KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

Website: www.ktunotes.in

MODULE 1

Introduction to Operating system

CONTENTS

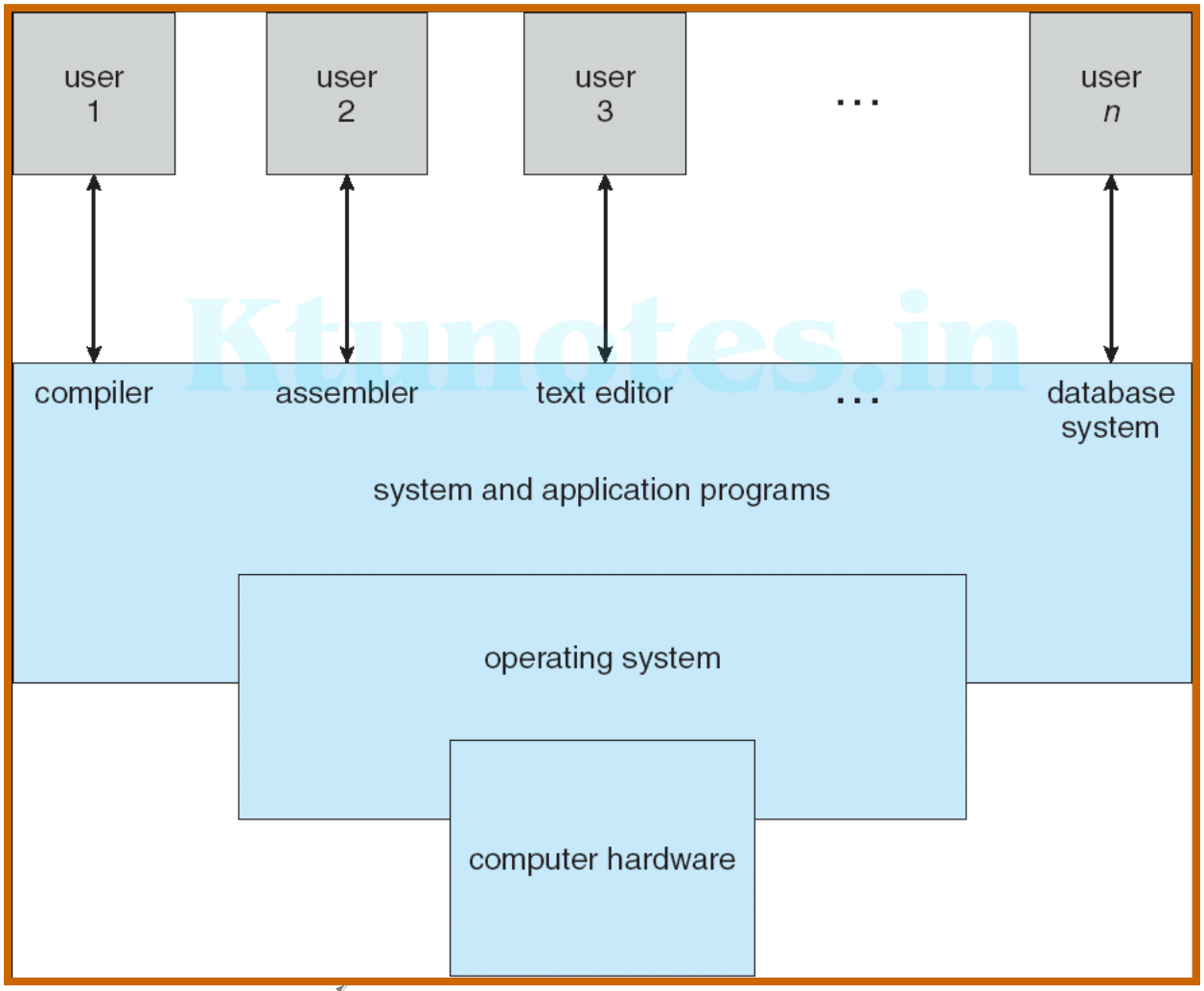
- Introduction: Operating system overview – Operations, Functions, Service – System calls,
- Types – Operating System structure - Simple structure, Layered approach, Microkernel, Modules– System boot process

Introduction to Operating system

An **operating system is a program that manages a computer's hardware.**

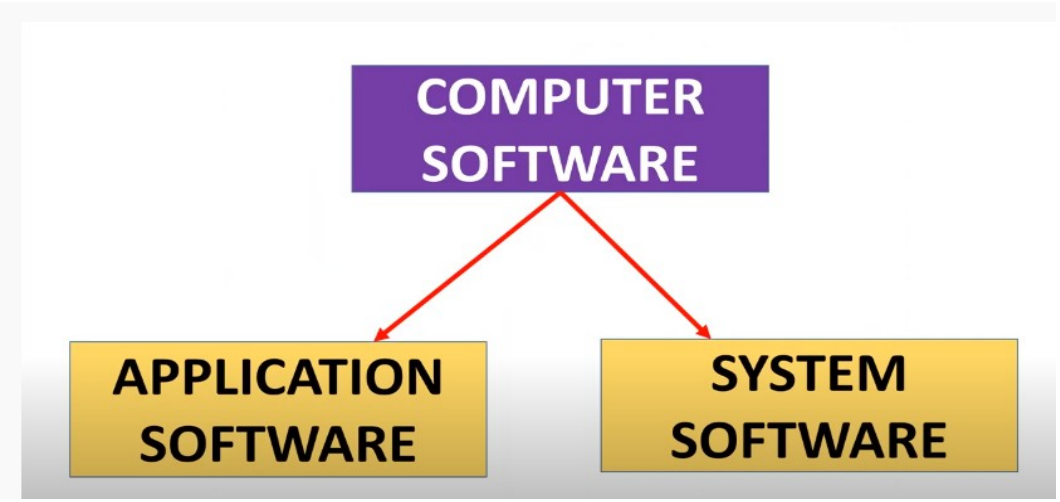
An Operating System is a program that acts as an **intermediary** between a **user** of a computer and the computer **hardware**.

Four Components of a Computer System (Computer System Structure)



Computer System Structure

- Computer system can be divided into four components
  - **Hardware** – provides basic computing resources
    - CPU, memory, I/O devices.
  - **Operating system**
    - Controls and coordinates use of hardware among various applications and users.
  - **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users.
    - Word processors, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers



Application Soft wares are programs that help the user to perform specific task. Eg: Word processors, web browsers, database systems, video games

System Software is a set of programs that control and manage the operations of computer hardware. It also helps application programs to execute correctly.

- To understand ,fully the operating system's role, we explore operating systems from two viewpoints:
  - The user
  - The system.

User View:

The user's view of the computer varies according to the interface being used

- **Single user computers** (e.g., PC, workstations). Such systems are designed for one user to monopolize its resources. The goal is to maximize the work (or play) that the user is performing. the operating system is designed mostly for **ease of use** and **good performance.**
- **Multi user computers** (e.g., mainframes, computing servers). These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization -- to assure that all available CPU time, memory, and I/O are used efficiently and that no individual users takes more than their air share

User View (Cont.)

- **Handheld computers** (e.g., smartphones and tablets). The user interface for mobile computers generally features a **touch screen.** The systems are resource poor, optimized for usability and battery life.
- **Embedded computers** (e.g., computers in home devices and automobiles) The user interface may have numeric keypads and may turn indicator lights on or off to show status. The operating systems are designed primarily to run without user intervention.

System View:

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

OS goals:

- Control/execute user/application programs.
- Make the computer system convenient to use.
- Ease the solving of user problems.
- Use the computer hardware in an efficient manner.

Click to add text

Functions of an Operating System

- Process Management
- Memory Management
- Storage Management
- I/O Management
- Protection and Security

Process Management

- A process is program in execution.

Eg: A word-processing program being run by an individual user on a PC

- A process needs certain resources to accomplish its task.
  - CPU time
  - memory
  - files
  - I/O devices.
- These resources are either given to the process when it is created or allocated to it while it is running.

Process Management

- The operating system is responsible for the following process management activities :
  - Scheduling processes and threads on the CPUs
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication

Memory Management

- Main memory is the only large storage device that the CPU is able to address and access directly.
- For a program to be executed, it must be mapped to absolute addresses and loaded into memory.
- As the program executes, it accesses program instructions and data from memory by generating these absolute addresses.
- When the program terminates, its memory space is declared available, and the next program can be loaded and executed.

- To improve both the utilization of the CPU and the speed of the computer's response to its users, general-purpose computers must keep several programs in memory, creating a need for memory management.

Memory Management

- The operating system is responsible for the following memory management activities :
  - Keeping track of which parts of memory are currently being used and who is using them
  - Deciding which processes (or parts of processes) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage management-File-System Management

Files represent programs (both source and object forms) and data.

Data files may be numeric, alphabetic, alphanumeric, or binary.

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

- Creating and deleting files
- Creating and deleting directories to organize files
- Supporting primitives for manipulating files and directories
- Mapping files onto secondary storage
- Backing up files on stable (non-volatile) storage media

Storage management-Mass Storage Management

- Main memory is too small to accommodate all data and programs, and because the data that it holds are lost when power is lost, the computer system must provide secondary storage to back up main memory.
- Most modern computer systems use disks as the principal 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

Storage management-Caching:

- Information is normally kept in some storage system (such as main memory).
- As it is used, it is copied into a faster storage system ,the cache on a temporary basis.
- When we need a particular piece of information, we first check whether it is in the cache. If it is, we use the information directly from the cache.
- If it is not, we use the information from the source, putting a copy in the cache under the assumption that we will need it again soon.

Storage management-Caching:

Cache memory is an extremely fast **memory** type that acts as a buffer between **RAM** and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed. **Cache memory** is used to reduce the average time to access data from the Main **memory**

Instruction Cache

- Most systems have an instruction cache to hold the instructions expected to be executed next.
- Without this cache, the CPU would have to wait several cycles while an instruction was fetched from main memory.

Data Cache

- Most systems have one or more high-speed data caches in the memory hierarchy.

Storage management-Caching:

- Because caches have limited size, **cache management is an important** design problem.

Migration of integer A from disk to register.

Storage management-Caching:

- In a hierarchical storage structure, the same data may appear in different levels of the storage system.
- For example, suppose that an integer A that is to be incremented by 1 is located in file B, and file B resides on magnetic disk.
- The increment operation proceeds by first issuing an I/O operation to copy the disk block on which A resides to main memory.
- This operation is followed by copying A to the cache and to an internal register.
- Thus, the copy of A appears in several places: on the magnetic disk, in main memory, in the cache, and in an internal register
- Once the increment takes place in the internal register, the value of A differs in the various storage systems.

Storage management-Caching:

- The value of A becomes the same only after the new value of A is written from the internal register back to the magnetic disk.
- In a multitasking environment, where the CPU is switched back and forth among various processes ,if several processes wish to access A, then each of these processes will obtain the most recently updated value of A.
- In a multiprocessor environment where, in addition to maintaining internal registers, each of the CPUs also contains a local cache .
- In such an environment, a copy of A may exist simultaneously in several caches.
- Since the various CPUs can all execute in parallel, we must make sure that an update to the value of A in one cache is immediately reflected in all other caches where A resides.
- This situation is called **cache coherency, and it is usually a hardware issue .**

Storage management-Caching:

- In a distributed environment, several copies (or replicas) of the same file can be kept on different computers.
- Since the various replicas may be accessed and updated concurrently, some distributed systems ensure that, when a replica is updated in one place, all other replicas are brought up to date as soon as possible

I/O Management

- One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user.
- Every operating systems has an I/O subsystem for managing its I/O devices.

Protection and Security

- If a computer system has multiple users and allows the concurrent execution of multiple processes, then access to data must be regulated.
- There must be mechanisms to ensure that files, memory segments, CPU, and other resources can be operated on by only those processes that have gained proper authorization from the operating system.
- For example, memory-addressing hardware ensures that a process can execute only within its own address space.
- The timer ensures that no process can gain control of the CPU without eventually relinquishing control.

Protection and Security

- Protection is any mechanism for controlling the access of processes or users to the resources defined by a computer system.
- A system can have adequate protection but still be prone to failure and allow inappropriate access.

- Job of security is to defend a system from external and internal attacks.

Protection and Security

- Attacks spread across a huge range and include viruses and worms, denial-of service attacks (which use all of a system's resources and so keep legitimate users out of the system), identity theft, and theft of service (unauthorized use of a system).
- Prevention of some of these attacks is considered an operating-system function .
- Protection and security require the system to be able to distinguish among all its users.
- Most operating systems maintain a list of user names and associated user identifiers (user IDs)

Protection and Security

- In some circumstances ,it is necessary to distinguish among sets of users rather than individual users.
- To accomplish this, define a group name and the set of users belonging to that group.
- Group functionality can be implemented as a system-wide list of group names and group identifiers.
- A user can be in one or more groups, depending on operating-system design.

Computer system can be categorize according to the number of general-purpose processors used:

- Single-Processor Systems
- Multiprocessor Systems
- Clustered Systems

Single-Processor Systems

- On a single processor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes.

- Almost all single processor systems have other special-purpose processors like device-specific processors, such as disk, keyboard, and graphics controllers.

- All of these special-purpose processors run a limited instruction set and do not run user processes.

- Sometimes, they are managed by the operating system, in that the operating system sends them information about their next task and monitors their status.

Multiprocessor Systems

- Two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices.

Multiprocessor systems have three main advantages:

1. **Increased throughput**

- By increasing the number of processors, more work done in less time.
- The speed-up ratio with *N processors* is not *N, however; rather, it is less than N.*
- *When multiple processors* cooperate on a task, a certain amount of overhead is incurred in keeping all the parts working correctly.
- Contention for shared resources also, lowers the expected gain from additional processors.

Multiprocessor Systems

2. **Economy of scale**

Multiprocessor systems can cost less than equivalent multiple single processor systems, because they can share peripherals, mass storage, and power supplies.

3. **Increased reliability**

If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down.

If we have ten processors and one fails, then each of the remaining nine processors can pick up a share of the work of the failed processor. Thus, the entire system runs only 10 percent slower, rather than failing altogether.

Multiprocessor Systems

The multiple-processor systems in use today are of two types:

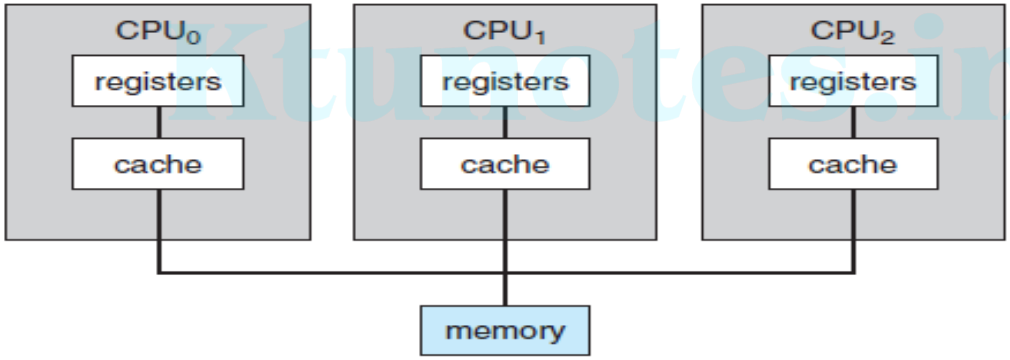
Asymmetric multiprocessing:

- **Each processor is assigned** a specific task.
- A ***boss processor controls the system; the other processors either*** look to the boss for instruction or have predefined tasks.
- This scheme defines a boss–worker relationship.
- The boss processor schedules and allocates work to the worker processors.

Multiprocessor Systems

Symmetric multiprocessing (SMP)

- Each processor performs all tasks within the operating system.
- All processors are peers; no boss–worker relationship exists between processors.

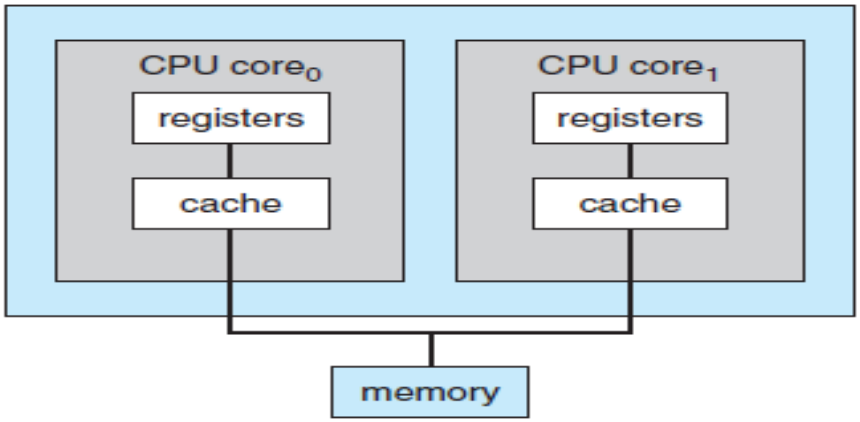


Symmetric multiprocessing architecture.

- Each processor has its own set of registers, as well as a private or local cache.
- All processors share physical memory.

Multiprocessor Systems

- Multiprocessing adds CPUs to increase computing power.
- Multiprocessor systems are termed as **multicore when** multiple computing **cores** on a single chip.
- They can be more efficient than multiple chips with single cores because on-chip communication is faster than between-chip communication.
- One chip with multiple cores uses significantly less power than multiple single-core chips



7 A dual-core design with two cores placed on the same chip.

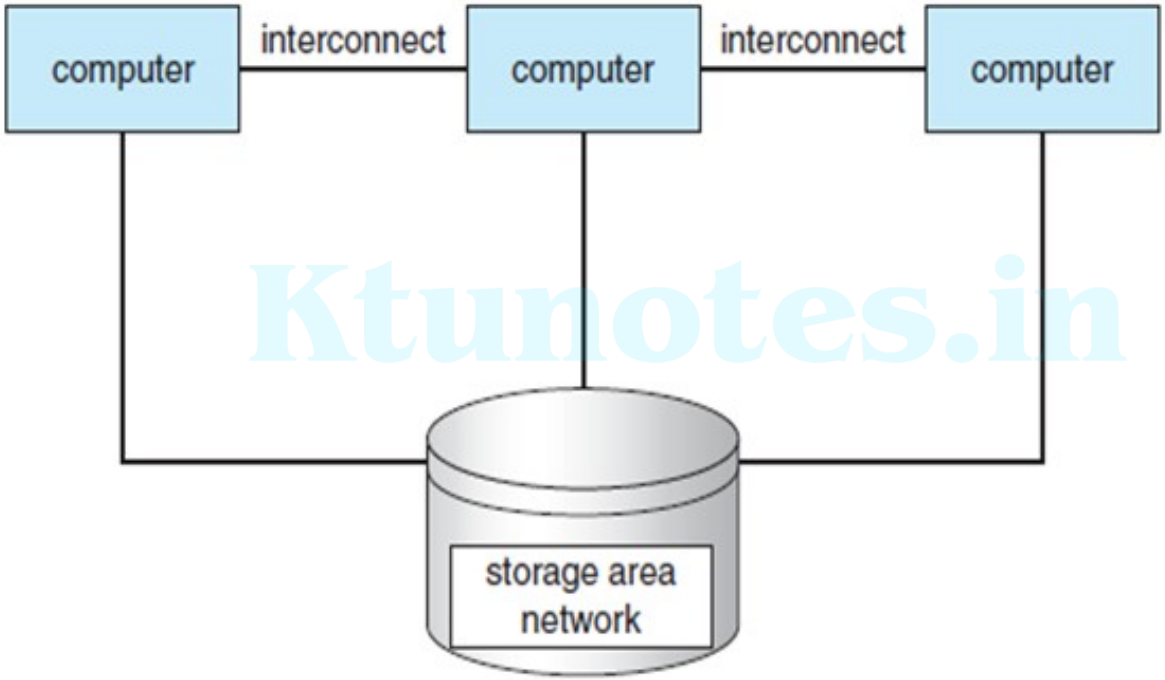
Clustered Systems

- A type of multiprocessor system is a **clustered system**, which gathers together multiple CPUs.
- Clustered systems differ from the multiprocessor systems in that they are composed of two or more individual systems or nodes joined together.
- **Each node may be a single processor system or a multicore** system.
- Clustered computers share storage and are closely linked via a local-area network LAN or a faster interconnect.
- Clustering is usually used to provide high-availability service.

Clustered Systems

- Service will continue even if one or more systems in the cluster fail.
- A layer of cluster software runs on the cluster nodes.
- Each node can monitor one or more of the others (over the LAN).
- If the monitored machine fails, the monitoring machine can take ownership of its storage and restart the applications that were running on the failed machine.
- The users and clients of the applications see only a brief interruption of service.

Clustered Systems



General structure of a clustered system.

Types of Operating System

- Single User
- Multi-user
- Multi-tasking
- Multiprocessing
- Batch Processing
- Real Time
- Distributed Systems

Single User

TWO TYPES:

- **Single user, single task**

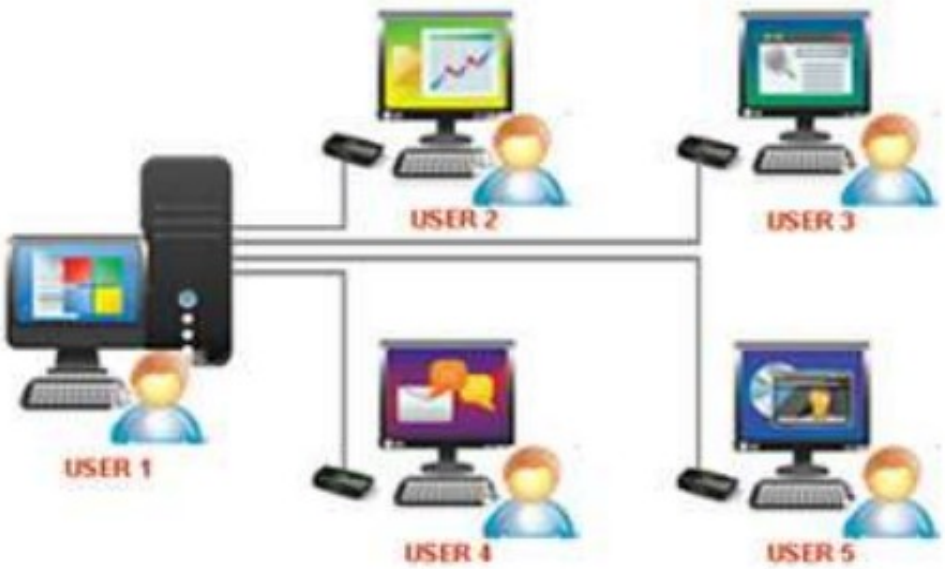
  Designed to manage the computer so that one user can effectively do one thing at a time. Example: The Palm OS for Palm handheld computers

- **Single user, multi tasking**

  Designed with a single user in mind but can deal with many applications running at the same time .Type of operating system most people use on their desktop and laptop computers today

Multi-user

- Allows many different users to take advantage of the computer's resources simultaneously
- Allows multiple users to access the computer system at the same time
- Time Sharing system and Internet servers as the multi user systems



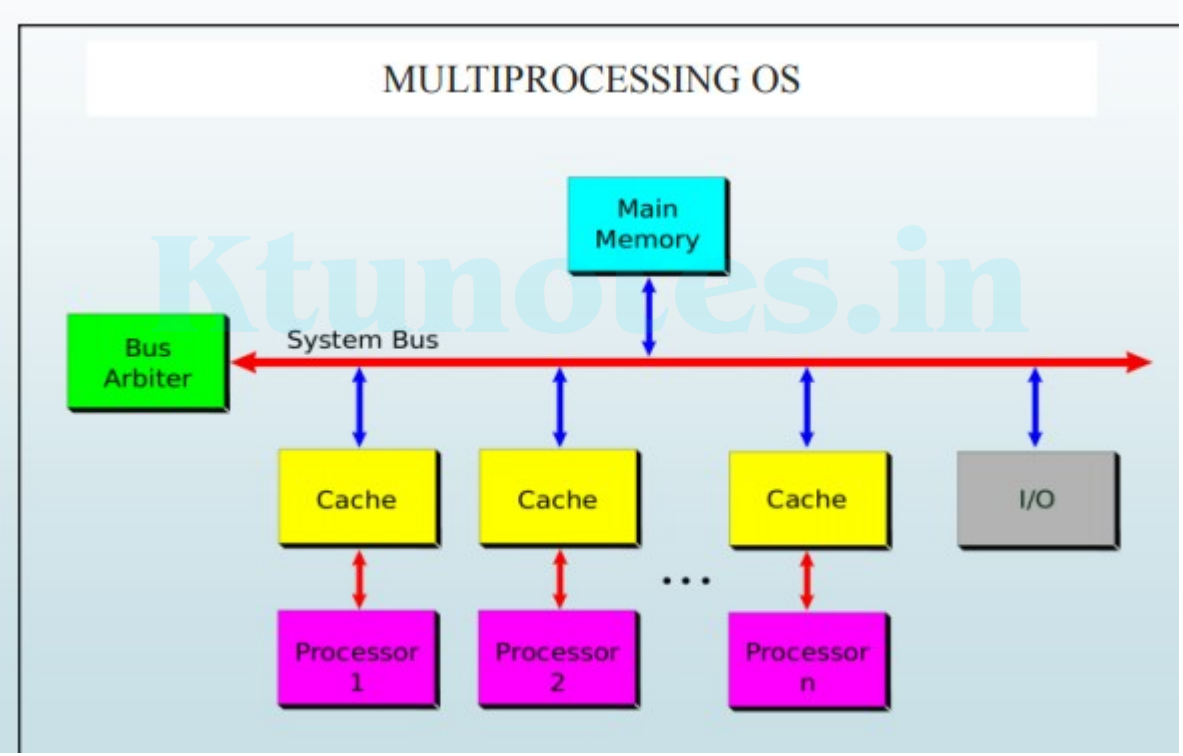
Multi-tasking

- Allows more than one program to run concurrently.
- The tasks share common processing resources, such as a CPU and main memory
- In the process, only one CPU is involved, but it switches from one program to another so quickly that it gives the appearance of executing all the programs at the same time.

MULTIPROCESSING

- Multiprocessing, in general, refers to the utilization of multiple CPUs in a single computer system
- Enables several programs to run concurrently
- The term also refers to the ability of a system to support more than one processor and/or the ability to allocate tasks between them

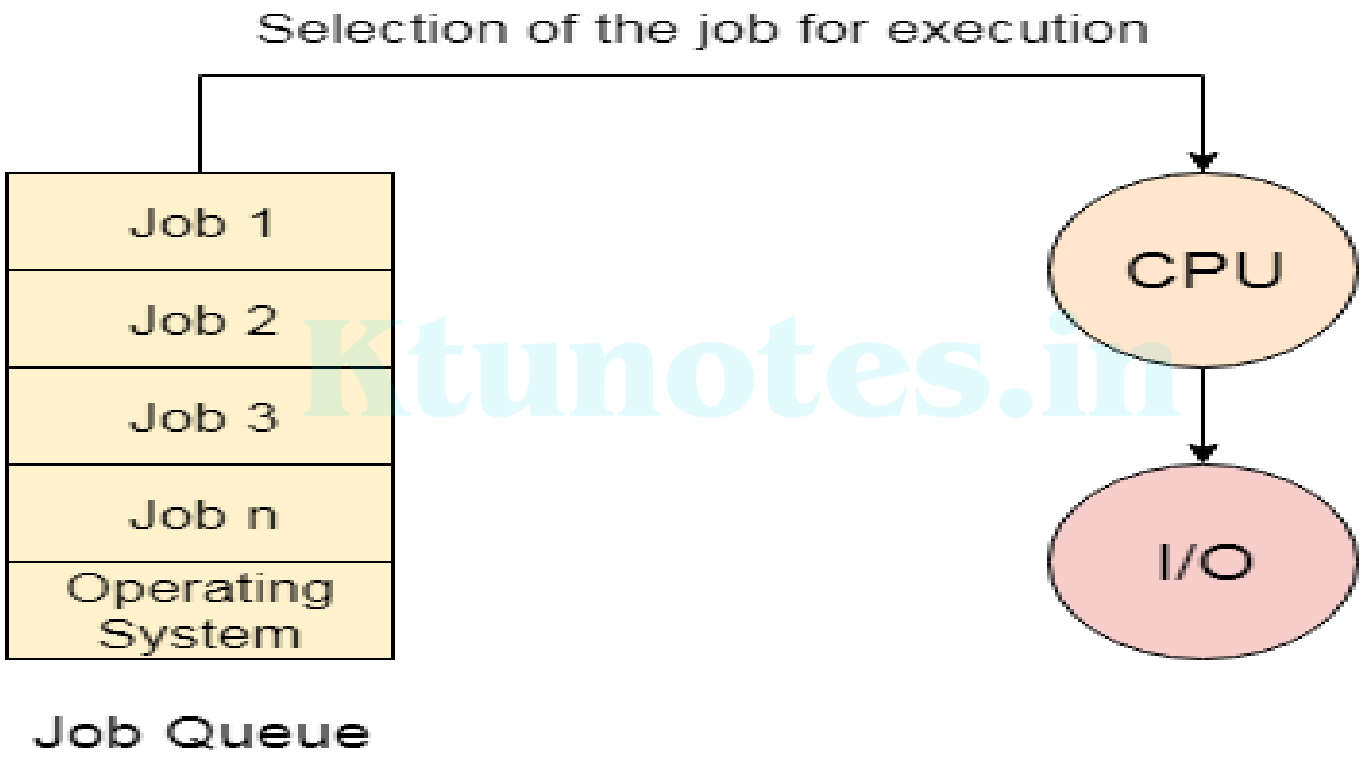
MULTI PROCESSING(cont..)



Batch Operating System

- **Batch Operating System**: the First operating system of the second-generation computer is the batch operating system.
- Batch operating system took the input on the punch card.
- Each punch card had the different form of data. System executed the jobs one by one in batch.
- When one job from the batch executed, then the second job has taken from it and so on.
- No need for user interaction.

Batch Operating System(cont..)



Real Time systems

- In Real Time systems, each job carries a certain deadline within which the Job is supposed to be completed, otherwise the huge loss will be there or even if the result is produced then it will be completely useless.
- The Application of a Real Time system exists in the case of military applications, if you want to drop a missile then the missile is supposed to be dropped with certain precision.

Distributed Operating System.

- Distributed Means data is stored and processed on multiple locations.
- When a data is stored on to the multiple computers, those are placed in different locations.

Operating-System Operations

- 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 operating system will sit quietly, waiting for something to happen.
- Events are almost always signaled by the occurrence of an interrupt or a 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**.**

- **Interrupts** are signals sent to the CPU by external devices, normally I/O devices. They tell the CPU to stop its current activities and execute the appropriate part of the operating system.

There are three types of interrupts:

- **Hardware Interrupts** are generated by hardware devices to signal that they need some attention from the OS. They may have just received some data (e.g., keystrokes on the keyboard); or they have just completed a task which the operating system previous requested, such as transferring data between the hard drive and memory.

- **Software Interrupts** are generated by programs when they want to request a system call to be performed by the operating system.

- **Traps** are generated by the CPU itself to indicate that some error or condition occurred for which assistance from the operating system is needed.
- For each type of interrupt, separate segments of code in the operating system determine what action should be taken.
- An **interrupt service routine (ISR)** is provided to deal with the interrupt.

Dual-Mode Operation

- **Dual-mode** operation allows OS to protect itself and other system components
- **User mode** and **kernel mode(** also called supervisor mode, system mode, or privileged mode)
  - **Mode bit** provided by hardware(kernel (0) or user (1).)
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

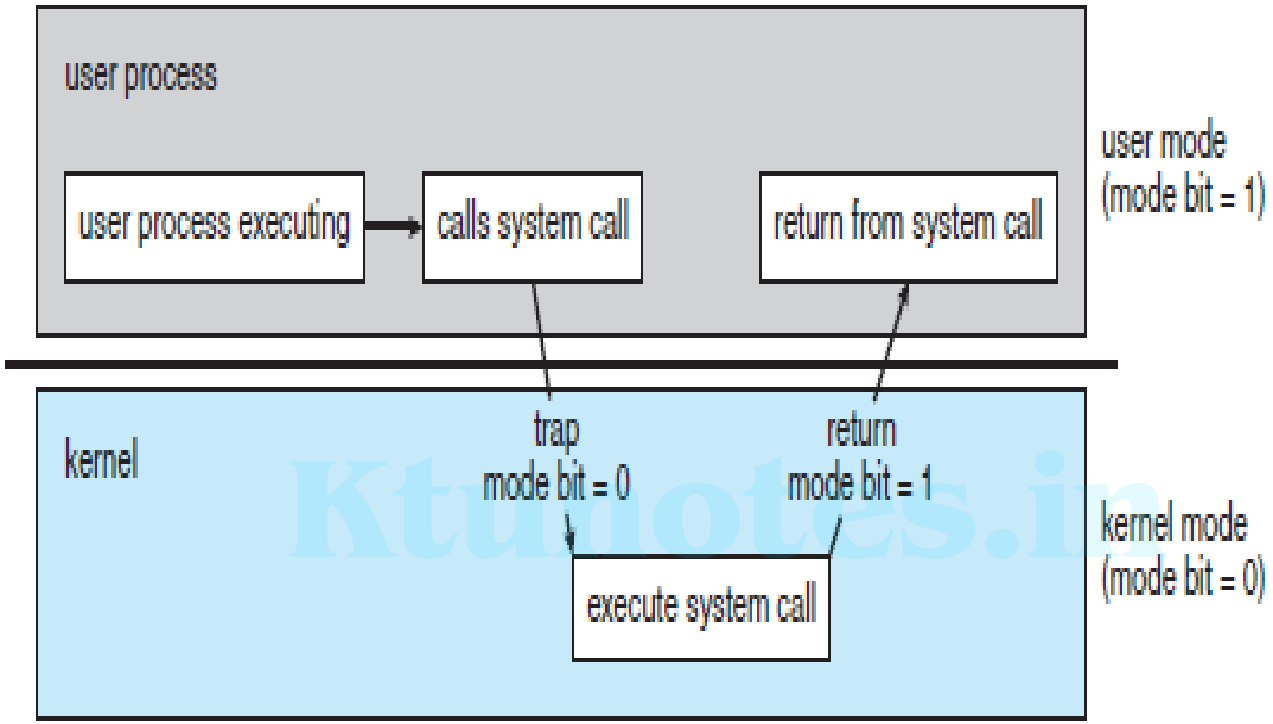


Figure 1.10 Transition from user to kernel mode.

- If the user attempts to run privileged instruction in user mode then it will treat instruction as illegal and traps to OS. Some of the privileged instructions are:

- Handling Interrupts
- To switch from user mode to kernel mode.
- Input-Output management.

Timer

- We must ensure that the operating system maintains control over the CPU.
- We cannot allow a user program to get stuck in an infinite loop or to fail to call system services and never return control to the operating system.
- To accomplish this goal, we can use a **timer.**
- A timer can be set to interrupt the computer after a specified period.

System Calls

- The mechanism used by an application program to request service from the operating system.

- System calls causes the processor to change mode (e.g. to "supervisor mode" or "protected mode").

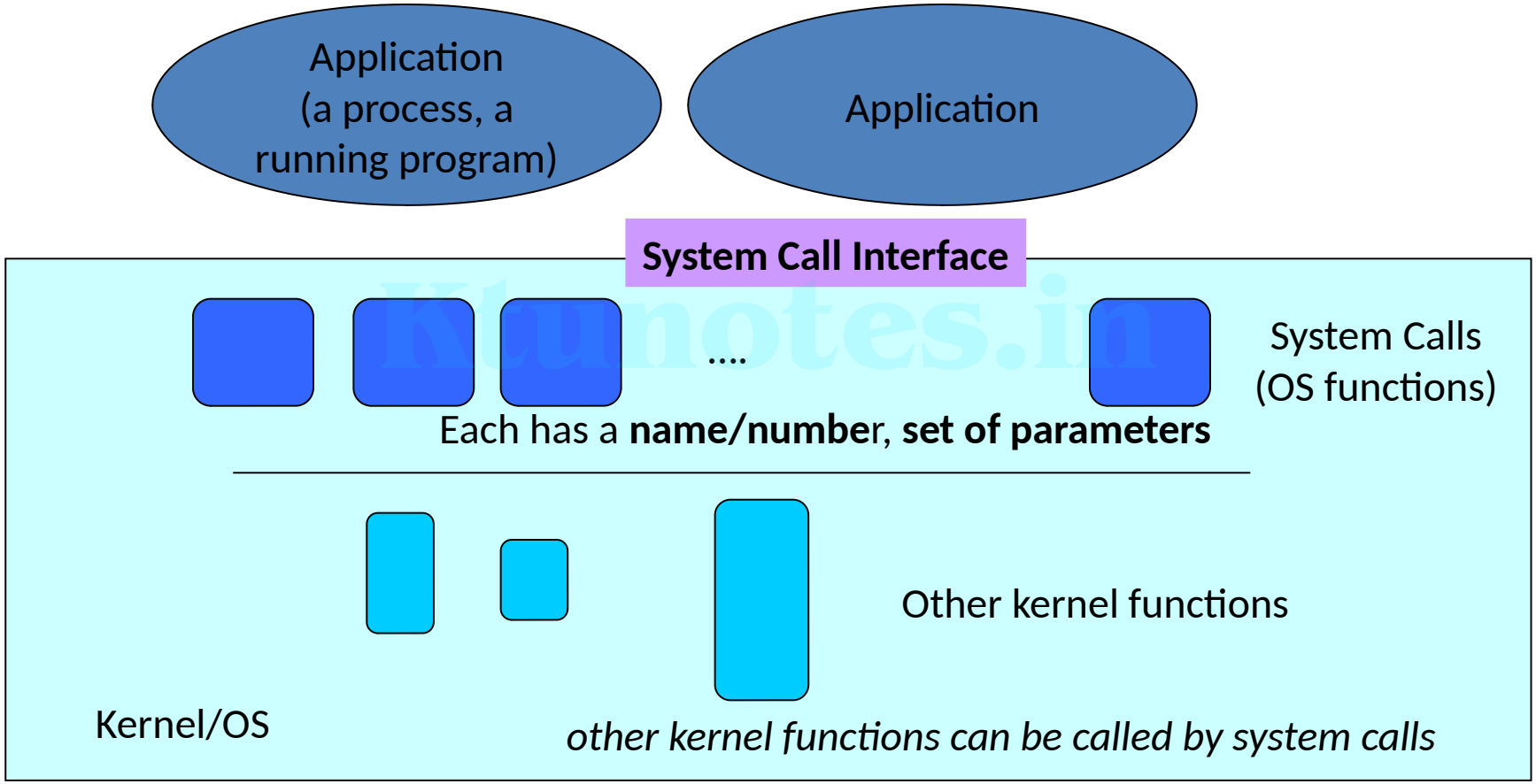
- This allows the OS to perform restricted actions such as accessing hardware devices or the memory management unit.

System Calls

- Systems calls are the programming interface to the services provided by the OS

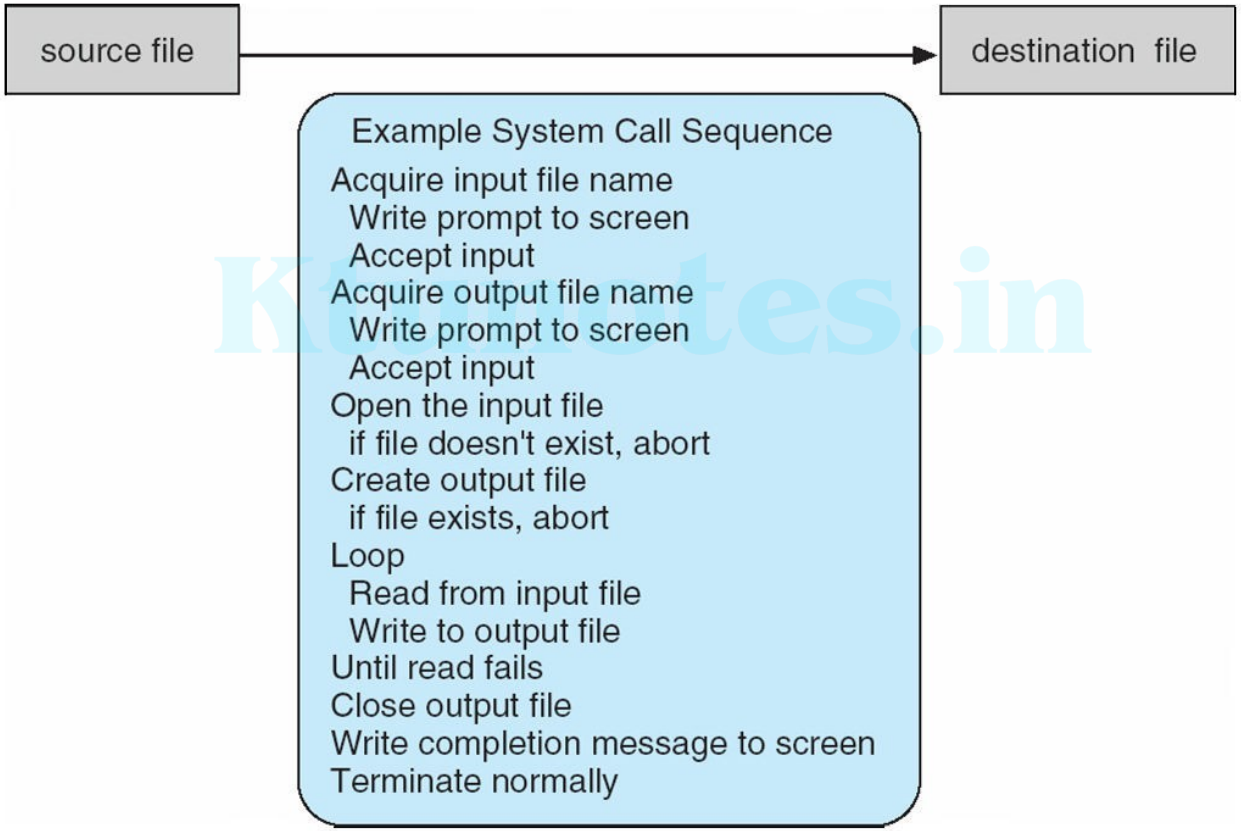
- Typically written in a high-level language (C or C++)

System Calls



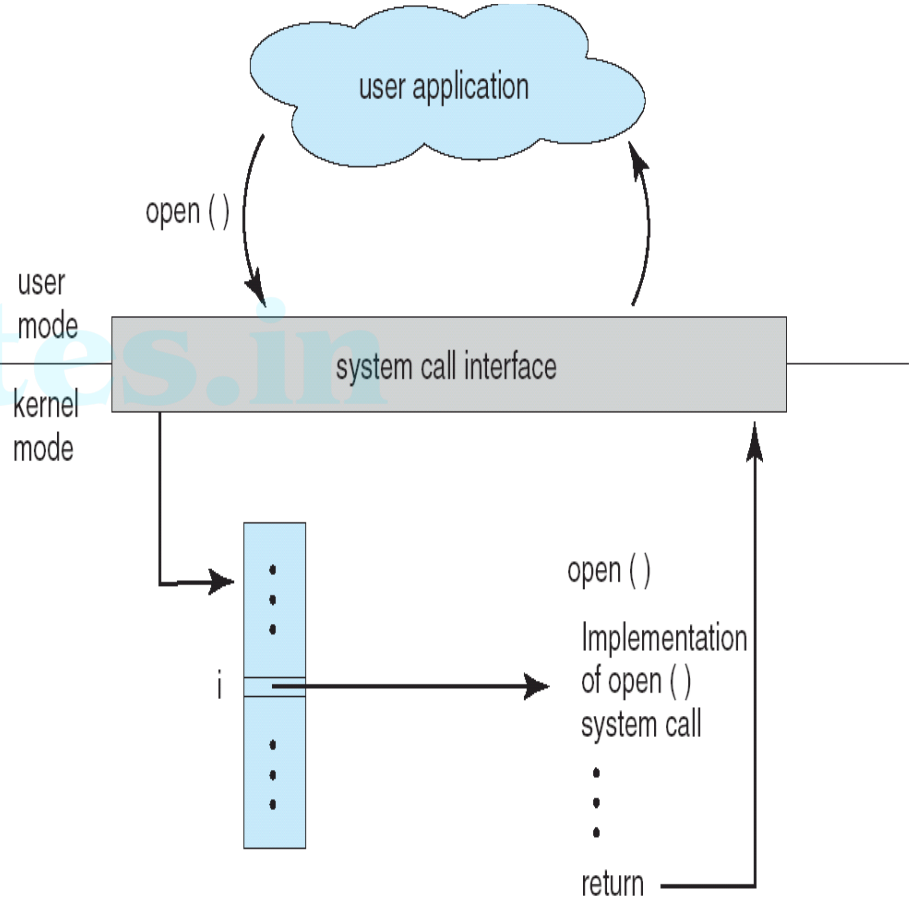
Example of System Calls

- System call sequence to copy the contents of one file to another file



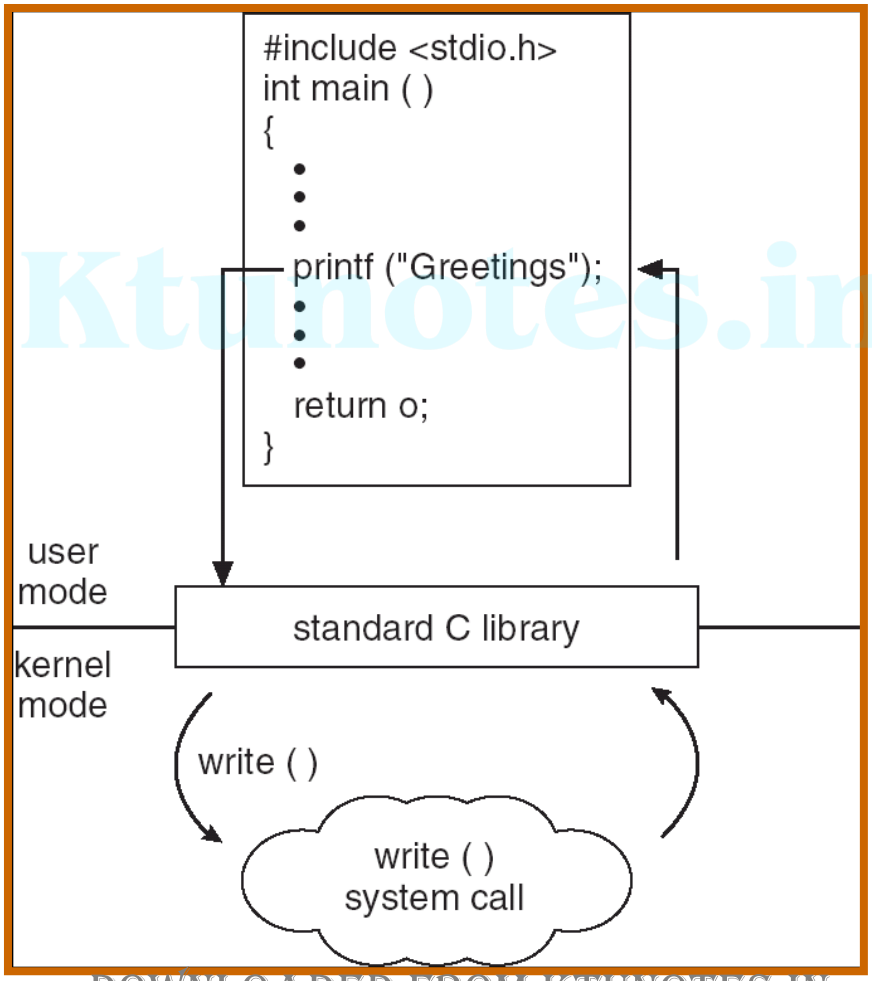
System Call Implementation and Calling

- Typically,
  - a **number** associated with each system call
  - Number used as an index to a table: System Call **table**
  - Table keeps **addresses** of system calls (routines)
  - System call runs and returns
- Caller does not know system call implementation
  - Just knows interface



Standard C Library Example

- C program invoking `printf()` library call, which calls the write() system call



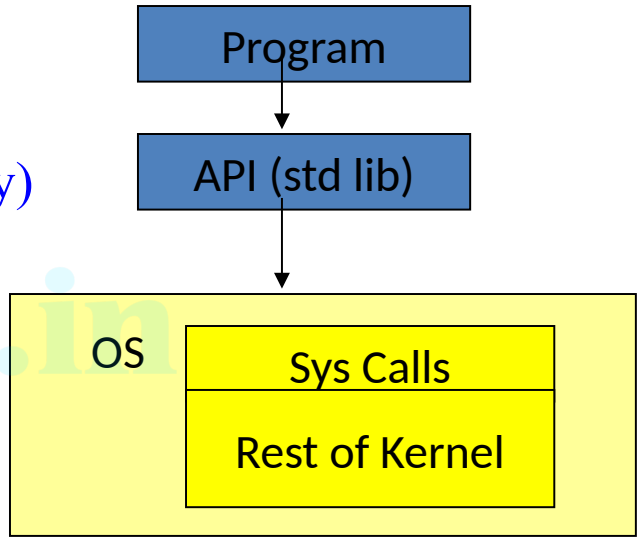
System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
  - Exact type and amount of information vary according to OS and call
- 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.

Accessing and executing System Calls

- System calls typically **not accessed directly** by programs
- Mostly accessed by programs via a high-level Application Program Interface (API) (i.e. a library) rather than direct system call use
- Three most common APIs are :
  - Win32 API for Windows,
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X),
  - Java API for the Java virtual machine (JVM)



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 and free memory
- File management
  - create file, delete file
  - open, close file
  - read, write, reposition
  - get and set file attributes

Types of System Calls

- Device management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- Information maintenance
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes
- Communications
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach and detach remote devices

Important System Calls Used in OS

wait()

In some systems, a process needs to wait for another process to complete its execution. This type of situation occurs when a parent process creates a child process, and the execution of the parent process remains suspended until its child process executes.

The suspension of the parent process automatically occurs with a wait() system call. When the child process ends execution, the control moves back to the parent process.

fork()

Processes use this system call to create processes that are a copy of themselves. With the help of this system Call parent process creates a child process, and the execution of the parent process will be suspended till the child process executes.

exec()

This system call runs when an executable file in the context of an already running process that replaces the older executable file. However, the original process identifier remains as a new process is not built, but stack, data, head, data, etc. are replaced by the new process.

kill():

The kill() system call is used by OS to send a termination signal to a process that urges the process to exit. However, a kill system call does not necessarily mean killing the process and can have various meanings.

exit():

The exit() system call is used to terminate program execution. Specially in the multi-threaded environment, this call defines that the thread execution is complete. The OS reclaims resources that were used by the process after the use of exit() system call.

EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

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

Operating System Structures

- Simple Structure
- Layered Systems
- Microkernels
- Modular

Simple Structure

- Many operating systems do not have well-defined structures.
- Frequently, such 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 carefully divided into modules.

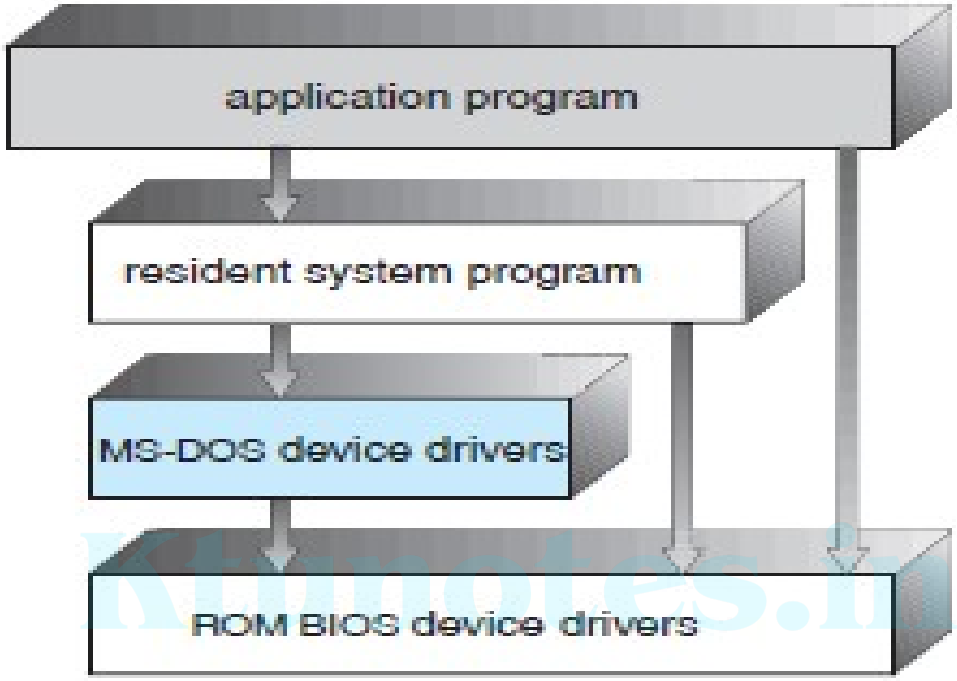


Figure 2.12 MS-DOS layer structure.

Device driver is a computer program that operates or controls a particular type of device that is attached to a computer.

BIOS(Basic Input output system) in a PC stored on a *ROM* chip

Simple Structure(cont..)

- In MS-DOS, the interfaces and levels of functionality are not well separated.
- For instance, application programs are able to access the basic I/O routines to write directly to the display and disk drives. Such freedom leaves MS-DOS vulnerable to errant (or malicious) programs, causing entire system crashes when user programs fail.
- The Intel 8088 for which it was written provides no dual mode and no hardware protection, the designers of MS-DOS had no choice but to leave the base hardware accessible.

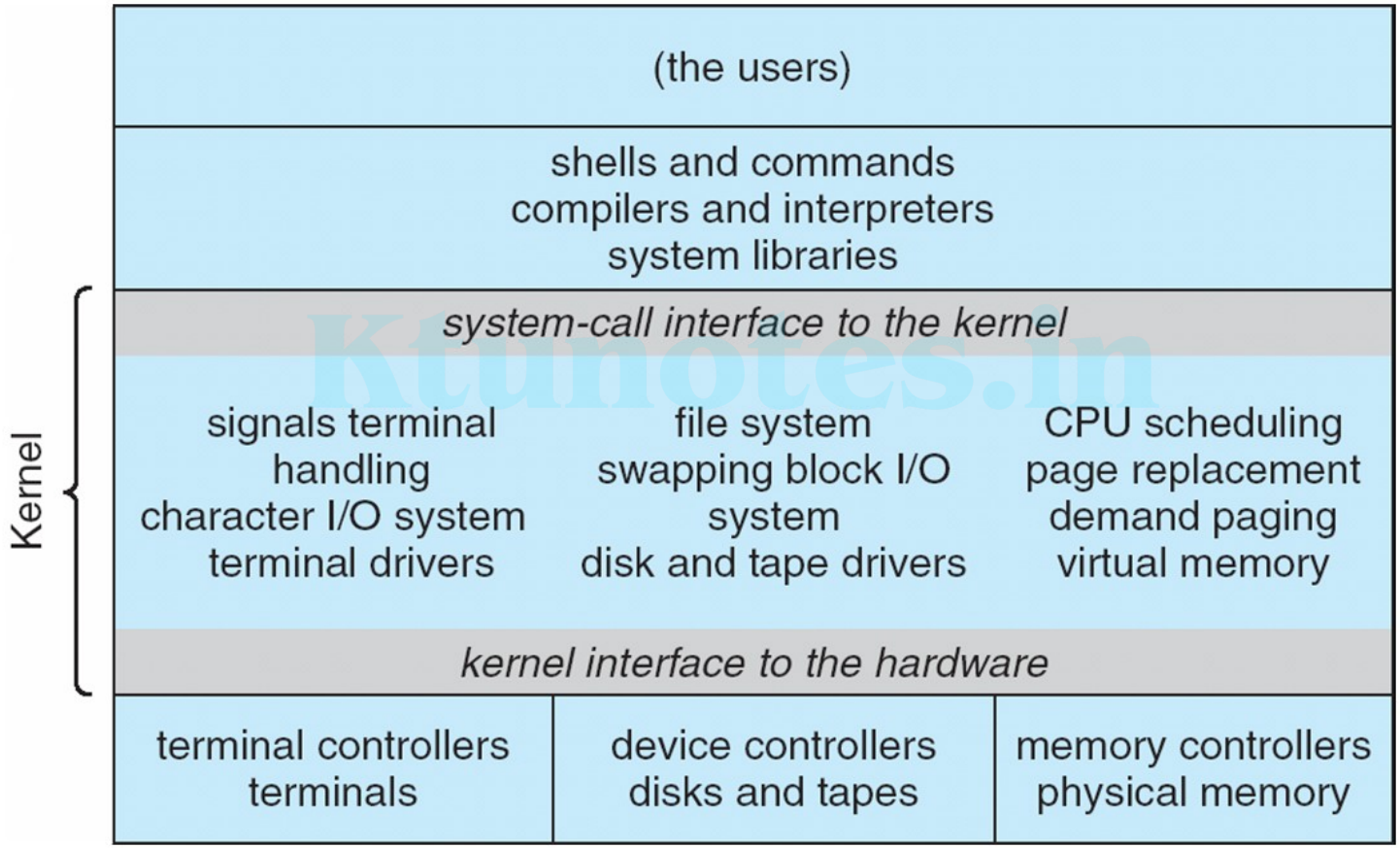
Monolithic Structure -- UNIX

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
  - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

Traditional UNIX System Structure

Beyond simple but not fully layered



Monolithic Structure – UNIX(cont..)

- This monolithic structure was difficult to implement and maintain.

- It had a distinct performance advantage, however: there is very little overhead in the system call interface or in communication within the kernel.

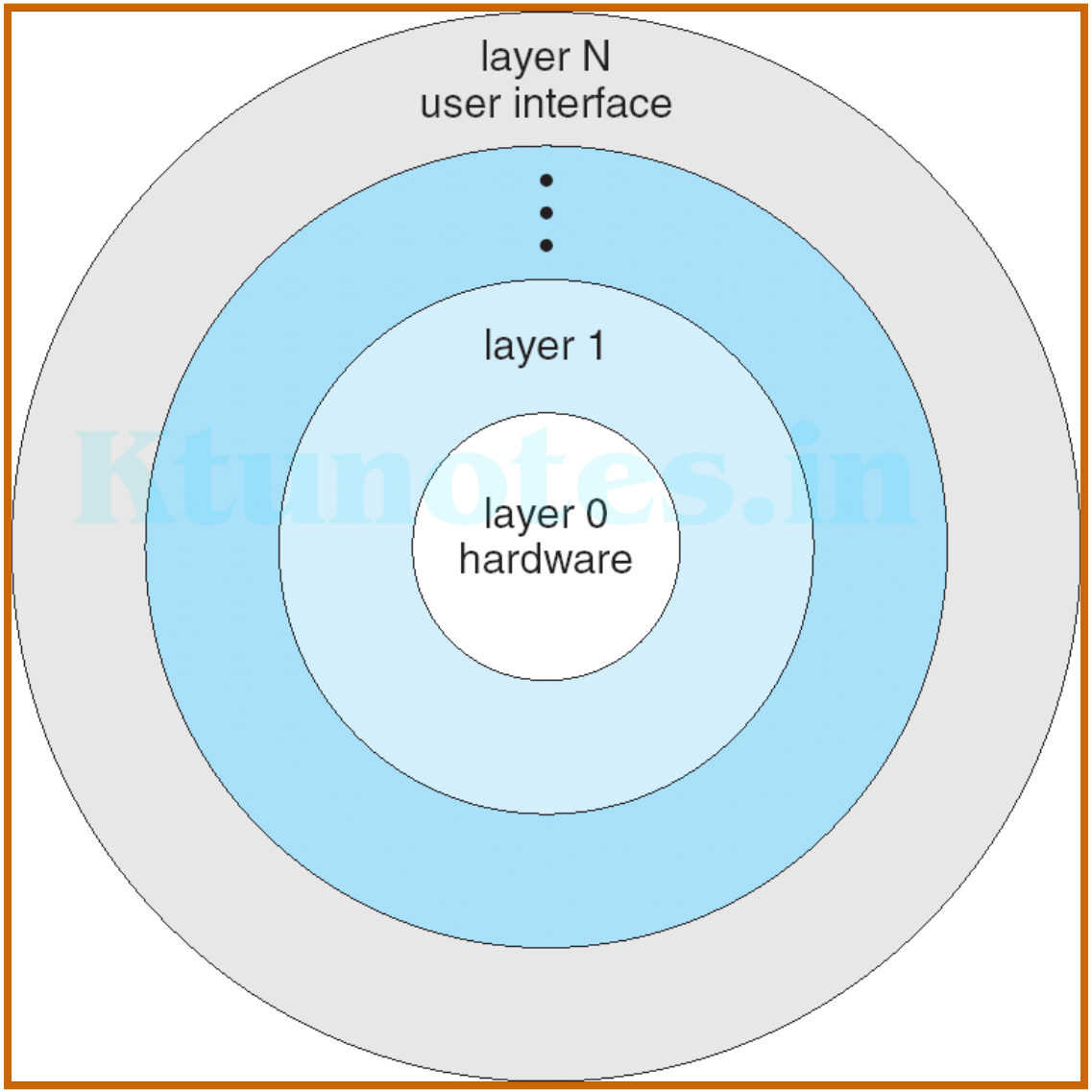
Layered Approach

- In a 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.

- Each layer uses functions (operations) and services of lower-level layers ie Layer n+1 uses services (exclusively) supported by layer n

- Easier to extend and evolve.

Layered Operating System



Layered Operating System

Advantage:

Simplicity of construction and debugging.

- Each layer 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, because, it uses only the basic hardware to implement its functions.
- Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on.
- If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged. Thus, the design and implementation of the system are simplified.

Layered Operating System

Disadvantage

It is difficult to appropriately define the various layers.

- Because a layer can use only lower-level layers, careful planning is necessary.
- For example, the device driver for the backing store must be at a lower level than the memory-management routines, because memory management requires the ability to use the backing store.

Layered Operating System

This structure is less efficient than other types.

- For instance, when a user program executes an I/O operation, it executes a system call that is trapped to the I/O layer, which calls the memory-management layer, which in turn calls the CPU-scheduling layer, which is then passed to the hardware.
- At each layer, the parameters may be modified, data may need to be passed, and so on. Each layer adds overhead to the system call. The net result is a system call that takes longer than does one on a non layered system.

Microkernel System Structure

- We have already seen that as UNIX expanded, the kernel became large and difficult to manage.
- In the mid-1980s, researchers at Carnegie Mellon University developed an operating system called **Mach that modularized** the kernel using the **microkernel approach.**
- **This method 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**.

Microkernel System Structure (cont..)

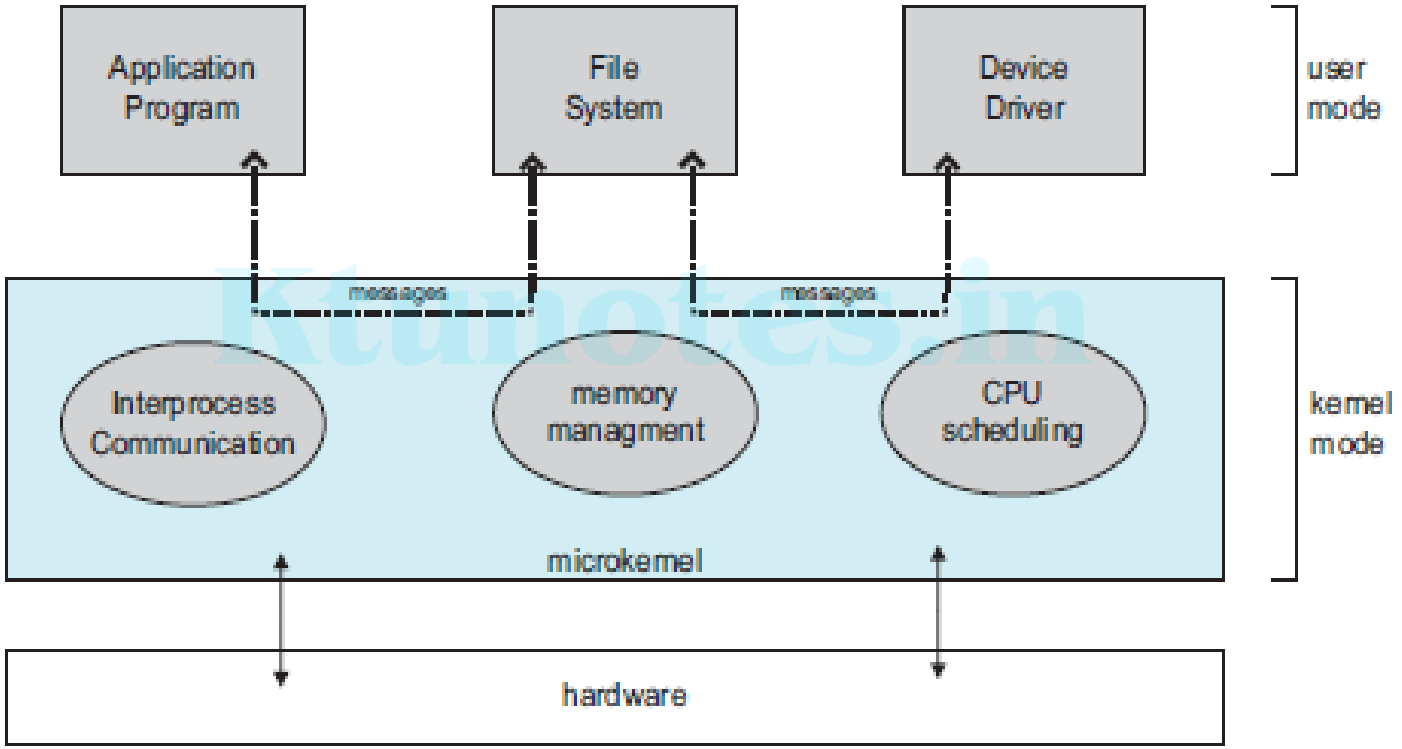


Figure 2.14 Architecture of a typical microkernel.

- The main function of the microkernel is to provide communication between the client program and the various services that are also running in user space. Communication is provided through **message passing.**

Benefits:

- One benefit of the microkernel approach is that it makes extending the operating system easier. 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 operating system is easier to port from one hardware design to another.
- The microkernel also provides more security and reliability, since most services are running as user—rather than kernel— processes. If a service fails, the rest of the operating system remains untouched.

Microkernel System Structure

- Benefits:(In short)
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Demerit:
  - Performance overhead of user space to kernel space communication

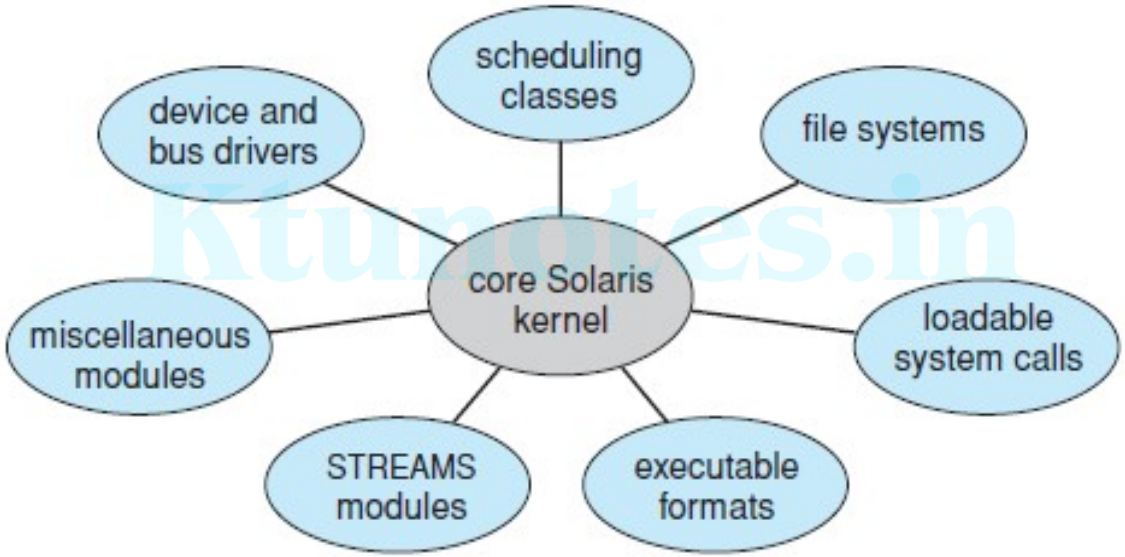
Modular

- Best current methodology for operating-system design involves using **loadable kernel modules.**
- **Here, the kernel has a set of core components** and links in additional services via modules, either at boot time or during run time.
- This type of design is common in modern implementations of UNIX, such as Solaris, Linux, and Mac OS X, as well as Windows.

Modular

- Kernel provides core services while other services are implemented dynamically, as the kernel is running.
- Resembles a layered system in that each kernel section has defined, protected interfaces; but it is more flexible than a layered system, because any module can call any other module.
- The approach is also similar to the microkernel approach in that the primary module has only core functions; but it is more efficient, because modules do not need to invoke message passing in order to communicate.

Modular-Example :Solaris operating system structure



Solaris loadable modules.

Modular-Example :Solaris operating system structure

The Solaris operating system structure, is organized around a core kernel with seven types of loadable kernel modules:

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

Operating System Services

- Operating systems provide an environment for execution of programs and services to programs and users
- One set of operating-system services provides functions that are helpful to the user:
  - **User interface** - Almost all operating systems have a user interface (**UI**).
    - Varies between **Command-Line (CLI), Graphics User Interface (GUI), Batch**
    - One is a **command-line interface (CLI), which uses text commands and a method for entering them (say,** a keyboard for typing in commands in a specific format with specific options).
    - Another is a **batch interface, in which commands and directives** to control those commands are entered into files, and those files are executed.
    - Most commonly, a **graphical user interface (GUI) is used**
  - **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device

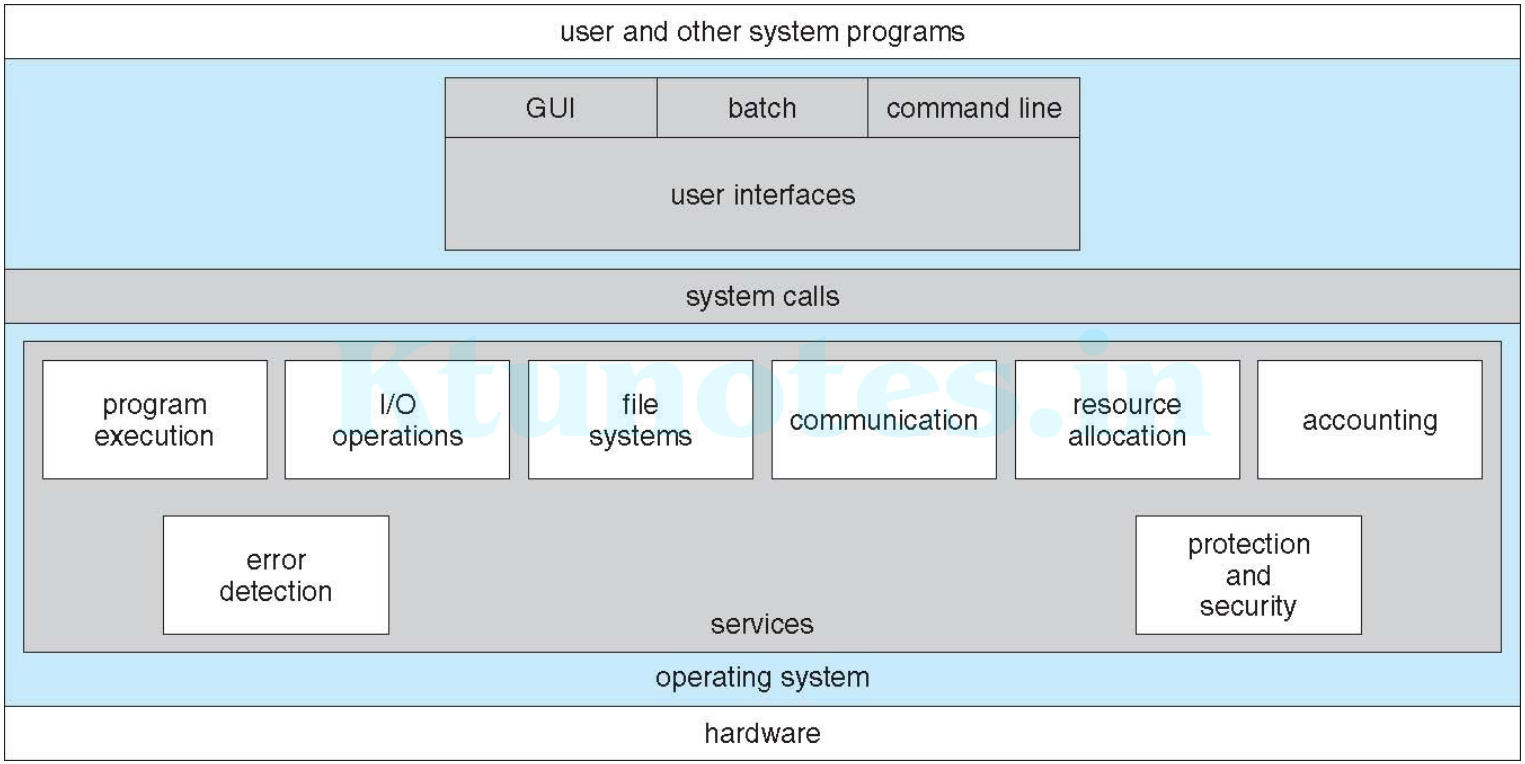
Operating System Services (Cont.)

- One set of operating-system services provides functions that are helpful to the user (Cont.):
  - **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
  - **Communications** – Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)
  - **Error detection** – OS needs to be constantly aware of possible errors
    - May occur in the CPU and memory hardware, in I/O devices, in user program
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Operating System Services (Cont.)

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources - CPU cycles, main memory, file storage, I/O devices.
  - **Accounting -** To keep track of which users use how much and what kinds of computer resources
  - **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - **Protection** involves ensuring that all access to system resources is controlled
    - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

A View of Operating System Services



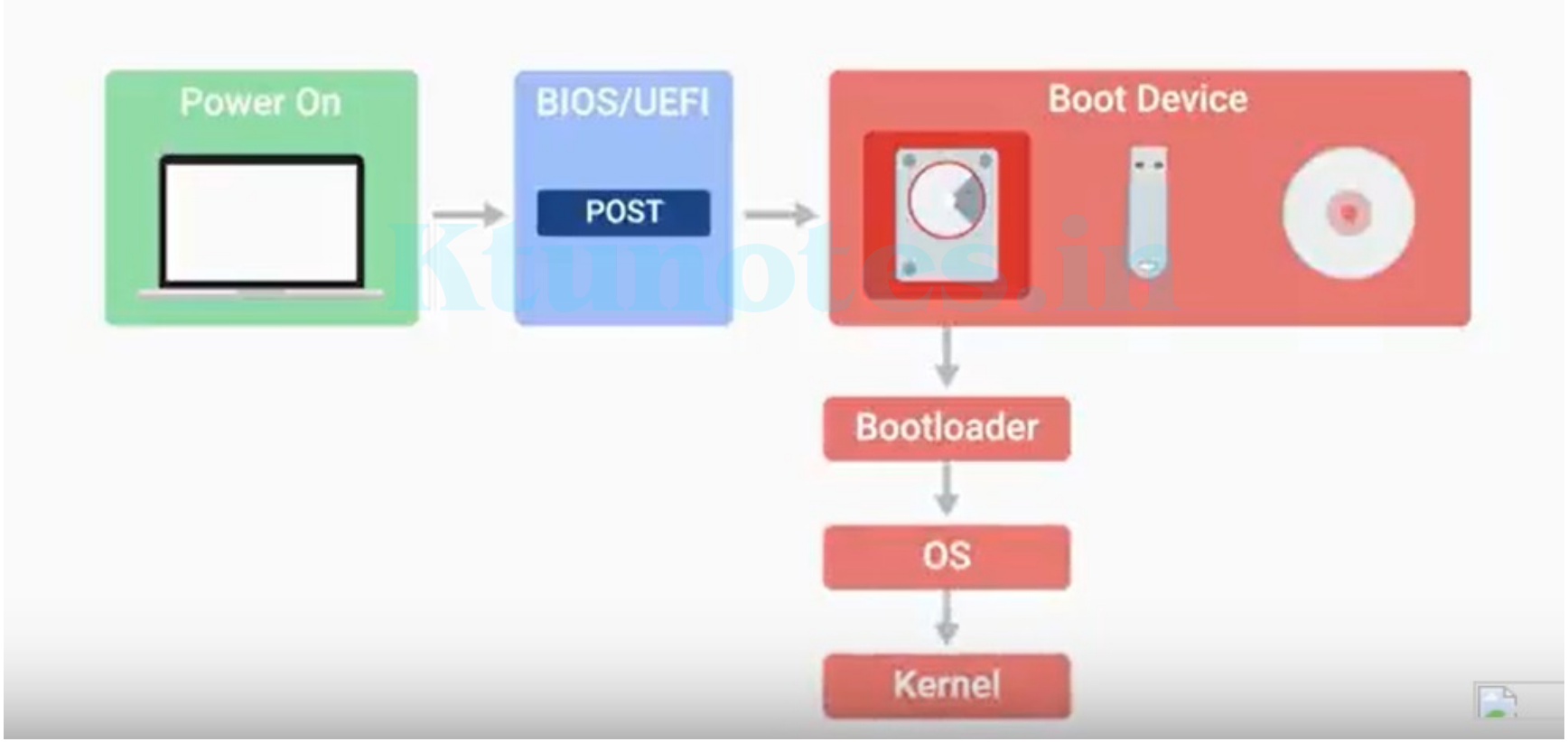
System Boot

- The procedure of starting a computer by loading the kernel is known as **booting** the system.
- On most computer systems, a small piece of code known as the **bootstrap program** or bootstrap loader locates the kernel, loads it into main memory, and starts its execution
- . This program is in the form of read-only memory (ROM), because the RAM is in an unknown state at system startup.
- All forms of ROM are also known as firmware

System Boot Cont..

- When power initialized on system, execution starts at a fixed memory location
  - Firmware ROM used to hold initial boot code
- Operating system must be made available to hardware so hardware can start it
  - Small piece of code – **bootstrap loader**, stored in **ROM** or **EPROM** locates the kernel, loads it into memory, and starts it
  - Sometimes two-step process where **boot block** at fixed location loaded by ROM code, which loads bootstrap loader from disk
- Common bootstrap loader, **GRUB(** is an example of an open-source bootstrap program for Linux systems) allows selection of kernel from multiple disks, versions, kernel options
- Kernel loads and system is then **running**

Booting Process



- BIOS(Basic Input Output System) is very **first step** when you press power button on your CPU. BIOS is a **small firmware** stored in ROM of mother board which is **manufacturer dependent code**. The function of BIOS is, it initializes some of the hardware on computer, checks integrity and finally it initializes first level boot loader .
- Boot loader is a program which is called by BIOS and once again **initializes boot related hardware** and finally boot loader is the one who exactly Knows(memory location) where the **Kernel image is stored in secondary memory**. It loads Kernel image from secondary storage to RAM.