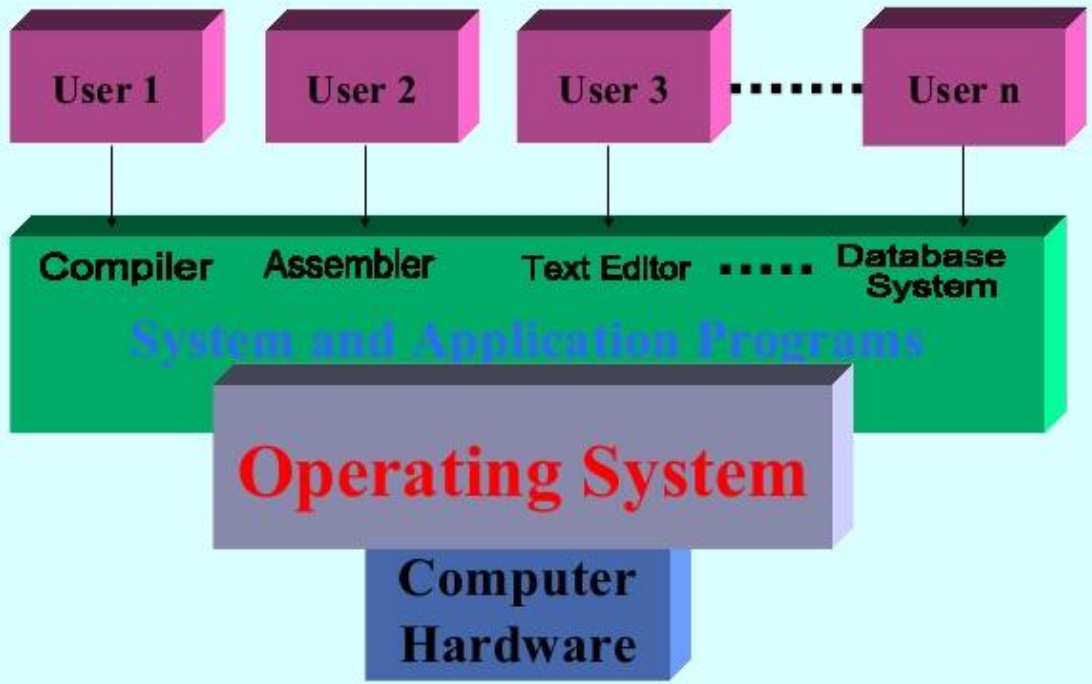
Operating System Structure

Abstract View of a Computer System



Views of OS

Three views of an operating system

- Application View: what services does it provide?
- System View: what problems does it solve?
- Implementation View: how is it built

Application View of an Operating System

The OS provides an execution environment for running programs.

The execution environment

- provides a program with the processor time and memory space that it needs to run.
- provides interfaces through which a program can use networks, storage, I/O devices, and other system hardware components.
  - Interfaces provide a simplified, abstract view of hardware to application programs.
- isolates running programs from one another and prevents undesirable interactions among them.

System View

The OS manages the hardware resources of a computer system. Resources include processors, memory, disks and other storage devices, network interfaces, I/O devices such as keyboard, mouse and monitor, and so on.

The operating system allocates resources among running programs.

It controls the sharing of resources among programs.

The OS itself also uses resources, which it must share with application programs.

Implementation View

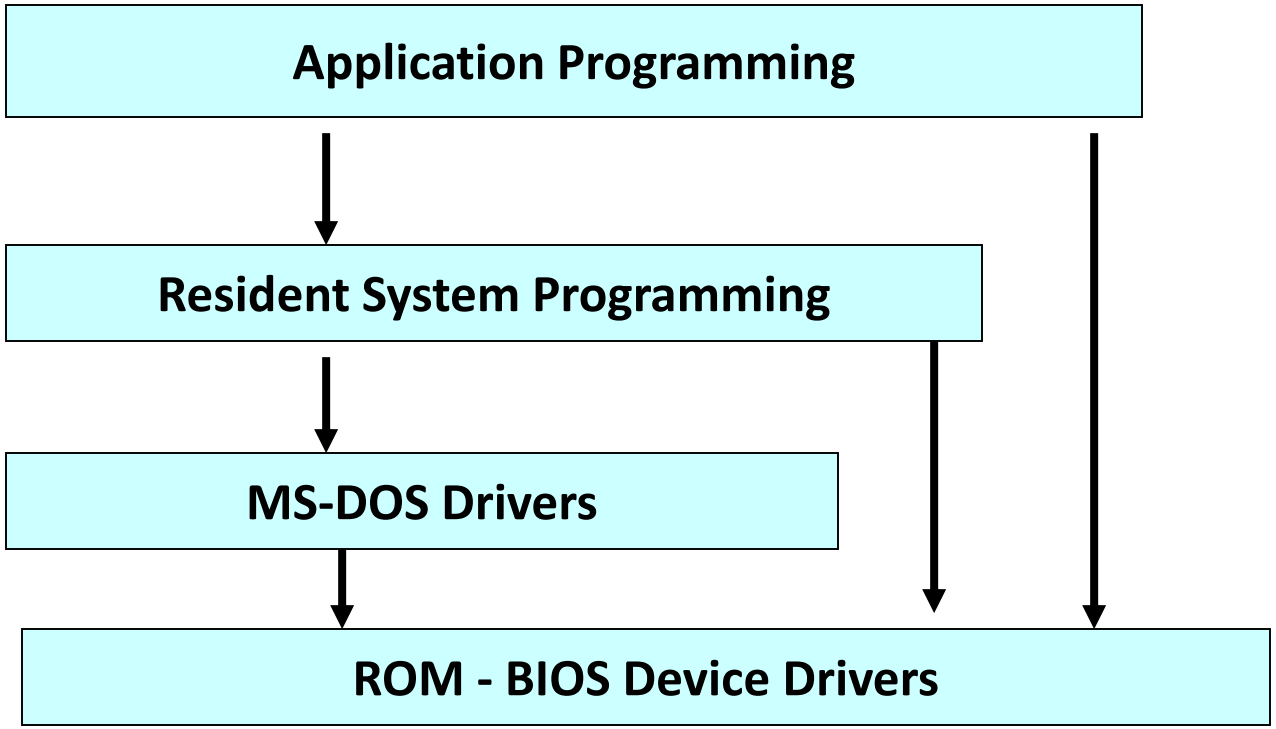
- The OS is a concurrent, real-time program.
- Concurrency arises naturally in an OS when it supports concurrent applications, and because it must interact directly with the hardware.
- Hardware interactions also impose timing constraints

OS Implementation

How An Operating System Is Put Together

A SIMPLE STRUCTURE:

Example of MS-DOS.

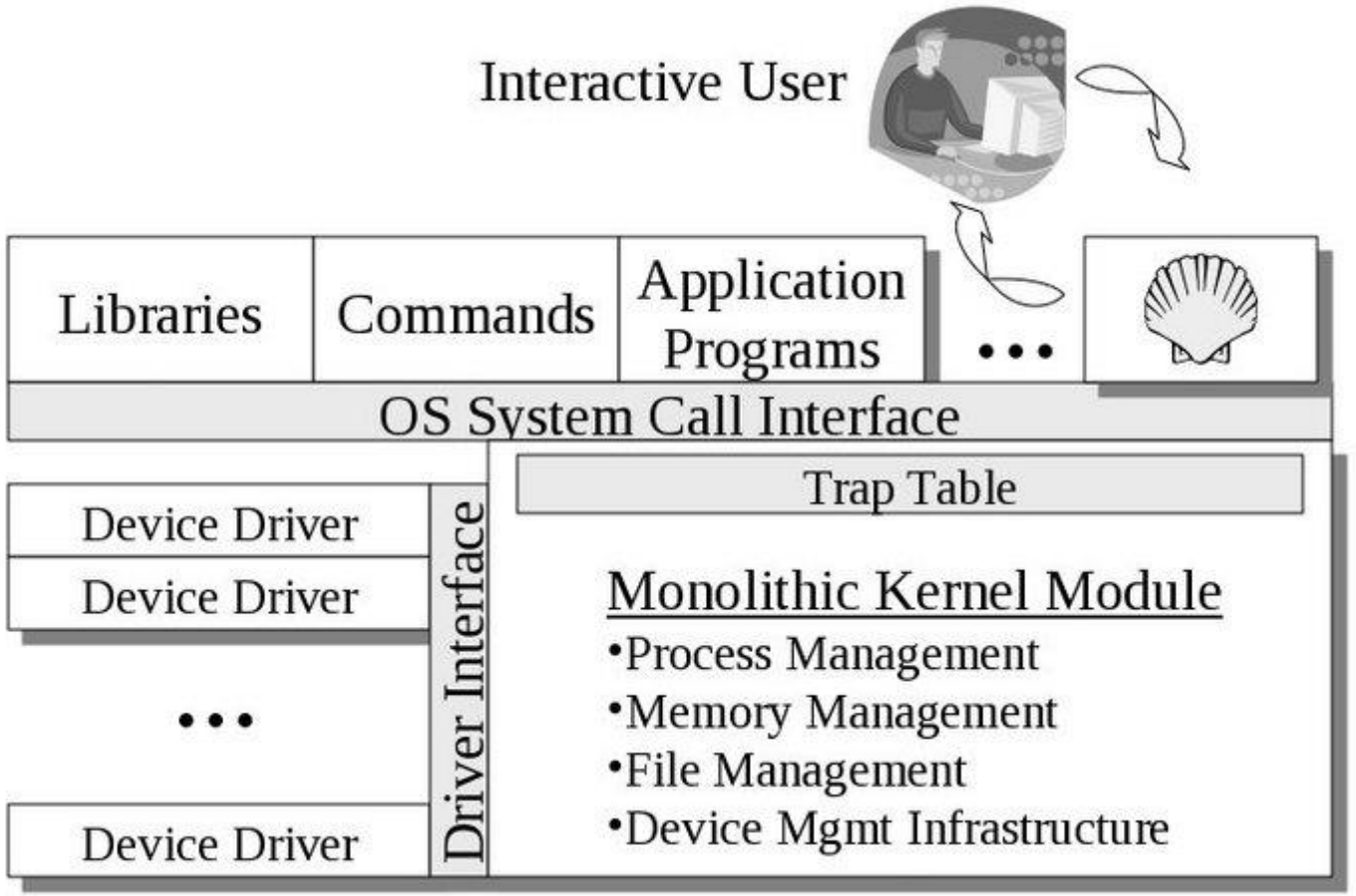


Simple Structure

- Operating systems such as MS-DOS and the original UNIX did not have well-defined structures.
- There was no CPU Execution Mode (user and kernel), and so errors in applications could cause the whole system to crash.

Monolithic Approach

•Functionality of the OS is invoked with simple function calls within the kernel, which is one large program.
•Device drivers are loaded into the running kernel and become part of the kernel.
e.g. OS/360, VMS and Linux.
Advantage:
Direct interaction between components makes monolithic OS highly efficient.
Disadvantages:
It is difficult to isolate the source of bugs and other errors because monolithic kernels group components together and also all code executes with unrestricted access to the system.



Layered Approach

- The layered approach to OS attempts to address the issue of OS becoming larger and more complex by grouping components that perform similar functions into layers.
- This approach breaks up the operating system into different layers.
- Each layer communicates exclusively with those immediately above and below it. Lower level layers provide services to higher level ones using an interface that hides their implementation.

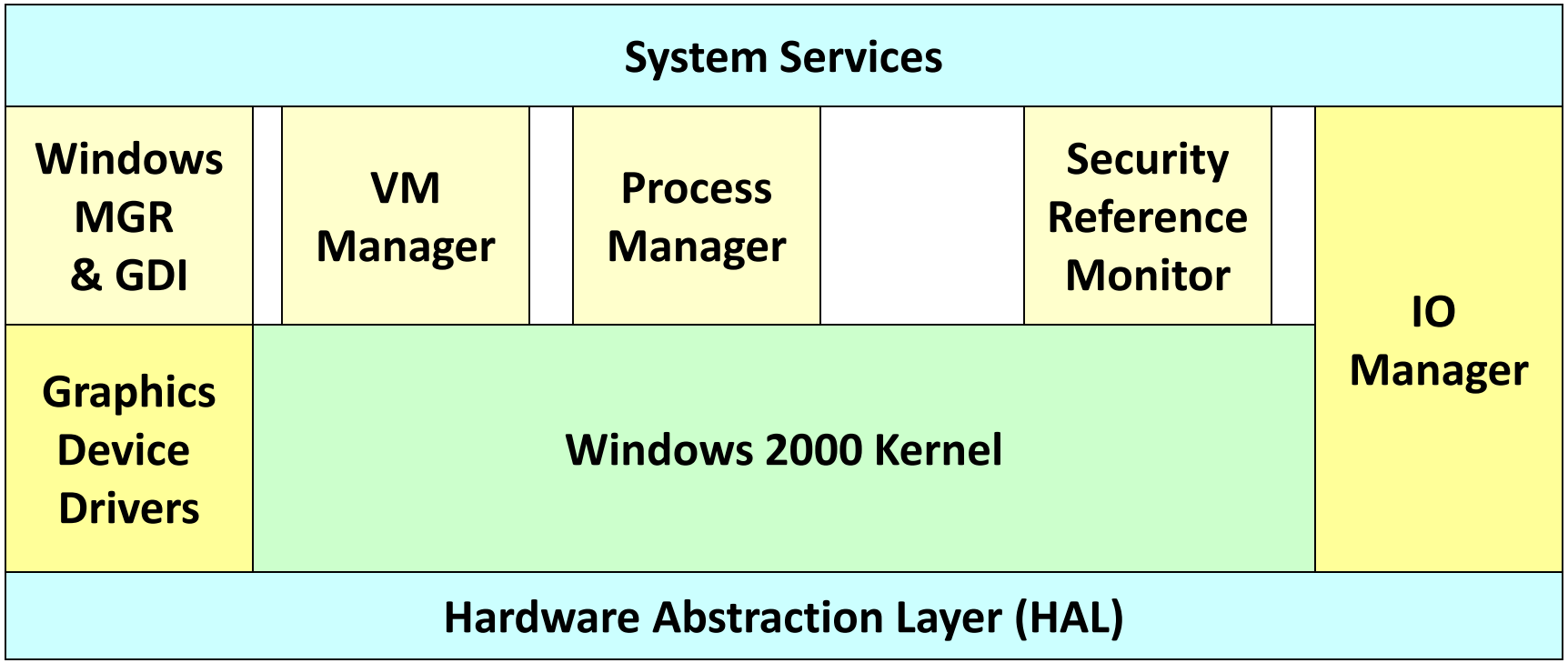
- Layered OS are more modular than monolithic OS because the implementation of each layer can be modified without requiring any modification to other layers. A modular system has self contained components that can be reused throughout the system.
- Each component hides how it performs its job and presents a standard interface that other components can use to request its services.
- With the layered approach, the bottom layer is the hardware, while the highest layer is the user interface.

- The main *advantage* is simplicity of construction and debugging.
- The main *difficulty* is defining the various layers.
- The main *disadvantage* is that the OS tends to be less efficient than other implementations.

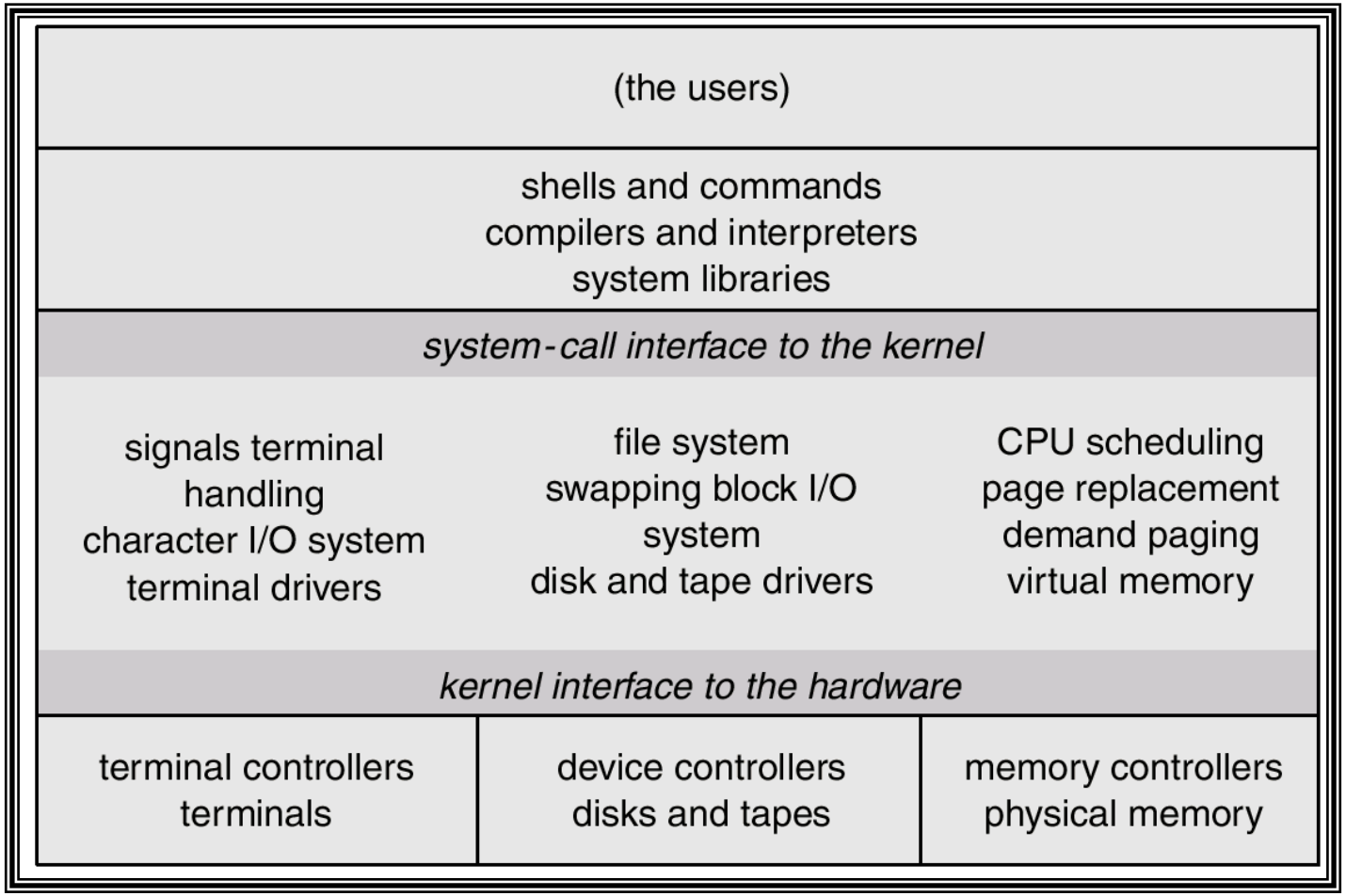
Hierarchical OS Model

Level	Name	Objects	Typical operations
5	Command Language interpreter	Environmental data	Statements in Command language
4	File system	Files, devices	Create, destroy, open, close, read and write
3	Memory management	Segments, pages	read, write, fetch
2	Basic I/O	Data blocks	read, write, allocate, free
1	Kernel	Process, semaphore	create, destroy, suspend, resume, signal, wait

Example - Windows 2000

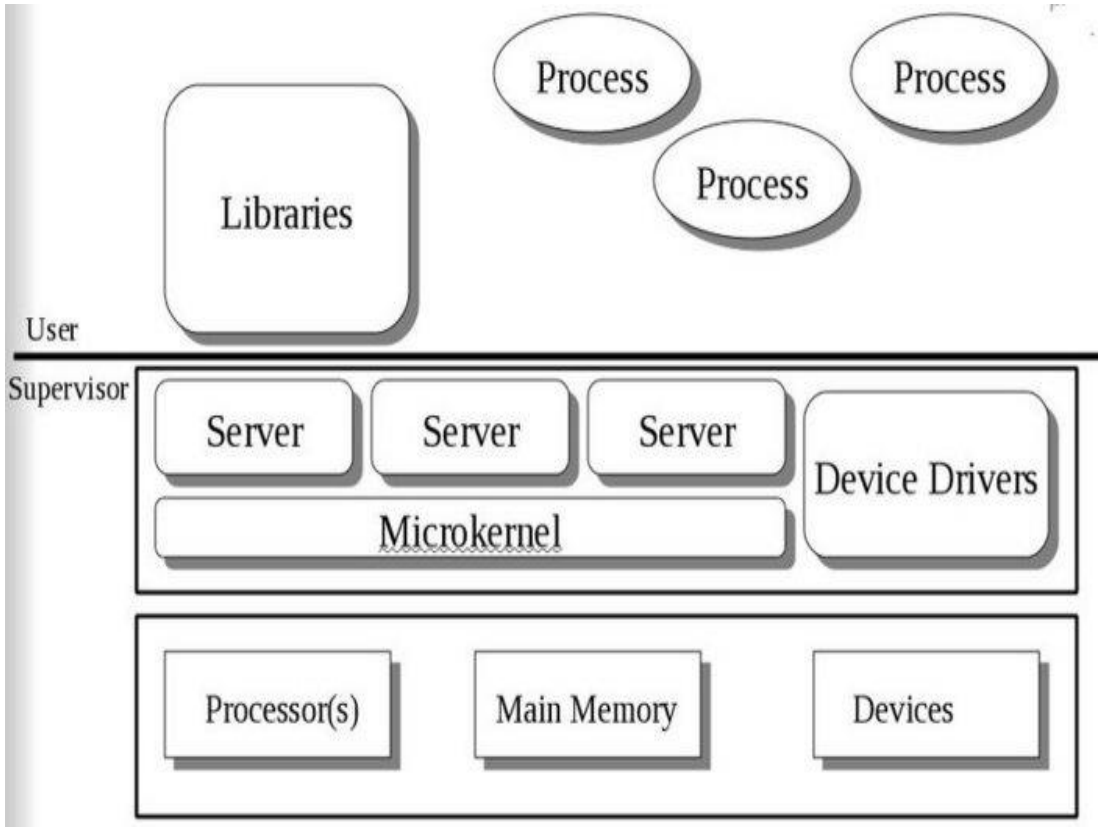


Unix


(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

Microkernels

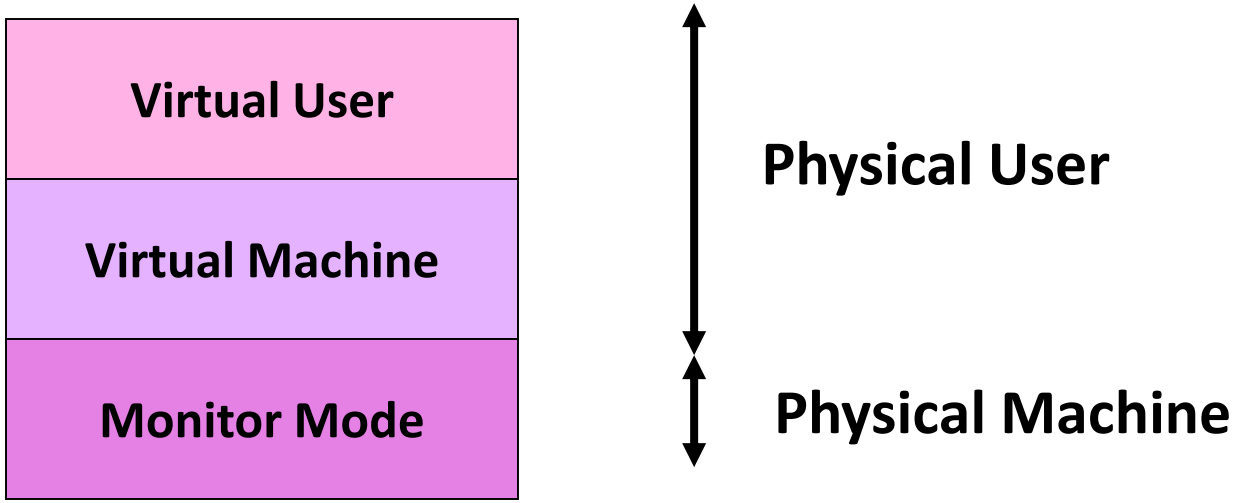
This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs.
•Generally they provide minimal process and memory management, and a communications facility.
•Communication between components of the OS is provided by message passing.
The *benefits* of the microkernel are as follows:
•Extending the operating system becomes much easier.
•Any changes to the kernel tend to be fewer, since the kernel is smaller.
•The microkernel also provides more security and reliability.
Main *disadvantage* is poor performance due to increased system overhead from message passing.



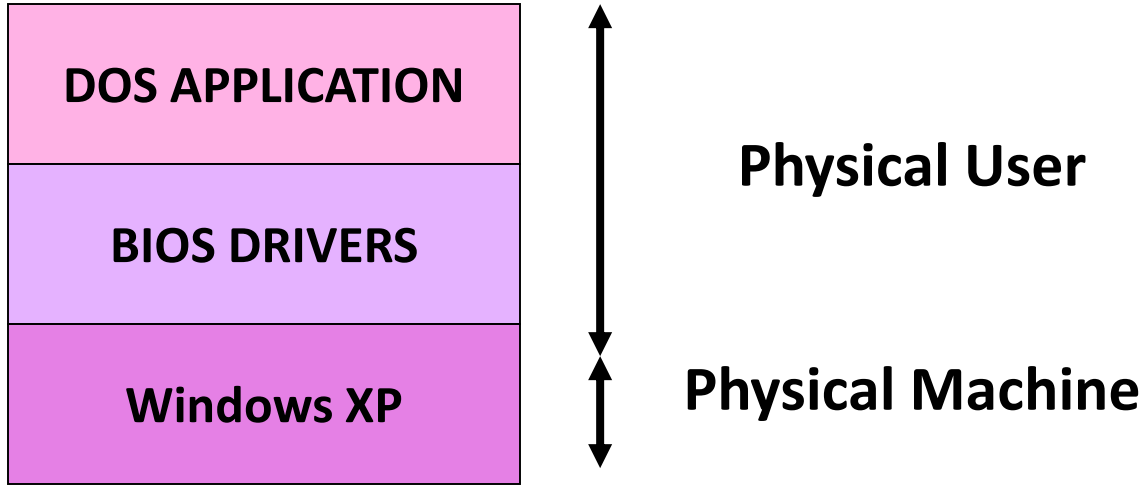
Virtual Machine

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

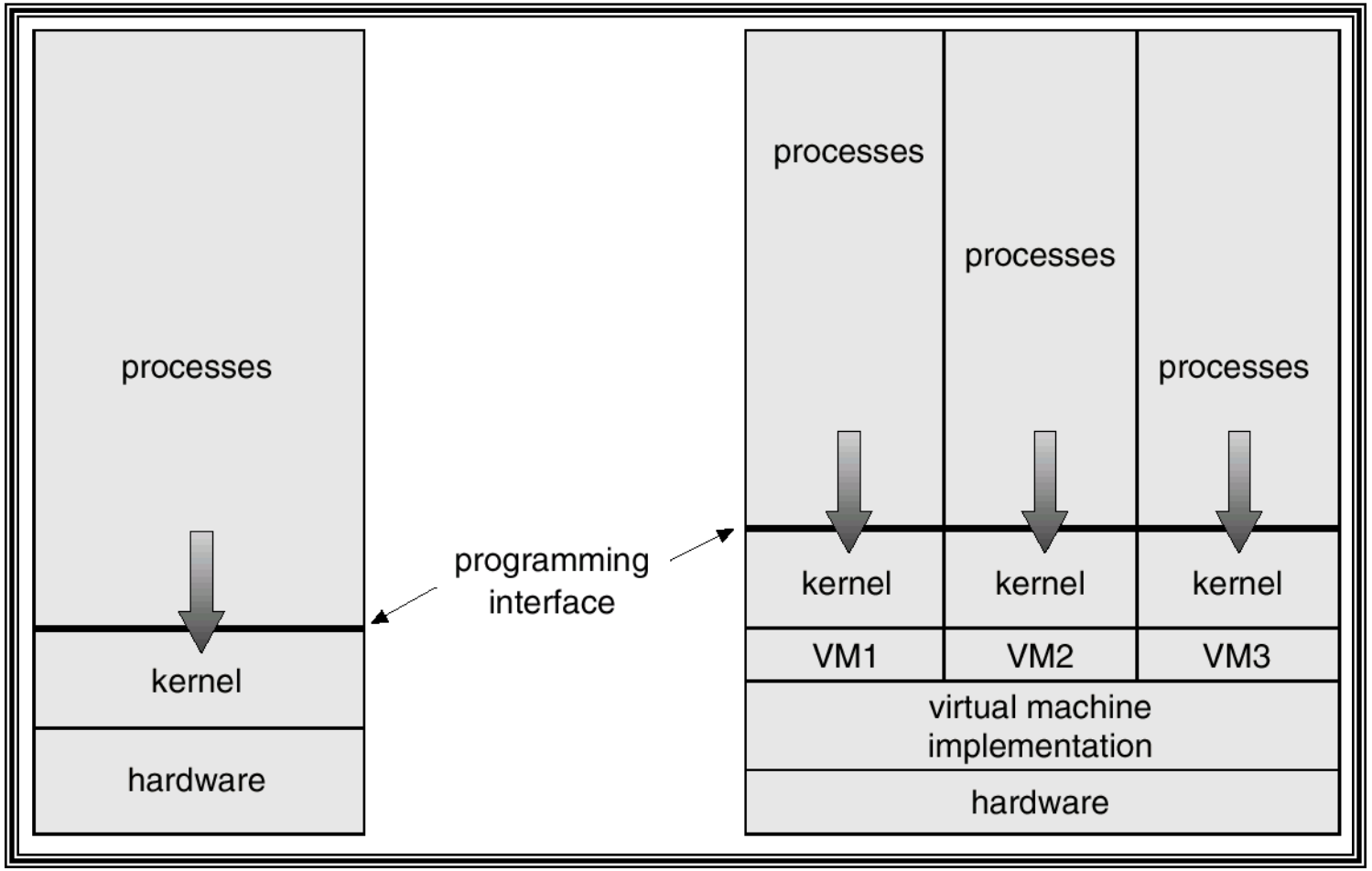
- The resources of the physical machine are shared. Virtual devices are sliced out of the physical ones. Virtual disks are subsets of physical ones.
- Useful for running different OS simultaneously on the same machine.
- Protection is excellent, but no sharing possible.
- Virtual privileged instructions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



Example of MS-DOS on top of Windows XP.



Virtual Machine



Example of Java Virtual Machine

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