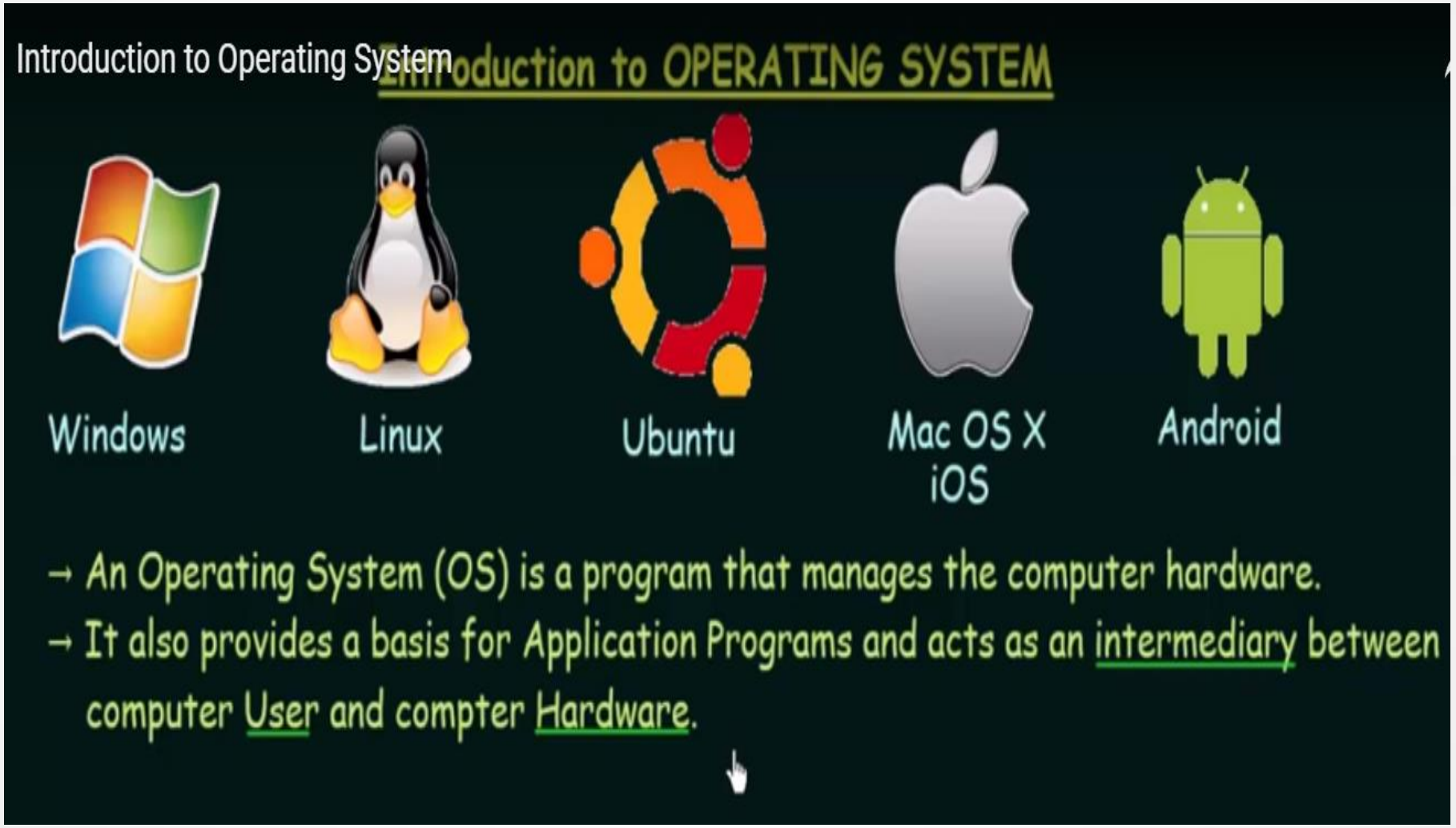
Operating System

Operating System's



Definition

"An operating system acts as an intermediary between the user of a computer and the computer hardware"

Or

"Operating system provides an environment in which a user can execute programs in a convenient and efficient manner"

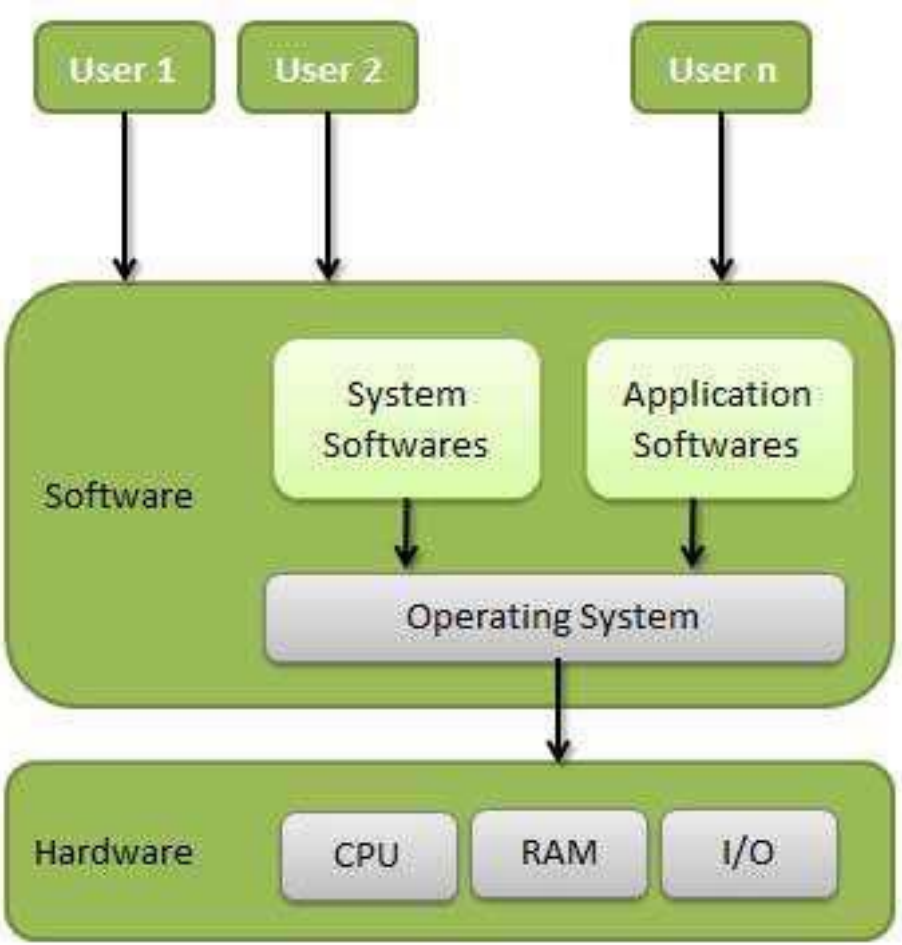
Definition

"An operating system is software that manages the computer hard ware"

Or

"Operating Systems are those programs that interface the machine with the application programs. The main function of these systems is to dynamically allocate the shared system resources to the executing programs."

Basic structure of OS



Objectives

a) **Convenience**

----An OS makes a computer more convenient to use

b) **Efficiency**

-----An OS makes a computer system resources to be used in an efficient manner.

c) **Ability to evolve**

A Major OS will evolve through over time for a number of reasons like:

--**Fixes**

--**New Services**

--**New types of Hardware**

--**Hardware Upgrades**

Convenience:The OS as a User/Computer Interface

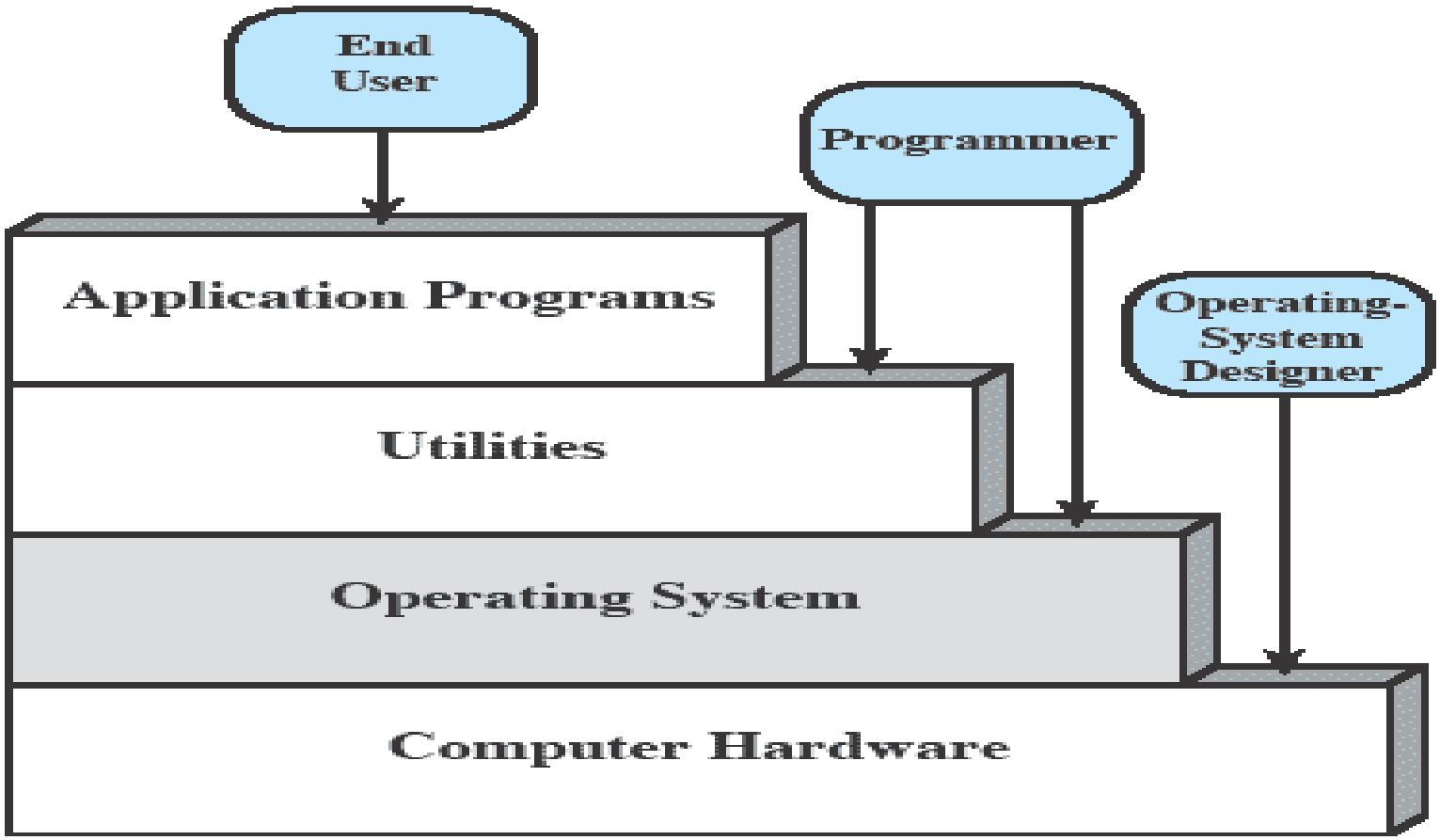


Figure 2.1 Layers and Views of a Computer System

Efficiency:The Operating System As a Resource Manager

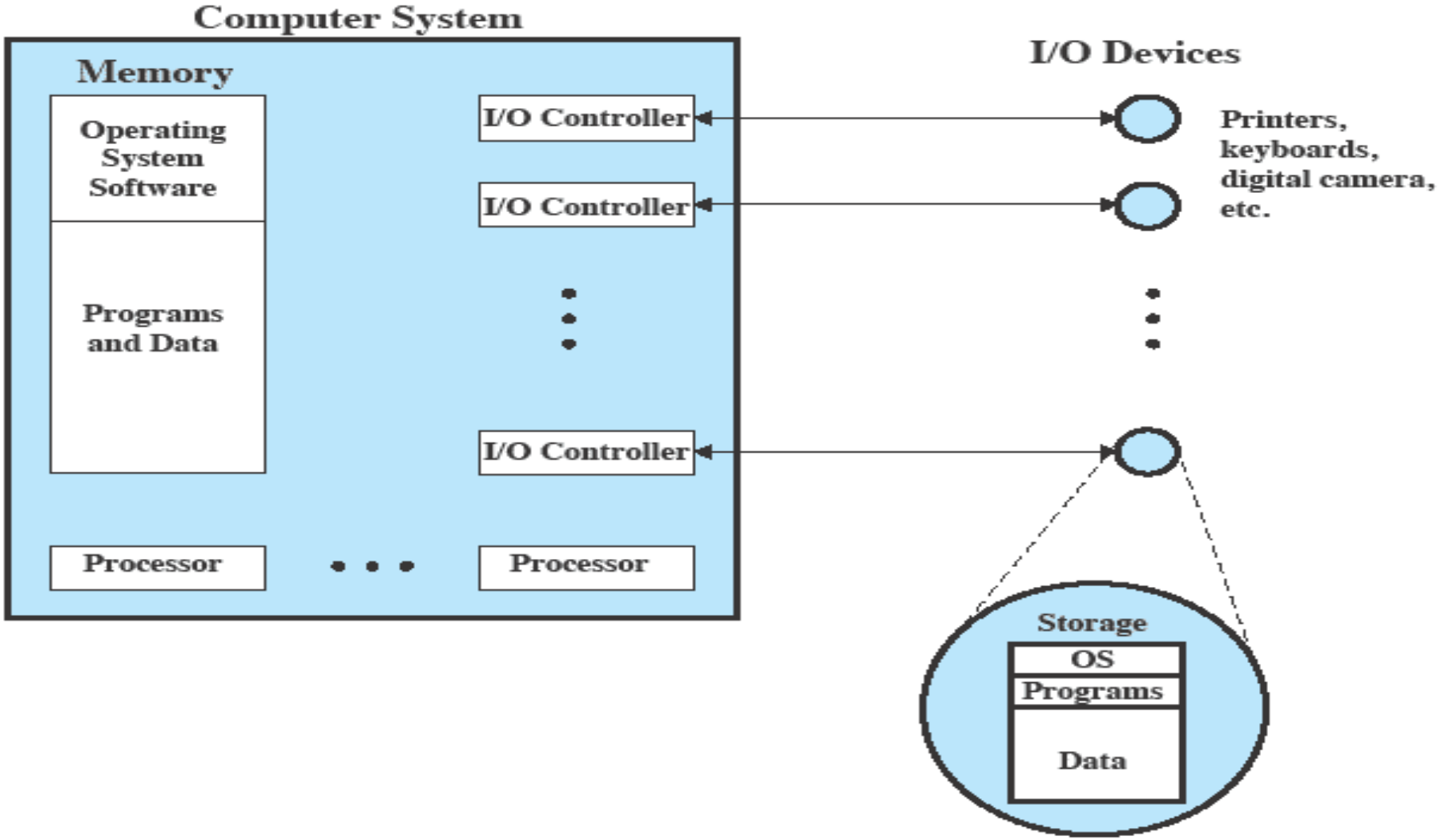


Figure 2.2 The Operating System as Resource Manager

Operating System Services

Operating System Services

An Operating System provides services to both the users and to the programs.

✓ It provides programs, an environment to execute.
✓ It provides users, services to execute the programs in a convenient manner.
Following are few common services provided by operating systems.

- ✓Program development
- ✓Program execution
- ✓ I/O operations
- ✓ File System manipulation
- ✓ Communication
- ✓ Error Detection
- ✓ Resource Allocation
- ✓ Protection

Operating-System Functions

Operating-System Functions

Following are some of important functions of an operating System.

- ✓ Memory Management
- ✓ Processor Management
- ✓ Device Management
- ✓ File Management
- ✓ Security Management

Operating-System Functions

Memory Management

Operating System does the following activities for memory management.

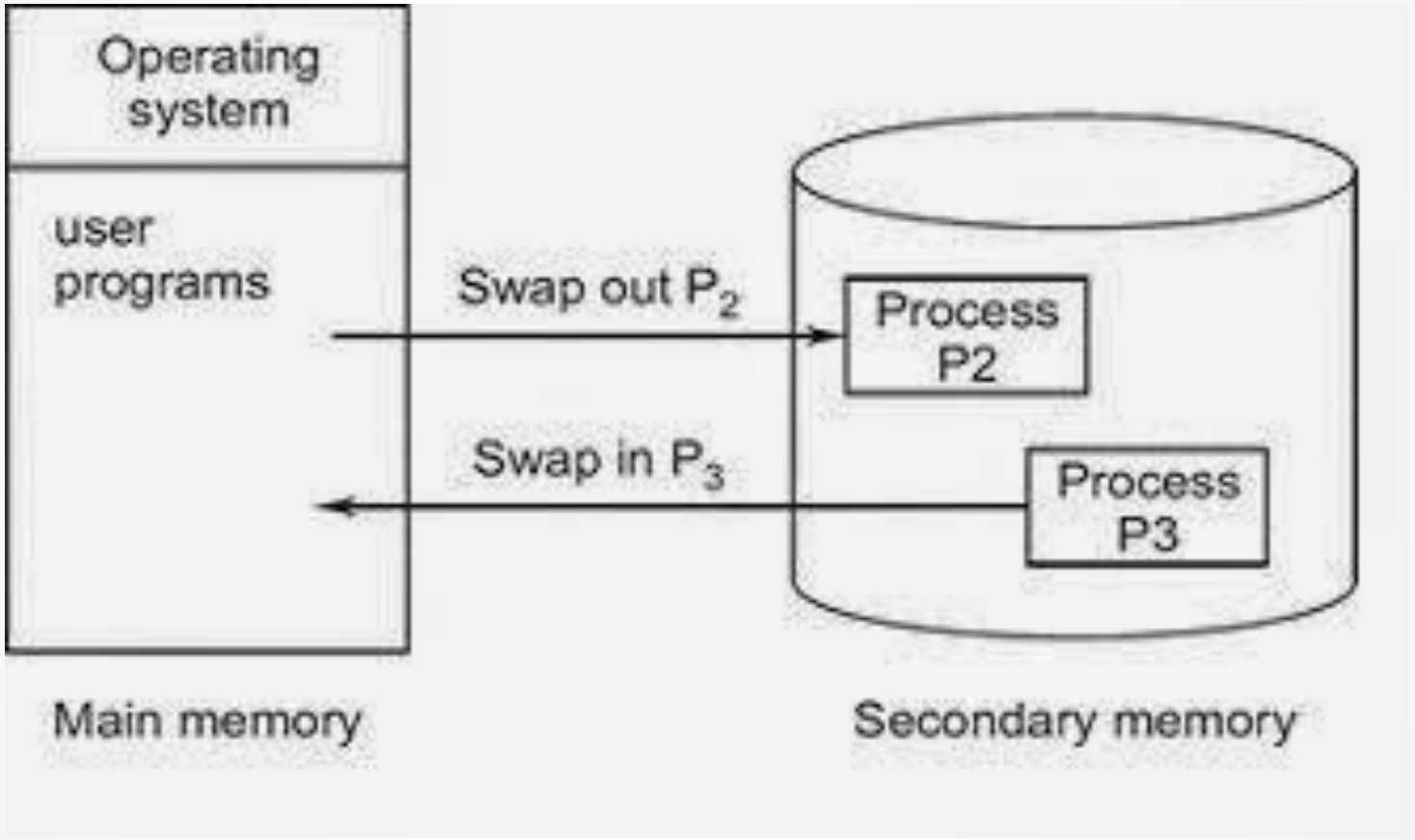
✓ **Keeps tracks of primary memory** i.e. what part of it are in use by whom, what part are not in use.

✓ In multiprogramming, OS decides which process will get memory when and how much.

✓ **Allocates the memory** when the process requests it to do so.

✓ De-allocates the memory when the process no longer needs it or has been terminated.

Operating-System Functions



Operating-System Functions

Processor Management

Operating System does the following activities for processor management.

✓ Keeps tracks of processor and status of process. Program responsible for this task is known as traffic controller.

✓ Allocates the processor (CPU) to a process.

✓ De-allocates processor when processor is no longer required.

Operating-System Functions

Device Management

OS manages device communication via their respective drivers.

Operating System does the following activities for device management.

✓ Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.

✓ Decides which process gets the device when and for how much time.

✓ Allocates the device in the efficient way.

✓ De-allocates devices.

Operating-System Functions

File Management

A file system is normally organized into directories for easy navigation and usage.

Operating System does the following activities for file management.

✓ Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.

✓ Decides who gets the resources.

✓ Allocates the resources.

✓ De-allocates the resources.

Operating-System Functions

Security management

By means of password and similar other techniques, preventing unauthorized access to programs and data.

By security management OS manages many tasks such as:-

- Alert messages
- Dialogue boxes
- Firewall
- Passwords

Types of Operating Systems

Types of Operating Systems

1) Batch operating system

2) Time-sharing operating systems

3) Distributed operating System

4) Network operating System

5) Real Time operating System

Types of Operating Systems

Batch operating system

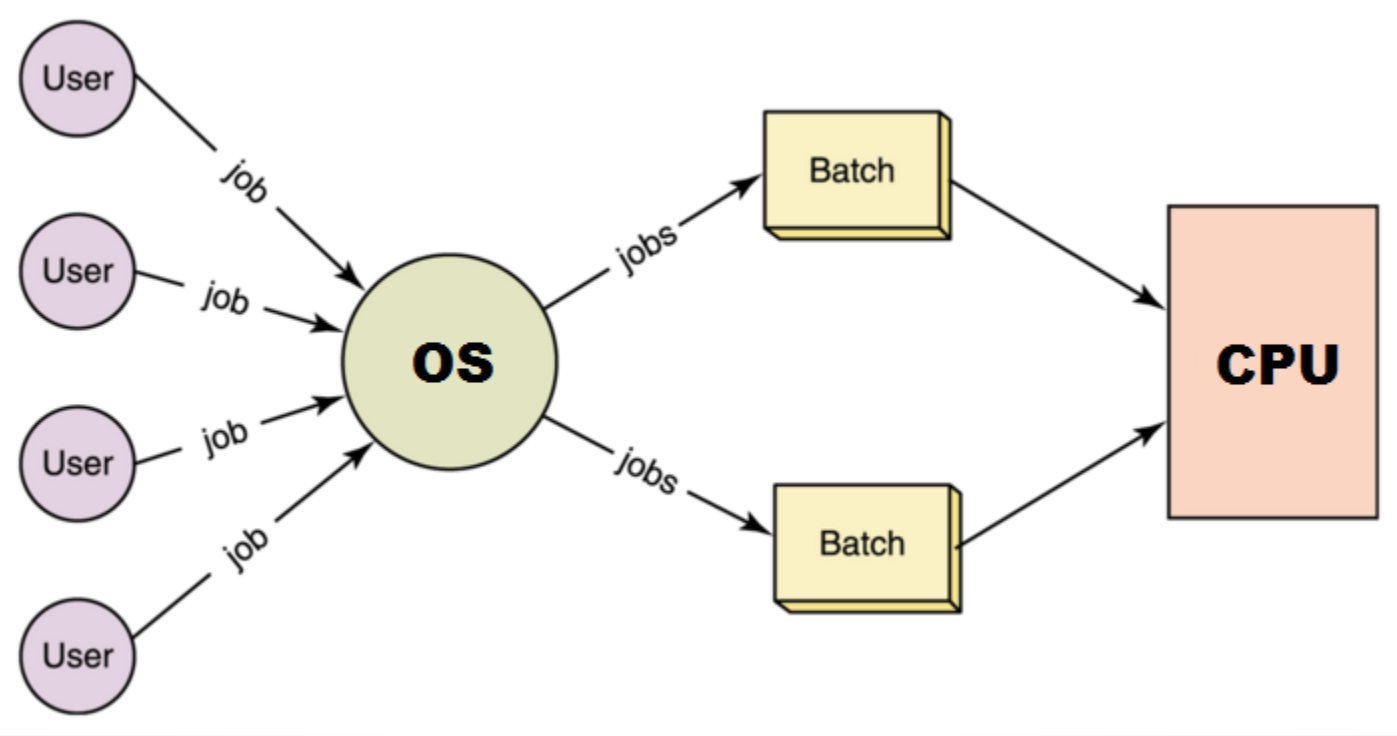
The users of batch operating system do not interact with the computer directly.

Each user prepares his job on an off-line device like punch cards and submits it to the computer operator.

The **problems** with Batch Systems are following.

- ✓ Lack of interaction between the user and job.
- ✓ CPU is often idle, because the speeds of the mechanical I/O devices are slower than CPU.
- ✓ Difficult to provide the desired priority

Batch operating system



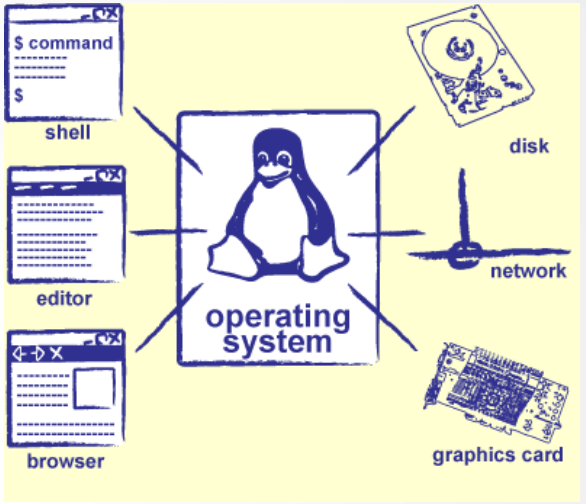
Types of Operating Systems

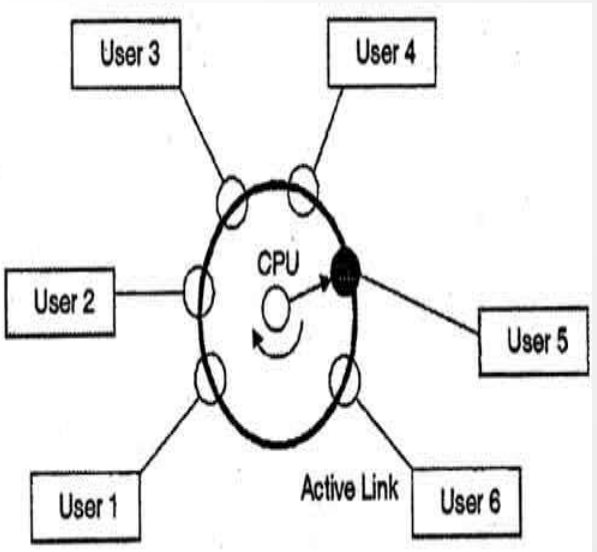
Time-sharing operating systems

✓Time sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time.

✓Time-sharing or multitasking is a logical extension of multiprogramming.

✓Processor's time which **is shared among multiple users** simultaneously is termed as time-sharing.





Types of Operating Systems

Advantages of Timesharing operating systems are following

- ✓ Provide advantage of quick response.
- ✓ Avoids duplication of software.
- ✓ Reduces CPU idle time.

Disadvantages of Timesharing operating systems are following.

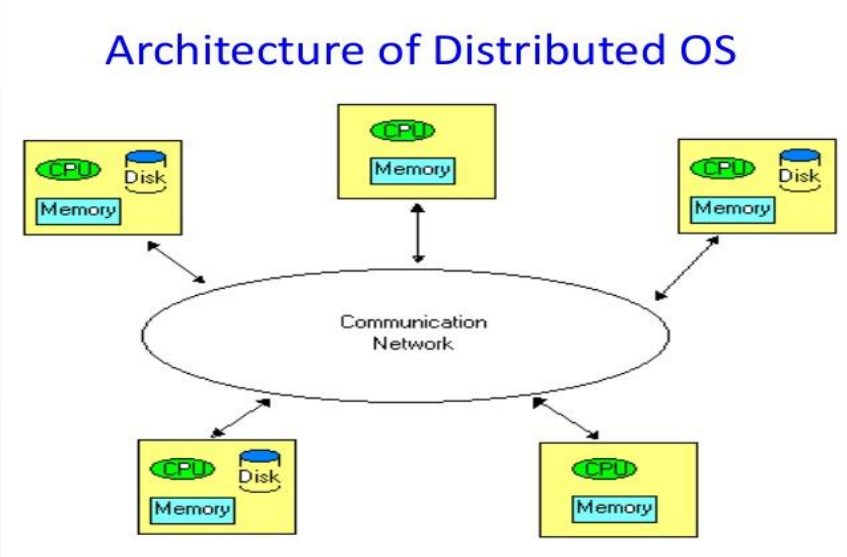
- ✓ Problem of reliability.
- ✓ Question of security and integrity of user programs and data.
- ✓ Problem of data communication.

Types of Operating Systems

Distributed operating System

✓Distributed systems use multiple central processors to serve multiple real time application and multiple users.

✓Data processing jobs are distributed among the processors accordingly to which one can perform each job most efficiently.



Types of Operating Systems

The advantages of distributed systems are following.

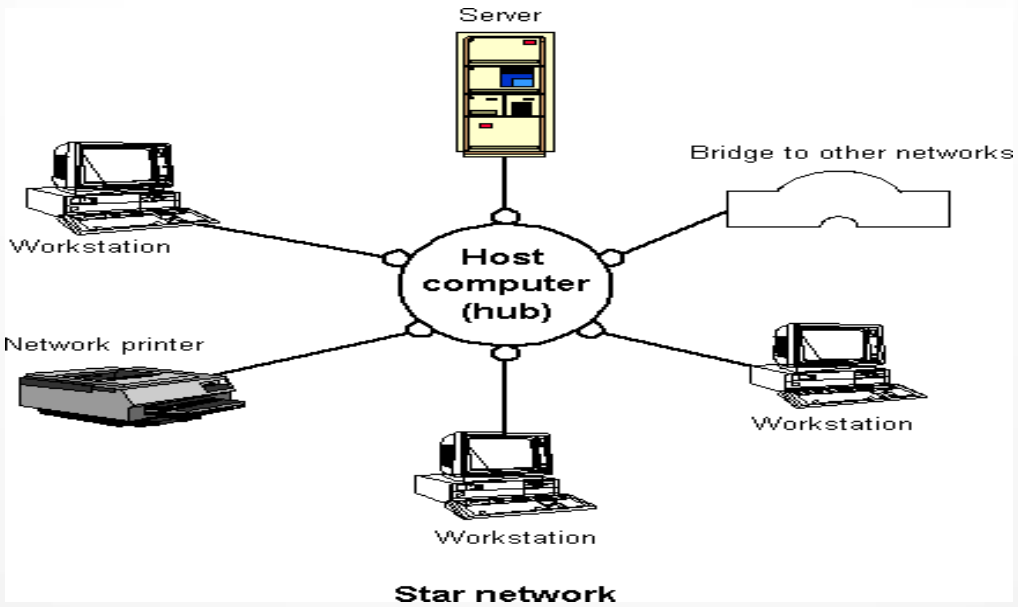
- ✓With resource sharing facility user at one site may be able to use the resources available at another.
- ✓Speedup the exchange of data with one another via electronic mail.
- ✓If one site fails in a distributed system, the remaining sites can potentially continue operating.
- ✓Better service to the customers.
- ✓Reduction of the load on the host computer.
- ✓Reduction of delays in data processing.

Types of Operating Systems

Network operating System

A network operating system (NOS) is a computer operating system that is designed primarily to support workstation, personal computer, and, in some instances, older terminal that are **connected on a local area network (LAN).**

Microsoft Windows Server, and Windows NT are examples of a NOS.



Star network

Types of Operating Systems

Real Time operating System (RTOS)

- ✓It responds to inputs immediately(Real-Time).
- ✓Here the task is completed within a specified time delay.
- ✓In real life situations like controlling traffic signal or a nuclear reactor or an aircraft,
- ✓The operating system has to respond quickly.
- ✓ Almost all the modern telecommunication systems make use of RTOS .

- ✓Example RTOS's

FreeRTOS.
Windows CE.etc

Distributed Systems

Distributed Systems

- A distributed system is a collection of physically separate, possibly heterogeneous computer systems that are networked to provide the users with access to the various resources that the system maintains.

- Distribute the computation among several physical processors.

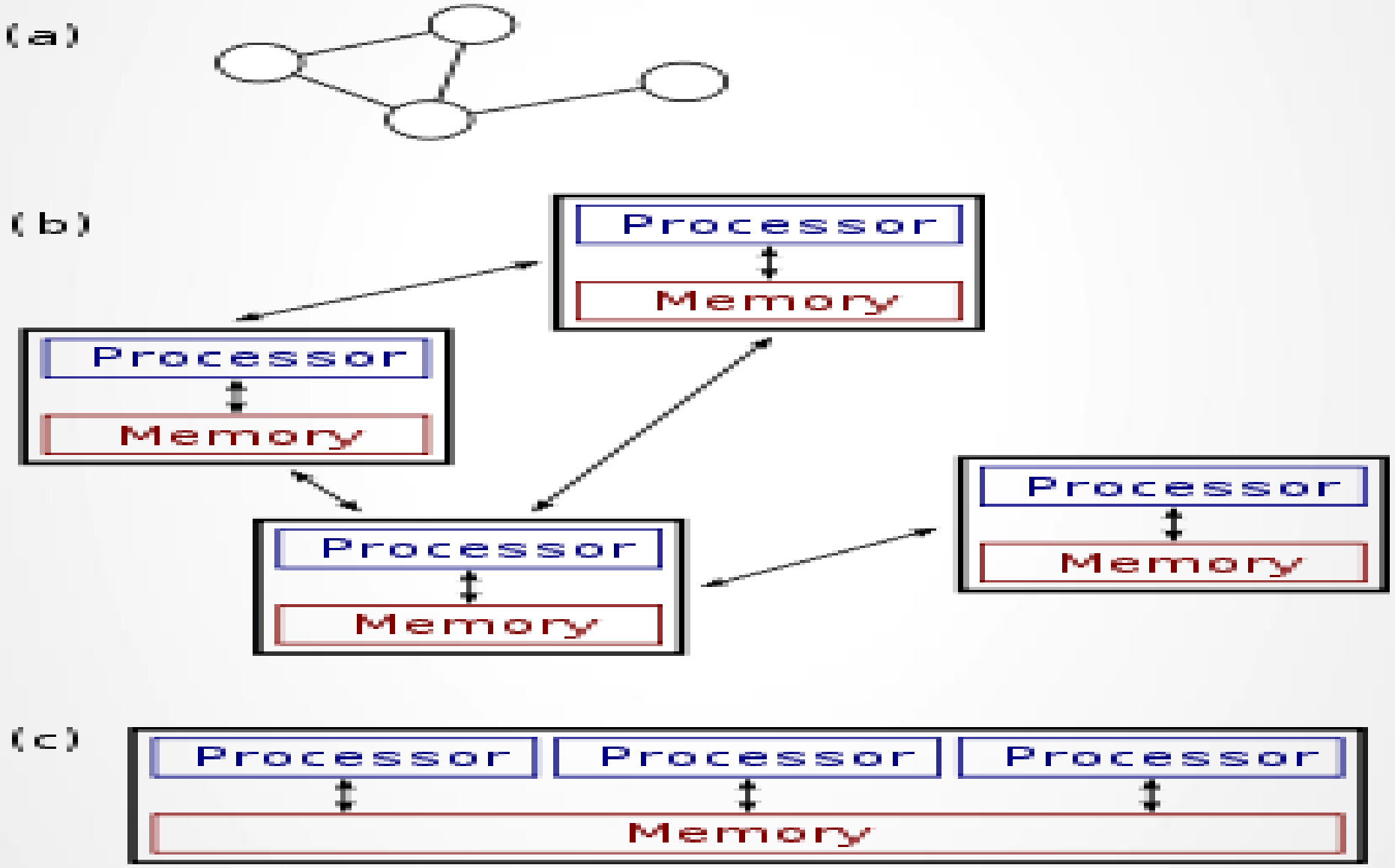
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.

- Advantages of distributed systems.
  - Resources Sharing
  - Computation speed up – load sharing
  - Reliability
  - Communications

Distributed Systems

- A network is a communication path between two or more systems.
- Distributed systems depend on networking for their functionality.
- Networks are characterized based on the distances between their nodes.
  - A local area network (LAN) connects computers within a room, a floor or a building.
  - A wide area network (WAN) links buildings, cities or countries.
  - A metropolitan area network (MAN) could link buildings within a city.

Distributed Systems



Distributed Systems

Computing Environments:

There are four types of computing environments::

1 Traditional Computing

2 Client-Server Computing

3 Peer-to-Peer Computing

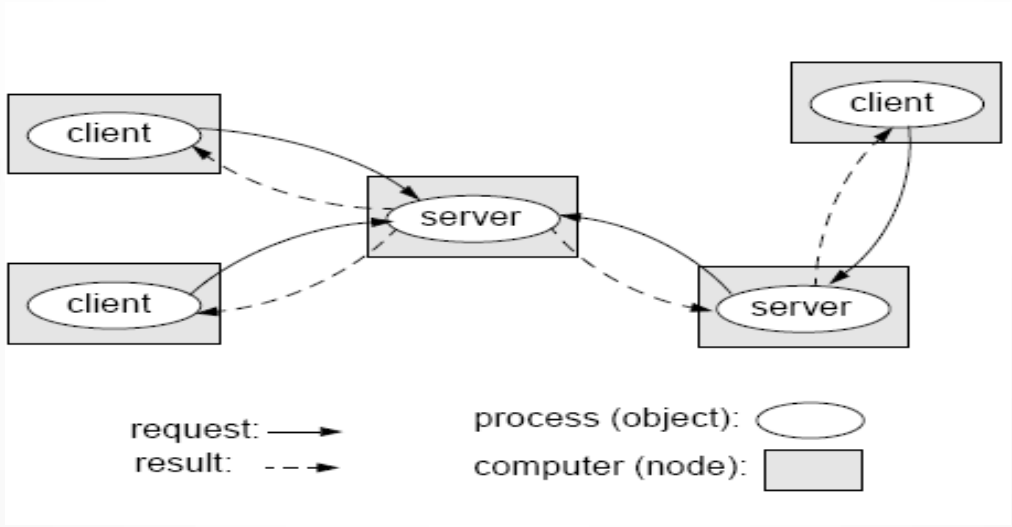
4 Web-Based **Computing**

client-server and peer-to-peer models of distributed systems.

Client-server model::

✓The client-server model firmly distinguishes the roles of the client and server.

✓Under this model, the client requests services that are provided by the server.



A client-server model can be defined as a centralized environment,

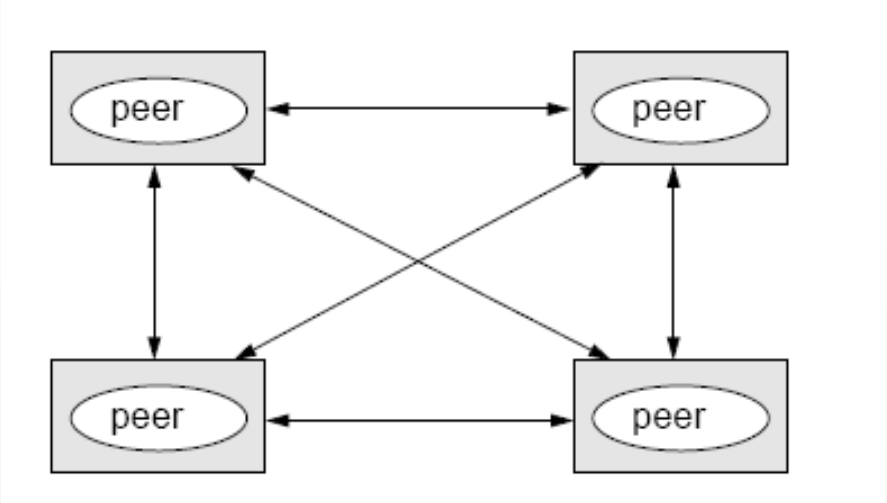
peer-to-peer model (P2P)

Another structure for a distributed system is the peer-to-peer (P2P) system model.

In this model, clients and servers are not distinguished from one another;

- The pattern of communication depends on the particular application.

This is the most general and flexible model of peer to peer (P2P).



Operating-System Operations

• Operating-System Operations

- **Interrupt-driven nature** of modern Operating Systems requires that erroneous processes not be able to disturb anything else.

- Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service

- Other process problems include infinite loop, processes modifying each other or the operating system

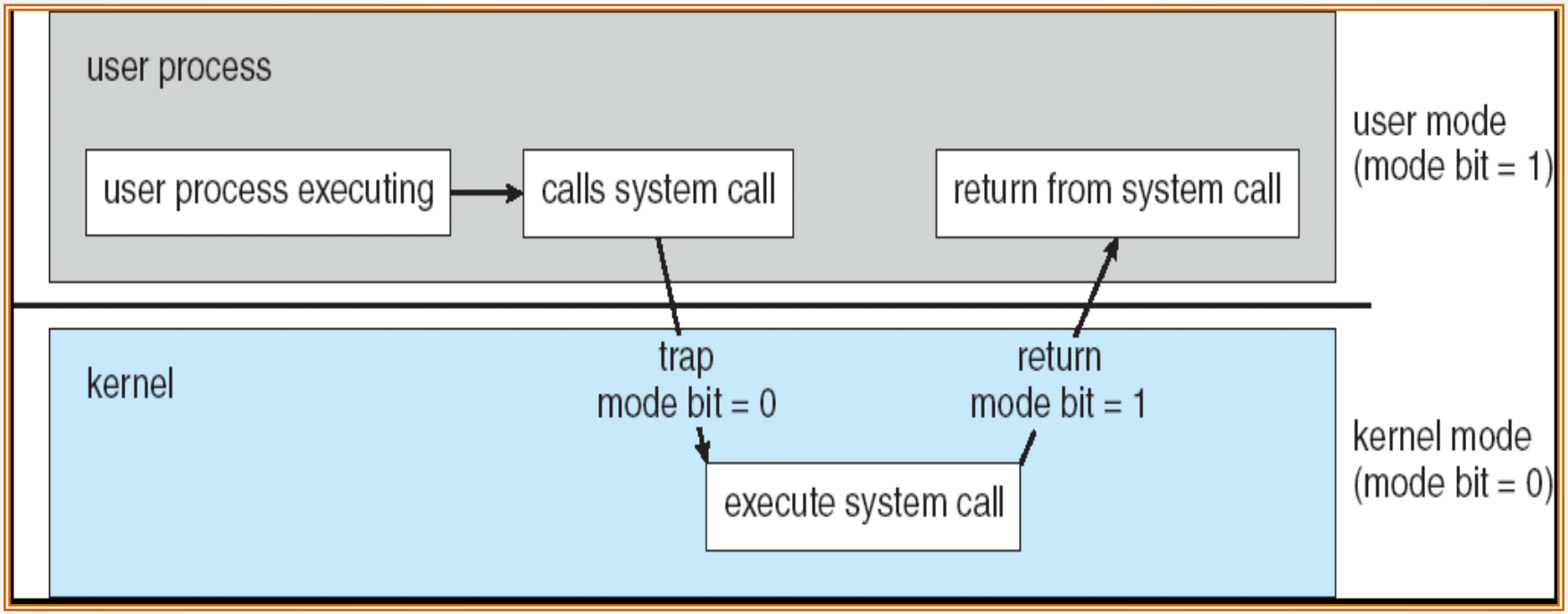
  - Dual-Mode Operation
  - Timer Operation

Operating System Operations

Dual Mode Operations

- There are two modes of executions:
  - User Mode (1)
  - Kernel Mode (supervisor mode, system mode, or privileged mode) (0)
- A ***mode bit*** is added to the hardware of computer to checkout the correct modes.
- Computer is executing on behalf of user a user application – ***user mode.***
- Requests a service from operating system – move to ***kernel mode***.

Transition from User to Kernel Mode



• Operating-System Operations

Timer:

- Timer to prevent infinite loop.
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

System calls

System calls

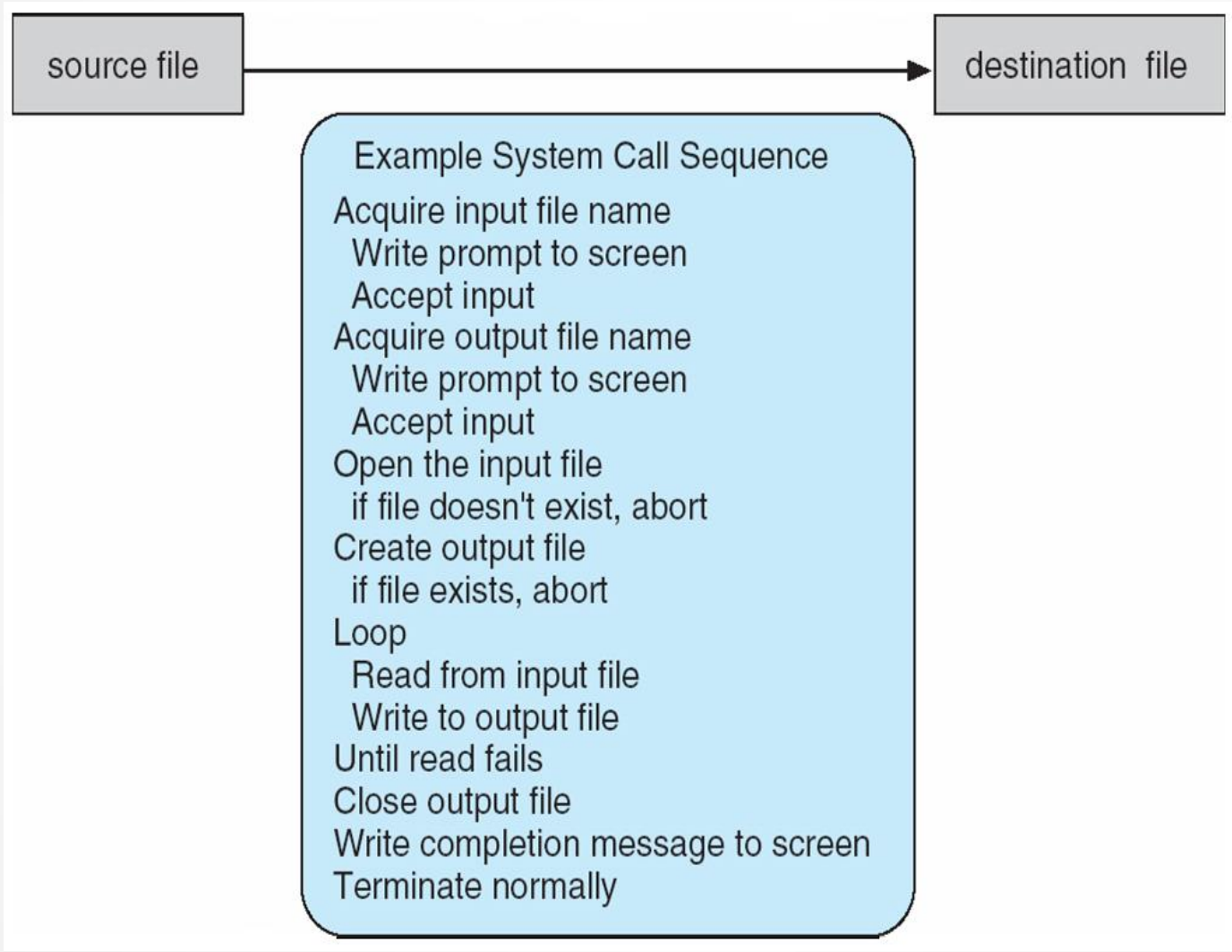
- System calls provide an interface to the services made available by an operating system.
  - --Generally available as **assembly-language instructions**.
- These calls are generally available as routines written in C and C++,
- Mostly accessed by programs via a high-level Application Program Interface (API)
- Three most common APIs are
  - Win32 API for Windows,
  - POSIX API (all versions of UNIX, Linux, and Mac OS X), and
  - Java API for the Java virtual machine (JVM)

System calls

A system call is used to complete a specific task:

- ✓ Reading input file name
- ✓ Opening a file
- ✓ Error message if any
- ✓ Abnormal termination
- ✓ Deletion of a file
- ✓ Prompting a message
- ✓ Reading data for the file
- ✓ Writing data to the file or console
- ✓ Closing the file
- ✓ Normal termination

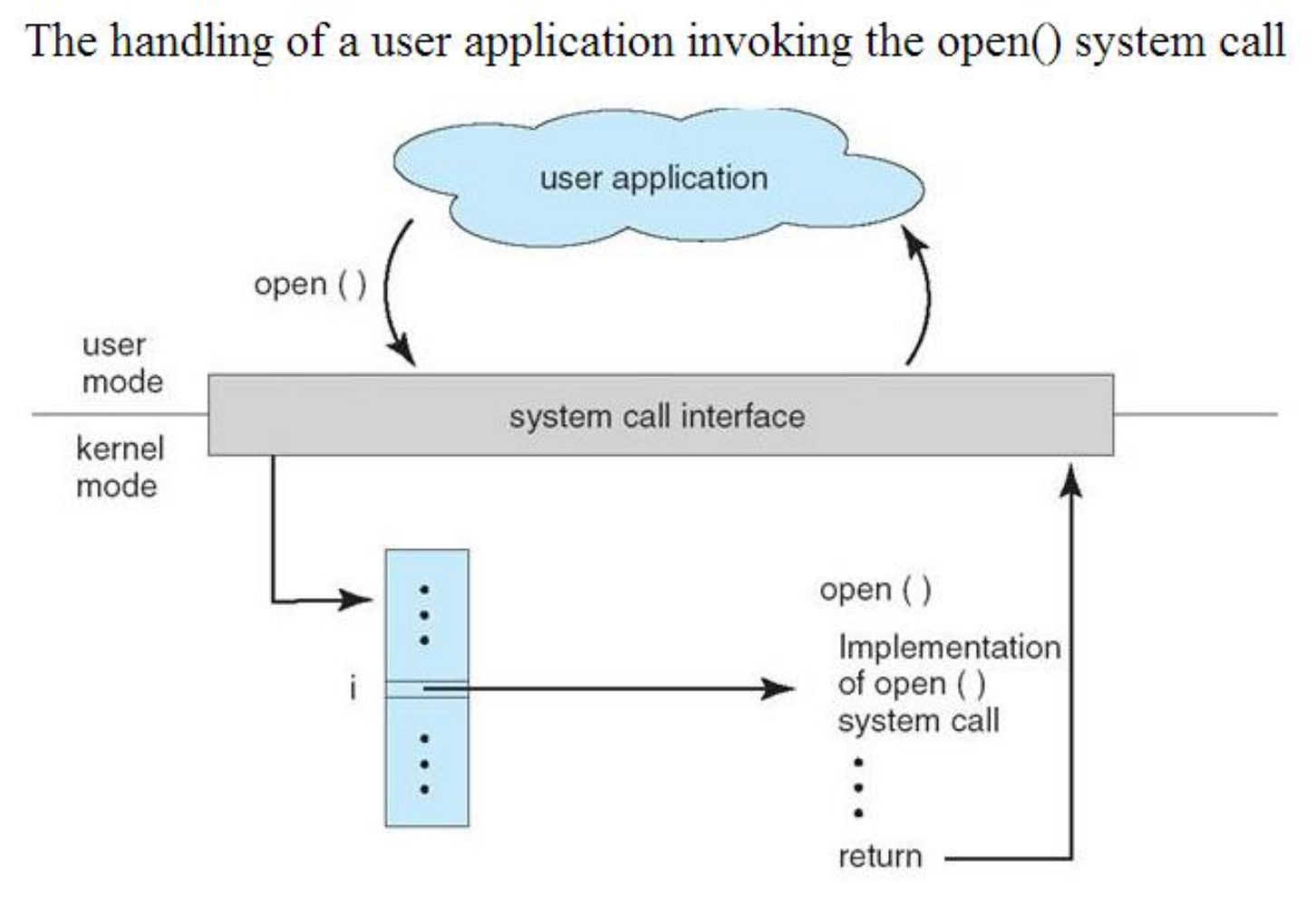
Example of System Calls



System Call Implementation:

- ✓Typically, **a number associated** with each system call
- ✓**System-call interface maintains a table indexed** according to these Numbers
- ✓The **system call interface invokes intended system call** in OS kernel and returns status of the system call and any return values.
- ✓The caller need know nothing about how the system call is implemented.
- ✓Just needs to obey API and understand what OS will do as a result Call.
- ✓Most details of OS interface hidden from programmer by API

API – System Call – OS Relationship



System Call Parameter Passing:

- Three general methods used to pass parameters to the OS

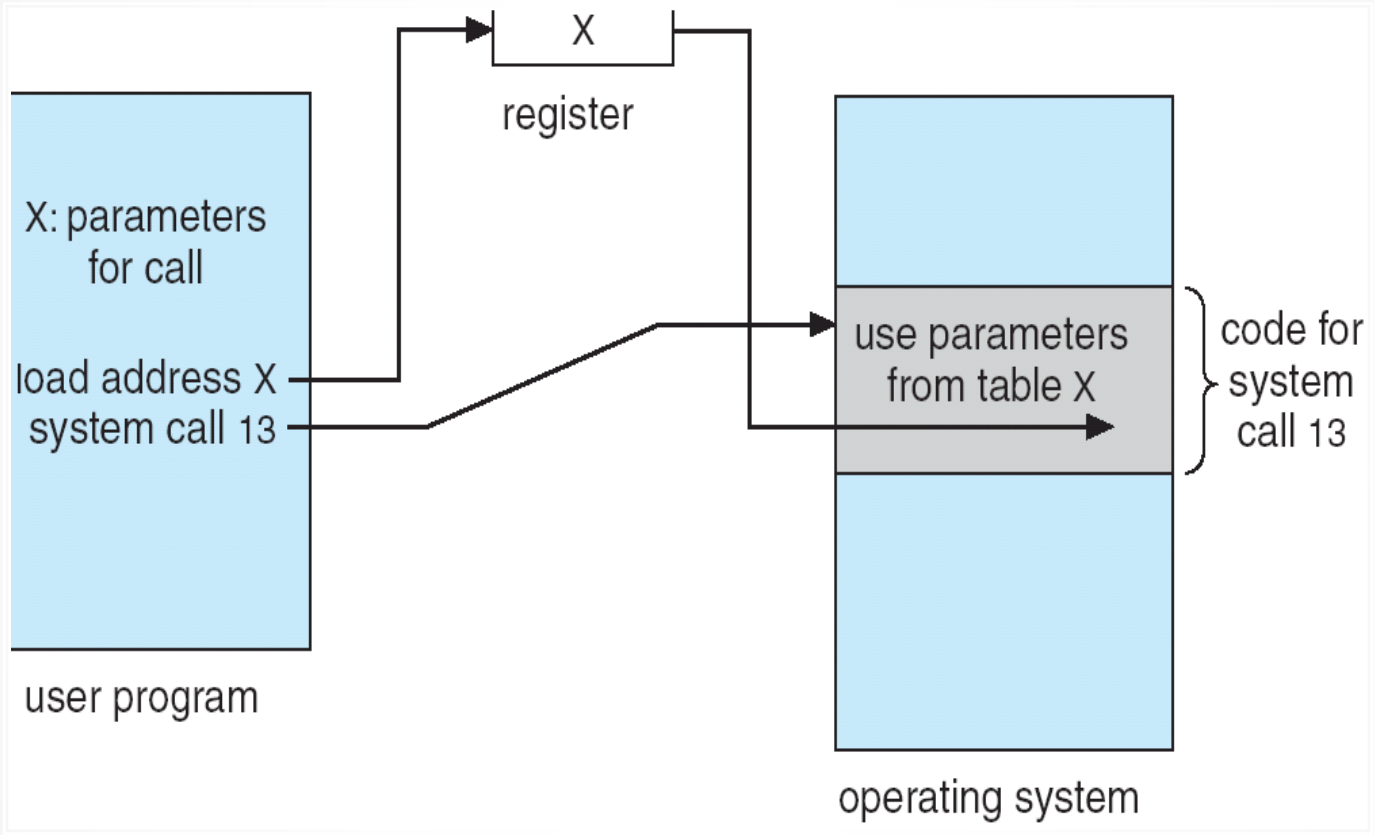
- Simplest: **pass the parameters in *registers***
  - In some cases, may be more parameters than registers

- Parameters stored in a *block*, or table, in memory, and address of block passed as a parameter in a register
  --This approach taken by Linux and Solaris

- Parameters placed, or *pushed*, onto the *stack* by the program and *popped* off the stack by the operating system

- Block and stack methods do not limit the number or length of parameters being passed

Passing of parameters as a table



Types of System Calls

- **Process control**
  - End, Abort
  - Load, Execute
  - Create process, Terminate process
  - Get process attributes, Set process attributes
  - Wait for time
  - Wait event, Signal event
  - Allocate memory, Free memory

- **File management**
  - Create file, Delete file
  - Open, Close
  - Read, Write, Reposition
  - Get file attributes, Set file attributes

Types of System Calls

❖ **Device management**

- ✓ Request device, Release device
- ✓ Read, Write, Reposition
- ✓ Get device attributes, Set device attributes
- ✓ Logically attach devices, Logically detach devices

❖ **Information maintenance**

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

❖ **Communications**

- ✓ Create communication connection, Delete communication connection
- ✓ Send messages, Receive messages
- ✓ Transfer status information
- ✓ Attach remote devices, Detach remote devices

Types of System Calls

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Special Purpose Systems

Special Purpose Systems

Classes of computers whose functions are limited and objective is to deal with limited computation domains-

- **Real Time Embedded Systems:**
- **Multimedia systems:**
- **Handheld systems:**

Special Purpose Systems

Real Time Embedded Systems:

✓**Embedded computers** are the most prevalent form of computers in existence.

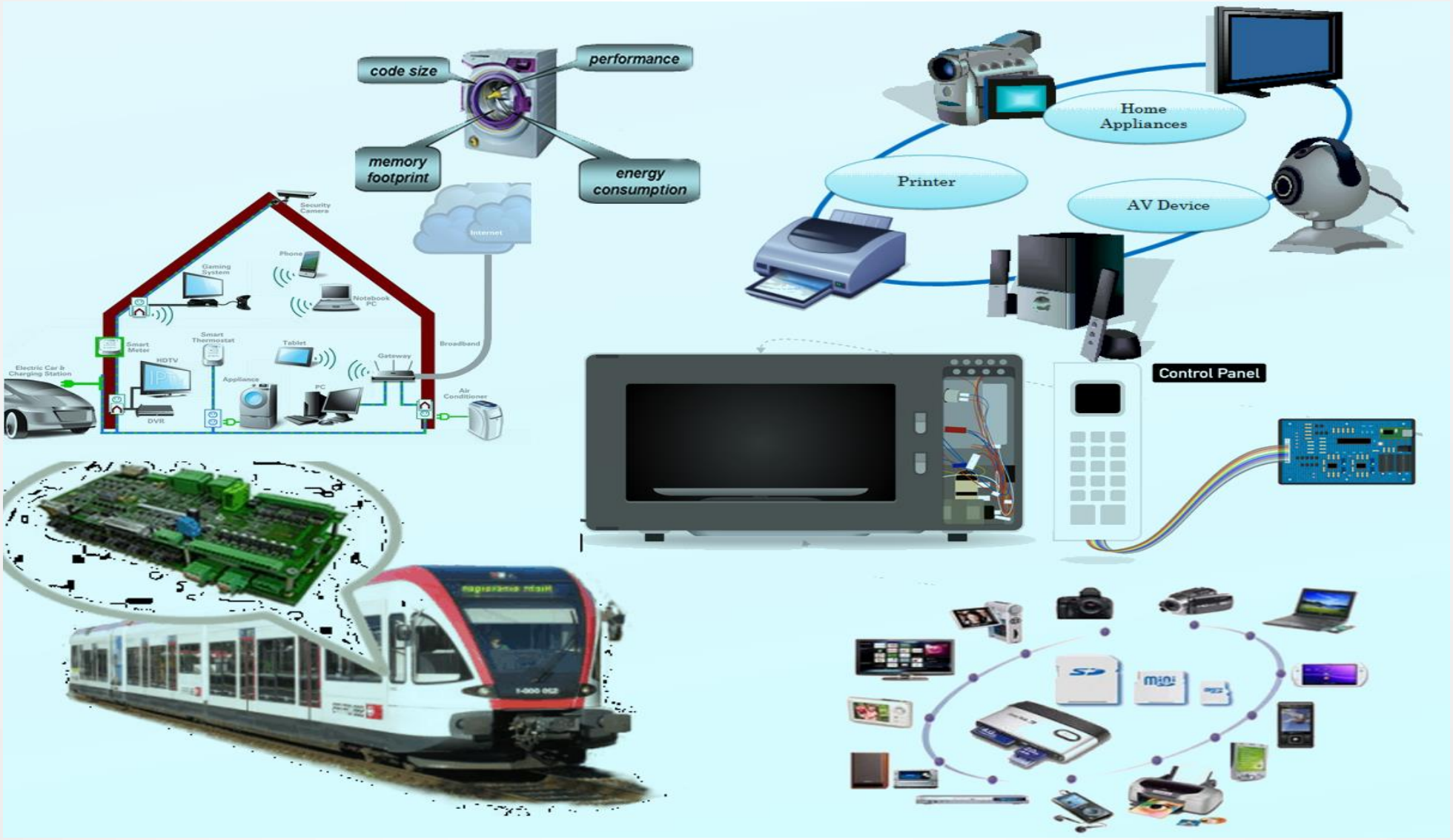
✓Embedded computers are devices found from car engines and manufacturing robots to VCR's and microwave ovens.

✓These have specific tasks to accomplish. Embedded systems almost always run real time operating system.

✓**A real-time system** has well-defined, fixed time constraints.

✓A real-time system functions correct only if it returns the correct result within its time constraints.

Special Purpose Systems



Multimedia systems:

A ***Multimedia System*** is a system capable of processing multimedia data and applications.

✓Multimedia data consist of **audio and video files** as well as **conventional files.**

✓These data differ from conventional data in that multimedia data—such as **frames of video—must be delivered (streamed)** according to certain **time restrictions** (for example, 30 frames per second).

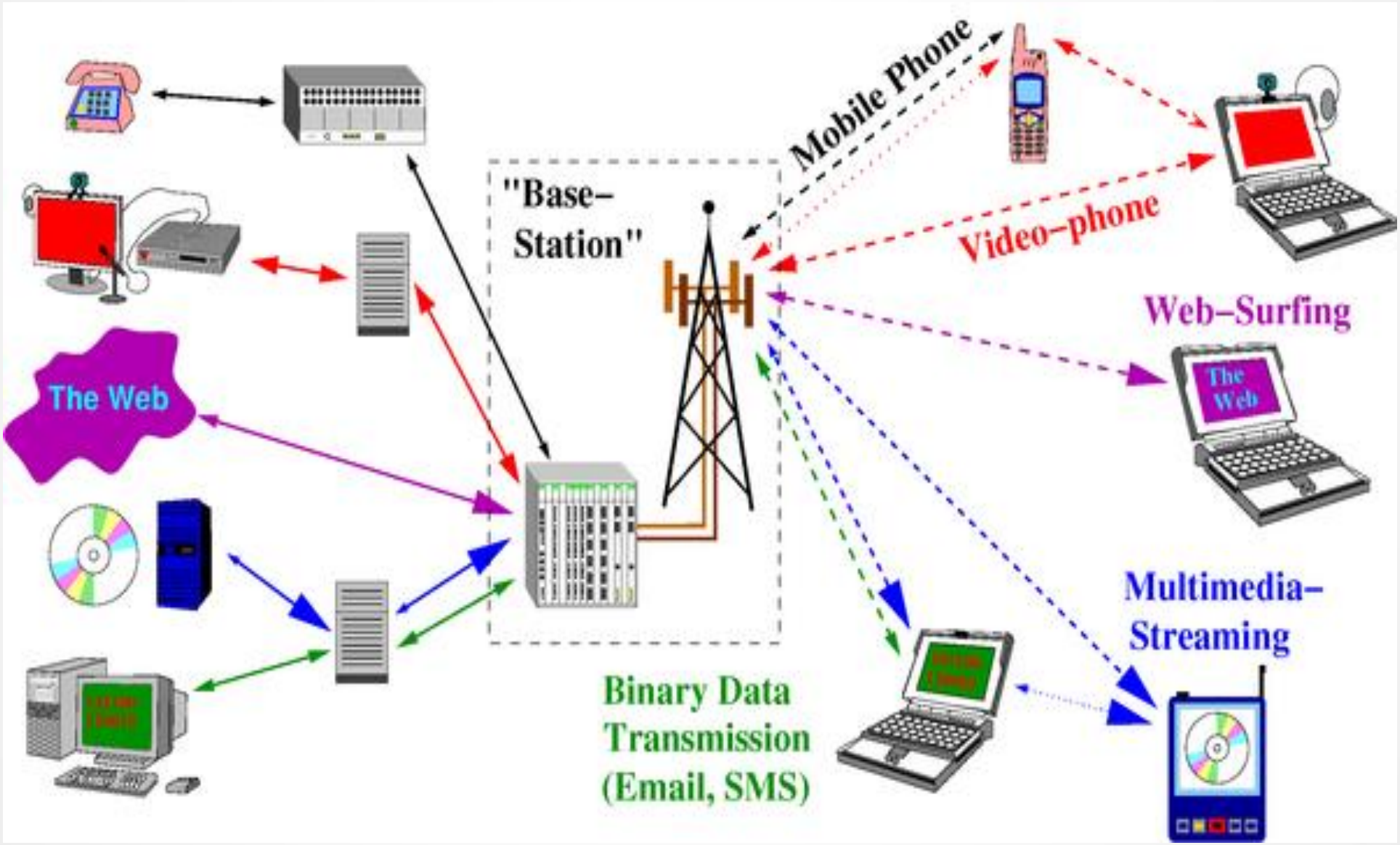
Special Purpose Systems

Multimedia systems:

Multimedia describes a wide range of applications that are in popular use today. These include

- ✓Audio files such as MP3
- ✓DVD movies,
- ✓Video conferencing
- ✓Short video clips of movie previews
- ✓News stories downloaded over the Internet.

Special Purpose Systems



Special Purpose Systems

Handheld systems:

Handheld systems include personal digital assistants (PDAs), such as Palmand Pocket-PCs, and cellular telephones, many of which use special-purpose embedded operating systems.

Why Handheld Systems..?

- **Advantage**
  - Portability
  - Price

- **Disadvantage**
  - Limited Memory
  - Slow processors
  - Small display screen
  - Smaller Keyboards

Operating System Structures

Types of OS Structures

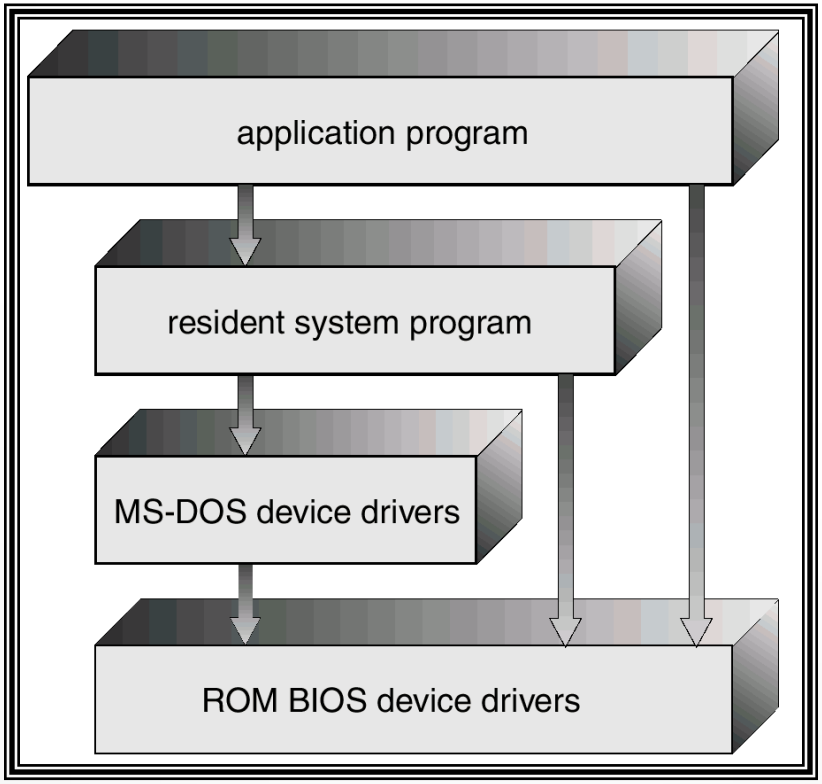
- Simple Structure
- Layered Approach
- Microkernel's
- Modules

Simple Structure

- Early commercial systems do not have well-defined structures.
- Frequently, such operating systems started as small, simple, and limited systems and then grew beyond their original scope.
- **MS-DOS is an example of such a system**.
- It was originally designed and implemented by a few people who had no idea that it would become so popular.
- It was written to provide the most functionality in the least space, so it was not divided into modules carefully.

Simple Structure

- In MS-DOS, the interfaces and levels of functionality are not well separated.
- Such freedom leaves MS-DOS vulnerable to errant (or malicious) programs, causing entire system crashes when user programs fail.



MS-DOS Layer Structure

Simple Structure

- Another example of limited structuring is the original UNIX operating system.

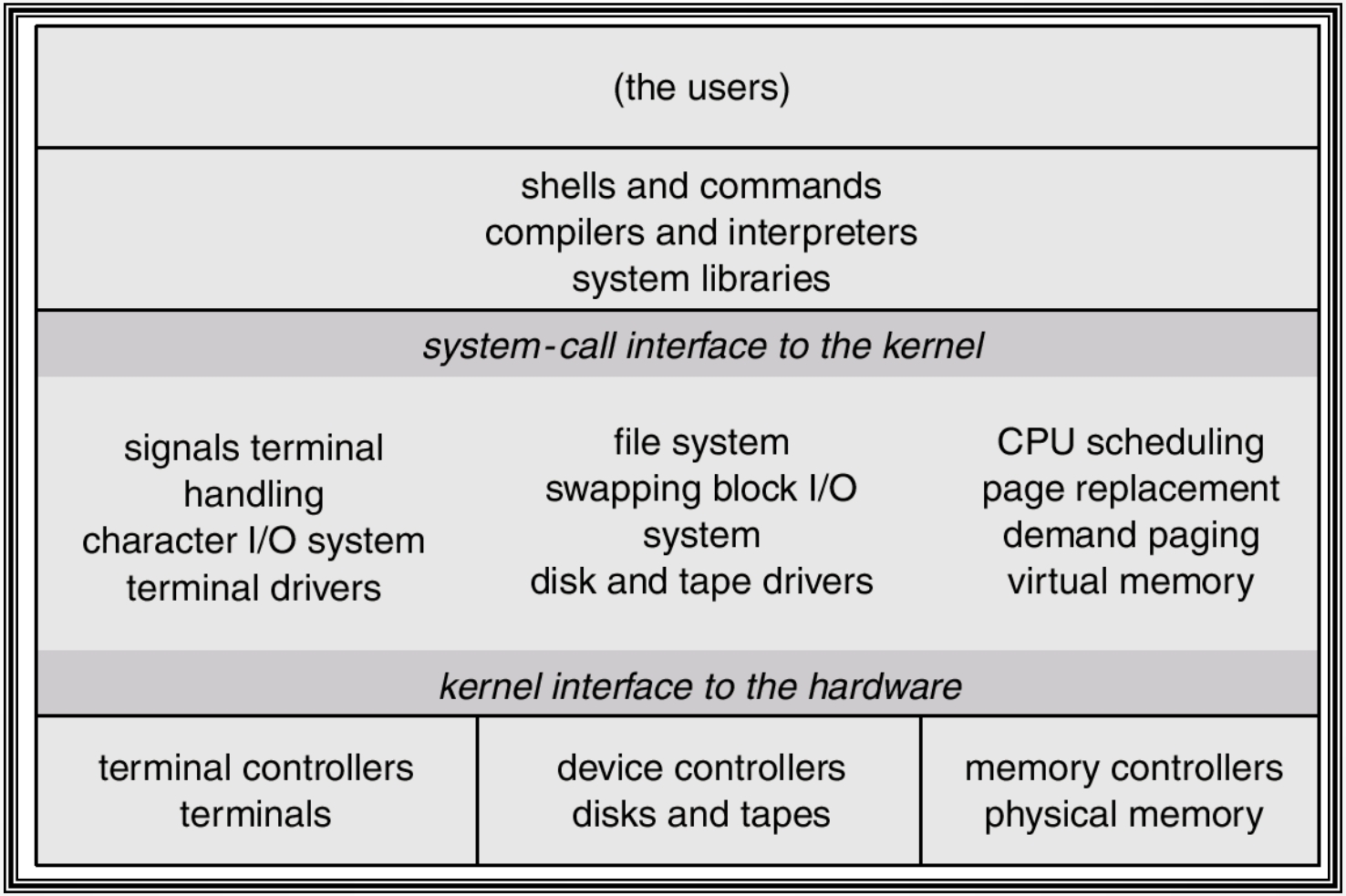
- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring.

- The UNIX OS consists of two separable parts.

Systems programs
The kernel

Consists of everything below the system-call interface and above the physical hardware

Simple Structure



UNIX System Structure

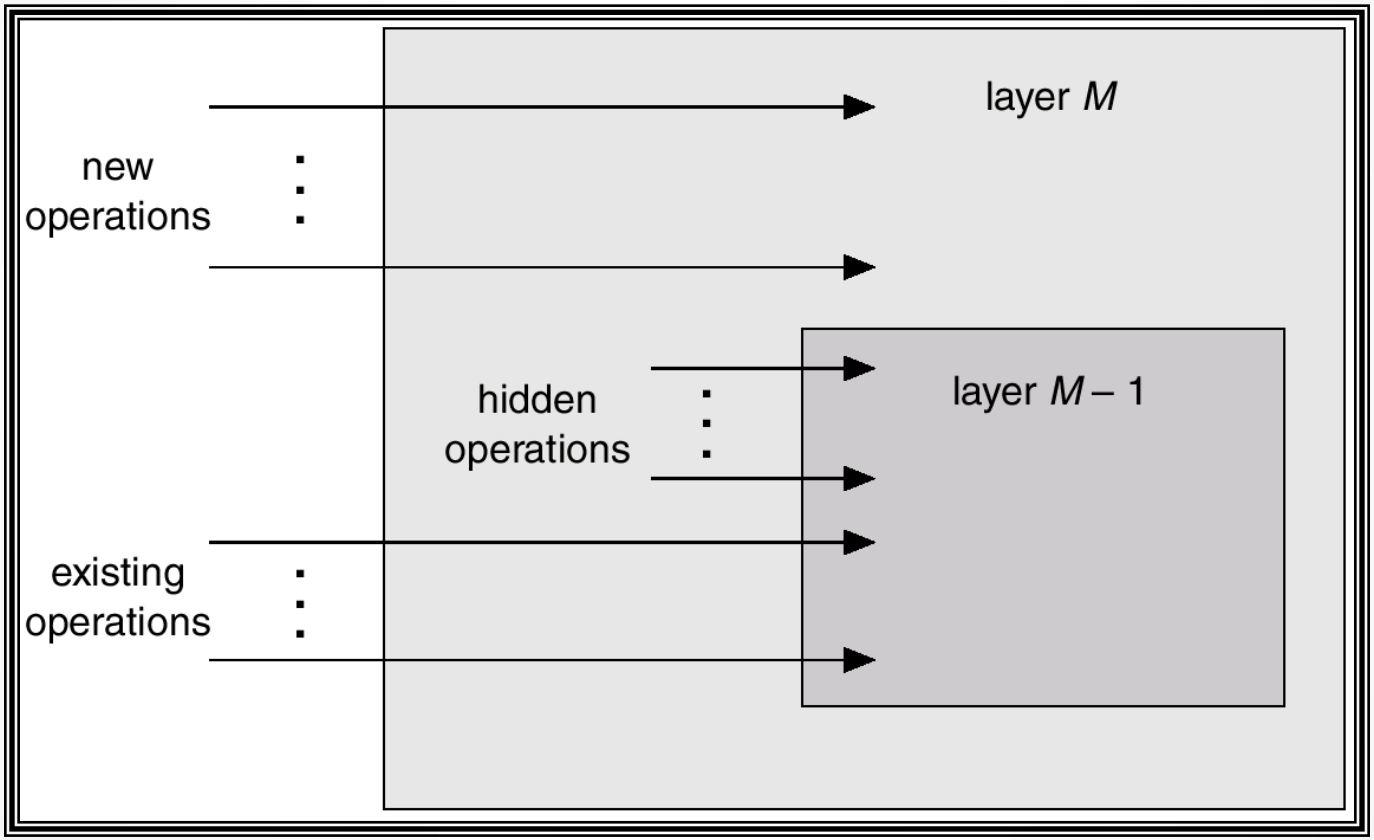
Layered Structure

- A system can be made modular in many ways.

  --One method is the **layered approach**.

- In Layered approach the operating system is broken up into number of layers(**levels**).
  - The bottom layer (layer 0) is the hardware;
  - The highest (layer N) is the user interface.
- A typical operating-system layer—say, layer M—consists of data structures and a set of routines that can be invoked by higher-level layers.
- Layer M, in turn, can invoke operations on lower-level layers.

Layered Structure

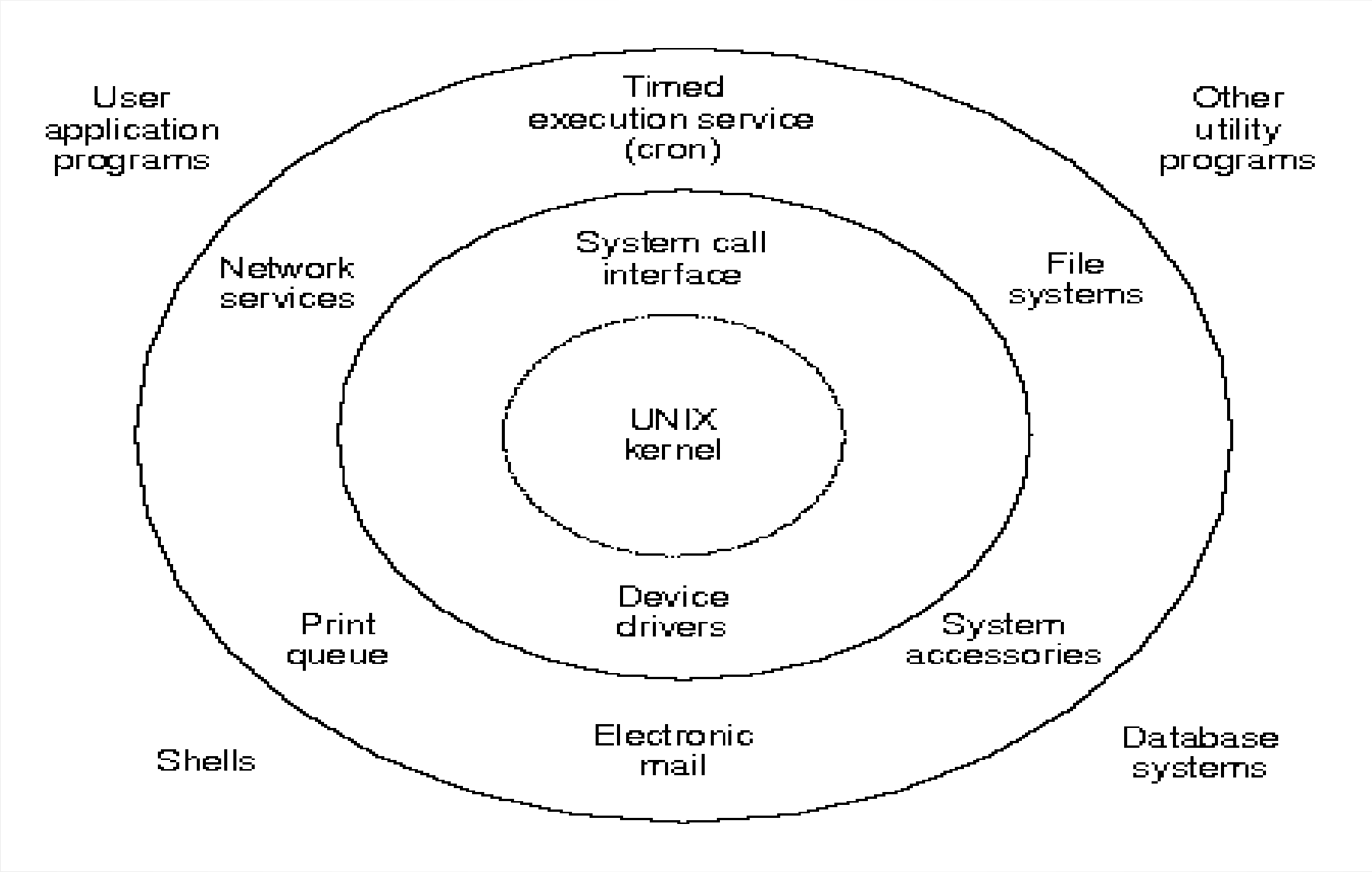


An Operating System Layer

Layered Structure

- The main advantage of the layered approach is simplicity of construction and debugging.
- The layers are selected so that each uses functions (operations) and services of only lower-level layers.
- This approach simplifies debugging and system verification.
- The first layer can be debugged without any concern for the rest of the system
- The major difficulty with the layered approach involves **appropriately defining the various layers**

Layered Approach



Microkernel's

In the mid-1980s, researchers at Mellon University developed an operating system called **Mach** that modularized the kernel using the **microkernel approach**.

This method of structures the operating system by removing all nonessential components from the kernel and implementing them as **system and user-level programs**.

--**The result is a smaller kernel**

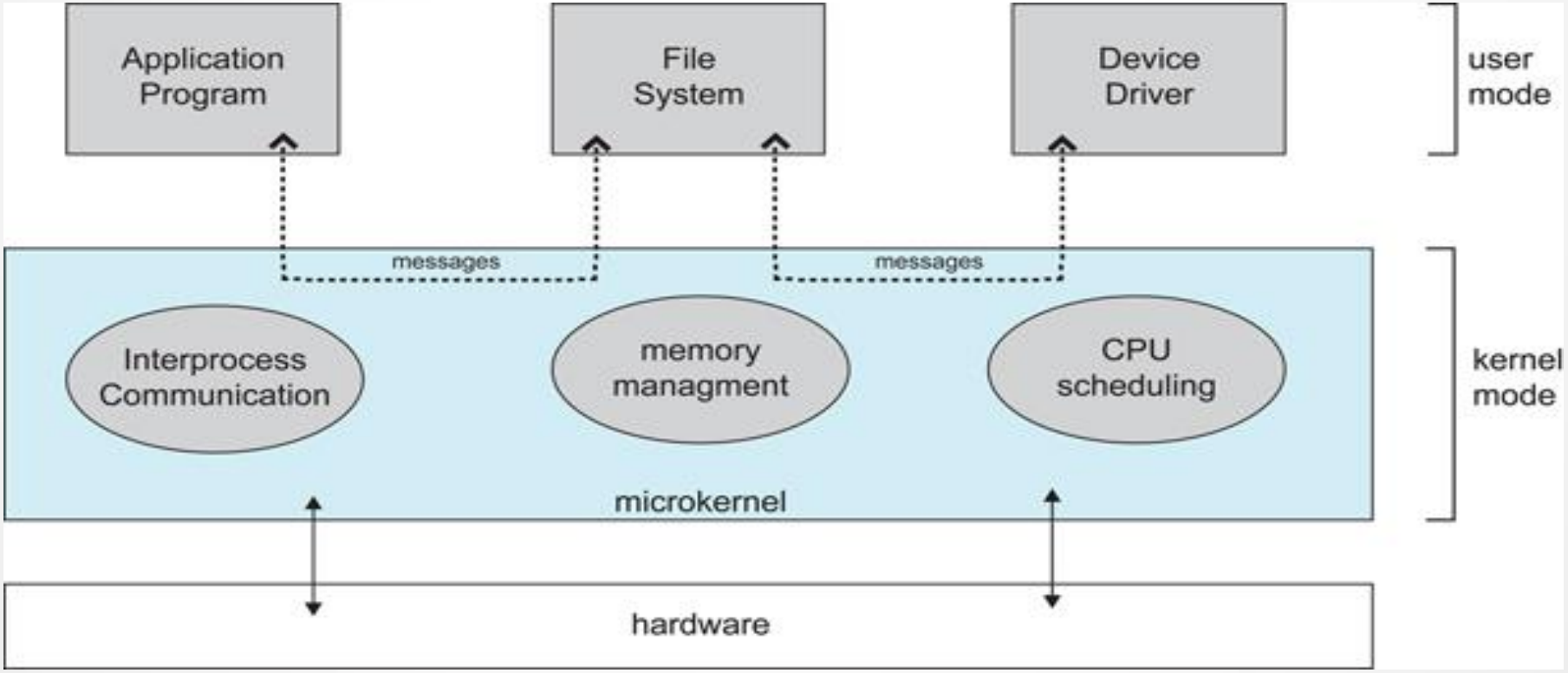
- Generally they provide minimal **process and memory management, and a communications facility.**

- Communication between the client program and the various services that are also running in user space. **Communication is provided by message passing**

Microkernel's

For example,

→if the client program wishes to access a file, it must interact with the file server. The **client program and service never interact directly**. Rather, they communicate **indirectly by exchanging messages with the microkernel**



Microkernel's

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.

Modules

✓Most modern operating systems implement kernel modules.

- Uses object-oriented approach.(best current methodology)

- Here, the kernel has a set of core components and dynamically links in additional services either during boot time or during run time.

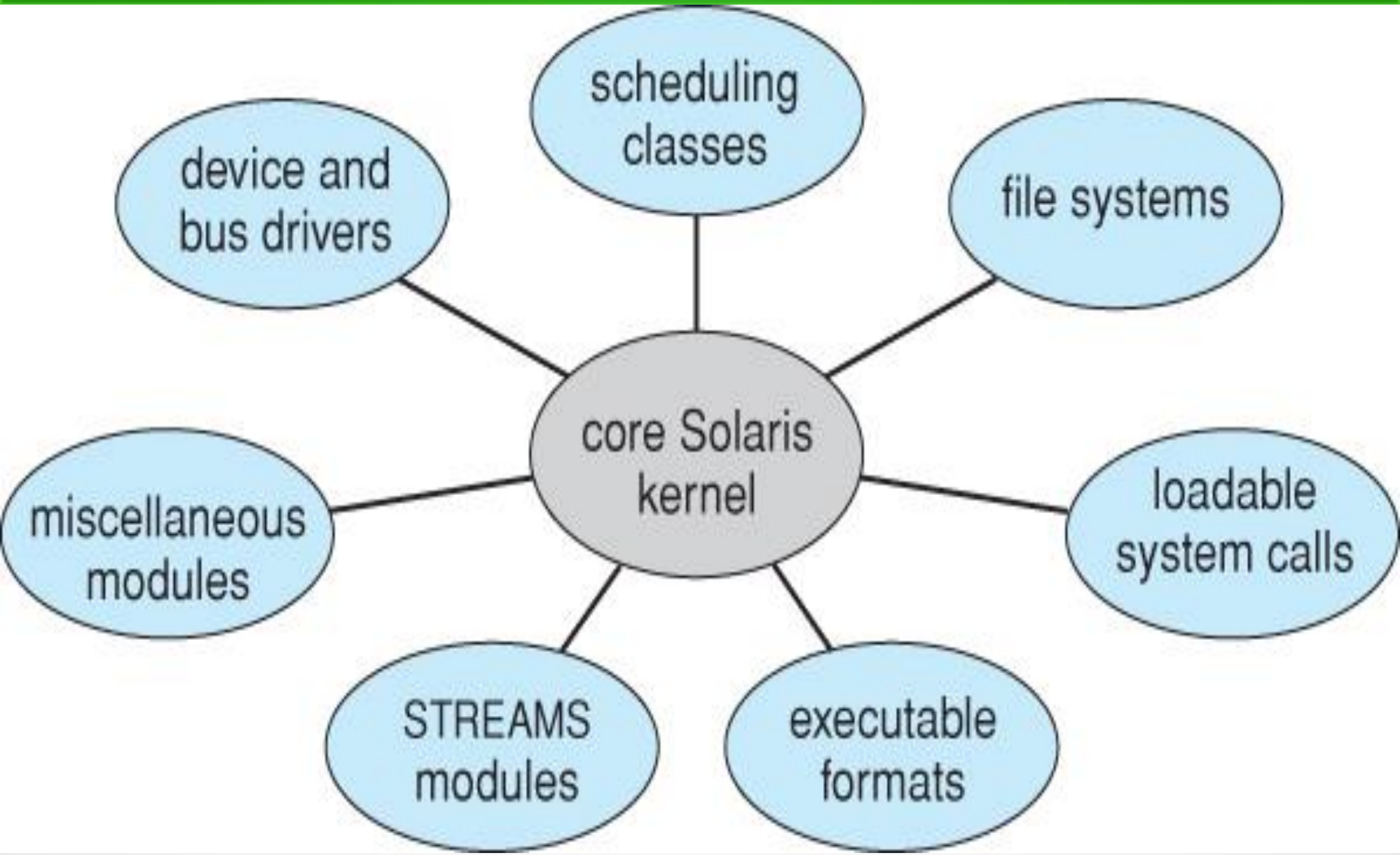
- Such a strategy uses dynamically loadable modules and is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X.

Modules

For example, the Solaris operating system structure is organized around a core kernel with **seven types** of loadable kernel modules:

1. **Scheduling classes**
2. **File systems Solaris loadable modules.**
3. **Loadable system calls**
4. **Executable formats**
5. **STREAMS modules**
6. **Miscellaneous**
7. **Device and bus drivers**

Modules


scheduling classes
device and bus drivers
file systems
core Solaris kernel
miscellaneous modules
loadable system calls
STREAMS modules
executable formats