OPERATING SYSTEMS

STRUCTURES

Jerry Breecher

OPERATING SYSTEM Structures

What Is In This Chapter?

- System Components
- System Calls
- How Components Fit Together
- Virtual Machine

SYSTEM COMPONENTS

These are the pieces of the system we'll be looking at:

Process Management
Main Memory Management
File Management
I/O System Management
Secondary Management
Networking
Protection System
Command-Interpreter System

PROCESS MANAGEMENT

A **process** is a **program** in execution: (A program is passive, a process active.)

A process has resources (CPU time, files) and attributes that must be managed.

Management of processes includes:

- Process Scheduling (priority, time management, . . .)
- Creation/termination
- Block/Unblock (suspension/resumption)
- Synchronization
- Communication
- Deadlock handling
- Debugging

OPERATING SYSTEM STRUCTURES

System Components

MAIN MEMORY MANAGEMENT

- Allocation/de-allocation for processes, files, I/O.
- Maintenance of several processes at a time
- Keep track of who's using what memory
- Movement of process memory to/from secondary storage.

FILE MANAGEMENT

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.

The operating system is responsible for the following activities in connections with file management:

- File creation and deletion.
- Directory creation and deletion.
- Support of primitives for manipulating files and directories.
- Mapping files onto secondary storage.
- File backup on stable (nonvolatile) storage media.

OPERATING SYSTEM STRUCTURES

System Components

I/O MANAGEMENT

- Buffer caching system
- Generic device driver code
- Drivers for each device - translate read/write requests into disk position commands.

SECONDARY STORAGE MANAGEMENT

- Disks, tapes, optical, ...
- Free space management (paging/swapping)
- Storage allocation (what data goes where on disk)
- Disk scheduling

OPERATING SYSTEM STRUCTURES

System Components

NETWORKING

- Communication system between distributed processors.
- Getting information about files/processes/etc. on a remote machine.
- Can use either a message passing or a shared memory model.

PROTECTION

- Of files, memory, CPU, etc.
- Means controlling of access
- Depends on the attributes of the file and user

How Do These All Fit Together?

In essence, they all provide services for each other.

SYSTEM PROGRAMS

- Command Interpreters -- Program that accepts control statements (shell, GUI interface, etc.)
- Compilers/linkers
- Communications (ftp, telnet, etc.)

System Tailoring

Modifying the Operating System program for a particular machine. The goal is to include all the necessary pieces, but not too many extra ones.

- Typically a System can support many possible devices, but any one installation has only a few of these possibilities.
- **Plug and play** allows for detection of devices and automatic inclusion of the code (drivers) necessary to drive these devices.
- A **sysgen** is usually a link of many OS routines/modules in order to produce an executable containing the code to run the drivers.

System Calls

A System Call is the main way a user program interacts with the Operating System.

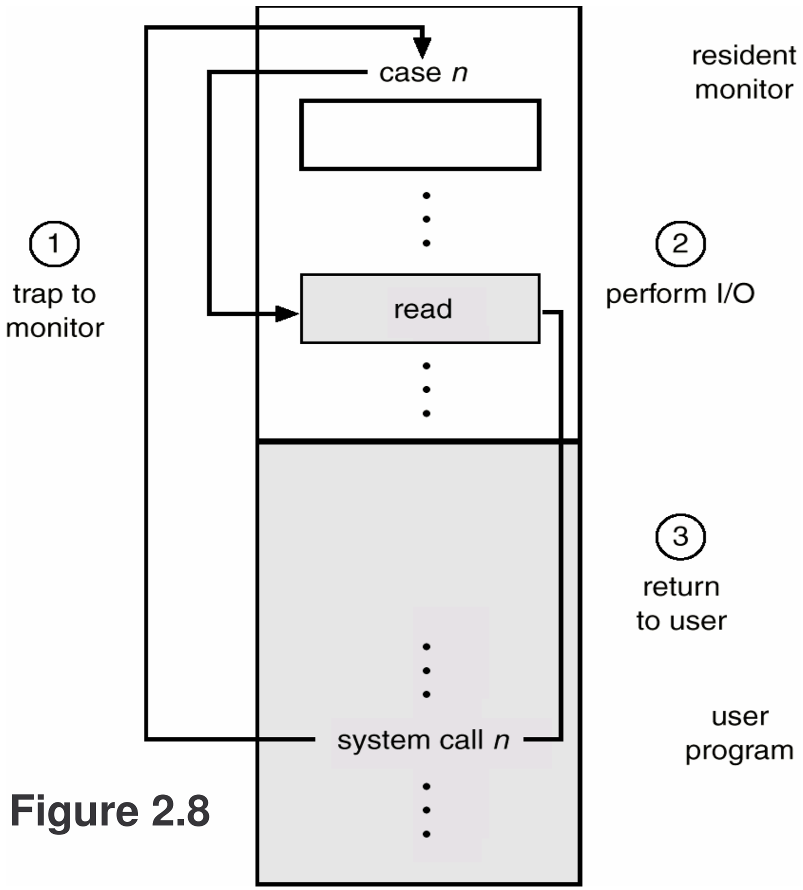


Figure 2.8

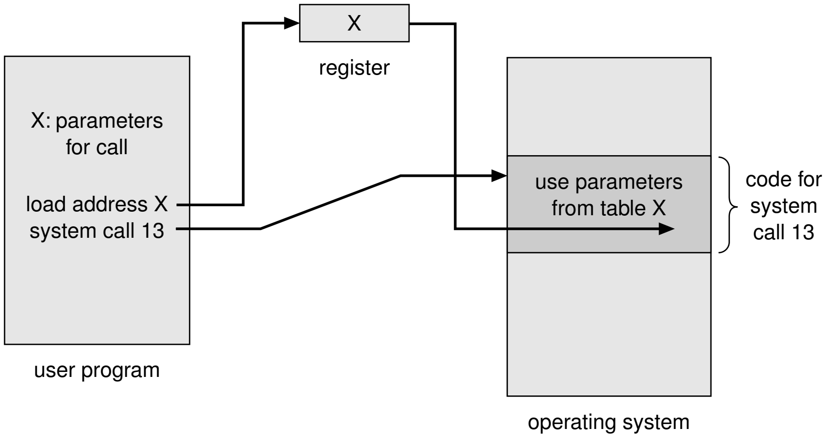


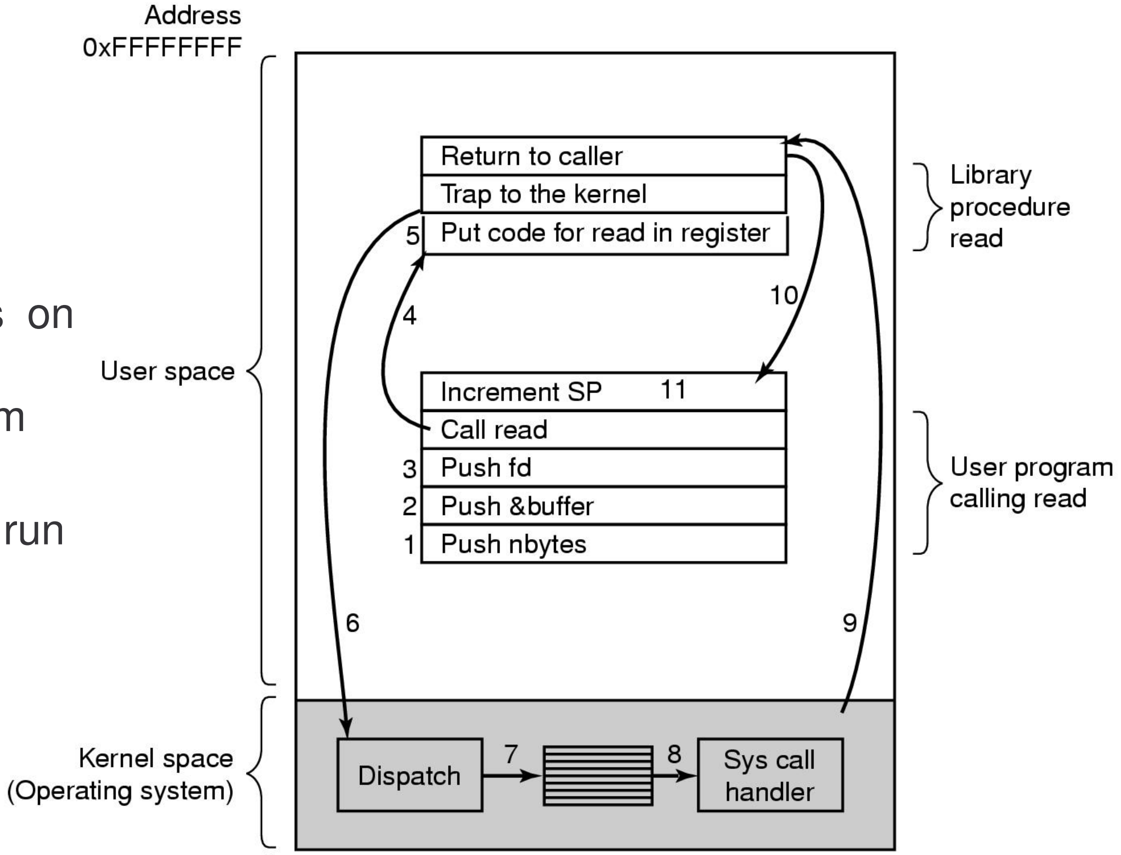
Figure 3.1

OPERATING SYSTEM STRUCTURES

System Calls

HOW A SYSTEM CALL WORKS

- Obtain access to system space
- Do parameter validation
- System resource collection (locks on structures)
- Ask device/system for requested item
- Suspend waiting for device
- Interrupt makes this thread ready to run
- Wrap-up
- Return to user



There are 11 (or more) steps in making the system call

read (fd, buffer, nbytes)

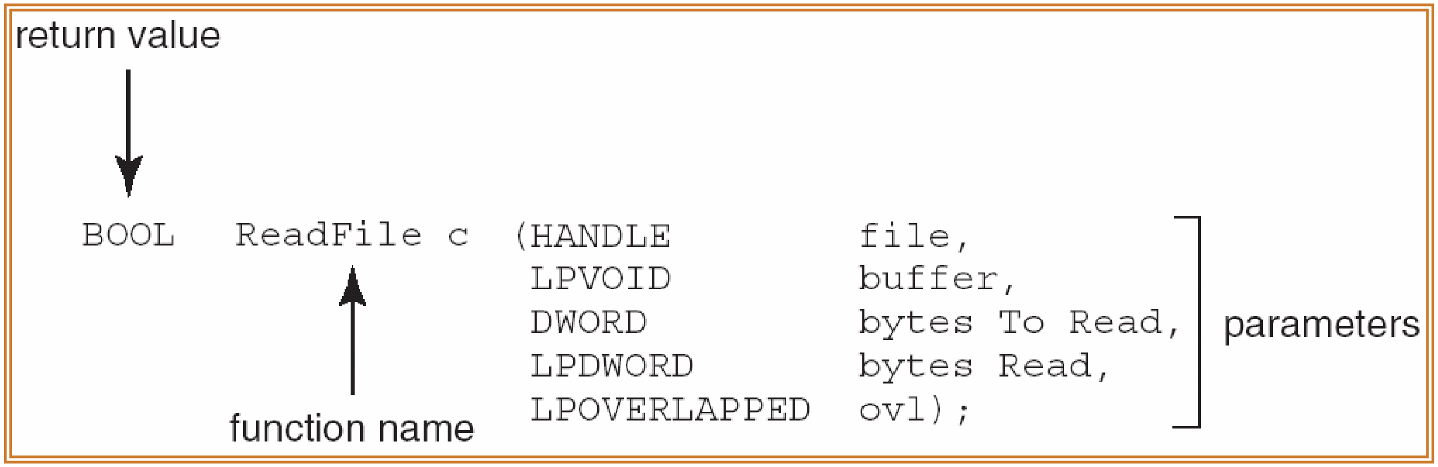
Linux API

OPERATING SYSTEM STRUCTURES

System Calls

Example of Windows API

Consider the ReadFile() function in the
Win32 API—a function for reading from a file.



```
BOOL   ReadFile c  (HANDLE        file,
                    LPVOID        buffer,
                    DWORD         bytes To Read,
                    LPDWORD       bytes Read,
                    LPOVERLAPPED  ovl);
```

A description of the parameters passed to ReadFile()

HANDLE file—the file to be read

LPVOID buffer—a buffer where the data will be read into and written from

DWORD bytesToRead—the number of bytes to be read into the buffer

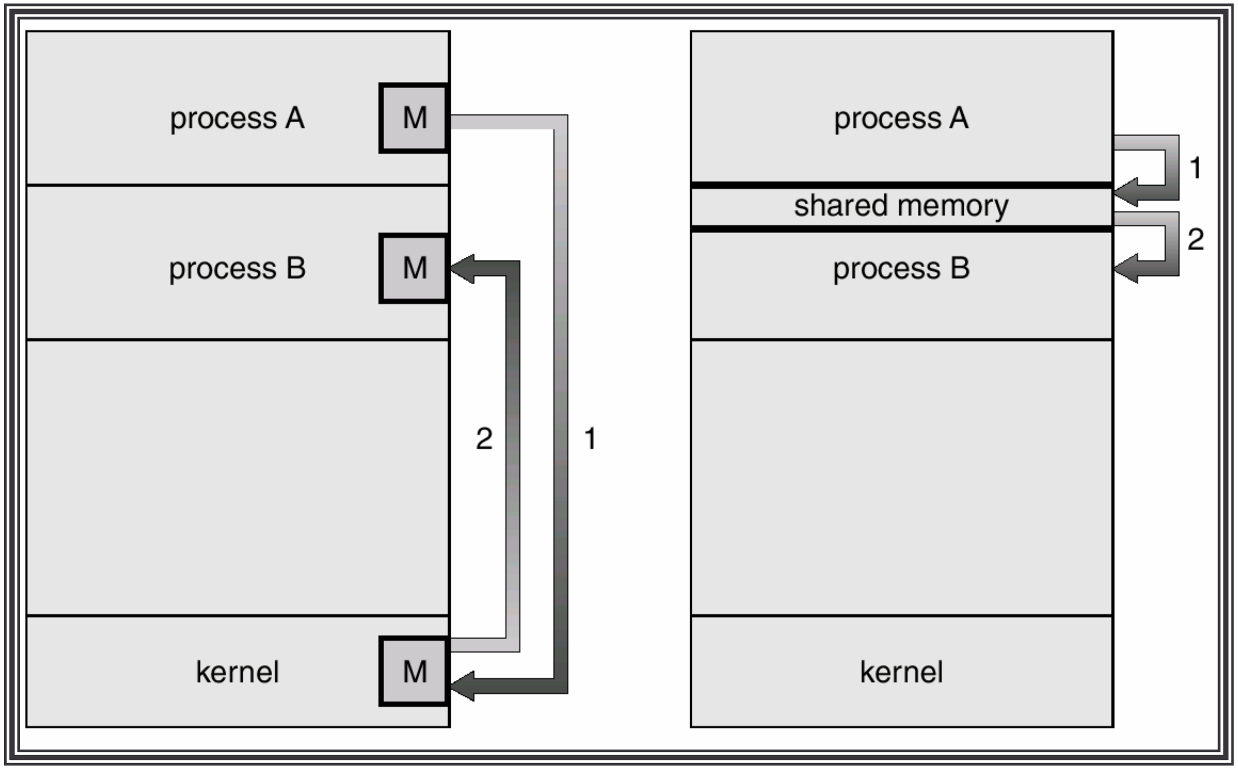
LPDWORD bytesRead—the number of bytes read during the last read

LPOVERLAPPED ovl—indicates if overlapped I/O is being used

OPERATING SYSTEM STRUCTURES

System Calls

Two ways of passing data between programs.



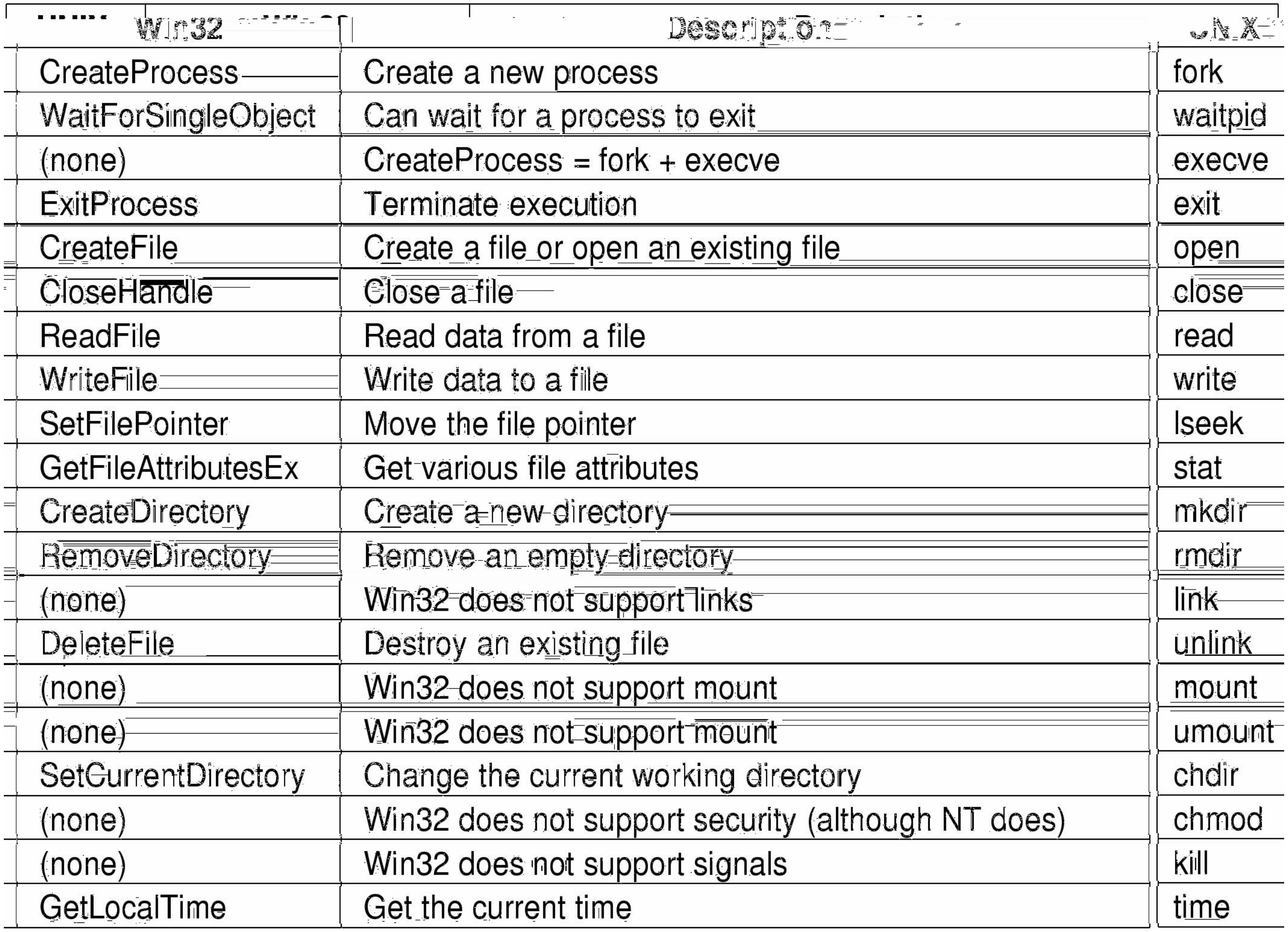
Msg Passing

Shared Memory

OPERATING SYSTEM STRUCTURES

System Calls

These are examples of various system calls.



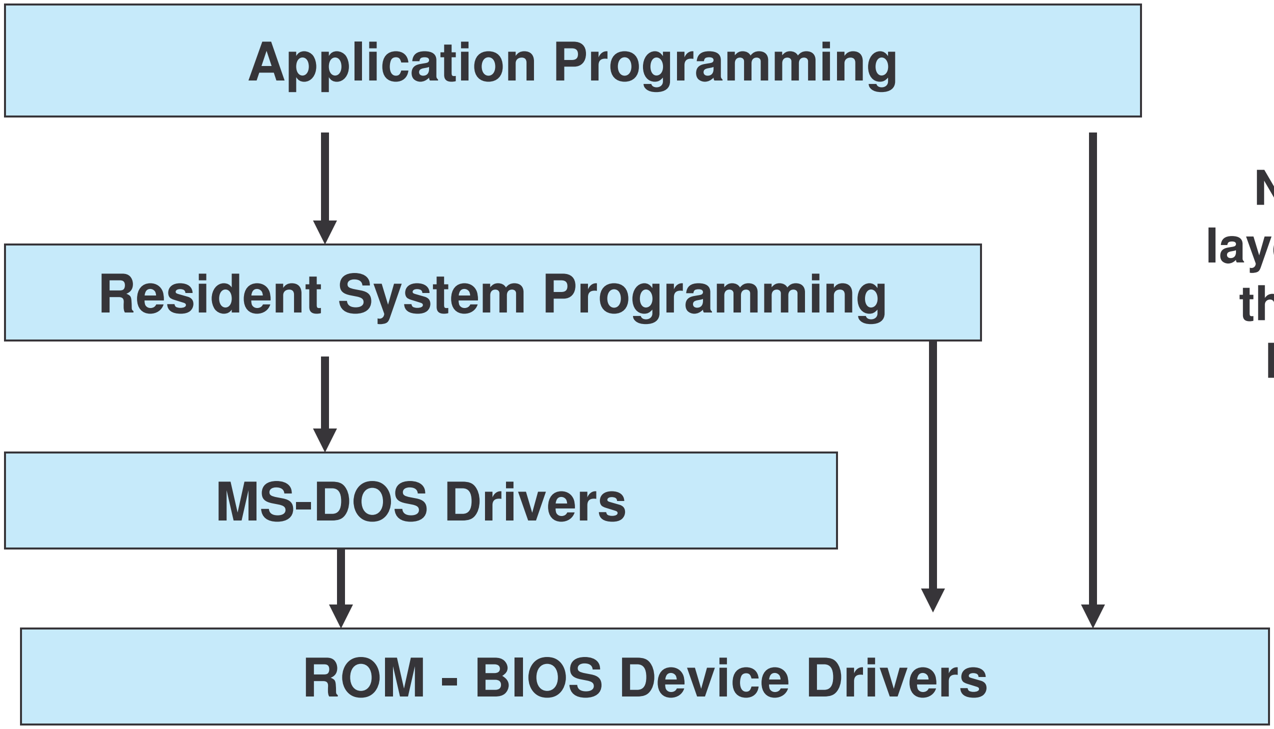
Win32	Description	UNIX
CreateProcess	Create a new process	fork
WaitForSingleObject	Can wait for a process to exit	waitpid
(none)	CreateProcess = fork + execve	execve
ExitProcess	Terminate execution	exit
CreateFile	Create a file or open an existing file	open
CloseHandle	Close a file	close
ReadFile	Read data from a file	read
WriteFile	Write data to a file	write
SetFilePointer	Move the file pointer	lseek
GetFileAttributesEx	Get various file attributes	stat
CreateDirectory	Create a new directory	mkdir
RemoveDirectory	Remove an empty directory	rmdir
(none)	Win32 does not support links	link
DeleteFile	Destroy an existing file	unlink
(none)	Win32 does not support mount	mount
(none)	Win32 does not support mount	umount
SetCurrentDirectory	Change the current working directory	chdir
(none)	Win32 does not support security (although NT does)	chmod
(none)	Win32 does not support signals	kill
GetLocalTime	Get the current time	time

OPERATING SYSTEM STRUCTURES

How An Operating System Is Put Together

A SIMPLE STRUCTURE:

Example of MS-DOS.



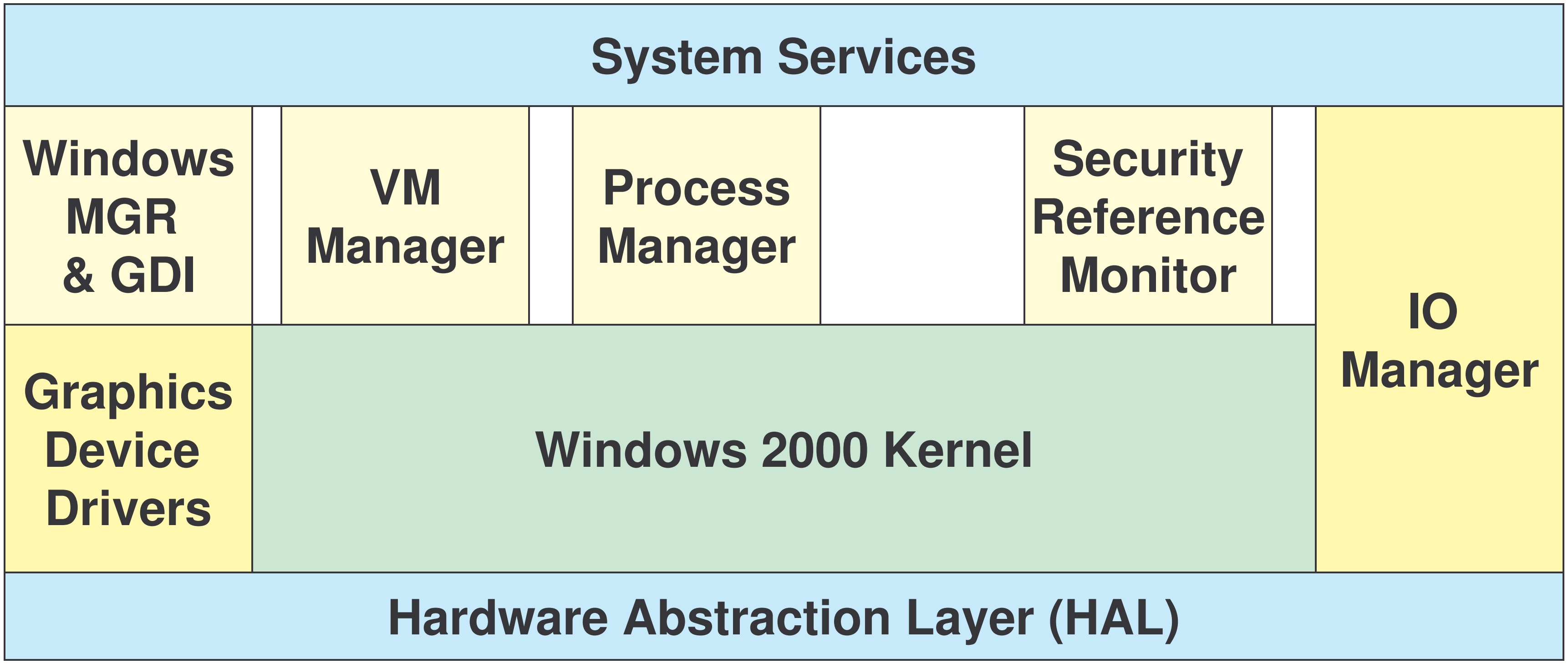
Note how all layers can touch the hardware. Bad News!!

OPERATING SYSTEM STRUCTURES

How An Operating System Is Put Together

A LAYERED STRUCTURE:

Example of Windows 2000.



System Services

Windows MGR & GDI

VM Manager

Process Manager

Security Reference Monitor

IO Manager

Graphics Device Drivers

Windows 2000 Kernel

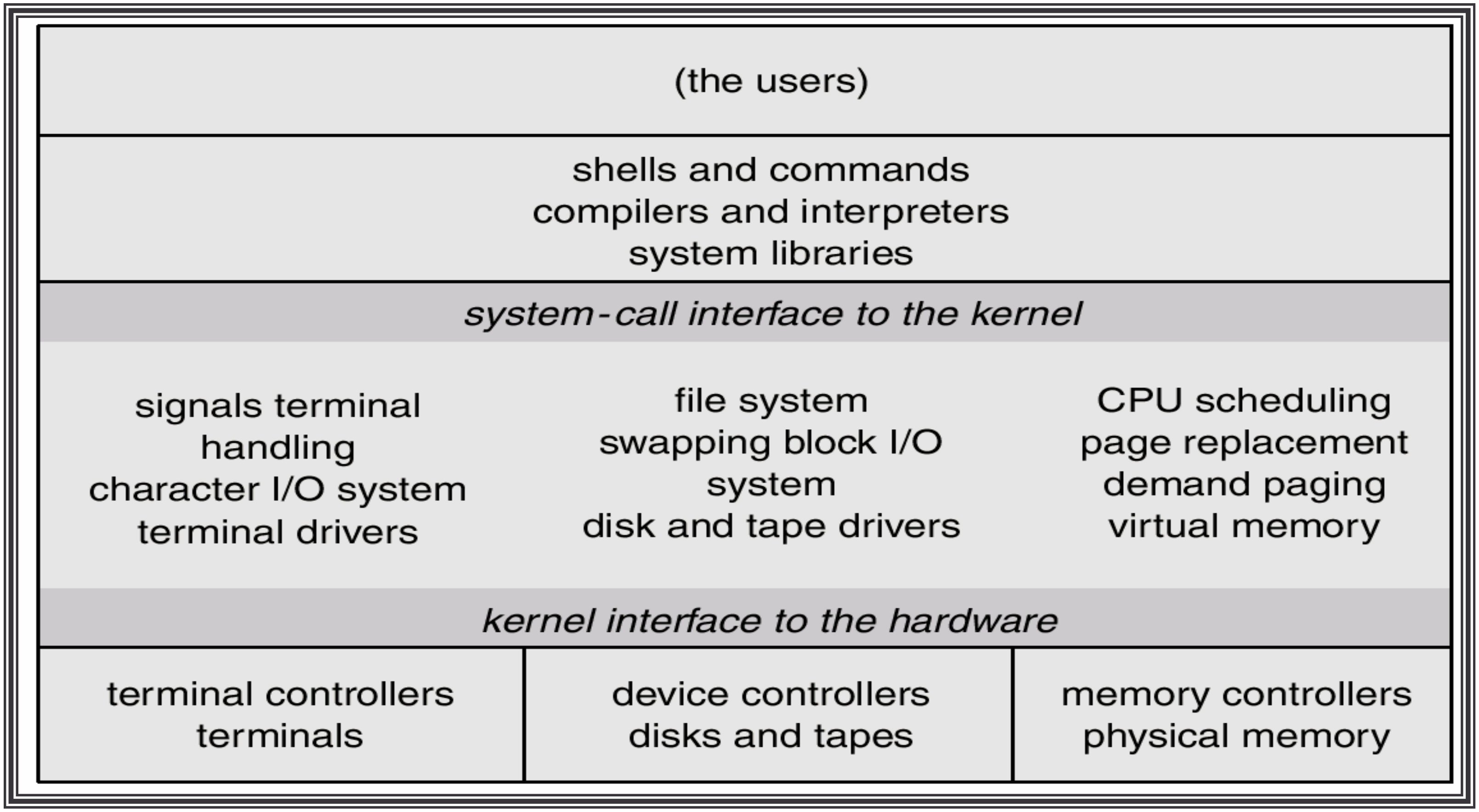
Hardware Abstraction Layer (HAL)

OPERATING SYSTEM STRUCTURES

How An Operating System Is Put Together

A LAYERED STRUCTURE:

Example of UNIX.



(the users)

shells and commands
compilers and interpreters
system libraries

system-call interface to the kernel

signals terminal handling
character I/O system
terminal drivers

file system
swapping block I/O system
disk and tape drivers

CPU scheduling
page replacement
demand paging
virtual memory

kernel interface to the hardware

terminal controllers
terminals

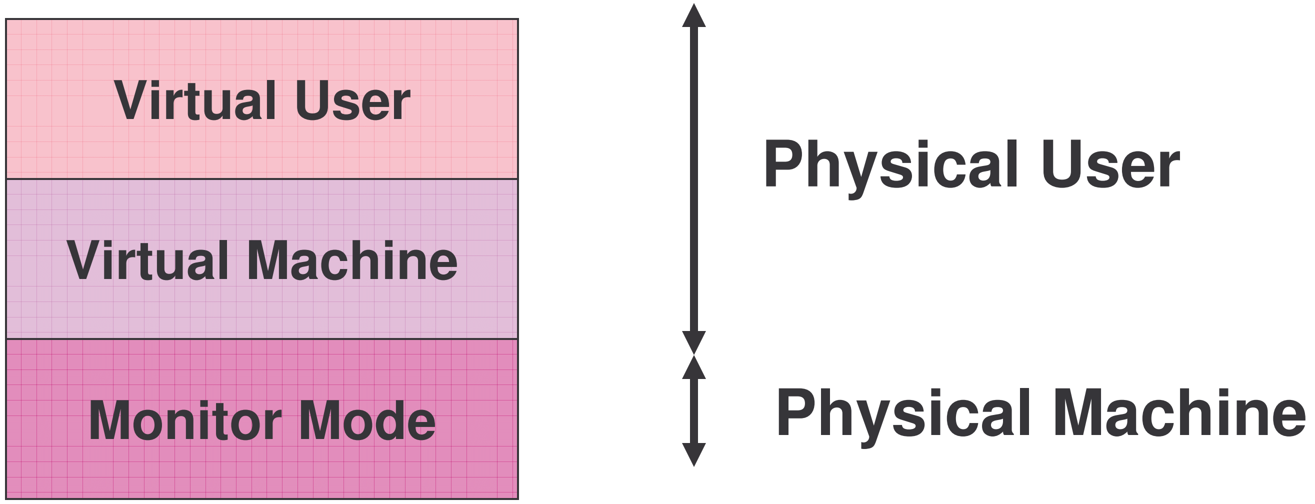
device controllers
disks and tapes

memory controllers
physical memory

Virtual Machine

In a Virtual Machine - each process "seems" to execute on its own processor with its own memory, devices, etc.

- The resources of the physical machine are shared. Virtual devices are sliced out of the physical ones. Virtual disks are subsets of physical ones.
- Useful for running different OS simultaneously on the same machine.
- Protection is excellent, but no sharing possible.
- Virtual privileged instructions are trapped.



Virtual User

Virtual Machine

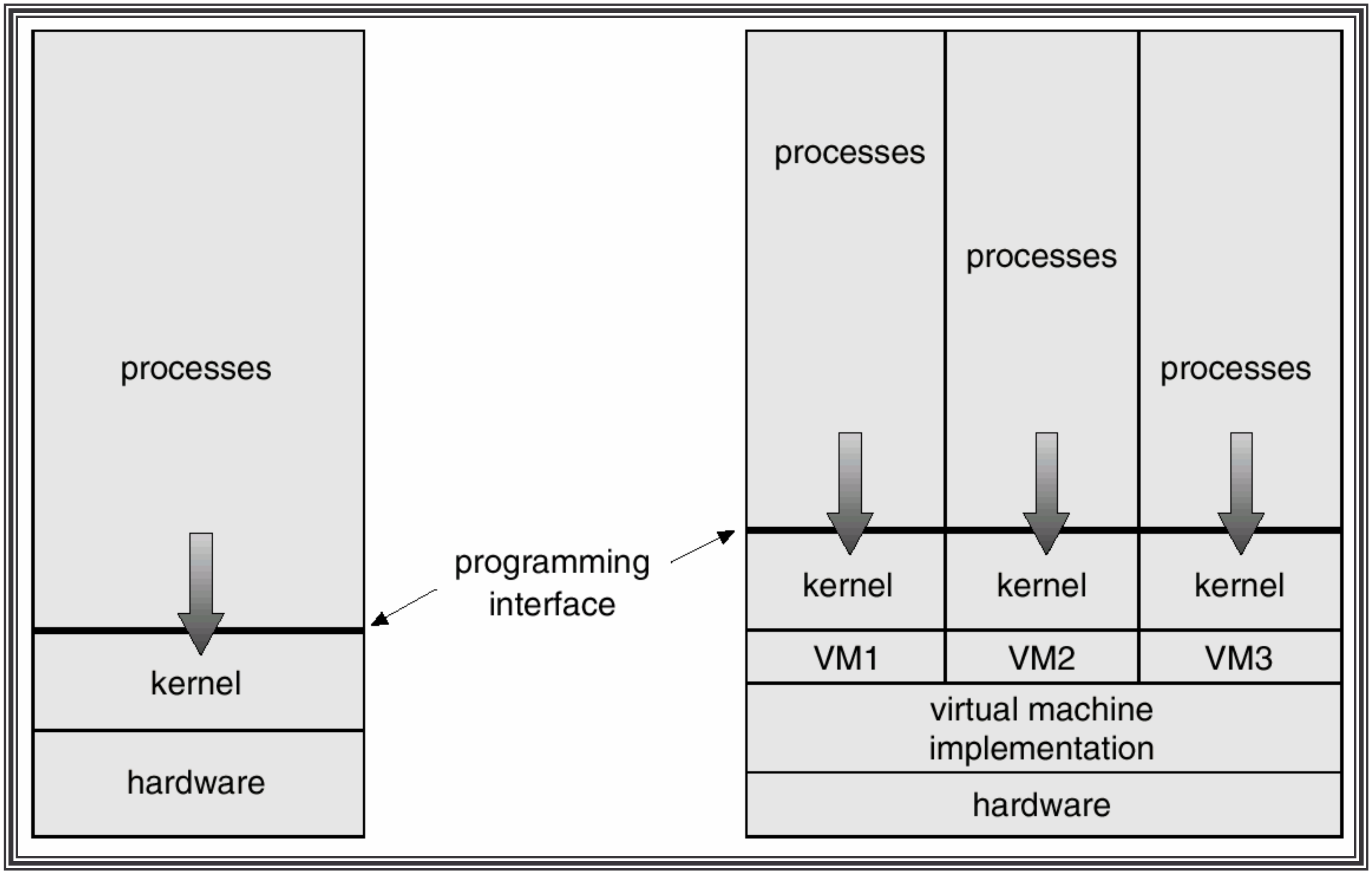
Monitor Mode

Physical User

Physical Machine

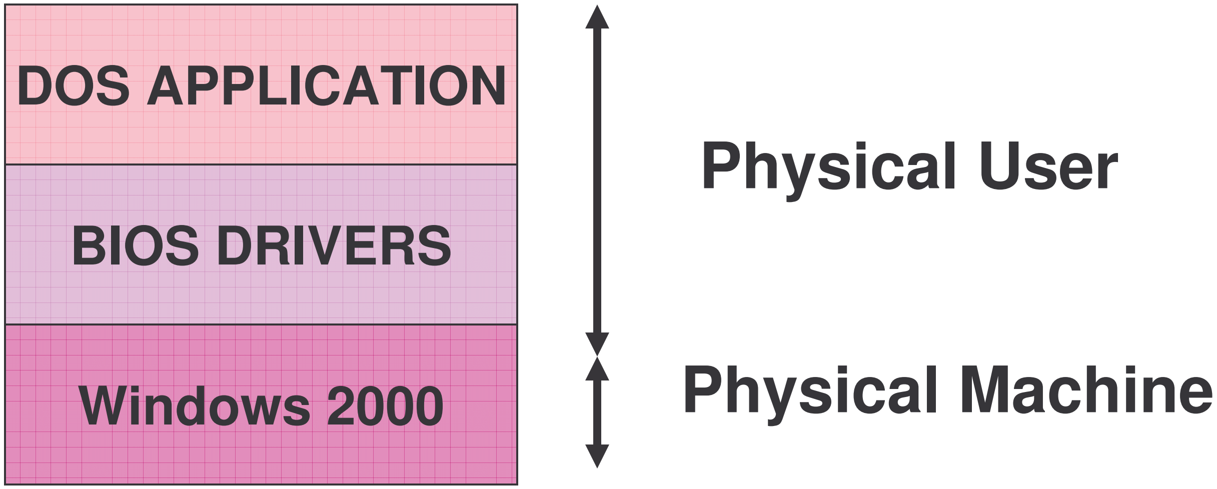
OPERATING SYSTEM STRUCTURES

Virtual Machine



Virtual Machine

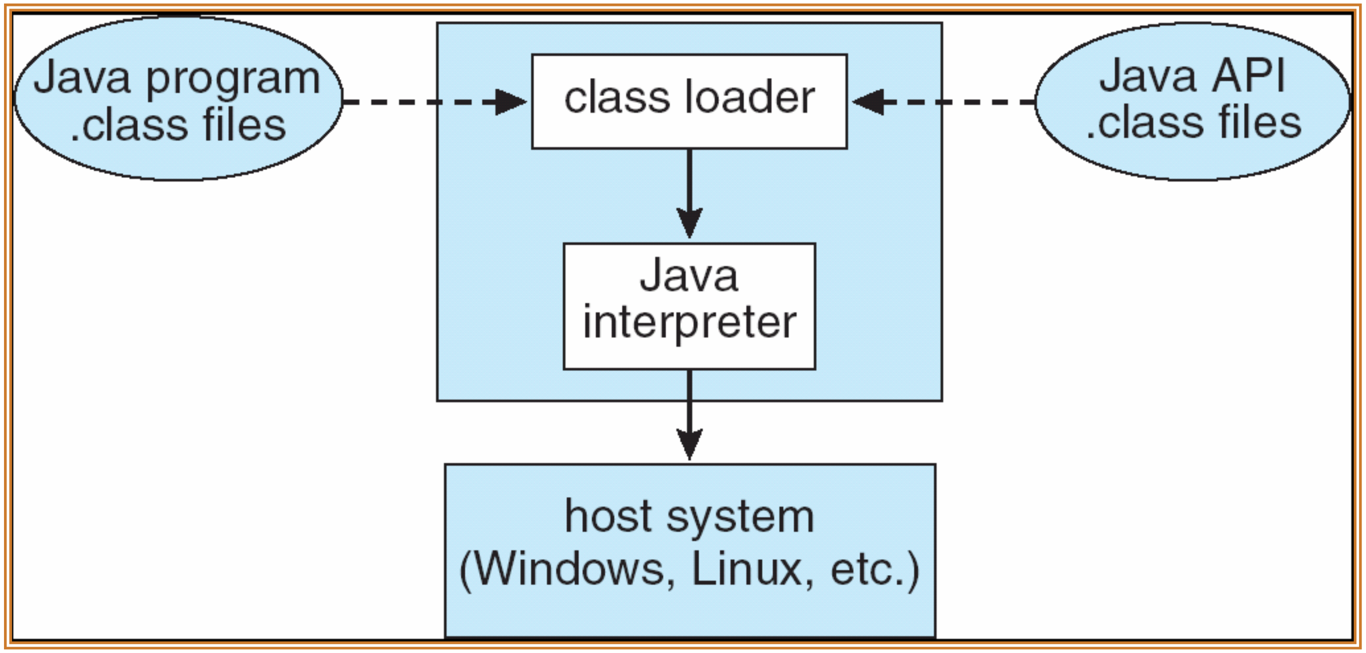
Example of MS-DOS on top of Windows 2000.



Virtual Machine

Example of Java Virtual Machine

The Java Virtual Machine allows Java code to be portable between various hardware and OS platforms.



OPERATING SYSTEM STRUCTURES

WRAPUP

We've completed our second overview of an Operating System – this at the level of a high flying plane.

We've looked at the basic building blocks of an operating system – processes, memory management, file systems, and seen how they all connect together.

Now we'll get into the nitty-gritty, spending considerable time on each of these pieces.