Ausgewählte Betriebssysteme
Windows 2000 & Linux

Betriebssysteme

Technische Universität Dresden

Outline of Lectures

- Introduction
- I/O Structure & IRQ Handling
- Memory management
- NT file system
- (Processes & Threads)

Resources for these Lectures

- *Inside Windows 2000*, 3rd Edition, D.A. Solomon, M.E. Russinovich
- www.sysinternals.com
- msdn.microsoft.com
- Driver Development Kit (www.microsoft.com/DDK/)
- Microsoft System Journal

Windows 2000 - An Introduction

“Ausgewählte Betriebsysteme”

Operating Systems

Computer Science Department

Technische Universität Dresden

Overview

- Architecture and components
  - Operating system layout
  - Subsystems
- Essential Mechanisms
  - Objects and handles
  - Registry
- (Startup and Shutdown)

Requirements and Design Goals

- Provide true 32-bit, preemptive, reentrant, virtual memory OS
- OS security (government and industry)
- Performance
- Portability: able to run on multiple HW platforms; Easy to port to new ones
- Reliability & Robustness: system should protect itself and applications
- …

Architecture Basics

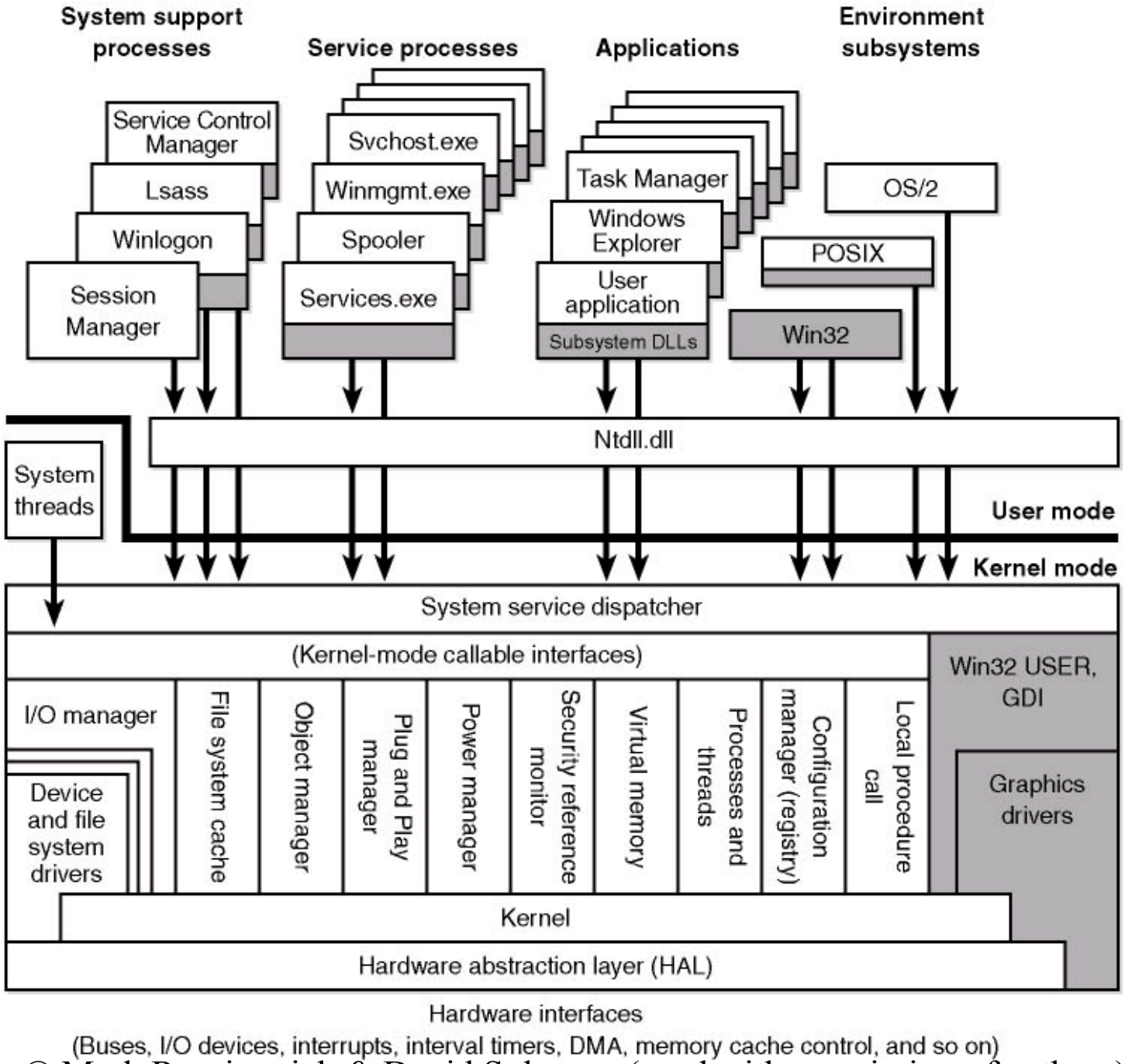
- No micro-kernel, but monolithic
- Modular design
- Separated into
  - architecture dependant / independent parts and
  - hardware dependant / independent parts
- Hardware dependant e.g.: (show example)
  - spin lock routines
  - context switch (save/restore context)
- Windows 2000 == Windows NT 5.0
  Windows XP == Windows NT 5.1

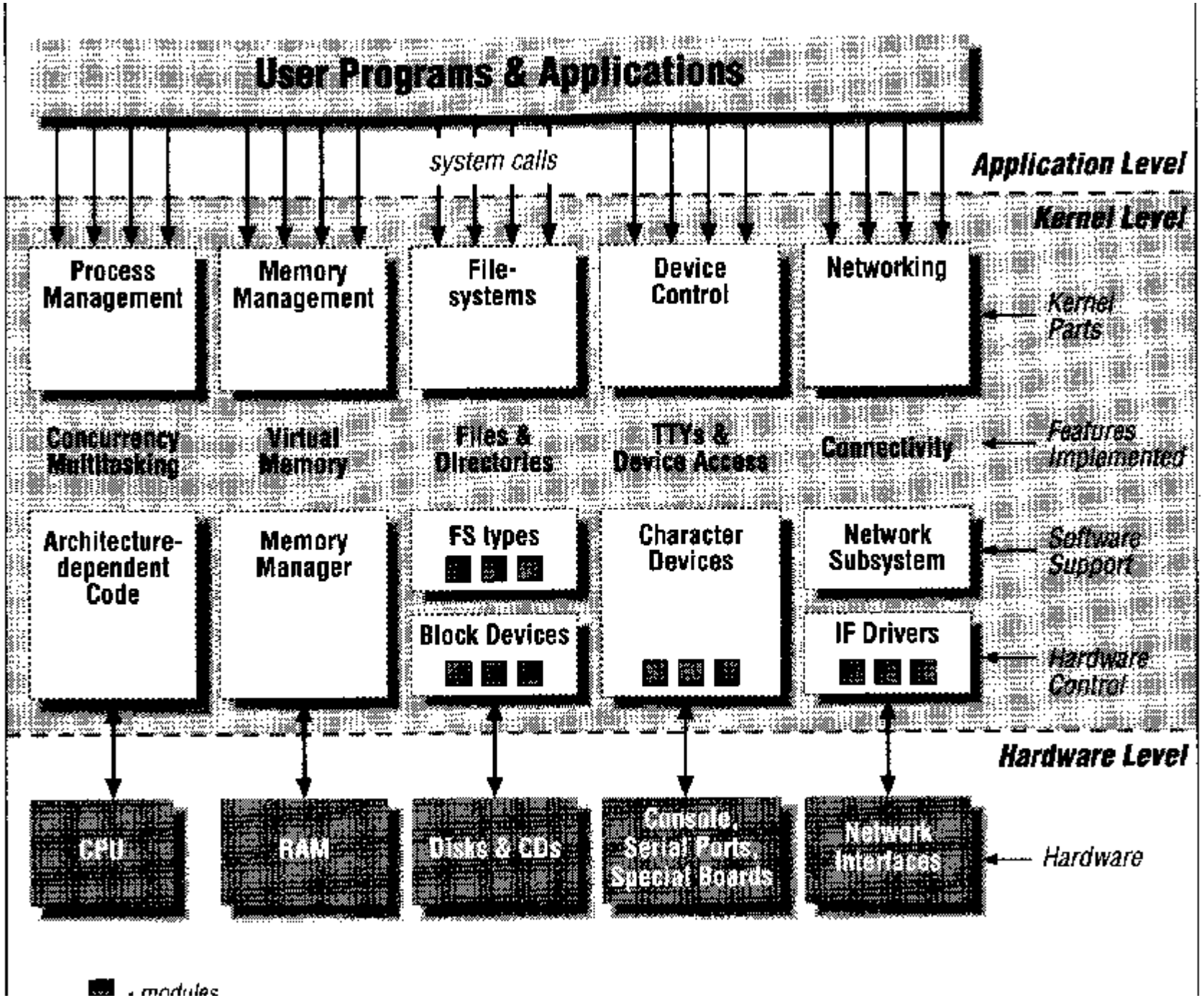
Notions

- Kernel/System component is a functional part of the operating system
- Environmental subsystem = Subsystem
- Subsystem is a runtime environment for applications
- Subsystem components are functional parts of a subsystem

Main System Components

- Object Manager.
- I/O manager.
- Device drivers.
- Memory management.
- Process management.
- Security subsystem.
- PnP manager.
- Kernel.




User Programs & Applications
system calls
Application Level
Kernel Level
Process Management
Memory Management
File-systems
Device Control
Networking
Kernel Parts
Concurrency Multitasking
Virtual Memory
Files & Directories
TTYs & Device Access
Connectivity
Features Implemented
Architecture-dependent Code
Memory Manager
FS types
Block Devices
Character Devices
Network Subsystem
IF Drivers
Software Support
Hardware Control
Hardware Level
CPU
RAM
Disks & CDs
Console, Serial Ports, Special Boards
Network Interfaces
Hardware
modules

Programs, Processes and Threads

- *Program* is static sequence of instructions
- *Process* is container for a set of resources
- *Threads* execute instance of program and
- Threads use resources of process

System Processes

- Idle (one thread per CPU) – ID 0
- System (contains majority of kernel-mode system threads) – ID 8
- Session manager (smss.exe)
- Win32 subsystem (csrss.exe)
- Logon process (winlogon.exe)
- Service control manager (services.exe)
- Local security authentication server (lsass.exe)

System Threads

- No user-mode binary → no address space
- Run in kernel-mode
- Can be attached to any process (default: System process)
- Can only be created from kernel-mode
- Used to perform actions, which need thread context:
  - Issue and wait for I/O
  - Polling a device

Examples for System Threads

- Memory Management
  - Write dirty pages
  - Swap processes
- Cache Manager
  - Read ahead / write behind I/O
- File Server
  - Respond to network requests
- Floppy Driver
  - Poll floppy device
    (more efficient then interrupt driven)

Logon (Winlogon)

- Handles interactive user logons and logoffs
- Notified by *Secure Attention Sequence (SAS)*
- Capture user name and password
- Send to security authentication server to verify login
- Identification in replaceable DLL - Graphical Identification and Authentication (GINA, msgina.dll)

Service Control Manager

- Service similar to UNIX daemon process
- Responsible for starting, stopping and interacting with service processes

Local Security Authentication

- Authenticate logon - see Startup
- Authenticates access to objects - see Objects and Handles

Executive

- Functions exported to user-mode via Ntdll.dll
- Functions only callable from kernel-mode
- Internal support functions for Ntoskrnl.exe
- Components:
  - Configuration manager; Process and thread manager; Security reference monitor; I/O manager; PnP manager; Power manager; Cache manager; Virtual memory manager
- Support functions:
  - Object manager; LPC facility; Rtl functions (string, data conversion, ...); executive support routines (memory allocation, resources, fast mutexes, ...)

Kernel

- Implements OS mechanisms
- Avoid policy making (except thread scheduling and dispatching)
- Hardware support
  - Interrupt handling
  - Exception dispatching
  - Multiprocessor synchronization
  - TLB and CPU cache support
  - Context switching

Hardware Abstraction Layer

- Loadable kernel module (Hal.dll)
- Functions documented in DDK
- Implements abstract layer for different architecture platforms
- Hides
  - I/O interfaces
  - Interrupt controllers
  - Multiprocessor communication mechanisms

Hardware Abstraction Layer

HAL File Name	Systems Supported
Hal.dll	Standard PCs
Halacpi.dll	Advanced Configuration and Power Interface (ACPI) PCs
Halapic.dll	Advanced Programmable Interrupt Controller (APIC) PCs
Halaacpi.dll	APIC ACPI PCs
Halmps.dll	Multiprocessor PCs
Halmacpi.dll	Multiprocessor ACPI PCs
Halborg.dll	Silicon Graphics Workstation
Halsp.dll	Compaq SystemPro

Device Drivers

- Loadable kernel modules
- Between hardware and I/O manager
- Can run in three contexts
  - User thread which initiated I/O function
  - Kernel-mode system thread
  - As a result of an interrupt

Overview

- Architecture and components
  - Operating system layout
  - Subsystems
- Essential Mechanisms
  - Objects and handles
  - Registry
- (Startup and Shutdown)

Subsystems

- Three different subsystems (Win32, POSIX, OS/2)
- Function calls cannot be mixed between subsystems
- Startup information in Registry (experiment: HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems)
- Role of subsystem is to expose subset of executive services to application programs

Subsystem Components

- API DLLs
  - export APIs defined by the subsystem
  - implement them by calling native services
- Functions in subsystem DLL do:
  - Entirely run in user mode inside the subsystem DLL (e.g. *GetCurrentProcessId; HeapLock*) or
  - Call windows 2000 executive or
  - Client/server request to subsystem process (e.g. Manage state of application)
- Subsystem process
  - maintain global state of subsystem
  - implement a few APIs that require system-wide state changes

Native Images

- .exe not linked against any subsystem
- Only a few, e.g.:
  - Smss.exe (session manager)
  - Csrss.exe (Win32 subsystem)

- Subsystem specified in exe header
  - See winnt.h (Zeile 4928; image header 4779)
  - Experiment: Notepad.exe; Cmd.exe

POSIX & OS/2

- POSIX subsystem
  - Only 1003.1 supported
  - Compile POSIX applications with platform SDK
  - Cannot create thread or window
  - Cannot use remote procedure calls or sockets
  - Linked against POSIX subsystem DLL (psxdll.dll)
- OS/2 subsystem
  - Only supports OS/2 1.2 16-bit applications

Win32 Subsystem

- Major components
  - Subsystem process (csrss.exe)
  - Kernel mode device driver (win32k.sys)
  - Subsystem DLLs (translate API functions into appropriate system service calls to Ntoskrnl.exe and win32k.sys)
  - Graphics device drivers
- Always running

Win32 Subsystem (2)

- Subsystem process
  - Supports console windows (drawing)
  - Creating and deleting processes and threads
  - Support for 16-bit virtual DOS machine
  - Other misc. functions and language support functions
- Kernel-mode driver
  - Window manager
  - Collects input from keyboard, mouse, etc
  - Pass user messages to applications
  - Graphics device interface

Ntdll.dll

- System support library (linked to subsystems)
- Contains two types of functions:
  - System service dispatch stubs to execute system services
  - Internal support functions used by subsystems

Ntdll.dll (2)

- System service dispatch stubs
  - Over 200 functions (*NtWriteFile* (wdm.h:9153))
  - Mostly accessible through Win32 API
  - Function contains architecture specific instruction that causes transition into kernel-mode
- Internal support functions
  - Image loader (*Ldr*); Heap manager; Win32 subsystem process communication (*Csr*); General runtime library routines (*Rtl*)

Overview

- Architecture and components
  - Operating system layout
  - Subsystems
- Essential Mechanisms
  - Objects and handles
  - Registry
- (Startup and Shutdown)

Objects and Handles

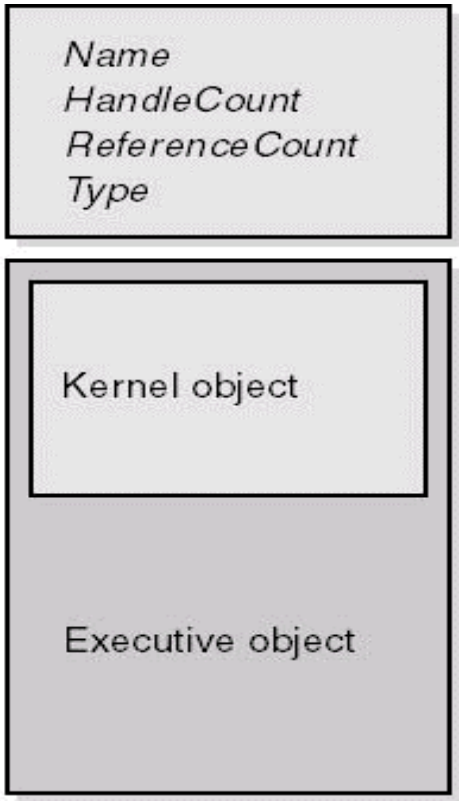
- Three types of Win32 objects
  - Win32 „kernel objects“ (events, mutexes, files, processes, threads, etc.)
    - Managed by Object manager
    - Handle values are private to each process
  - Win32 „GDI objects“ (pens, brushes, fonts, etc.)
    - Managed by Win32 subsystem
    - Handle values are valid system-wide
  - Win32 „user objects“ (windows, menus, etc.)
    - Managed by Win32 subsystem
    - Handle values are valid system-wide

Object Manager

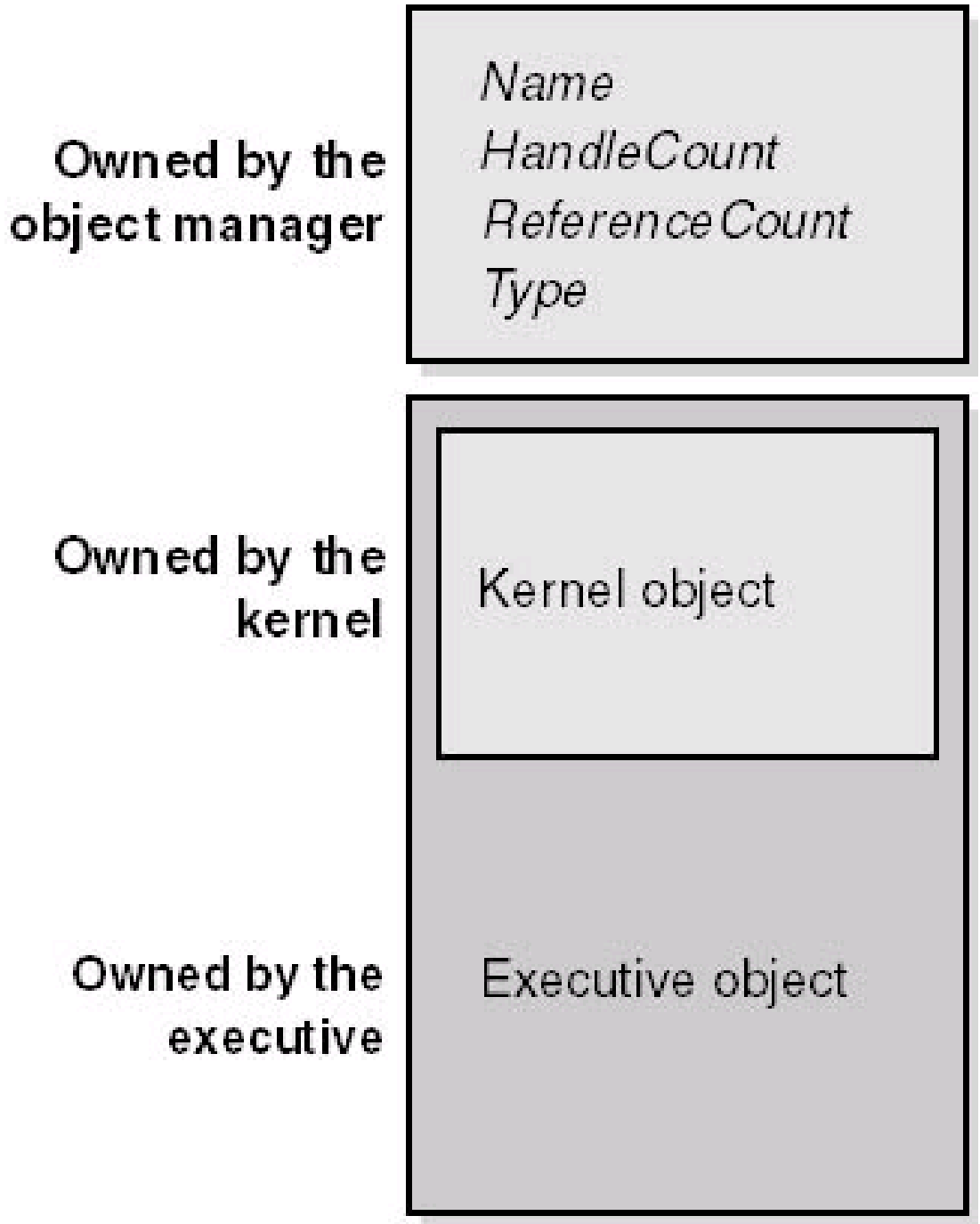
- Centralized resource control:
  - uniform mechanism for using system resources
  - isolate protection to one location
  - establish object-naming scheme
  - provide mechanisms to charge for resource usage

Kernel- vs. Executive Objects

- Object header
  - used by object manager to manage objects regardless of type
- Kernel objects
  - primitive set of objects implemented by kernel
  - provide fundamental capabilities (e.g. synchronization)
- Executive objects
  - implemented by various components in executive

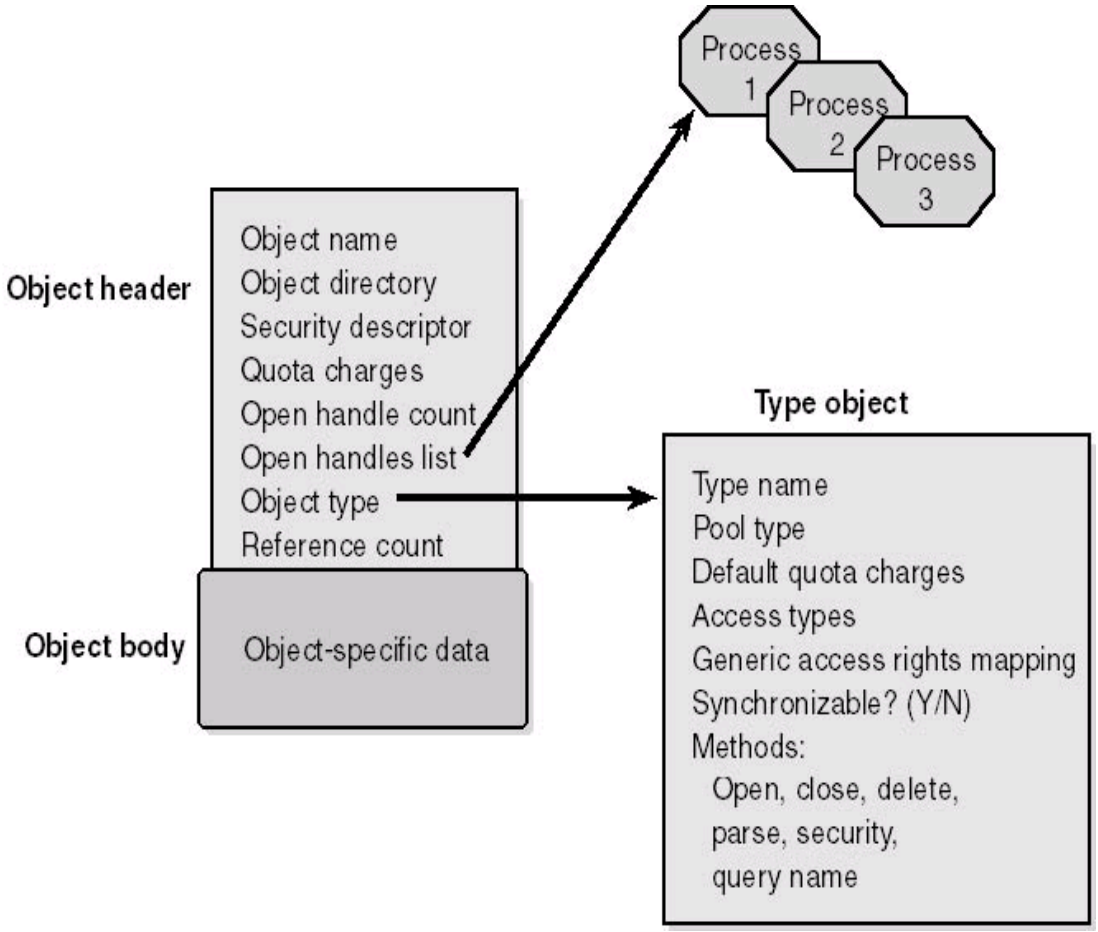


picture © Mark Russinovich & David Solomon
(used with permission of authors)

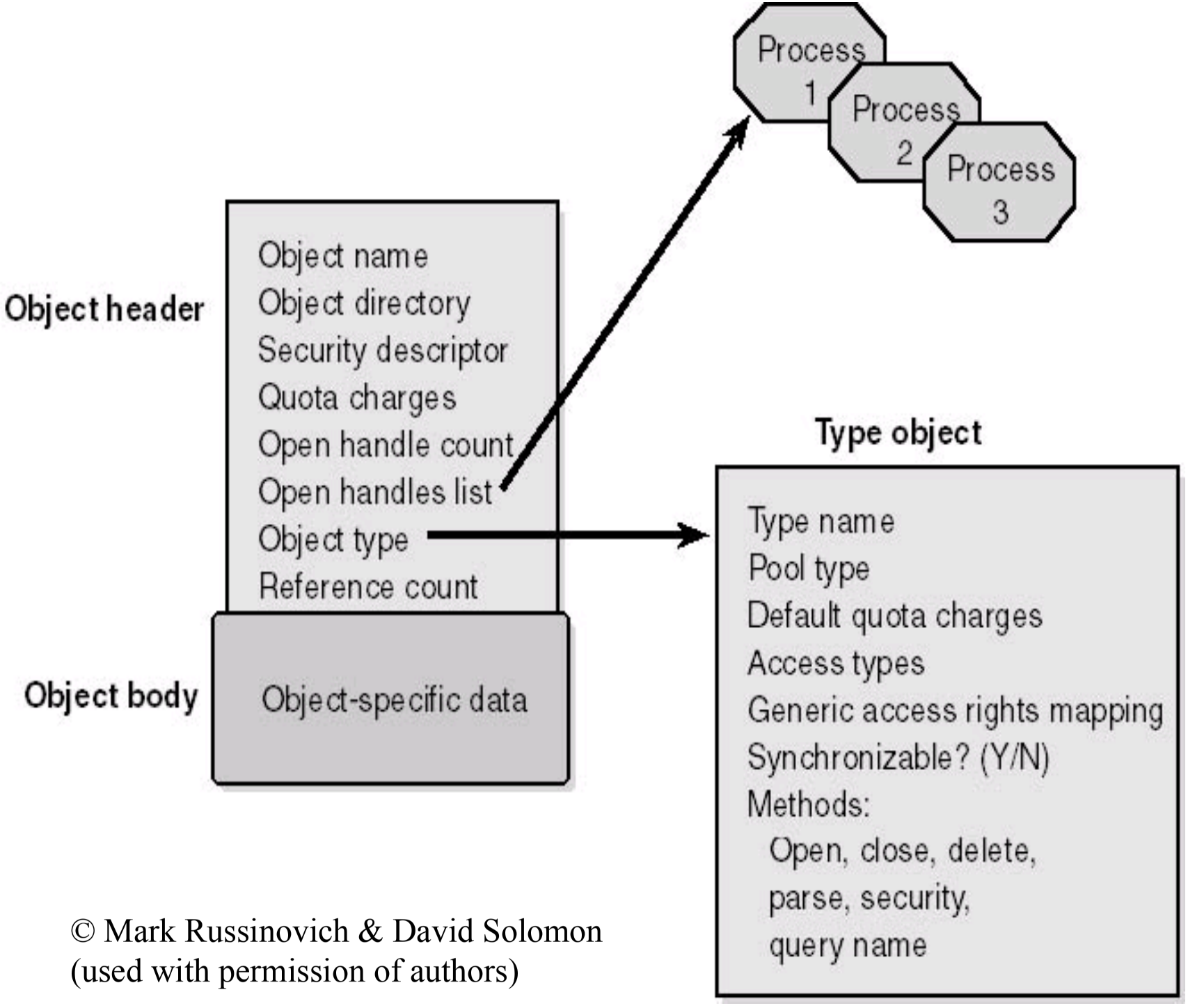


Object Structure

- Object body unique to its object type
- Object manager stores object attributes once for each object type
- Object type cannot be manipulated from user-mode



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



Standard Object Header Attributes

- Object name
- Object directory
- Security descriptor
- Quota charges (how much is a process charged)
- Open handle count
- Open handle list (list of processes with handle for object)
- Object type
- Reference count

Object Names

- A way to distinguish objects from another
- A method for finding a particular object
- Allows processes to share objects
- Only time name is needed: create object, open object, search for object
- Otherwise handle is used
- Names are global to computer but not across network

Type Object

- Object header contains data common to all objects, but with different values
- Object type contains data common to objects of same type (with same values)
- Type Object Attributes:
  - Type name
  - Pool type (paged/non-paged memory)
  - Default quota charges
  - Access types (allowed access request methods)
  - Generic access rights mapping (map „rwxa“ to type-specific rights)
  - Synchronization (whether threads can wait on object)
  - Methods (that are called automatically by object manager)

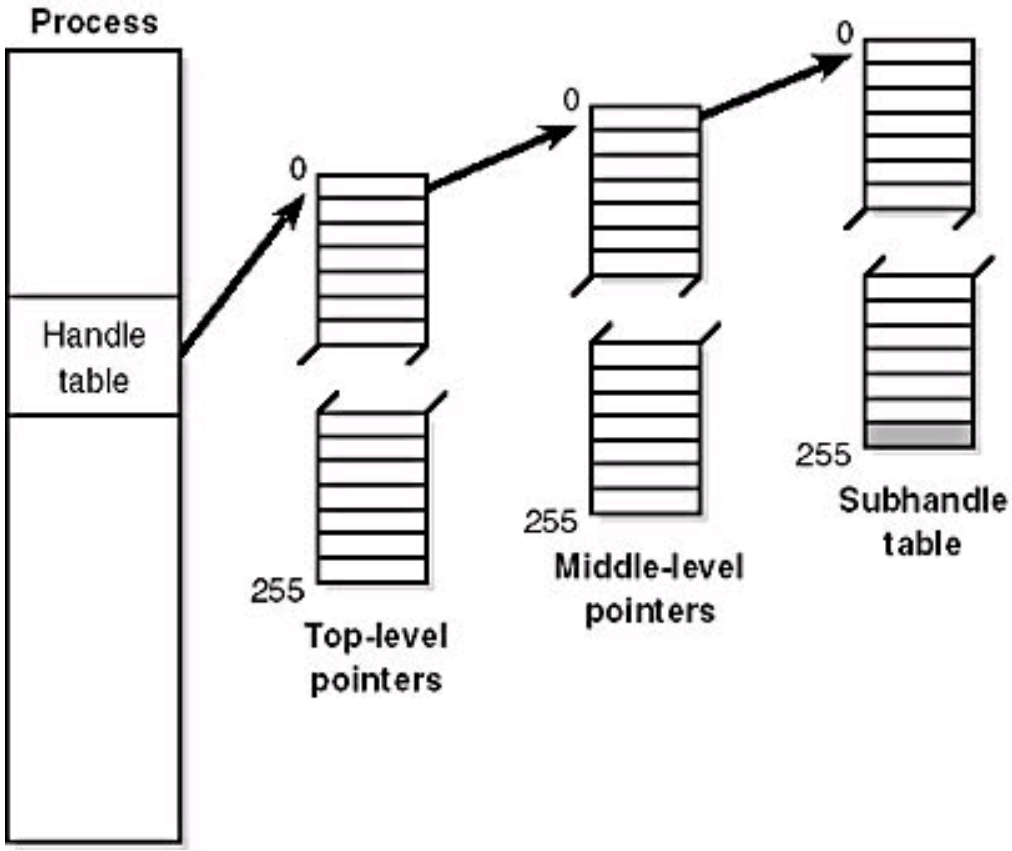
Object Methods

- Executive component can register methods when creating an object type (experiment)
- Control manipulation of all objects of that type
- Object manager calls these methods at well-defined points in life of object
  - Open, close, delete, query name, parse, security
  - Parse: find object outside of object managers name-space (e.G. Registry and FS)

Object Handles

- Process creates/opens object by name: receives handle
- Handle can be inherited
- Consistent interface to reference objects
- Object manager has exclusive right to create handles and associated objects
- Handle is index into process-specific handle table

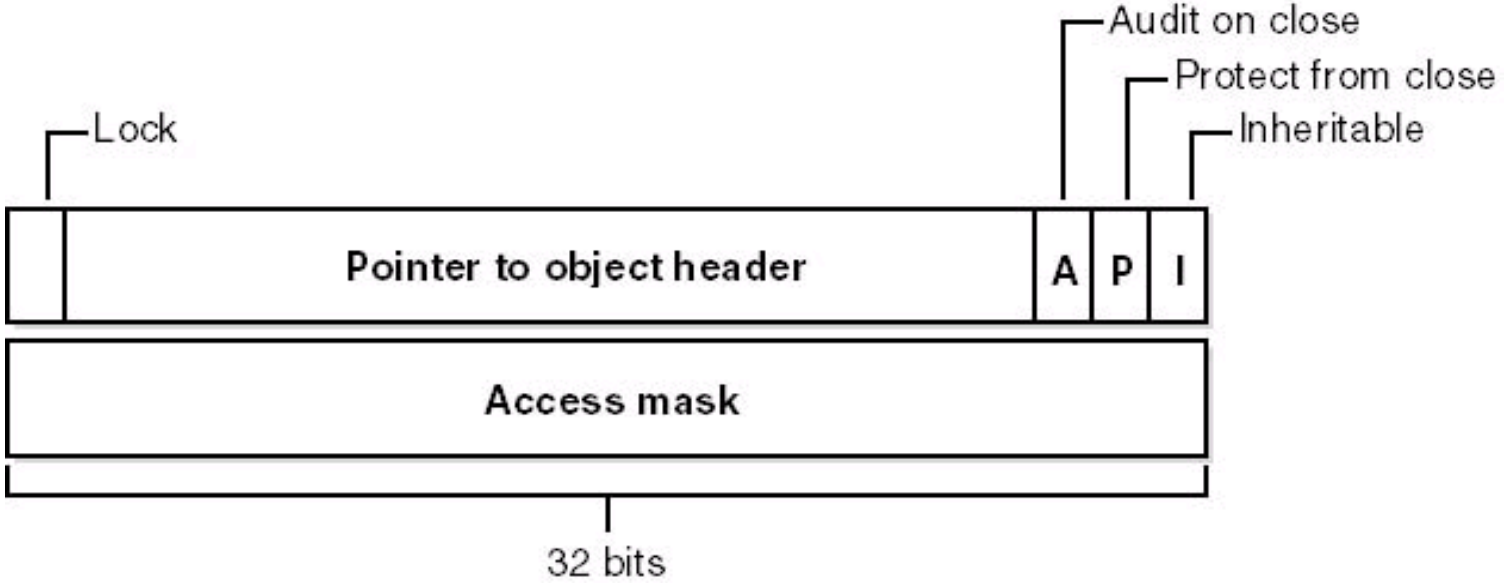
Handle Table



- Handle is 32-bit value
- Lower 24-bit are used in 8-bit chunks as indices into tables
- Subhandle table entries are 64-bit wide

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

Subhandle Table Entry



- Object headers always 32-bit aligned: 3 lower bits are flags
- Entry locked when object manager translates handle into object pointer
- Whole table only locked for create or close
- System components use kernel handle table

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

Object Security

- Process handle table
  - Is unique for each process
  - Is in system address space → cannot be modified from user mode

  → Is trusted
- Security checks when handle table entry is created
  - Process specifies *desired access rights*
  - Permissions are checked by security reference monitor
  - If access permitted, set of granted access rights returned and stored in object handle
  - Handle table entry indicates „validated“ access rights

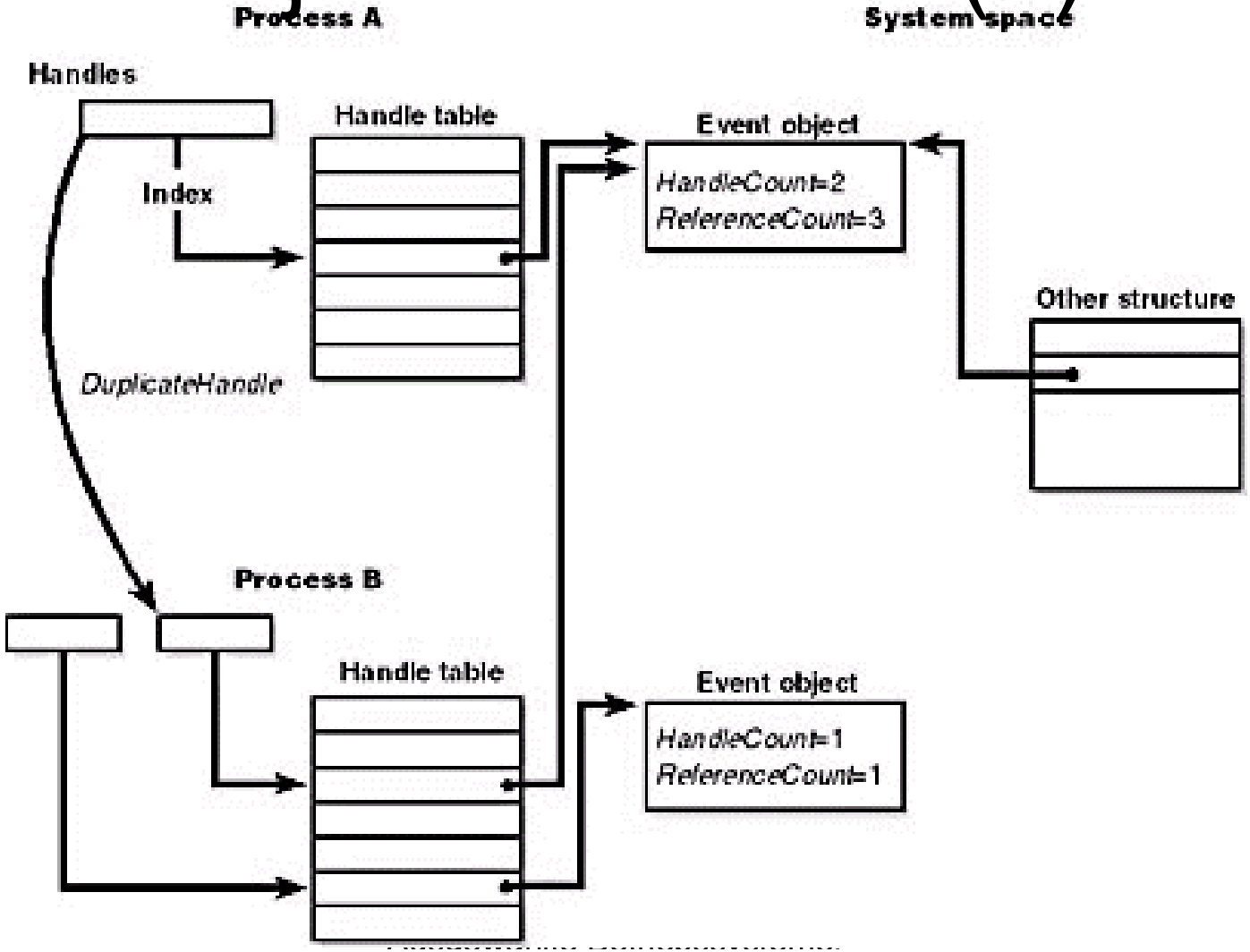
Object Security (2)

- APIs that take handle, look into handle table before performing function
  - No need to check file ACL, process or thread access token
  - Object manager checks whether operations correspond to access mask

Object Retention

- Every time object is opened, object handle number is incremented and decremented on close (usually from user-mode)
- If handle number falls to zero, object's name is removed from name-space
- Reference count is incremented each time a pointer to object is handed out (usually from kernel-mode)
- If reference count falls to zero, object is removed from memory

Object Retention (2)



Resource Accounting

- Used to limit resource usage of processes
- Quota charge is value by which the process' page and/or non-paged pool quota is incremented (open) or decremented (close)
- Quota handling supported, but not enforced
- "Soft quotas": if quota reaches limit, memory manager is asked to expand quota

Executive Objects

- Created by environmental subsystem on behalf of user or by parts of OS component during normal operation
- Subsystem export sub- or superset of executive objects
  - e.g. Win32 mutexes are based on executive mutexes
  - POSIX subsystem uses executive objects to represent POSIX-style processes, pipes, etc.

Overview

- Architecture and components
  - Operating system layout
  - Subsystems
- Essential Mechanisms
  - Objects and handles
  - Registry
- (Startup and Shutdown)

Registry

- Repository for system/user configuration information
  - contains information required to boot and configure as well as current running dynamic status information
  - network accessible - basis for remote configuration (not control) (information has to be reread)
- Most registry parameters can be adjusted using graphical utility (e.g. regmon)
- Can be compared to information in (Linux):
  - /etc: software settings
  - /proc: hardware information

Registry Organization

Root Key	Abbreviation	Description	Link
HKEY_CURRENT_USER	HKCU	Points to the user profile of the currently logged-on user	Subkey under HKEY_USERS corresponding to currently logged-on user
HKEY_USERS	HKU	Contains subkeys for all loaded user profiles	Not a link
HKEY_CLASSES_ROOT	HKCR	Contains file association and COM registration information	HKLM\SOFTWARE\ Classes
HKEY_LOCAL_MACHINE	HKLM	Placeholder— contains other keys	Not a link
HKEY_CURRENT_CONFIG	HKCC	Current hardware profile	HKLM\SYSTEM\Current ControlSet\Hardware Profiles\Current
HKEY_PERFORMANCE_DATA	HKPD	Performance counters	Not a link

Registry Organization (2)

- Five main hives for local machine information
  - \HKEY_LOCAL_MACHINE\System
    - Controls booting and running the system
    - Available during bootstrap
  - \HKLM\Hardware
    - Hardware configuration data, resource usage, etc.
    - Completely volatile (not saved across boots)
  - \HKLM\Software
    - Per machine software data not critical for booting
  - \HKLM\SAM
    - Account and group database
  - \HKLM\Security
    - System wide security policies

Registry Hives

- A hive is a piece of registry database stored in its own file
  - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet \Control\hivelist
- Registry mechanism guarantees atomicity
  - Use transactional logging to ensure structural integrity
- Registry hives are read into paged pool
  - A system registry quota limits usage

Registry Security and Recovery

- Registry Security
  - Registry hives are in system directory
  - Registry keys have full security descriptors
- Recovering from registry problems (“last known good”)
  - “Successful boot” is determined when someone logs in
  - Before that, system hive of registry used for previous successful boot can be restored easily
  - To restore damaged registry hives, use emergency repair procedure
  - To backup system registry hives run Rdisk.exe

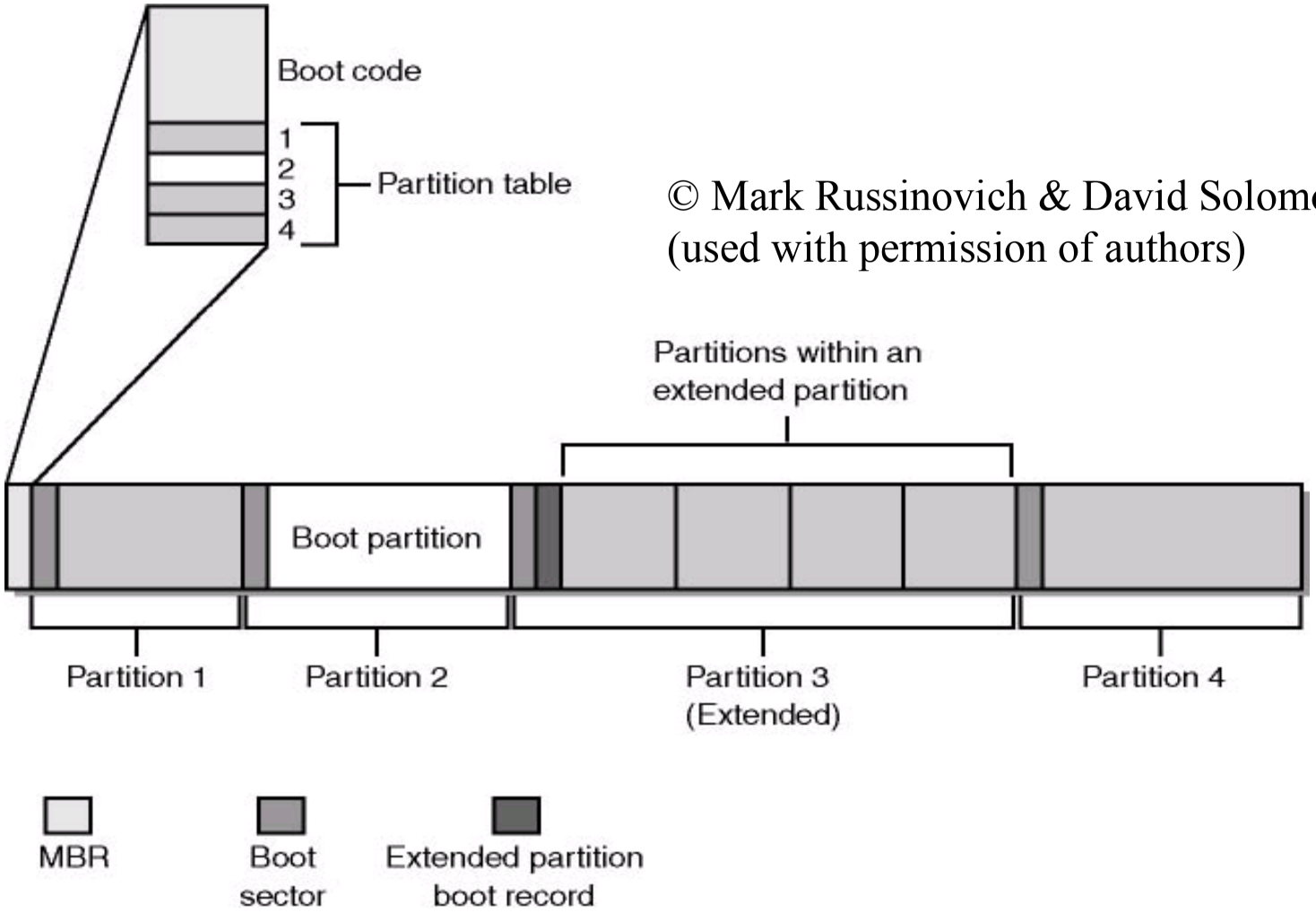
Overview

- Architecture and components
  - Operating system layout
  - Subsystems
- Essential Mechanisms
  - Objects and handles
  - Registry
- (Startup and Shutdown)

Startup and Shutdown

- Master Boot Record and boot partition
- Ntldr
- Ntdetect.com
- Ntoskrnl.exe
- Component initialization
- Logon
- Shutdown
- Crash

Master Boot Record and Boot Partition



Master Boot Record and Boot Partition (2)

- MBR contains boot code and partition table; started by BIOS on IBM-compatible PC
- Boot partition must be formatted with supported FS (FAT, FAT32, NTFS)
- Boot partition contains two files: Ntldr and Ntdetect
- Boot.ini file contains start menu options
- Boot sector contains startup code depending on FS

Ntldr

- Boot sector gives Windows 2000 info about logical disk drives and reads Ntldr
- If Ntldr not found: Error and system halted
- Ntldr starts in real mode, but switched first to protected mode to be able to address whole memory
- Creates page tables for first 16MB and enables paging

Ntldr (2)

- Boot.ini is loaded (boot code contains code to access IDE-based disks and display read-only)
- If boot or system drive is SCSI, a file Ntbootdd.sys is loaded first (read-only SCSI FS driver)
- Ntldr displays boot menu if more than one entry exists
- Menu entry directs Ntldr to Windows system directory
- If boot.ini contains DOS entry, the bootsec.dos file is loaded and its MBR is executed in real-mode
- boot.ini entries may have arguments (e.g. /DEBUG)

Ntdetect

- Next Ntdetect is loaded and executed
- Runs in 16-bit real-mode
- Uses BIOS to query the computer hardware
- Results stored in internal data structures, which will later be HKLM\Hardware\Description
- Switches back to Ntldr

Ntldr (3)

- Clears screen and displays “Starting Windows” with empty progress bar
- Loads necessary files from boot partition

1. Kernel image and HAL
2. Reads System hive
3. Scans in-memory System hive to locate boot device drivers
4. Explicitly load FS drivers
5. Load boot drivers - no initialization yet
6. Prepare CPU registers to run Ntoskrnl.exe

Ntoskrnl.exe

- Start phase 0 - all interrupts disabled
- Initialize all CPUs
- Boot CPU performs system-wide initialization
- Interrupt controller of each CPU configured
- Call initialization routines for main components
  - Memory manager, object manager, security reference manager, process manager, PnP manager
- Proceed with Idle loop, which start phase 1 (enable interrupts first)
- Display windows 2000 start screen

Ntoskrnl.exe (2)

- Power manager initialized
- System time read and stored as boot time
- Remaining processors started
- Security manager initialized
- Memory manger initialized
- Ntdll.dll mapped into system address space
- Cache manager initialized
- Configuration manager initializes registry
- File-system driver data structures initialized

Ntoskrnl.exe (3)

- PnP BIOS called
- I/O manager loads installed drivers
- Kernel mode paging enabled
- Power management data structures initialized
- Create Session manager subsystem process
- Initialization process waits for session manager 5 seconds
- If not started after this time the system is crashed
- Initialization process will be zero page thread

Session Manager

- Is user-mode process, but regarded trusted component (e.g. can create security tokens)
- Does not use Win32 but native API (starts Win32 subsystem)
- Finish registry initialization
- Runs programs configured to start during boot
- Delayed file renaming
- Open known Dlls

Session Manager (2)

- Load registry hives from disk
- Create system environment variables
- Load win32k.sys (changes to VGA mode)
- Start subsystem processes (incl. Csrss.exe)
- Start logon
- Create debug ports and threads to listen to them

Session Manager (3)

- Waits for process handles of csrss.exe and winlogon
- If on of them terminates, the session manager crashes the system (system depends on their existence)
- After user logged on the startup is regarded as successfully completed
- Last Known Good is replaced with Current Control Set (except user boot LKG or error)

Shutdown

- User‘s window processes are terminated
- User‘s console processes are terminated
- Systems windows (GUI) and services (console) terminated
- Power manager set new power level -> I/O manager shuts down drivers
- Registry flushed to disk
- Modified files written to disk
- Power manager decides what to do (Shutdown, reboot, power down)

System Crashes

- Reasons:
  - kernel-mode operation incurs unhandled exception
  - a call to kernel support routine results in reschedule if IRQ level is to high to initiate scheduler
  - a page fault on memory backed by paging or memory mapped file at IRQ level above scheduling level
  - device driver or kernel function calls *KeBugCheckEx* explicitly
  - hardware error, such as machine check or NMI

Layout of BSOD

- Blue Screen Of Death (BSOD)
  - STOP code with parameters of *KeBugCheckEx* plus text-equivalent of STOP code
  - if one of the parameter is address, the module the address belongs to is displayed
  - can create crash dump file (System properties)
  - analyze it with Dumpchk or kernel memory space analyzer (Kanalyze.exe) or kernel debugger (display processes, threads, drivers, etc.)