CHAPTER 2
OPERATING SYSTEM STRUCTURES

Contents of chapter 2

- Components / Functions of Operating System
- Services of Operating System
- System Structure
- Operating System Structure/Architecture
- System Calls
- System Programs

System Calls

- System calls provide the interface between a process and the Operating System.
- These calls are generally available as assembly language instructions, and they are usually listed in the various manuals used by the assembly language programmers.
- Systems calls can be grouped roughly into five major categories : process control, file management, device management, information maintenance, and communications.

1. Process Control :

- A process or job executing one program my want to load on and execute another program. Create/ Terminate, Load/execute

  Wait/Signal, Allocate/ free memory, Wait for time
- Create : system call is used when we create a new job or process.
- Terminate : A running program needs to be able to halt its execution either normally(end) or abnormally(abort).
- Load/execute
- Wait/Signal :We may want to wait for a specific event to occur (wait event)
- Allocate/ free memory
- Wait for time : We may want to wait for a certain time (wait time) for a process to finish its execution.

2. File Management :

- Create/delete; open/close; read/write/seek; attributes.
- We first need to be able to create or delete files.
- We may also read, write or reposition (rewind or skip to the end of the file).
- Finally, we need to close the file.
- For either files or directories, we need to be able to determine the values of various attributes, and to reset them if necessary.
- File attributes include the file name, a file type, protection codes, accounting information and so on.
- At least two system calls, get file attribute and set file attribute are required for this function.

3. Device Management :

- Request/release device; read/write/seek, attributes.
- If a system has multiple users, we must first request the device to ensure exclusive use of it.
- After we are finished with the device, we must release it.
- Once the device has been requested (and allocated to us) we can read, write and reposition the device.
- get device attributes, set device attributes
- Logically attach or detach devices

4. Information Maintenance :

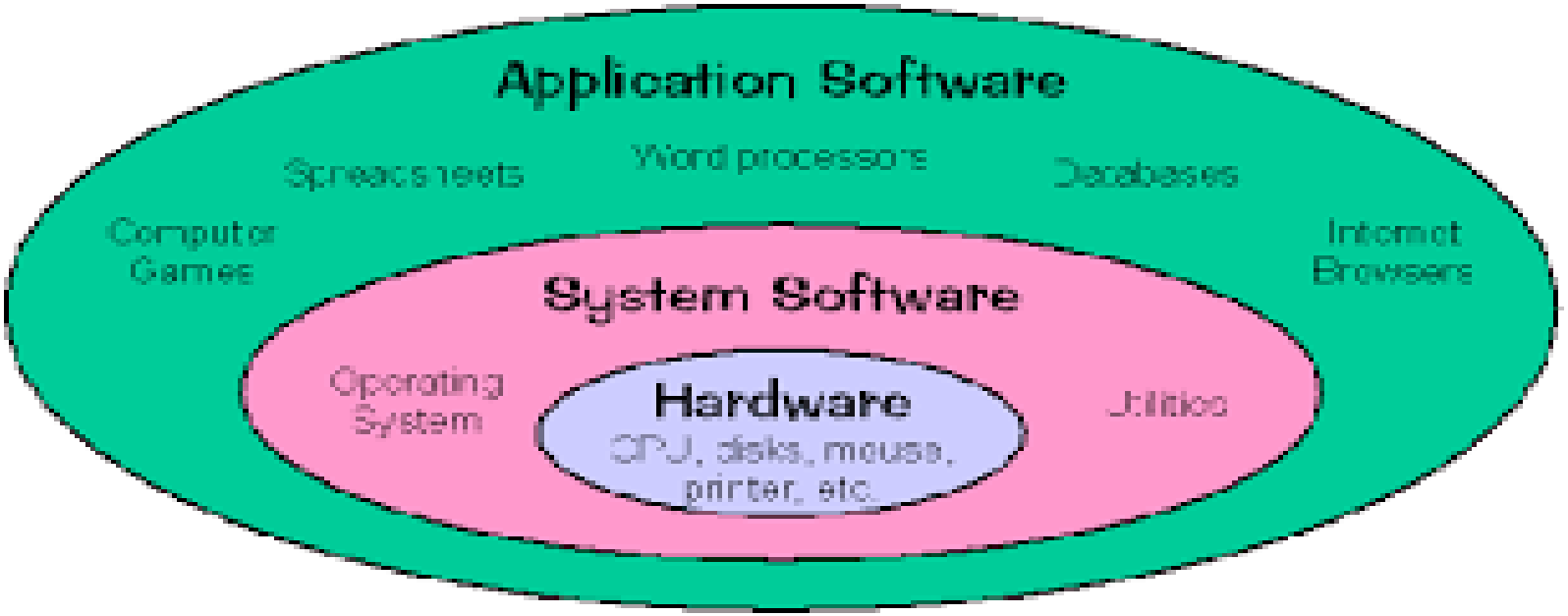
- Get time or date, set time/date
- Get system data, set system data
- Get process, file or device attributes
- Set process, file or device attributes

5. Communications :

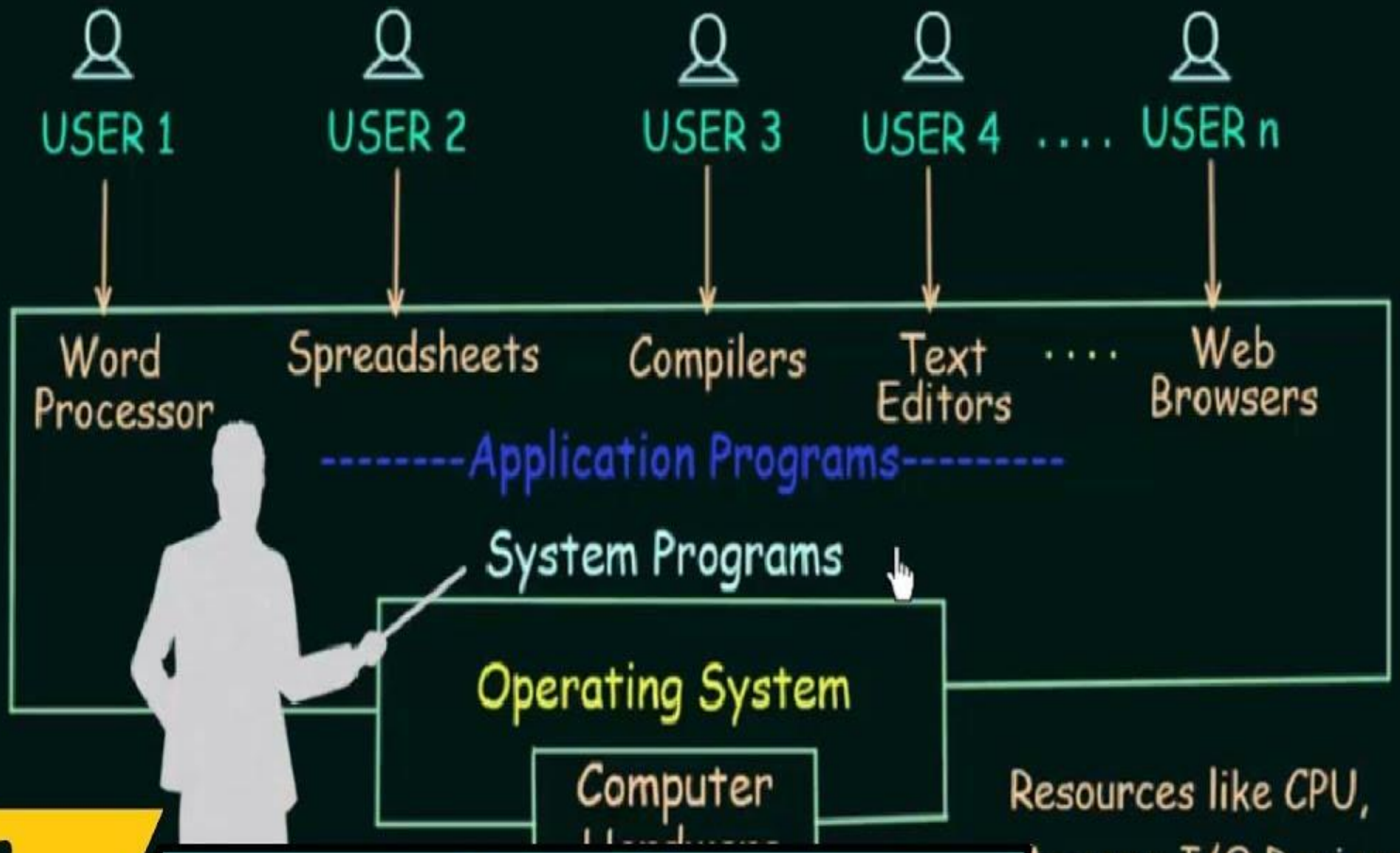
- Create, delete communication connection.
- Send, receive messages.
- Transfer status information.
- Attach or detach remote devices.

System Programs

- Aspect of modern system is the collection of system programs.
- The logical computer hierarchy shows that, at the lowest level is the hardware.
- Next are the operating system, then the system programs, and finally the application programs.



System Programs



11 Operating System

- System programs provide a convenient environment for program development and execution.
- Some of them are simply user interfaces to system calls; others are considerably more complex.
- They can be divided into these categories:

1. File Management : These programs create, delete, copy, rename, print, dump list and generally manipulate files and directories.
2. Status Information : Some programs simply ask the system for the date, time, amount of available memory or disk space, number of users, or similar status information.

   That information is then formatted, and is printed to the terminal or other output device or file.

3. File Modification : Several text editors may be available to create and modify content of files stored on disk or tape.
4. Programming-language support: compilers, assemblers and interpreters for common programming languages (such as C, C++, JAVA, VISUAL BASIC and PERL) are often provided to the user with the OS.

   Some of these programs are now priced and provided separately.
5. Program loading and execution : Once a program is assembled or compiled, it must be loaded into memory to be executed.

   The system may provide absolute loader, relocatable loaders, linkage editors, and overlay loaders.

   Debugging systems for either higher-level language or machine-level language are also needed.

6. Communications : These programs provide the mechanism for creating virtual connections among processes, users and different computer systems. They allow users to send messages to another's screen to browse web pages, to send electronic mail message, to log in remotely, or to transfer files from one machine to another.
7. System utilities or application programs : Most OS are supplied with programs that solve common problems, or perform common operations such programs include web browsers, word processors and text formatters, spreadsheets, DB systems, plotting & statistics-analysis package and games. These programs are known as System Utilities.

System Structure

- An operating system contains a kernel, command processor or shell and Graphical User Interface (GUI).

Kernel

- It is the control module of an operating system.
- It is that part of the OS that loads first and remains in the main memory.
- It is a bridge between applications and the actual data processing done at the hardware level.
- It is responsible for memory management, process and task management, and disk management. Thus, kernel's primary function is to manage the computer's resources and allow other programs to run and use these resources.
- Kernel provides the lowest level abstraction layer for various resources like processes and I/O devices that application software must control to perform it function.

- When a user gives command for performing any operation, then the request goes to the shell. The shell translates these human programs into machine language and then transfers the request to the kernel.
- The kernel receives the request from shell, processes it and displays the result on the screen.
- All these functions are performed by kernel in a transparent manner.

Shell

- A shell is a software that provides an interface for users of an operating system which provides access to the services of a kernel.
- The name shell originated from shells being an outer layer of interface between the user and the internals of the operating system (the kernel).
- OS shells are divided into two categories: command line and graphical.
- Command line shell is used in MS-DOS and graphical shells are used in windows that provides GUI facilities.

Command Processor

- It is that part of an operating system which receives and executes operating system commands.
- Whenever a command prompt is displayed the command processor waits for a command.
- After the user enters a command, the command processor makes sure that command is valid and then executes it or issues an error message.
- For an OS with graphical user interface, the command processor interprets mouse operations and executes the appropriate command. It is also known as command line interpreter.

Graphical User Interface

- It is provided so that the user need not remember tedious syntax of the command language, instead pointing to an option by means of a mouse can do his/her job.

BASICS OF OPERATING SYSTEM ARCHITECTURE

- An operating system architecture is usually organized into two part or modes, **kernel** or **supervisor mode** and **user** mode.
- This segregation is done in order to provide certain privileges to operating system that are not given to application programs.
- **Kernel mode** executes the operating system processes and **user mode** executes application programs of users.
- In other words, if the instruction is from the application program, the machine is said to be in the user mode. If it is executing any operating system instruction, the machine is said to be in the kernel mode.
- At any time, a single bit in PSW (Program Status Word) indicates whether instruction is user mode or kernel mode instruction as CPU can execute only one instruction at a time.
- A system can enter kernel mode from user mode by using any of these following methods:

  1. System Call
  2. Traps
  3. Interrupts

Computer System Operation

- For a computer to start running – for instance, when it is powered up or rebooted – it needs to have an initial program to run. This initial program, or bootstrap program, tends to be simple.
- Typically, it is stored in read-only memory (ROM) such as firmware or EEPROM within the computer hardware. It initializes all aspects of the system, from CPU registers to device controllers to memory contents.
- The bootstrap program must know how to load the OS and to start executing that system.
- To accomplish this goal, the bootstrap must locate and load into memory the operating system kernel.
- The OS then starts executing the first process, such as "**init**", and waits for some event to occur.

What is Kernel?

- A kernel is an important part of an OS that manages system resources.
- It also acts as a bridge between the software and hardware of the computer.
- It is one of the first program which is loaded on start-up after the bootloader.
- The Kernel is also responsible for offering secure access to the machine's hardware for various programs.
- It also decides when and how long a certain application uses specific hardware.

Interrupt

- The occurrence of an event is usually signaled by an **interrupt** from hardware or the software.
- Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus.
- Software may trigger an interrupt by executing a special operation called a system call(also called a monitor call).
- Modern Operating Systems are interrupt driven.
- If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an OS will sit quietly, waiting for something to happen.
- Events are almost always signaled by the occurrence of an **interrupt** or a **trap**.

Trap

- A trap (or an exception) is a software-generated interrupt caused either by an error (for example, division by zero or invalid memory access) or by a specific request from a user program that an operating-system service be performed.
- The interrupt-driven nature of an OS defines that system's general structure.
- For each type of interrupt, separate segments of code in the OS determine what action should be taken.
- An interrupt service routine is provided that is responsible for dealing with the interrupt.
- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location.
- The fixed location usually contains the starting address where the service routine for the interrupt is located.
- The interrupt service routine executes; on completion, the CPU resumes the interrupted computation.

ARCHITECTURE OF AN OPERATING SYSTEM

- The fundamental structure of an operating system that defines the interconnection between the system components.
- An operating system can have different architectures:
  1. Monolithic
  2. Microkernel Architecture
  3. Layered Architecture
  4. Client server architecture
  5. Virtual Machine architecture

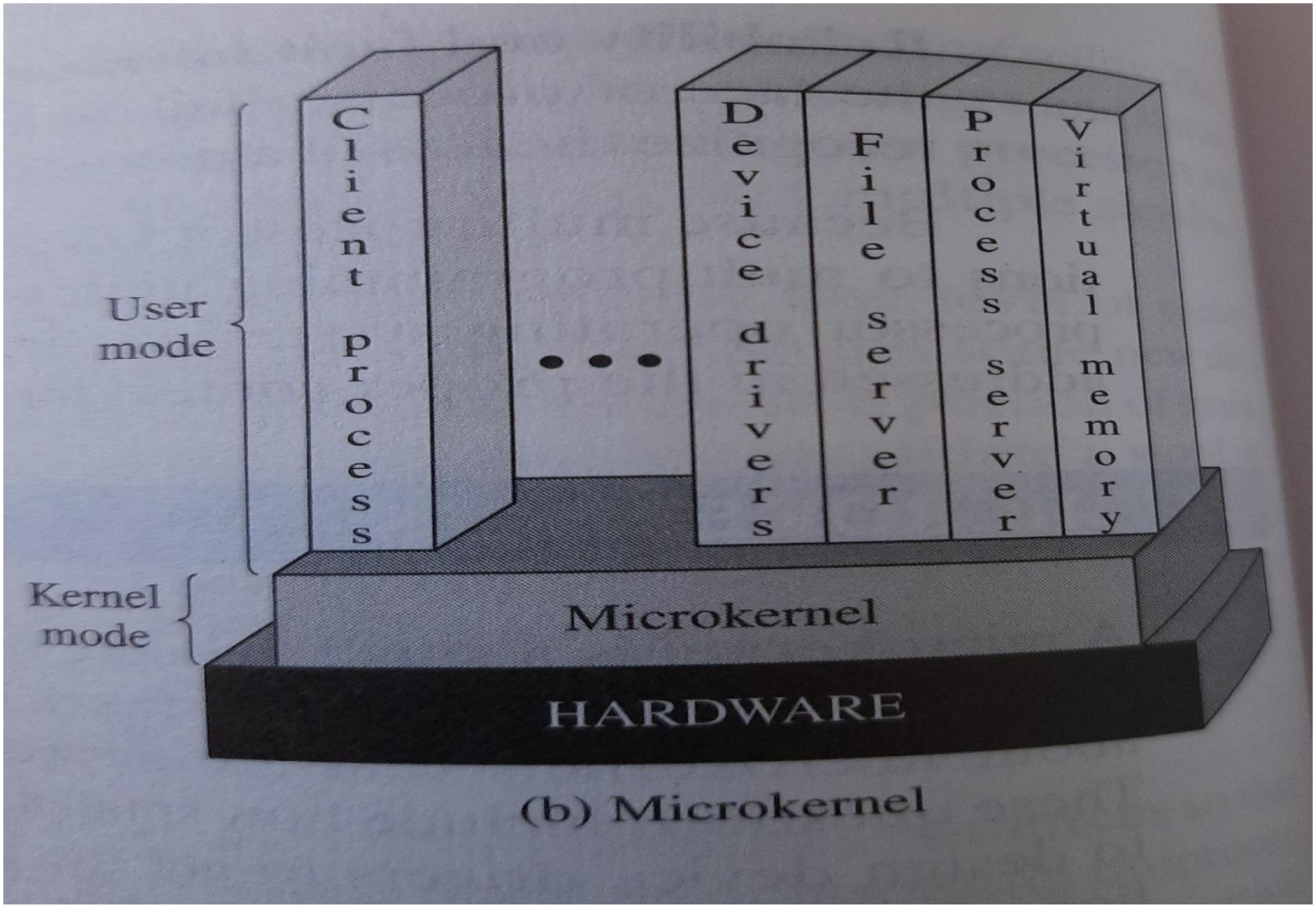
Monolithic Architecture

- *A large kernel containing virtually the complete OS.*
- A **monolithic kernel** is an operating system architecture where the entire operating system is working in kernel space. This increases the size of the kernel as well as the operating system.
- **Advantages of Monolithic Kernel**
- Some of the advantages of monolithic kernel are −
- The execution of the monolithic kernel is quite fast as the services such as memory management, file management, process scheduling etc., are implemented under the same address space.
- A process runs completely in a single address space in the monolithic kernel.
- The monolithic kernel is a static single binary file.
- **Disadvantages of Monolithic Kernel**
- Some of the disadvantages of monolithic kernel are −
- If any service fails in the monolithic kernel, it leads to the failure of the entire system.
- To add any new service, the entire operating system needs to be modified by the user.

Microkernel Architecture

- The microkernel contains basic requirements such as memory, process scheduling mechanisms and basic inter process communication.
- The only software executing at the privileged level i.e. kernel mode is the microkernel.
- The other functions of the operating system are removed from the kernel mode and run in the user mode. These functions may be device drivers, file servers, application interprocess communication etc.
- The microkernel makes sure that the code can be easily managed because the services are divided in the user space.
- This means that there is less code running in the kernel mode which results in increased security and stability.

Microkernel Architecture Diagram



Benefits of Microkernels

- Some of the benefits of microkernels are –
- Microkernels are modular and the different modules can be replaced, reloaded, modified, changed etc. as required. This can be done without even touching the kernel.
- All new services are added to user space and consequently do not require modification of the kernel.
- When the kernel does have to be modified, the changes tend to be fewer, because the microkernel is a smaller kernel.
- The resulting OS is easier to port from one hardware to another.
- Microkernels contain fewer system crashes as compared to monolithic systems. Also, the crashes that do occur can be handled quite easily due to the modular structure of microkernels.
- Microkernel also provides more security and reliability, since most services are running as user rather than kernel processes.
- If the service fails rest of the OS remains untouched.

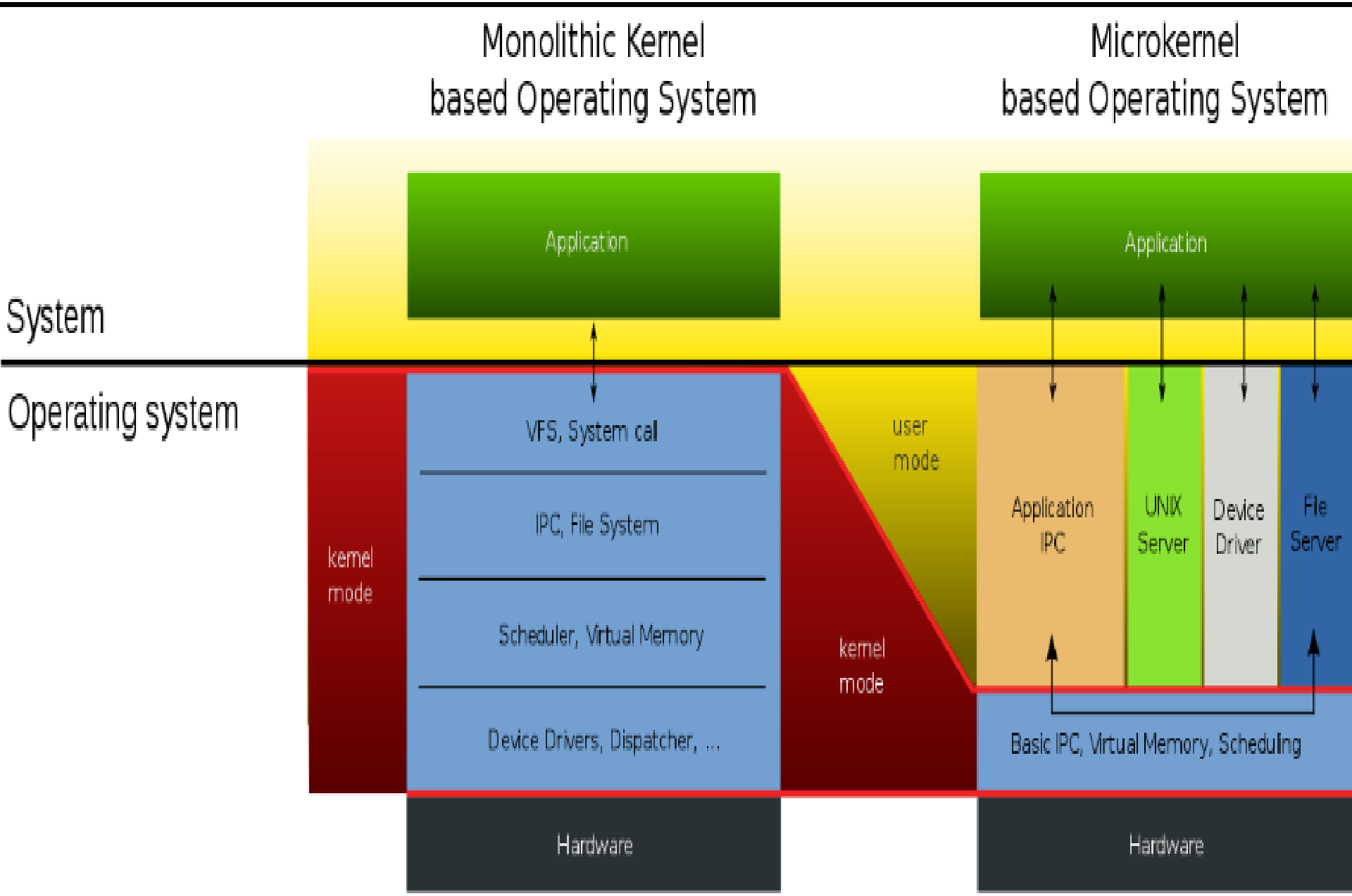
Disadvantages of Microkernel architecture

- Providing services in a **microkernel system** are expensive compared to the normal monolithic **system.**

Differences Between Microkernel and Monolithic Kernel

Difference Between Microkernel and Monolithic Kernel

Parameters	Monolithic kernel	MicroKernel
Basic	It is a large process running in a single address space	It can be broken down into separate processes called servers.
Code	In order to write a monolithic kernel, less code is required.	In order to write a microkernel, more code is required
Security	If a service crashes, the whole system collapses in a monolithic kernel.	If a service crashes, it never affects the working of a microkernel.
Communication	It is a single static binary file	Servers communicate through IPC.
Example	Linux, BSDs, Microsoft Windows (95,98, Me), Solaris, OS-9, AIX, DOS, XTS-400, etc.	L4Linux, QNX, SymbianK42, Mac OS X, Integrity, etc.


Monolithic Kernel
based Operating System
Microkernel
based Operating System
System
Operating system
Application
Application
VFS, System cal
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
kernel
mode
user
mode
kernel
mode
Application
IPC
UNIX
Server
Device
Driver
File
Server
Basic IPC, Virtual Memory, Scheduling
Hardware
Hardware

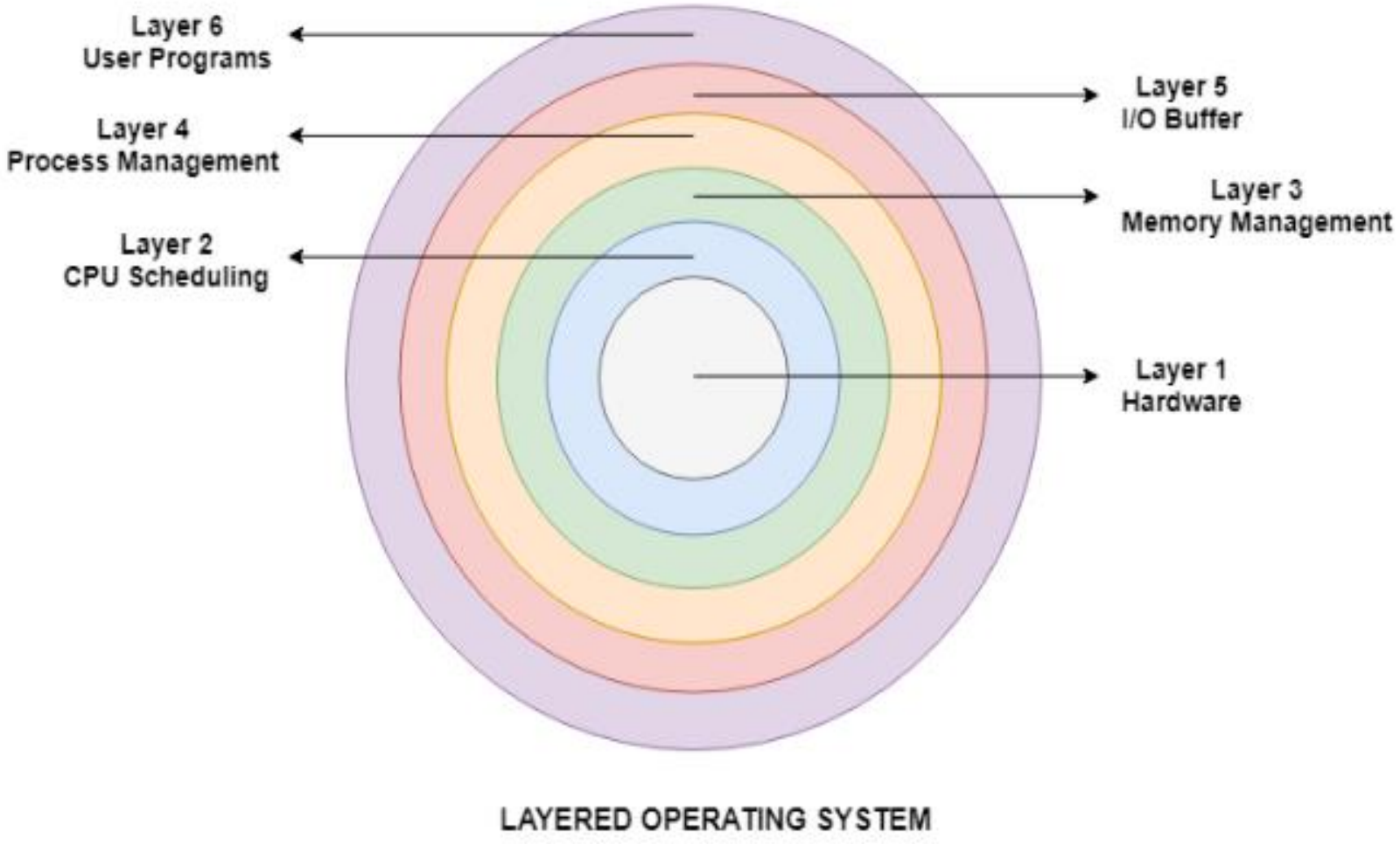
Summary:

- A kernel is an important part of an OS that manages system resources.
- A microkernel is a software or code which contains the required minimum amount of functions, data, and features to implement an operating system.
- In Monolithic Kernel approach, the entire operating system runs as a single program in kernel mode
- A Microkernel is the most important part for correct implementation of an operating system.
- A microkernel comprises only the core functionalities of the system.
- A monolithic kernel is a large process running in a single address space, whereas Microkernel can be broken down into separate processes called servers.
- Microkernel architecture is small and isolated therefore it can function better
- Providing services in a microkernel system are expensive compared to the normal monolithic system

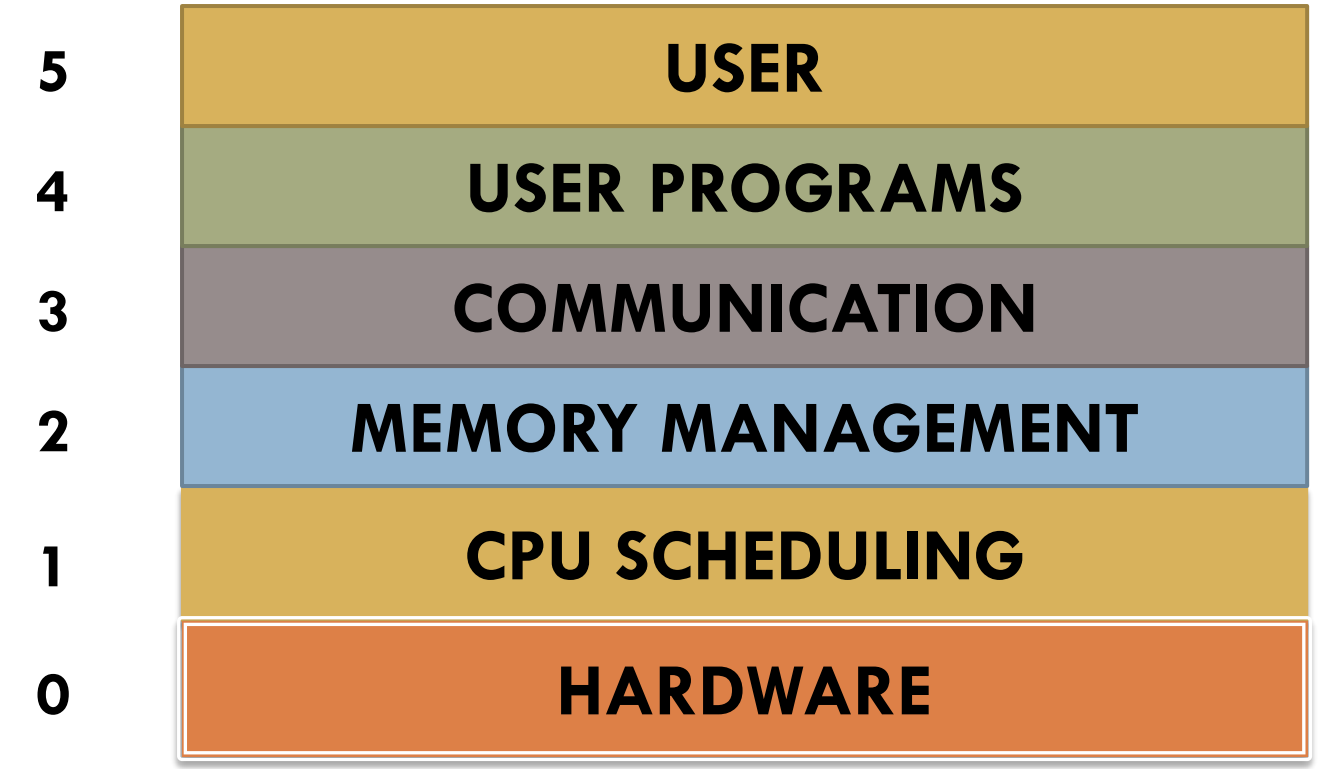
Layered Architecture

- Layering provides a distinct advantage in an operating system. All the layers can be defined separately and interact with each other as required. Also, it is easier to create, maintain and update the system if it is done in the form of layers. Change in one layer specification does not affect the rest of the layers.
- Each of the layers in the operating system can only interact with the layers that are above and below it. The lowest layer handles the hardware and the uppermost layer deals with the user applications.

Layered OS Representation



Layered System



Six layers of Layered System

Layer 1 : Hardware

- This layer interacts with the system hardware and coordinates with all the peripheral devices used such as printer, mouse, keyboard, scanner etc. The hardware layer is the lowest layer in the layered operating system architecture.

Layer 2 : CPU Scheduling

- This layer deals with scheduling the processes for the CPU. There are many scheduling queues that are used to handle processes. When the processes enter the system, they are put into the job queue. The processes that are ready to execute in the main memory are kept in the ready queue.

Layer 3 : Memory Management

- Memory management deals with memory and the moving of processes from disk to primary memory for execution and back again. This is handled by the third layer of the operating system.

Layer 4 : Process Management

- This layer is responsible for managing the processes i.e assigning the processor to a process at a time. This is known as process scheduling. The different algorithms used for process scheduling are FCFS (first come first served), SJF (shortest job first), priority scheduling, round-robin scheduling etc.

Layer 5 : I/O Buffer

- I/O devices are very important in the computer systems. They provide users with the means of interacting with the system. This layer handles the buffers for the I/O devices and makes sure that they work correctly.

Layer 6 : User Programs

- This is the highest layer in the layered operating system. This layer deals with the many user programs and applications that run in an operating system such as word processors, games, browsers etc.

ADVANTAGES AND DISADVANTAGES LAYERED ARCHITECTURE

ADVANTAGES

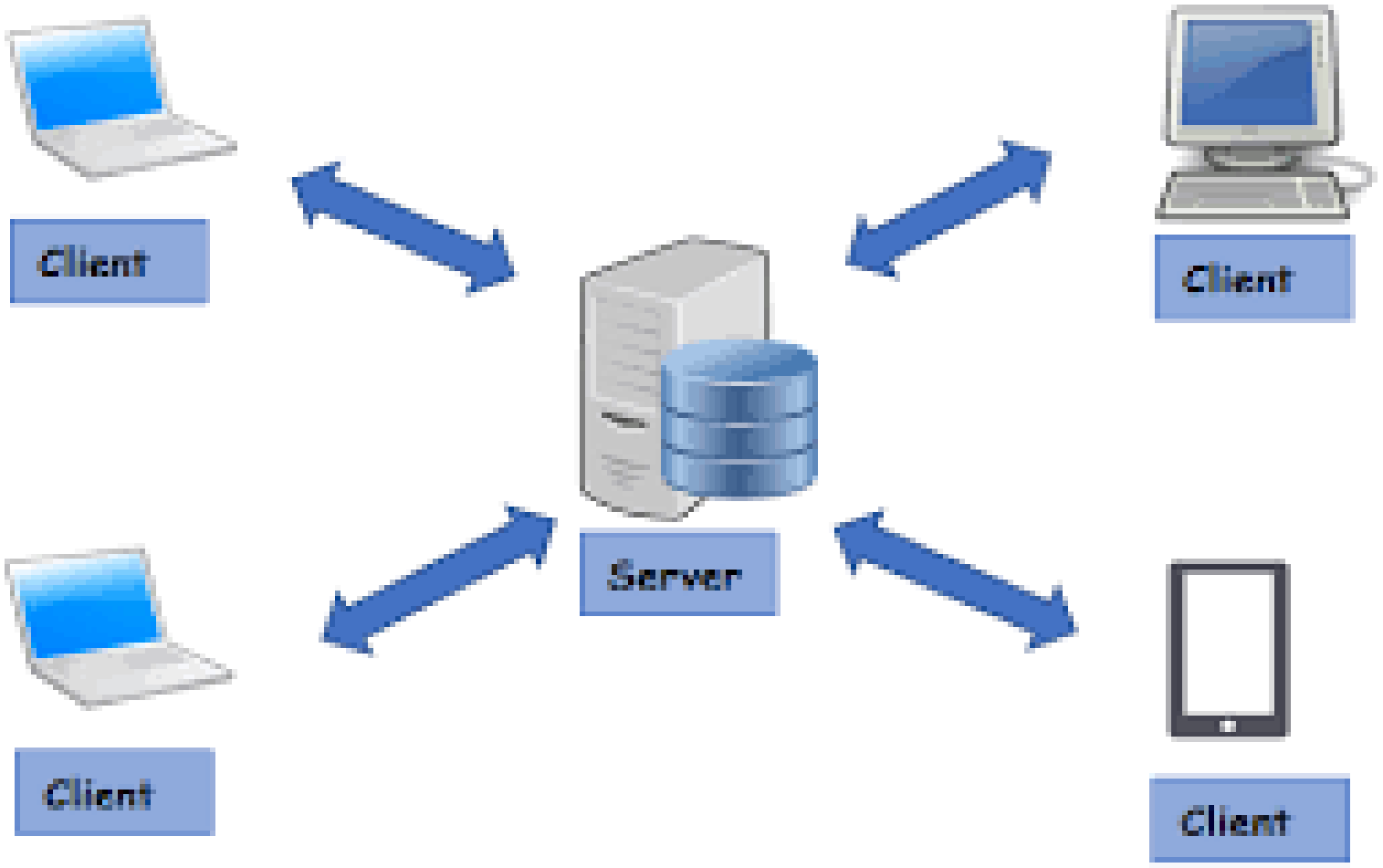
- Layered architecture also helps you to test the components independently of each other.

DISADVANTAGES

- There might be a negative impact on the **performance** as we have the extra overhead of passing through layers instead of calling a component directly.

Client-Server Architecture

- **Client-server architecture, architecture** of a computer network in which many clients (remote processors) request and receive service from a centralized **server** (host computer).
- **Client** computers provide an interface to allow a computer user to request services of the **server** and to display the results the **server** returns.
- **Examples** of computer applications that use the **client-server model** are email, network printing, and the World Wide Web.


Client
Client
Server
Client
Client

Advantages of Client-Server Architecture

- **Centralization of contro**l: access, resources and integrity of the data are controlled by the **dedicated server** so that a program or unauthorized client cannot damage the system. This centralization also facilitates task of updating data or other resources (better than the networks P2P).
- **Scalability**: You can increase the capacity of clients and servers separately. Any element can be increased (or enhanced) at any time, or you can add new nodes to the network (clients or servers).
- **Easy maintenance**: distribute the roles and responsibilities to several standalone computers, you can replace, repair, upgrade, or even move a server, while customers will not be affected by that change (or minimally affect). This independence of the changes is also known as encapsulation.
- There are technologies sufficiently developed, designed for the paradigm of C / S to ensure security in transactions, interface friendliness, and ease of use.

Disadvantages of Client-Server Architecture

- Traffic congestion has always been a problem in the paradigm of C / S. When a large number of simultaneous clients send requests to the same server might cause many problems for this (to more customers, more problems for the server).
- When a server is down, customer requests cannot be met.

Examples of Client-Server Architecture

- **Examples** of servers include web servers, mail servers, and file servers. Each of these servers provide resources to **client** devices, such as desktop computers, laptops, tablets, and smartphones.

VIRTUAL MACHINES (VVVImp Q)

- The fundamental idea behind virtual machine is to abstract the hardware of a single computer into several different execution environment, thereby creating the illusion that each separate execution environment is running its own private computer. By using CPU scheduling and virtual memory techniques.
- The concept of a virtual machine is to provide an interface that looks like independent **hardware,** to multiple different Operating Systems running simultaneously on the same physical **hardware**. Each OS believes that it has access to and control over its own **CPU**, RAM, I/O devices, hard drives, etc.
- Each process is provided with a virtual copy of the underlying computer.
- There are several reasons for creating a virtual machine, all of which are fundamentally related to being able to share the same hardware yet run several different execution environment concurrently.

Virtual machine representation

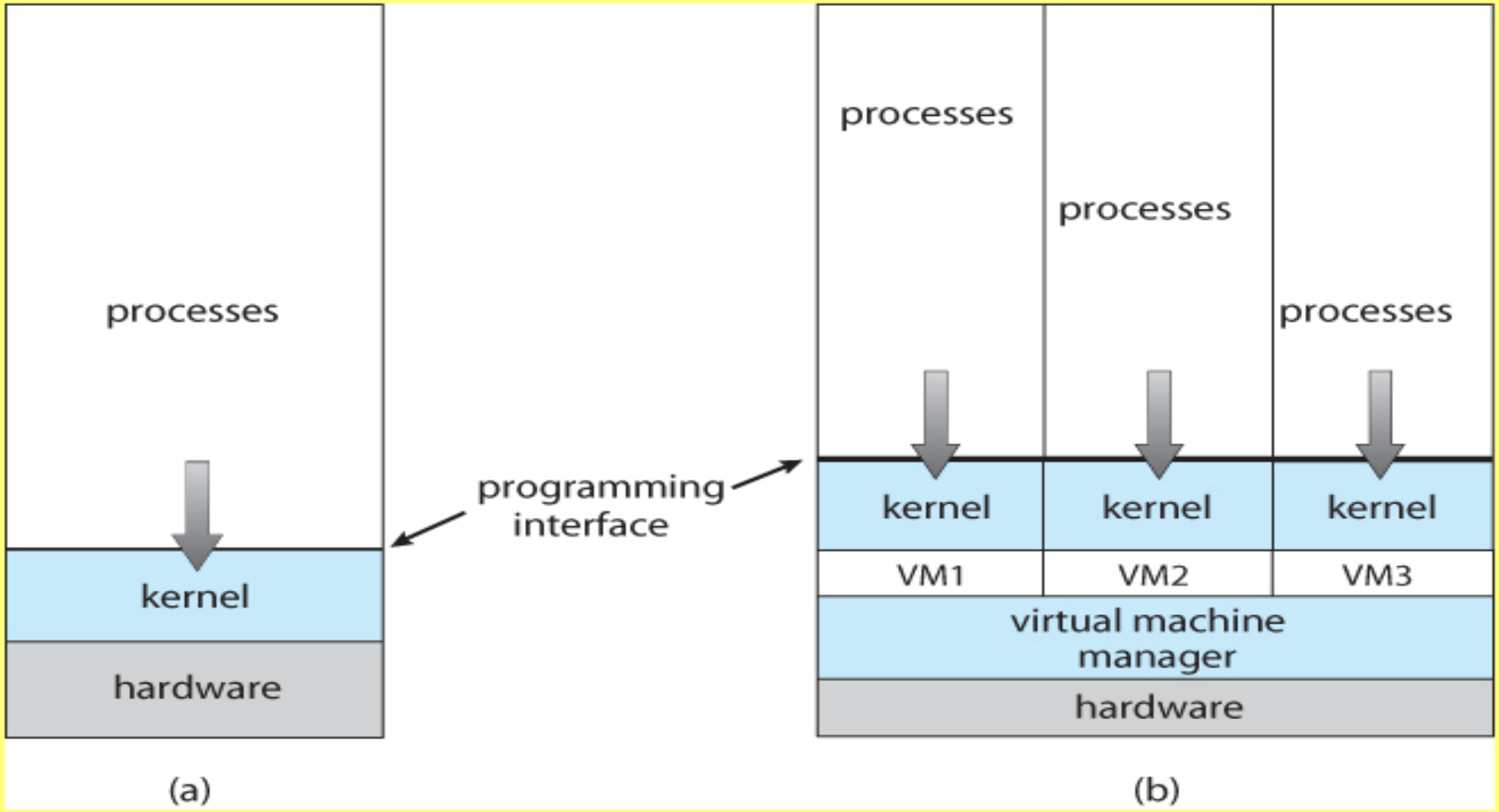


Figure 16.1 - System models. (a) Nonvirtual machine. (b)Virtual machine.

Benefits

- There are two primary advantages:
- First, by completely protecting system resources, the virtual machine provides a robust level of security.
- Second, the virtual machine allows system development to be done without disrupting normal system operation.
- Each virtual machine is completely isolated from all other virtual machines, so we have no security problems as the various system resources are completely protected.

The Java Virtual Machine

- Java was designed from the beginning to be platform independent, by running Java only on a Java Virtual Machine, JVM, of which different implementations have been developed for numerous different underlying HW platforms.
- Java source code is compiled into Java byte code in .class files. Java byte code is binary instructions that will run on the JVM.
- The JVM implements memory management and garbage collection.
- Java byte code may be interpreted as it runs, or compiled to native system binary code using just-in-time (JIT) compilation. Under this scheme, the first time that a piece of Java byte code is encountered, it is compiled to the appropriate native machine binary code by the Java interpreter. This native binary code is then cached, so that the next time that piece of code is encountered it can be used directly.
- Some hardware chips have been developed to run Java byte code directly, which is an interesting application of a real machine being developed to emulate the services of a virtual one!

Implementation