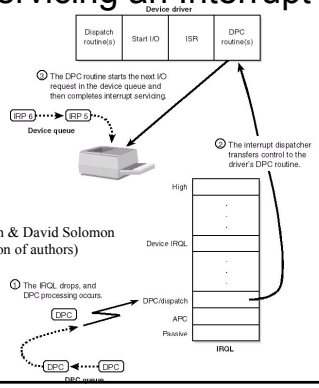
I/O to Single-Layered Driver



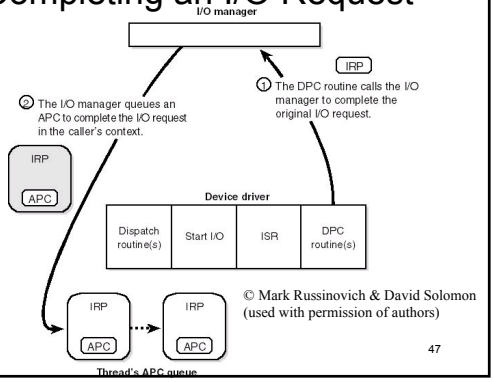
Servicing an Interrupt



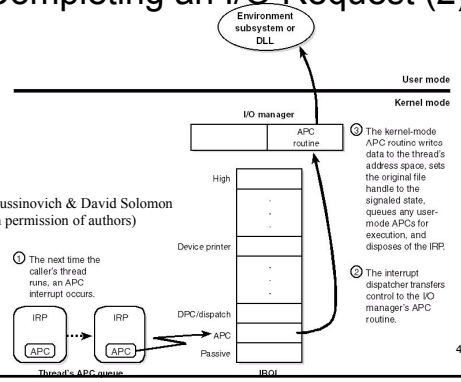
Servicing an Interrupt (2)

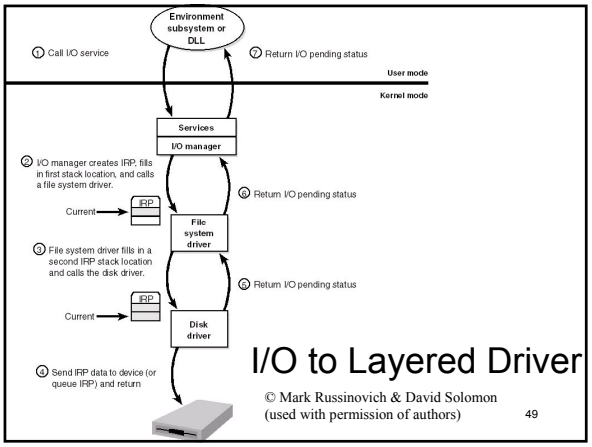


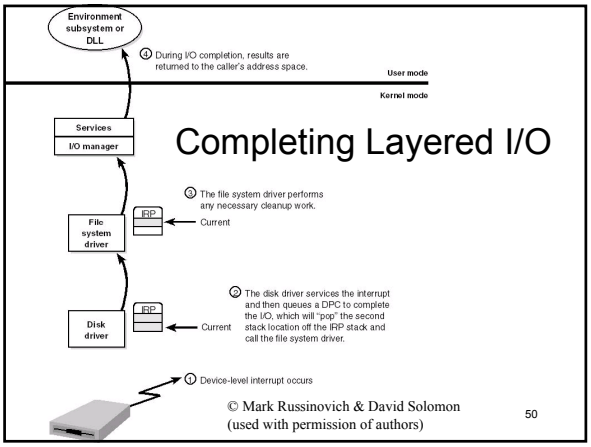
Completing an I/O Request



Completing an I/O Request (2)







Synchronization

- Drivers have to synchronize using kernel-synchronization routines
- On single processor by raising IRQL
- On multiple processors also using spin locks