

## **Chapter 3: Operating-System Structures**

- System Components
- Operating System Services
- System Calls
- System Programs
- System Structure
- Virtual Machines

## **Common System Components**

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

## Process Management

- A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- Processes can create sub-processes to execute concurrently.
- A program by itself is not a process; a program is a passive entity, whereas a process is an active entity.
- The execution of a process must progress in a sequential fashion. The CPU executes one instruction of the process after another until the process completes.
- Operating System processes: Those execute system code.
- User processes: Those that execute user code.

## Process Management (Cont.)

- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - Process suspension and resumption.
  - Provision of mechanisms for:
    - 1) process synchronization
    - 2) process communication
  - Deadlock handling

## Main-Memory Management

- Memory is a large array of words or bytes, each with its own address. It is a repository (storage) of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

## 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.
- A file consists of a sequence of bits, bytes, lines, or records whose meanings are defined by their creators.
- 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.

## I/O System Management

- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices
- The O.S. hides the peculiarities of specific hardware devices from the user.

## Secondary-Storage Management

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling

## Networking (Distributed Systems)

- A *distributed* system is a collection of processors that do not share memory or a clock. Each processor has its own local memory and clock.
- The processors in the system are connected through a communication network.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

## Protection System

- *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.

## Command-Interpreter System

- Command-Interpreter system is a system program, which is the interface between the user and the operating system.
- Command-Interpreter system is known as the shell.
- Some operating systems provide a user-friendly interface (mouse-based window) such as, Macintosh and Microsoft Windows.
- Some operating systems provide text interface (commands are typed on keyboard) such as MS-DOS and Unix shells.

## Command-Interpreter System (Cont.)

- Many commands are given to the operating system by control statements which deal with:
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access
  - protection
  - networking
- The program that reads and interprets control statements is called variously:
  - control-card interpreter
  - command-line interpreter
  - shell (in UNIX)
- Its function is to get and execute the next command statement.

## Operating System Services

- Program execution – system capability to load a program into memory and to run it.
- I/O operations – since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
- File-system manipulation – program capability to read, write, create, and delete files.
- Communications – exchange of information between processes executing either on the same computer or on different systems tied together by a network. Implemented via *shared memory* or *message passing*.
- Error detection – ensure correct computing by detecting errors in the CPU (such as power failure) and memory hardware, in I/O devices (such as connection failure), or in user programs.

## Additional Operating System Functions

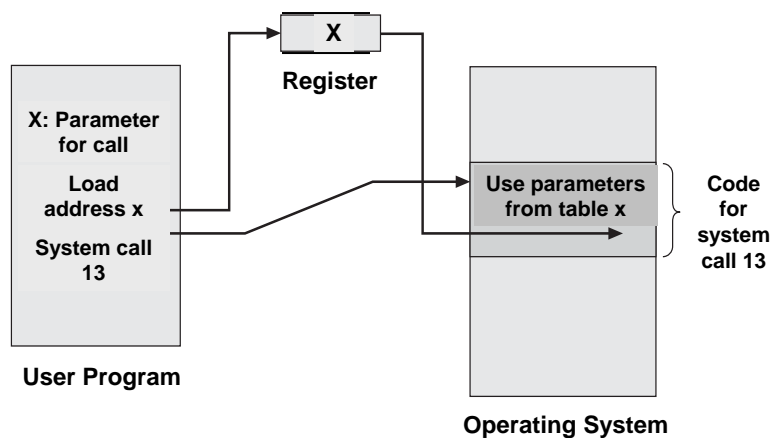
Additional functions exist not for helping the user, but rather for ensuring efficient system operations.

- Resource allocation – allocating resources, such as CPU cycles, main memory, file storage, I/O devices, to multiple users or multiple jobs running at the same time.
- Accounting – keep track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
- Protection – ensuring that all access to system resources is controlled.

## System Calls

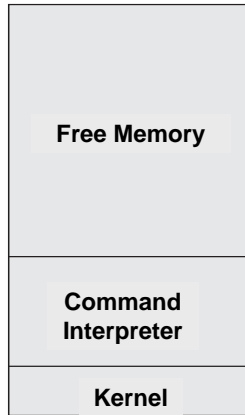
- System calls provide the interface between a running program and the operating system.
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C. Bliss, PL/360, PERL)
- Three general methods are used to pass parameters between a running program and the operating system.
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - *Push* (store) the parameters onto the *stack* by the program, and *pop* off the stack by operating system.

## Passing of Parameters As A Table



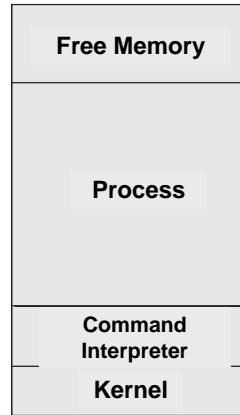
## MS-DOS Execution

At System Start-up



(a)

Running a Program

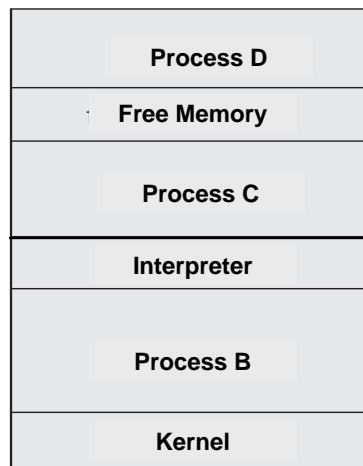


(b)

Operating System Concepts

3.17

## UNIX Running Multiple Programs

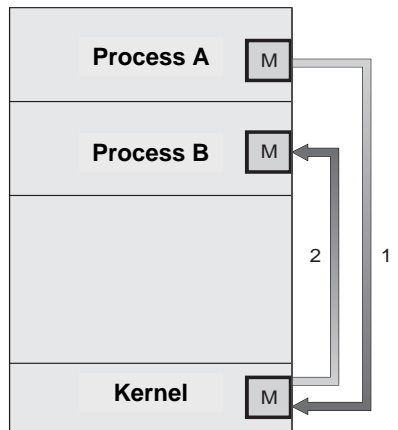


Operating System Concepts

3.18

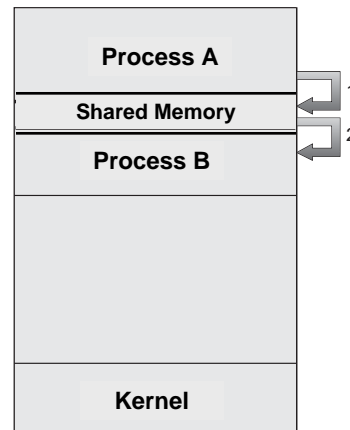
## Communication Models

### Message Passing



(a)

### Shared Memory



(b)

Operating System Concepts

3.19

## System Calls Categories

- System calls can be grouped into 5 categories:
  1. Process Control: end, abort, load, execute, create process, terminate process, allocate and free memory.
  2. File Manipulation: create file, delete file, open file, close file, read file, and write file.
  3. Device Manipulation: request device, release device, read, write.
  4. Information Maintenance: get time or date, set time or date, get process or file or device.
  5. Communications: create or delete communication connection, send and receive messages.

Operating System Concepts

3.20

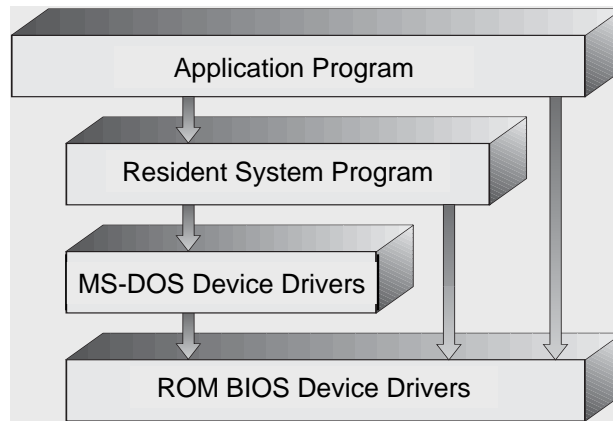
## System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into several categories:
  - File manipulation: create, delete, copy, rename, print files.
  - Status information: Some programs ask the system for date and time, disk space, number of users.
  - File modification: Text editors to create and modify the content of files stored on disk.
  - Programming language support: Compilers and assemblers are provided to the user with the O.S.
  - Program loading and execution: After a program is assembled or compiled, it must be loaded into memory to be executed. The system may provide loaders, linkage editors and debuggers.
  - Communications: Programs provide mechanism for creating virtual connections among processes, users, and computer systems, such as sending messages and transferring files.

## System Structure – Simple Approach

- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.
  - No dual mode and no hardware protection (Intel 8088) in MS-DOS.

## MS-DOS Layer Structure



Operating System Concepts

3.23

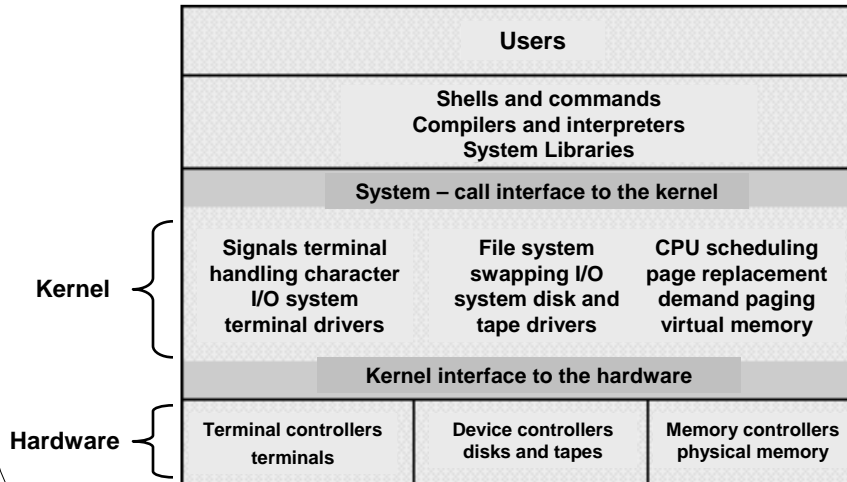
## System Structure – Simple Approach (Cont.)

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts.
  - Systems programs
  - The kernel
    1. Consists of everything below the system-call interface and above the physical hardware
    2. Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

Operating System Concepts

3.24

## UNIX System Structure



Operating System Concepts

3.25

## System Structure – Layered Approach

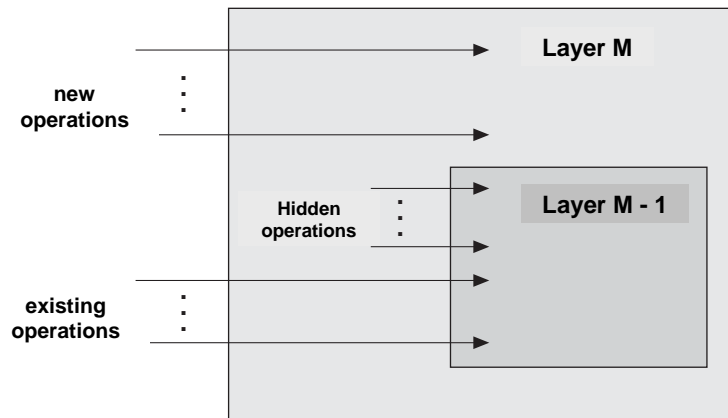
- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- Main advantage of layered approach:
  - First layer can be debugged without any concern, because it uses only basic hardware.
  - Once the first layer is debugged, its correct functioning while second layer is worked on and so on.
  - If an error occur we know in which layer.
  - Each layer is implemented using those operations provided by lower-level layers.
  - A layer does not need to know how the low-level operations are implemented, it needs to know what these operations do.
  - Each layer hides the existence of data structures, operations, and hardware from higher-level layer.

Operating System Concepts

3.26

## An Operating System Layer

It consists of data structures and a set of routines that can be invoked by higher-level layers.



Operating System Concepts

3.27

## Layered Structure of the THE OS

- A layered design was first used in THE operating system. Its six layers are as follows:

-----  
Layer 5: user programs

-----  
Layer 4: buffering for I/O devices

-----  
Layer 3: operator-console device driver

-----  
Layer 2: memory management (virtual memory)

-----  
Layer 1: CPU scheduling

-----  
Layer 0: hardware  
-----

Operating System Concepts

3.28

## Venus Layer Structure

- It consists of 7 layers as follows:

-----  
Layer 6: user programs

-----  
Layer 5: device drivers and schedulers

-----  
Layer 4: virtual memory

-----  
Layer 3: I/O channel

-----  
Layer 2: CPU scheduling

-----  
Layer 1: instruction interpreter

-----  
Layer 0: hardware

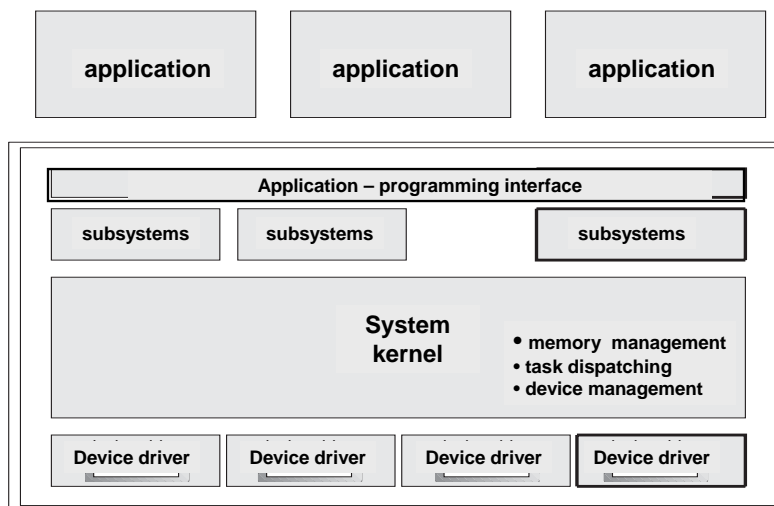
## Venus Layer Structure (Cont.)

- The low layers (layer 4 to layer 0) put into micro-code.
- Advantage: Additional speed of execution and clarity between micro-coded layers and higher layers.
- Order is important in layers.
- Each layer adds overhead to the system call, and the result is a system call takes longer than one does on a non-layered system.

## OS/2 Layer Structure

- OS/2 is a descendant of MS-DOS that adds multitasking and dual mode operation.
- Access to low-level facilities directly by user is not allowed.
- OS/2 provides more control over the hardware and more knowledge of which resources each user program is using than the MS-DOS.

## OS/2 Layer Structure (Cont.)



## Virtual Machines (VM)

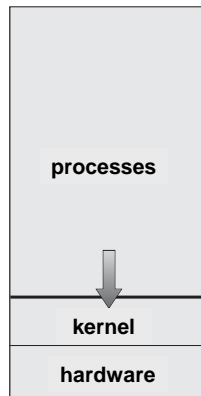
- A *virtual machine* takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- A virtual machine provides an interface *identical* to the underlying bare hardware.
- Example of disk systems in virtual machines: suppose you have 3 disks drives in physical machine and you want 7 disks drives in virtual machine. The solution is to provide virtual disks, which are identical in all respects except size, called minidisks in IBM's VM. The sum of the sizes of all minidisks must be less than the actual amount of physical disk space available.

## Virtual Machines (Cont.)

- The resources of the physical computer are shared to create the virtual machines.
- Implementation:
  - Two modes for protection.
  - In virtual machine, we must have a virtual user mode and a virtual monitor mode.
  - Both modes run in a physical user mode.

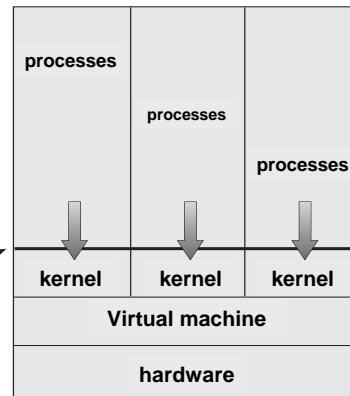
## System Models

### Non-virtual Machine



(a)

### Virtual Machine



(b)

## Advantages/Disadvantages of Virtual Machines

- The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- The virtual machine concept is difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine.
- Sharing minidisk: Files can be shared.
- Sharing using a network of virtual machines, each of which can send information over the virtual communication networks. Note that virtual minidisks and communication networks are modeled after physical disks and communication networks. The virtual minidisks and communication networks are implemented in software.
- So, two ways for sharing in virtual: minidisks and communication networks.

## **Advantages/Disadvantages of Virtual Machines (Cont.)**

- Modifying or changing an O.S. is difficult. Do it at night; the machine should be stopped, since the O.S. runs on and controls the entire machine.
- Virtual machines solves system compatibility problems: For example, MS-DOS programs on Intel CPU-based systems. Users like to use Sun Microsystems and DEC (faster processors). Solution is to create a virtual Intel machine on top of the Sun processors.

## **System Design Goals**

- User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast.
- System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient.

## System Implementation

- Traditionally written in assembly language, operating systems can now be written in higher-level languages.
- Code written in a high-level language:
  - can be written faster.
  - is more compact.
  - is easier to understand and debug.
- An operating system is far easier to *port* (move to some other hardware) if it is written in a high-level language.