Introduction to Operating Systems

Contents

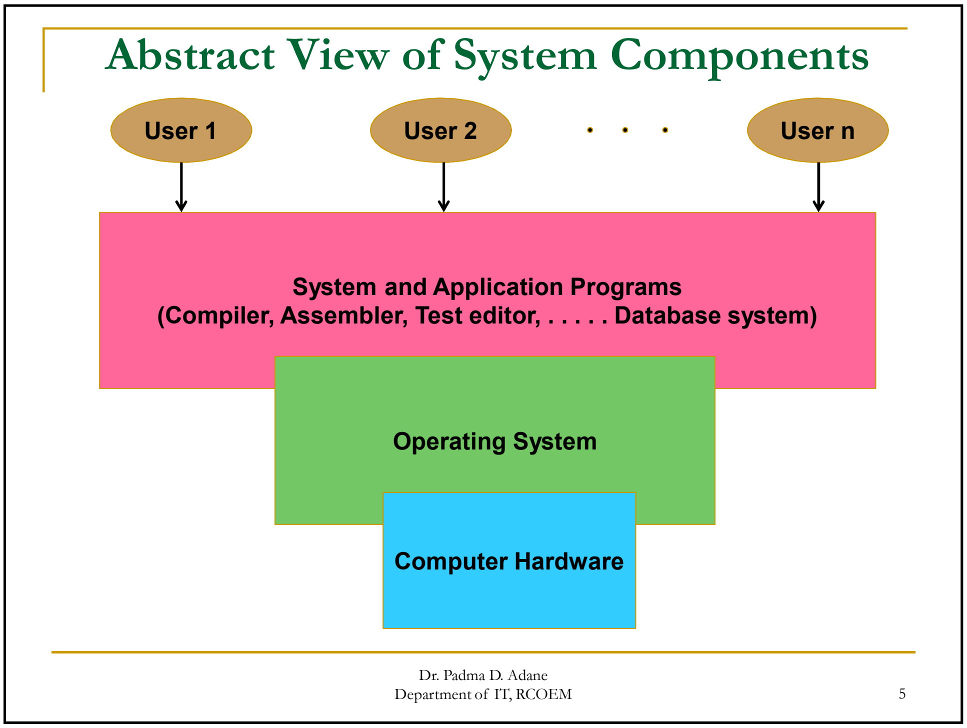
- Introduction
- System Components
- Services
- System programs and System Calls
- Evolution

Introduction

- A program that acts as an intermediary or interface between a user of a computer and the computer hardware
- Goals of Operating System
  - Provide an environment for execution of user programs
  - Make the computer system convenient to use
  - Use the computer hardware in a efficient manner

Introduction

- Operating System is also known as:
  - Resource allocator – manages and allocates resources.
  - Control program – controls the execution of user programs and operations of I/O devices .
  - Kernel – the one program running at all times (all else being application programs).



Abstract View of System Components

User 1

User 2

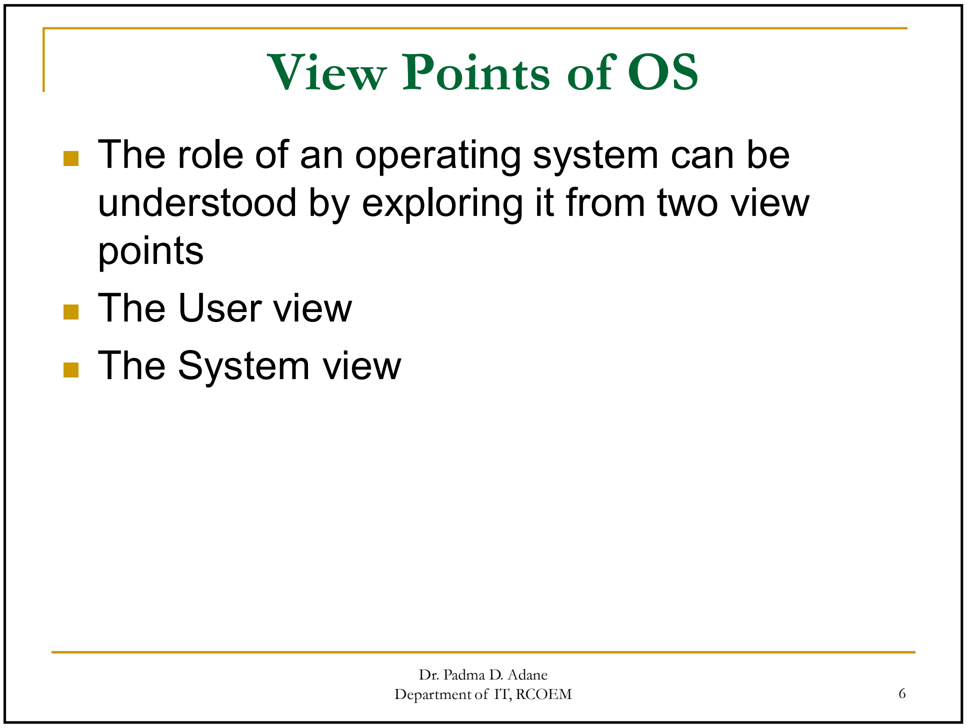
. . .

User n

System and Application Programs
(Compiler, Assembler, Test editor, ... Database system)

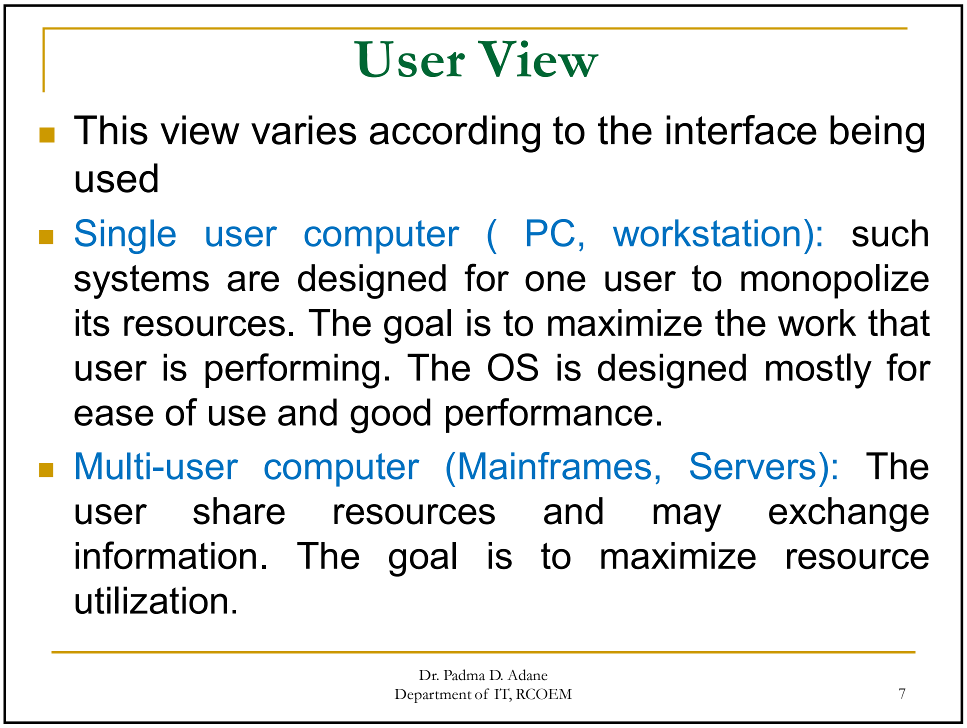
Operating System

Computer Hardware



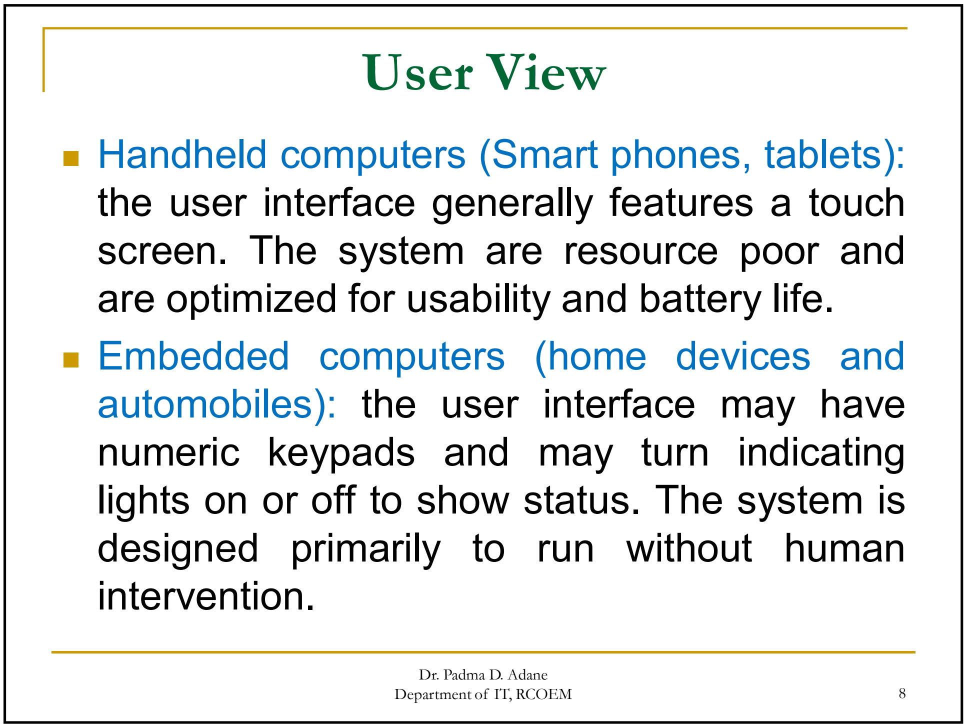
View Points of OS

- The role of an operating system can be understood by exploring it from two view points
- The User view
- The System view



User View

- This view varies according to the interface being used
- Single user computer (PC, workstation): such systems are designed for one user to monopolize its resources. The goal is to maximize the work that user is performing. The OS is designed mostly for ease of use and good performance.
- Multi-user computer (Mainframes, Servers): The user share resources and may exchange information. The goal is to maximize resource utilization.



User View

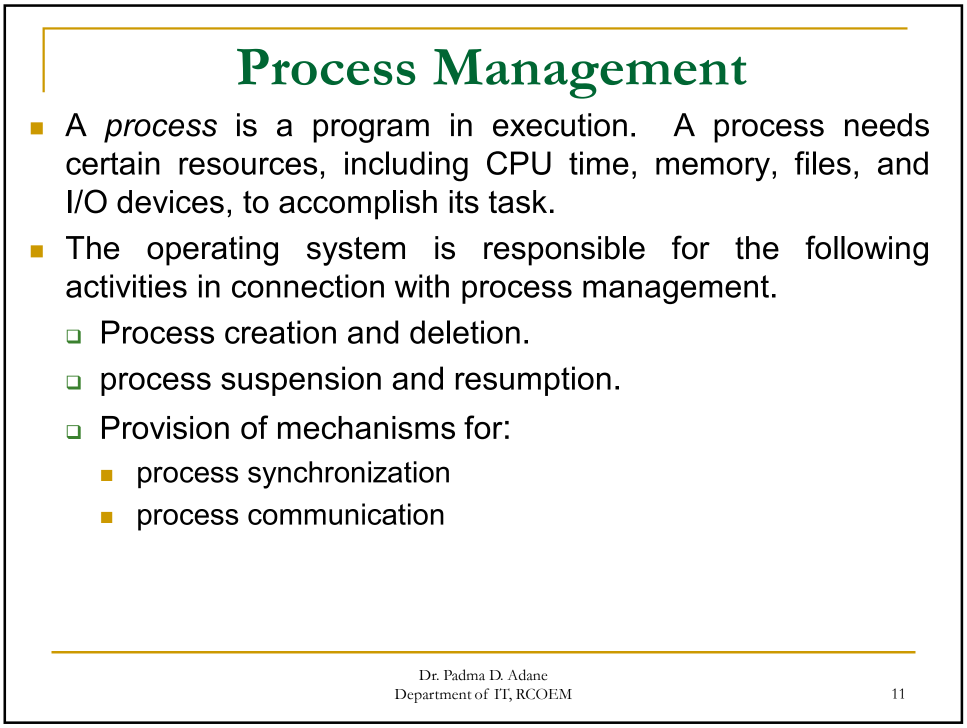
- Handheld computers (Smart phones, tablets): the user interface generally features a touch screen. The system are resource poor and are optimized for usability and battery life.
- Embedded computers (home devices and automobiles): the user interface may have numeric keypads and may turn indicating lights on or off to show status. The system is designed primarily to run without human intervention.

System View

- The system is more intimately involved with the hardware.
- Resource allocator:
  - Manages all resources
  - Resolves conflicting requests for efficient and fair resource utilization.
- Control program:
  - Controls execution of programs to prevent errors and improper use of the computer

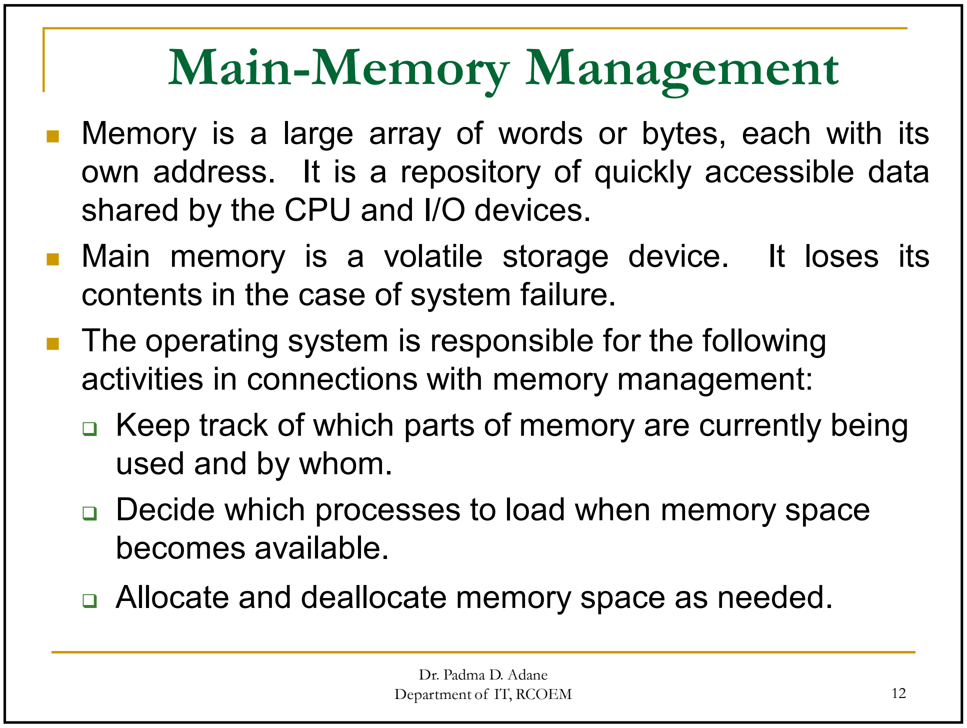
Common Operating System Components

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



Process Management

- A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - process suspension and resumption.
  - Provision of mechanisms for:
    - process synchronization
    - process communication



Main-Memory Management

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

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connections with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.

I/O System Management

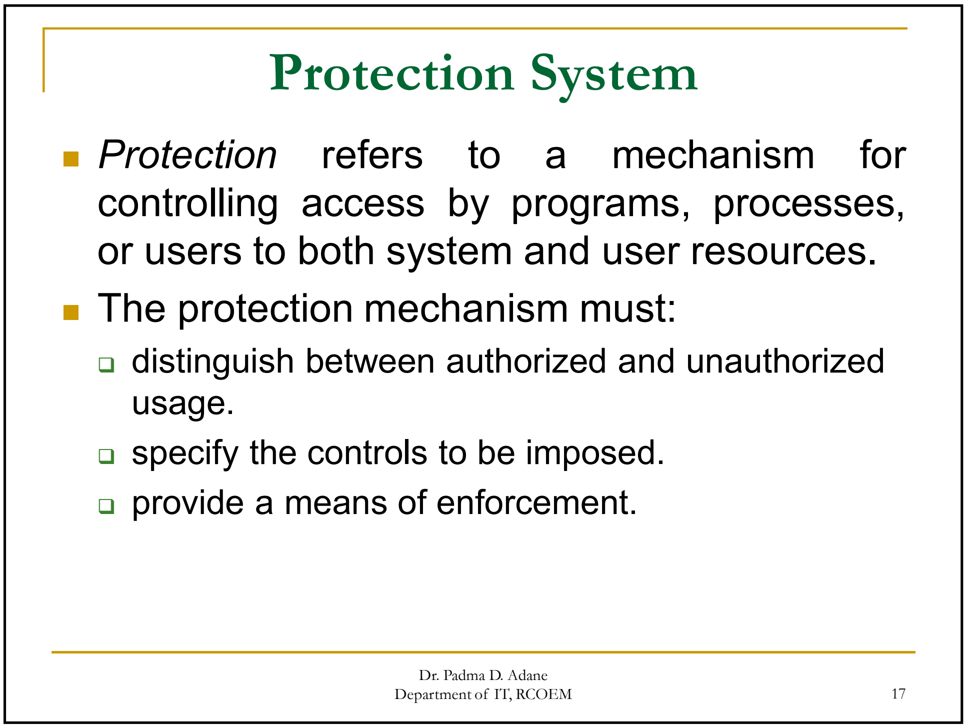
- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

Secondary-Storage Management

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

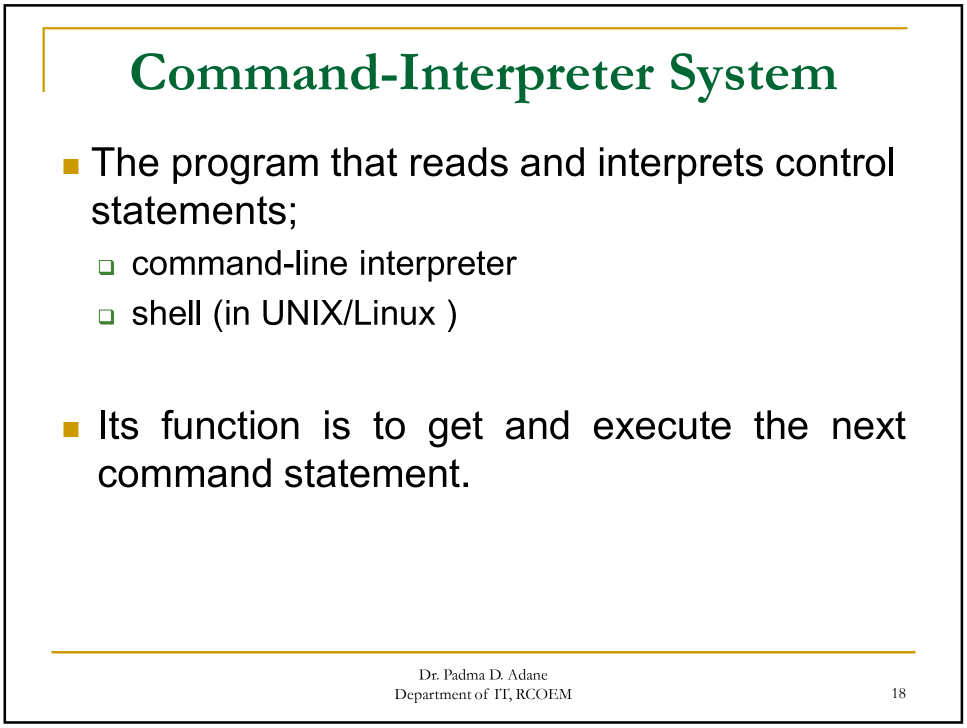
Networking (Distributed Systems)

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



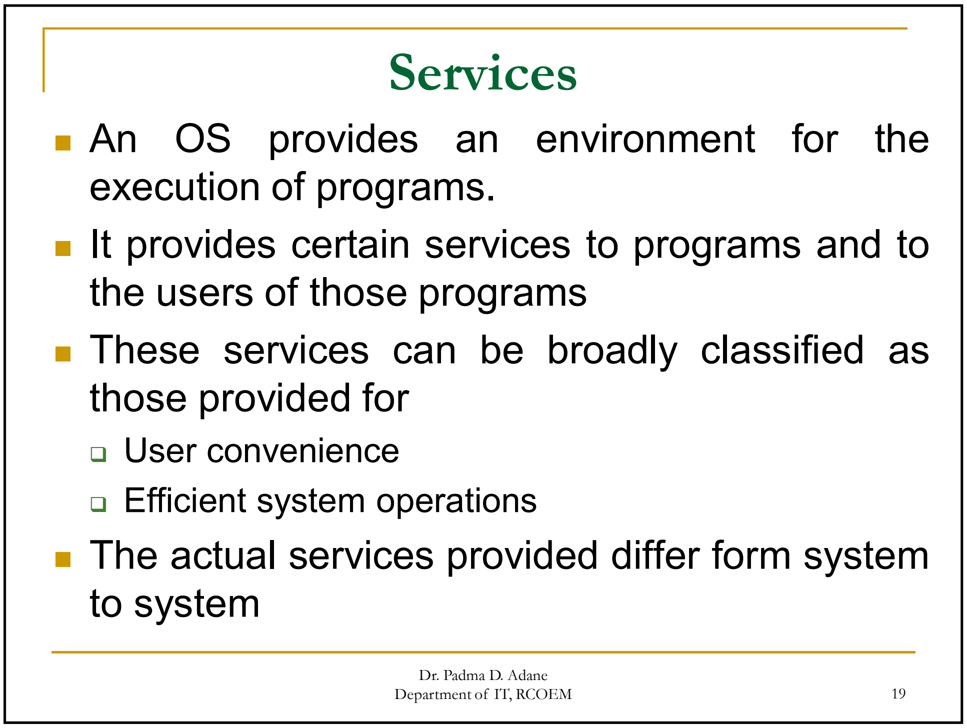
Protection System

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



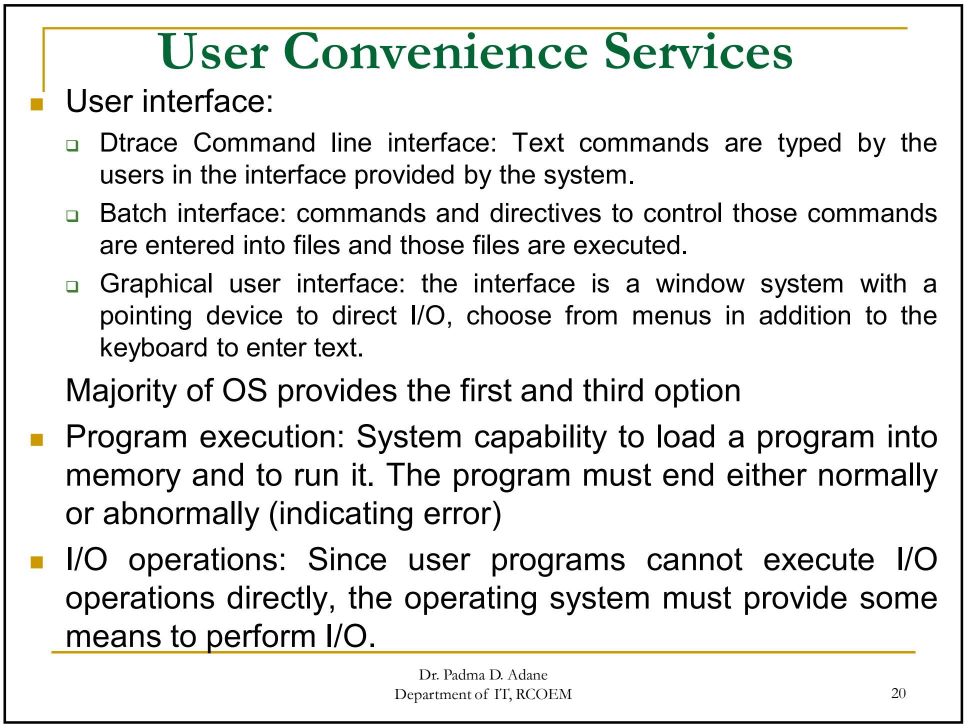
Command-Interpreter System

- The program that reads and interprets control statements;
  - command-line interpreter
  - shell (in UNIX/Linux)
- Its function is to get and execute the next command statement.



Services

- An OS provides an environment for the execution of programs.
- It provides certain services to programs and to the users of those programs
- These services can be broadly classified as those provided for
  - User convenience
  - Efficient system operations
- The actual services provided differ form system to system



User Convenience Services

- User interface:
  - Dtrace Command line interface: Text commands are typed by the users in the interface provided by the system.
  - Batch interface: commands and directives to control those commands are entered into files and those files are executed.
  - Graphical user interface: the interface is a window system with a pointing device to direct I/O, choose from menus in addition to the keyboard to enter text.

  Majority of OS provides the first and third option
- Program execution: System capability to load a program into memory and to run it. The program must end either normally or abnormally (indicating error)
- I/O operations: Since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.

User Convenience Services

- File system manipulation: Capability to read, write, create, delete and access files. Few systems supports more than one file system.
- Communication: Exchange of information between processes executing either on the same computer or on different systems tied together by a network. Implemented via shared memory or message passing.
- Error detection: Ensure appropriate actions in case of detecting errors in the CPU and memory hardware, in I/O devices, or in user programs. Debugging facilities can greatly enhance system efficiency.

Services for Efficient system operations

- Resource allocation – Allocating resources to multiple users or multiple jobs running at the same time. Typical resources include CPU cycles, Main memory space, File storage, I/O devices).
- Accounting – Keeping track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
- Protection – Ensuring that all access to system resources is controlled

System Calls

- System calls provide the interface between a running program and the operating system.
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, C++)

System Calls: An example

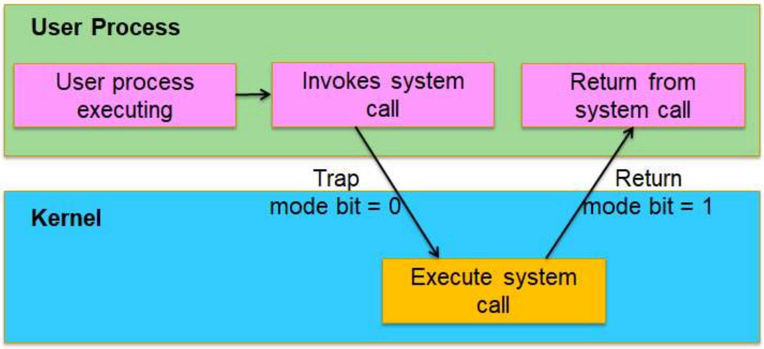
- A program to read data from one file and copy to another file
- Acquiring file name
  - Command line arguments
  - Prompting user to input file name
- Opening the Input file
  - File may not exists
  - File may not have read permission
- Opening the output file
  - File may already exists
    - Abort
    - Overwrite
    - delete
- Read from input file
- Write to the output file
- Close both files
- Terminate the program

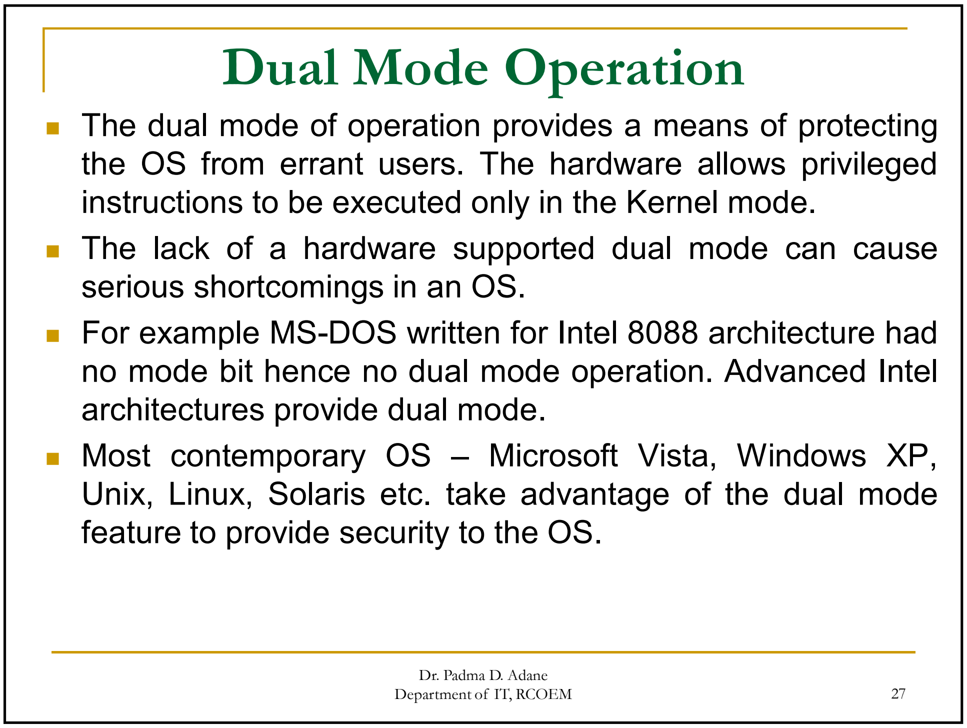
Dual Mode Operation

- The OS operates in two modes: user mode and the kernel mode (supervisor/ system/ privileged mode)
- Most system provides a hardware support to differentiate between the two modes. A bit, called the mode bit, is added to the hardware of the computer to indicate the current mode; kernel (0), user (1).
- At system boot time, it starts in kernel mode. The OS is then loaded and starts user applications in user mode.

Dual Mode Operation

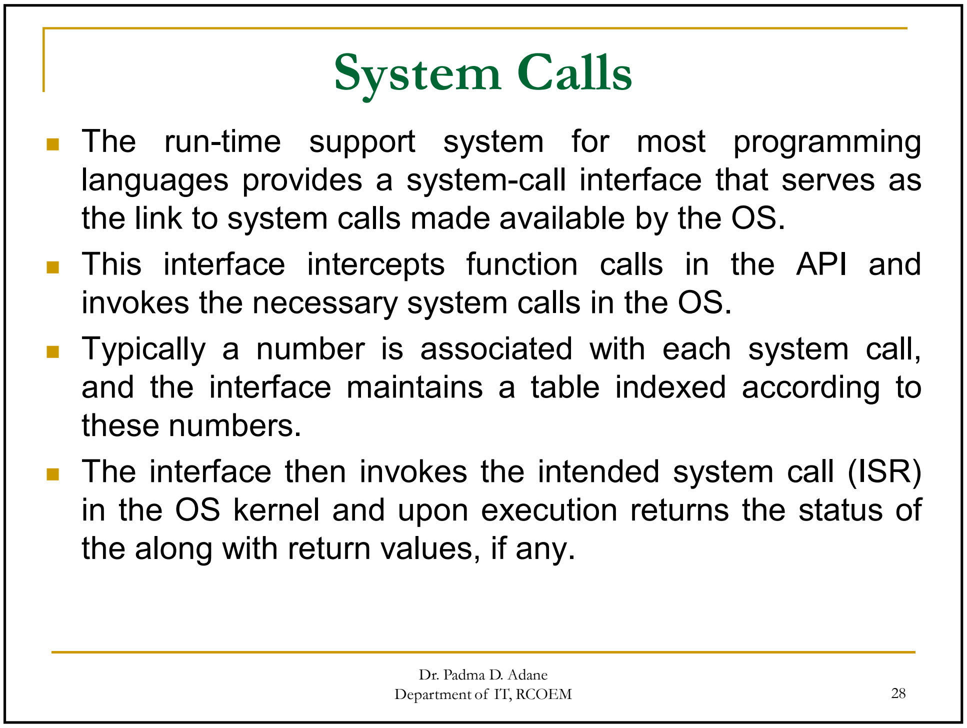
- Almost every event, whether it is a specific request from a user program or an error/exception, results in a trap, a software generated interrupt, and the system switches the mode.





Dual Mode Operation

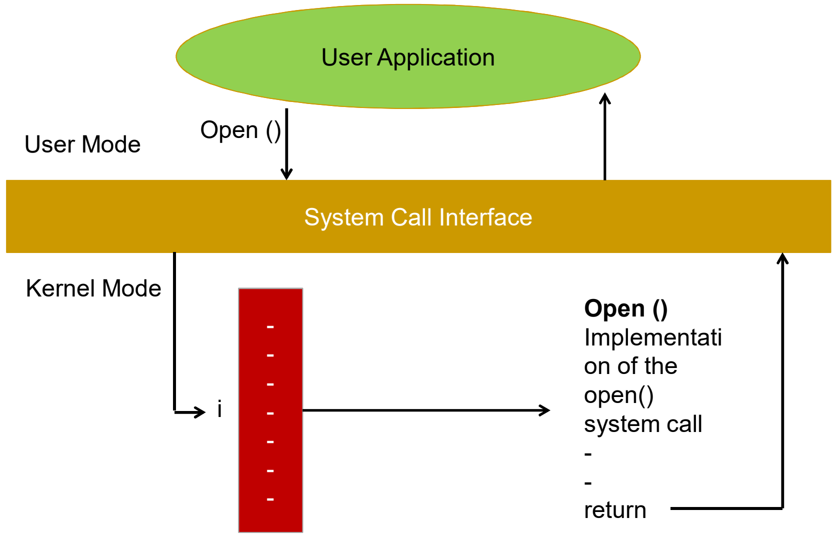
- The dual mode of operation provides a means of protecting the OS from errant users. The hardware allows privileged instructions to be executed only in the Kernel mode.
- The lack of a hardware supported dual mode can cause serious shortcomings in an OS.
- For example MS-DOS written for Intel 8088 architecture had no mode bit hence no dual mode operation. Advanced Intel architectures provide dual mode.
- Most contemporary OS – Microsoft Vista, Windows XP, Unix, Linux, Solaris etc. take advantage of the dual mode feature to provide security to the OS.



System Calls

- The run-time support system for most programming languages provides a system-call interface that serves as the link to system calls made available by the OS.
- This interface intercepts function calls in the API and invokes the necessary system calls in the OS.
- Typically a number is associated with each system call, and the interface maintains a table indexed according to these numbers.
- The interface then invokes the intended system call (ISR) in the OS kernel and upon execution returns the status of the along with return values, if any.

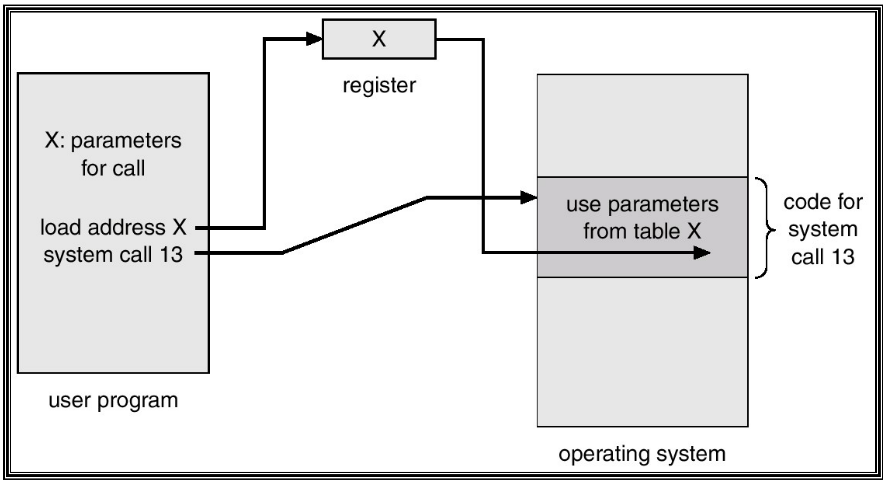
Handling Invocation of Open() System call



System Calls

- Invoking a system call may require certain parameter to be passed to it.
- Three general methods are used to pass parameters between a running program and the operating system
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register
  - Push (store) the parameters onto the *stack* by the program, and *pop* off the stack by operating system

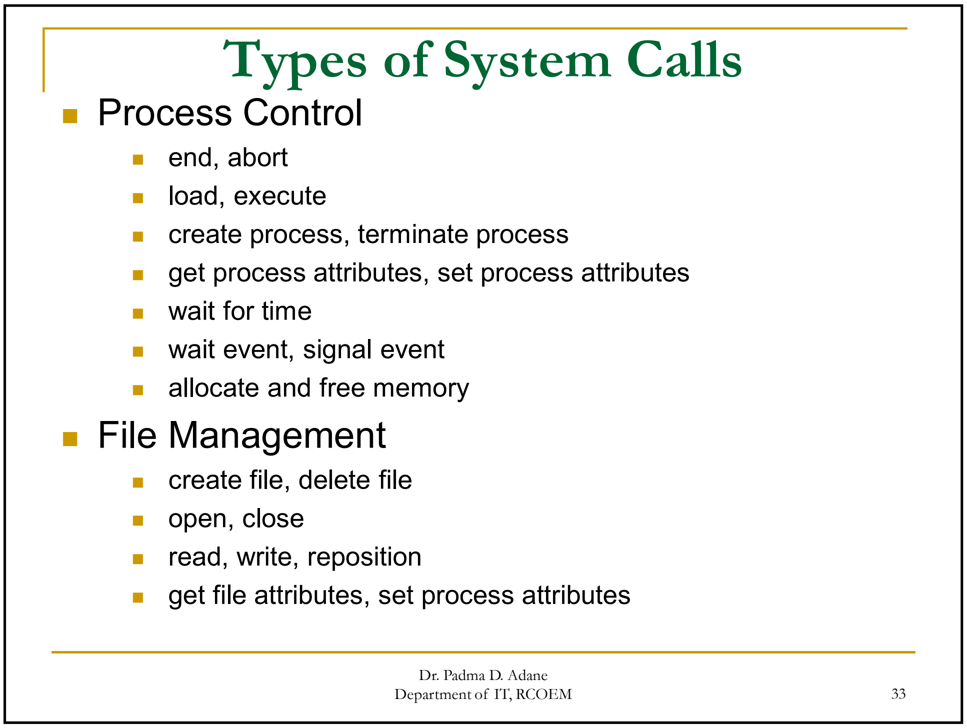
System Calls



Passing of Parameters As A Table

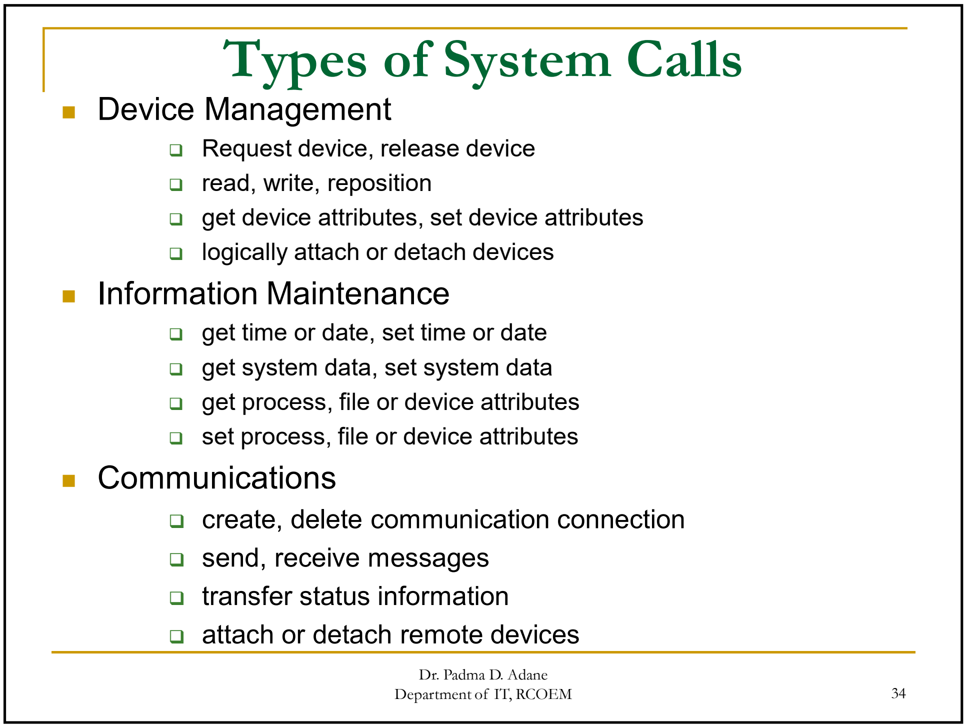
Types of System Calls

- System calls can be grouped roughly into six major groups:
  - Process Control
  - File Management
  - Device Management
  - Information Maintenance
  - Communications
  - Protection



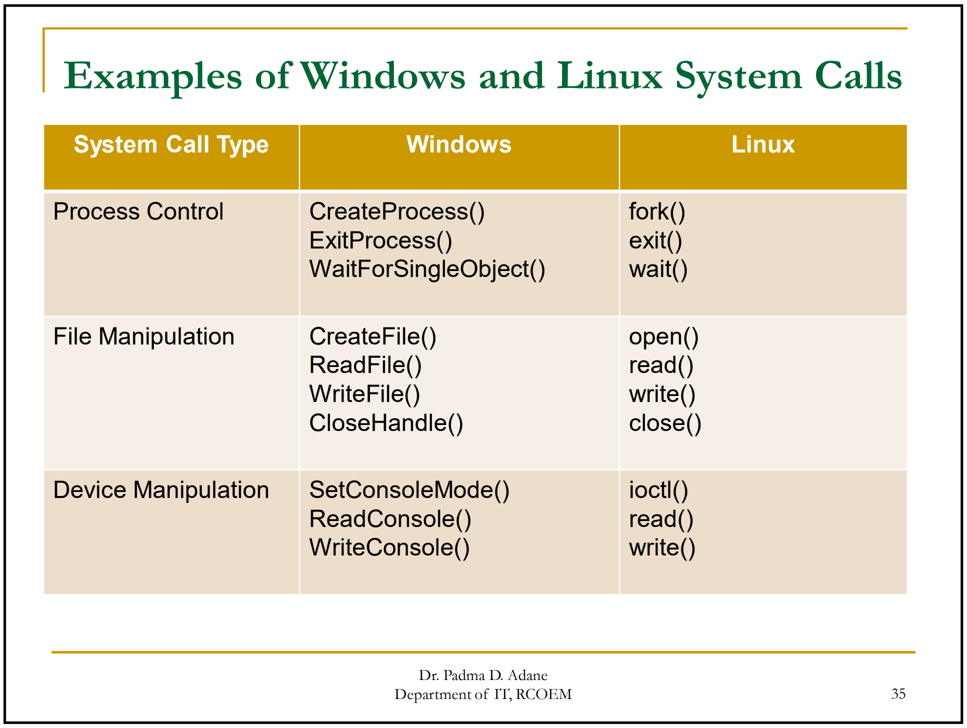
Types of System Calls

- 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
  - read, write, reposition
  - get file attributes, set process 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 process, file or device attributes
  - ❑ set process, file or device attributes
- Communications
  - ❑ create, delete communication connection
  - ❑ send, receive messages
  - ❑ transfer status information
  - ❑ attach or detach remote devices



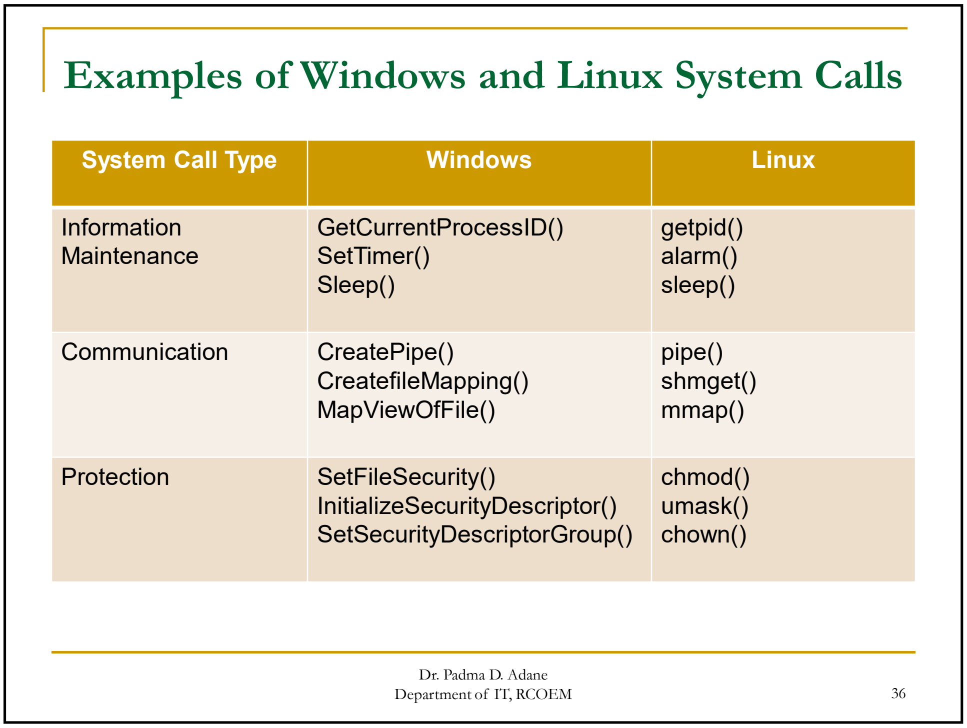
Examples of Windows and Linux System Calls

System Call Type	Windows	Linux

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()



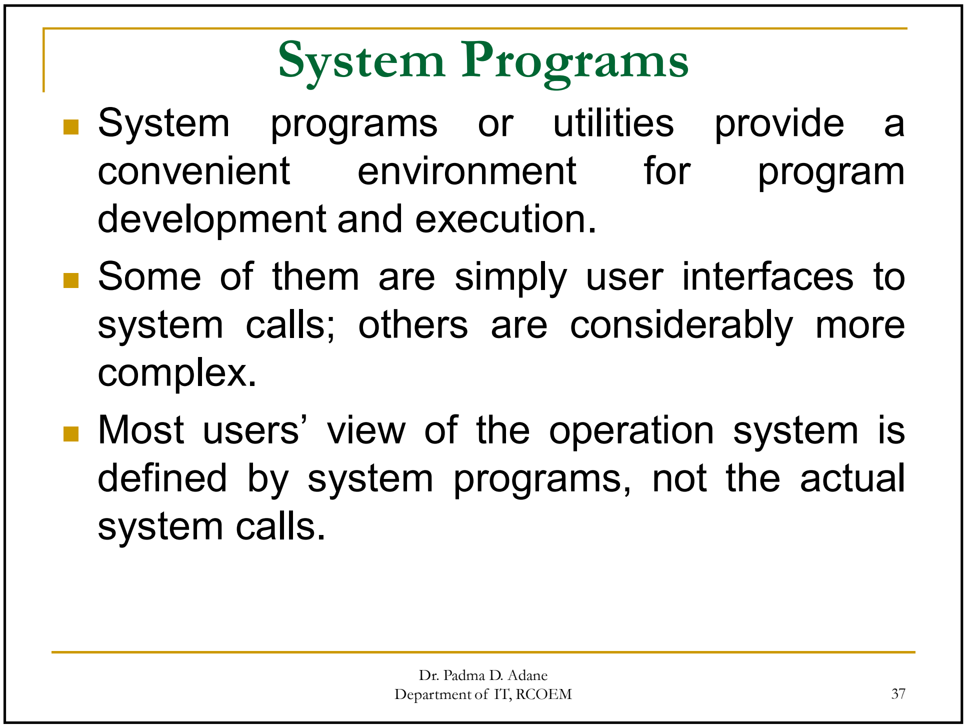
Examples of Windows and Linux System Calls

System Call Type	Windows	Linux

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		getpid()
		alarm()
		sleep()

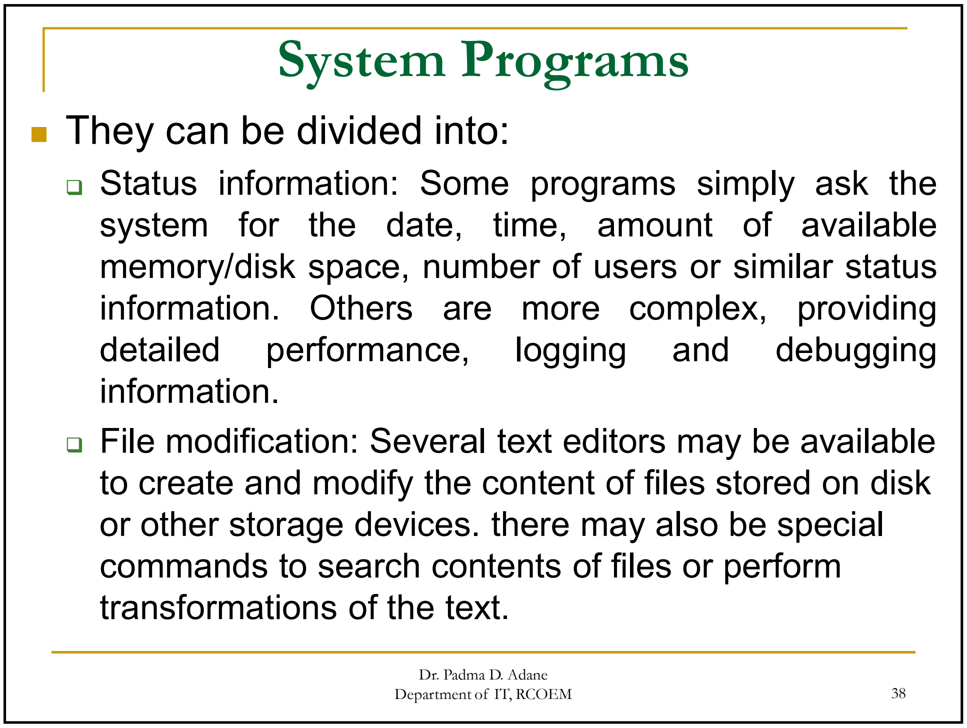
Communication	CreatePipe()
	CreatefileMapping()
	MapViewOfFile()
		pipe()
		shmget()
		mmap()

Protection	SetFileSecurity()
	InitializeSecurityDescriptor()
	SetSecurityDescriptorGroup()
		chmod()
		umask()
		chown()



System Programs

- System programs or utilities provide a convenient environment for program development and execution.
- Some of them are simply user interfaces to system calls; others are considerably more complex.
- Most users’ view of the operation system is defined by system programs, not the actual system calls.



System Programs

- They can be divided into:
  - Status information: Some programs simply ask the system for the date, time, amount of available memory/disk space, number of users or similar status information. Others are more complex, providing detailed performance, logging and debugging information.
  - File modification: Several text editors may be available to create and modify the content of files stored on disk or other storage devices. there may also be special commands to search contents of files or perform transformations of the text.

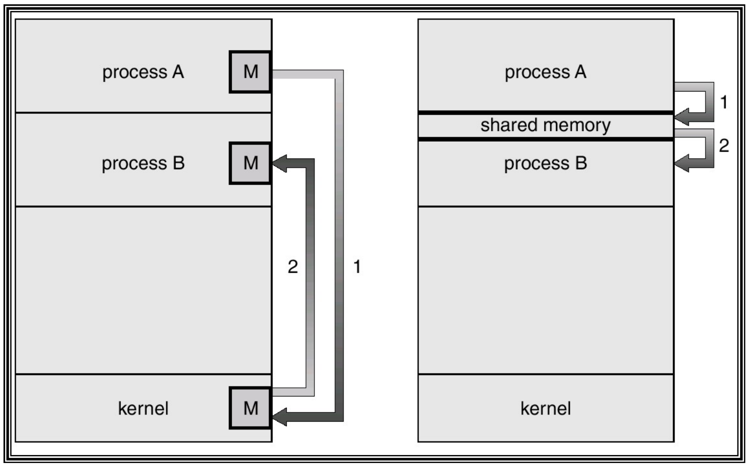
System Programs

- File manipulation: These programs create, delete, copy, rename, print, dump, list files and directories.
- Programming language support: Compilers, assemblers, debuggers and interpreters for common programming languages are often provided with the OS
- 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 loaders, relocatable loaders, linkage editors etc. Debugging systems for either higher-level languages or machine language are needed as well.

System Programs

- Communications: These programs provide the mechanism for creating virtual connections among processes, users and computer systems
- In addition to system programs, most OS are supplied with programs that are useful in performing common operations; web browsers, word processors, spreadsheets, database systems, games.

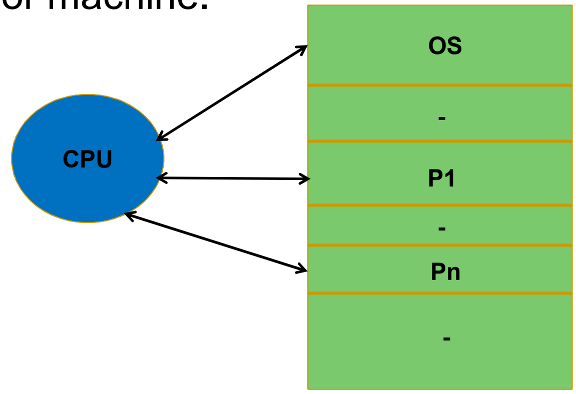
Communication Models

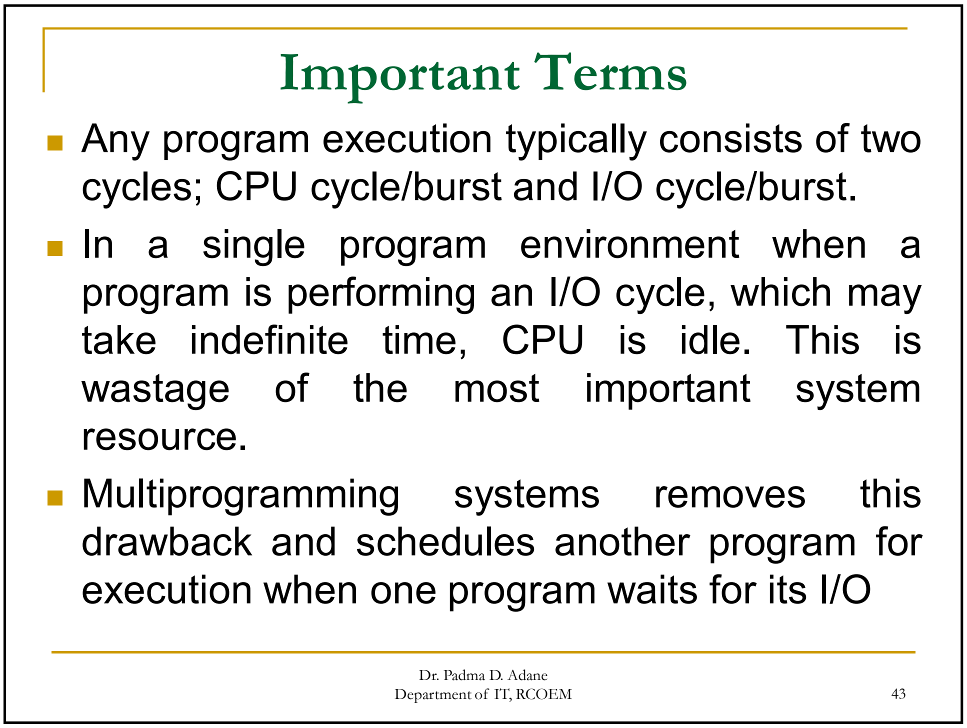


Message Sharing **Shared Memory**

Important Terms

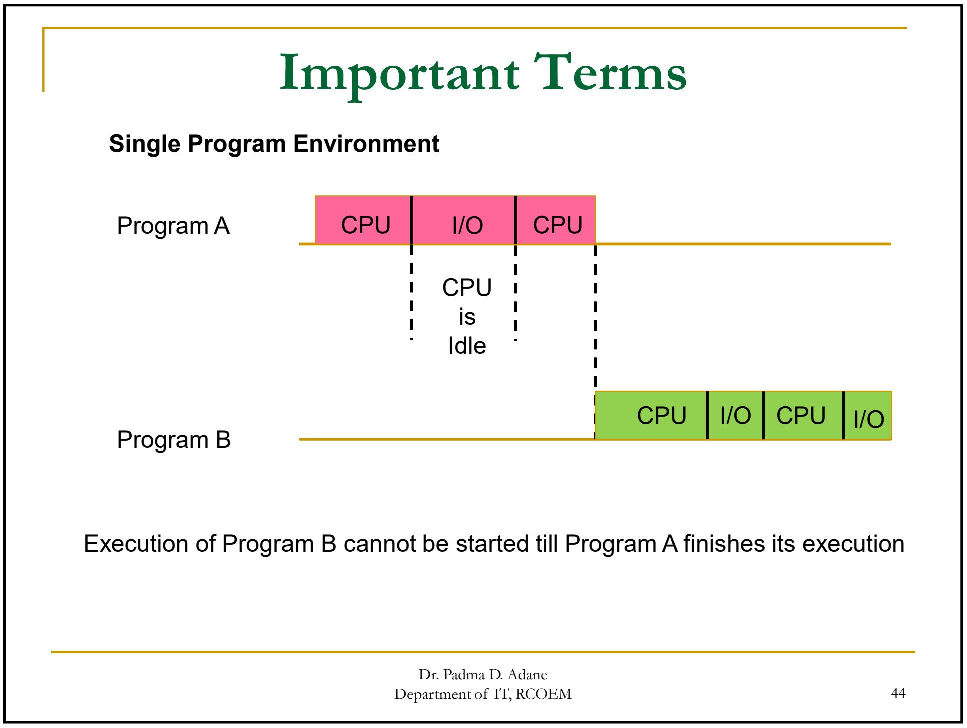
- **Multi-programming:** The ability of an operating system to execute more than one program on a single processor machine.





Important Terms

- Any program execution typically consists of two cycles; CPU cycle/burst and I/O cycle/burst.
- In a single program environment when a program is performing an I/O cycle, which may take indefinite time, CPU is idle. This is wastage of the most important system resource.
- Multiprogramming systems removes this drawback and schedules another program for execution when one program waits for its I/O



Important Terms

Single Program Environment

Program A: CPU | I/O | CPU

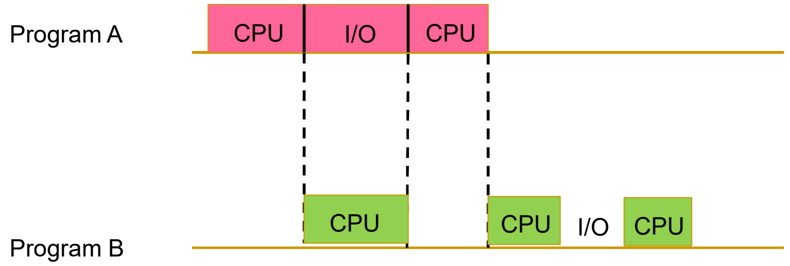
CPU is Idle

Program B: CPU | I/O | CPU | I/O

Execution of Program B cannot be started till Program A finishes its execution

Important Terms

Multi-Programming Environment



Execution of Program B can be started during the I/O time of Program A. Thus, Overall throughput of the system increases and waiting time reduces.

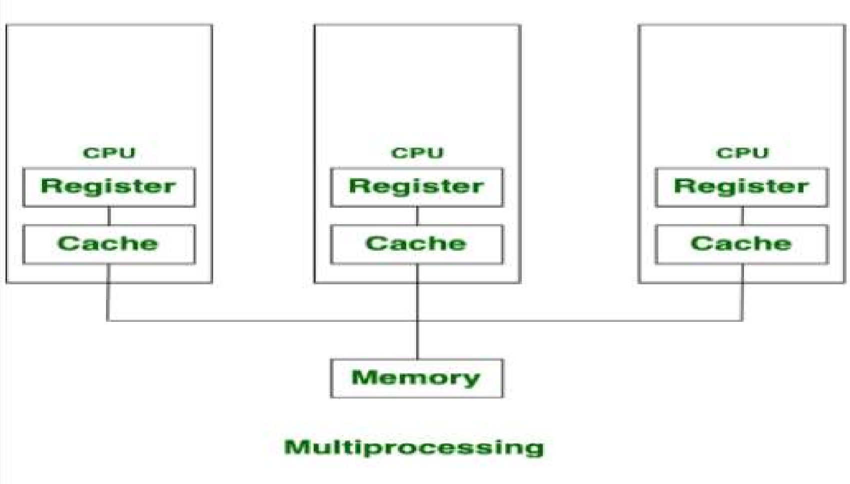
Important Terms

- **Multitasking:** Multitasking is a logical extension of multi programming. It refers to execution of multiple tasks (say processes, programs, threads etc.) at a time. In the modern operating systems, we are able to play MP3 music, edit documents in Microsoft Word, surf the Google Chrome all simultaneously, this is accomplished by means of multi tasking. The major way in which multitasking differs from multi programming is that multi programming works solely on the concept of context switching whereas multitasking is based on time sharing alongside the concept of context switching.

Important Terms

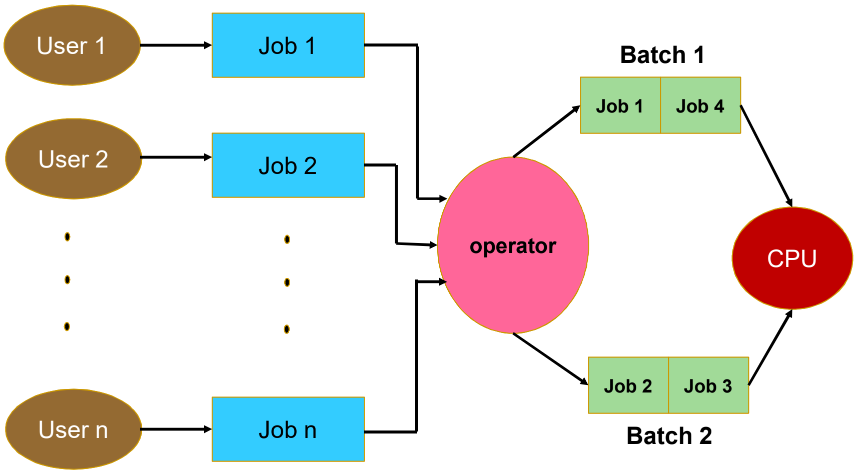
- **Multiprocessing:** It is the use of two or more CPUs (processors) within a single Computer system. Since there are multiple processors available, multiple processes can be executed at a time. These multi processors share the computer bus, sometimes the clock, memory and peripheral devices also. The main advantage of multiprocessor system is to get more work done in a shorter period of time i.e. higher throughput. It also provides increased reliability in the sense that if one processor fails, the work does not halt. It still works with slight reduction in throughput.

Important Terms

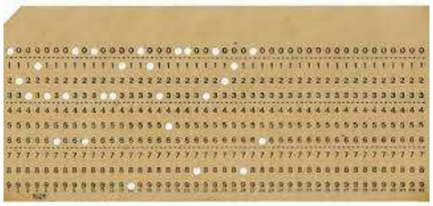


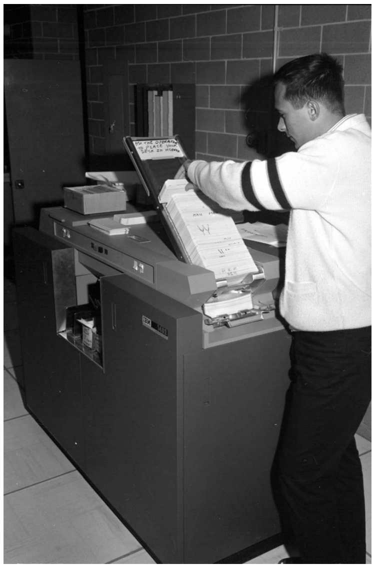
Types of Operating Systems

- **Batch Operating System:** The users of a 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. To speed up processing, jobs with similar needs are batched together and run as a group. The programmers leave their programs with the operator and the operator then sorts the programs with similar requirements into batches.



User 1
User 2
User n
Job 1
Job 2
Job n
operator
Batch 1
Job 1
Job 4
Batch 2
Job 2
Job 3
CPU





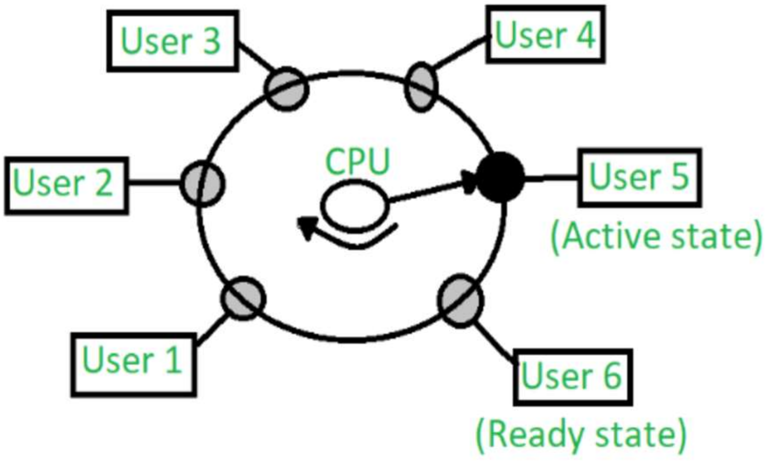
Types of Operating Systems

- **Advantages of Batch Operating System**
  - Multiple users can share the batch systems.
  - The idle time for the **batch system** is very less.
  - It is easy to manage large work repeatedly in batch systems.
- **Disadvantages of Batch Operating System**
  - Batch systems are hard to debug.
  - Lack of interaction between the user and the job.
  - Involvement of mechanical I/O devices slows down the overall process.
  - The other jobs will have to wait for an unknown time if any job fails.

Types of Operating Systems

Time Sharing Operating System:

- This is a type of multitasking system in which the processor's time is equally divided between all processes to be executed i.e. each of the n processes get $1/n^{th}$ of the CPU time.
- The CPU is allocated to processes in a First come First basis.
- If a process needs more time than the allotted time slot, it has to join the end of the queue and wait till next time slot is allotted to it.
- The objective of a batch OS is to maximize CPU throughput, whereas, the objective of a time sharing system to minimize the user response time.



Types of Operating Systems

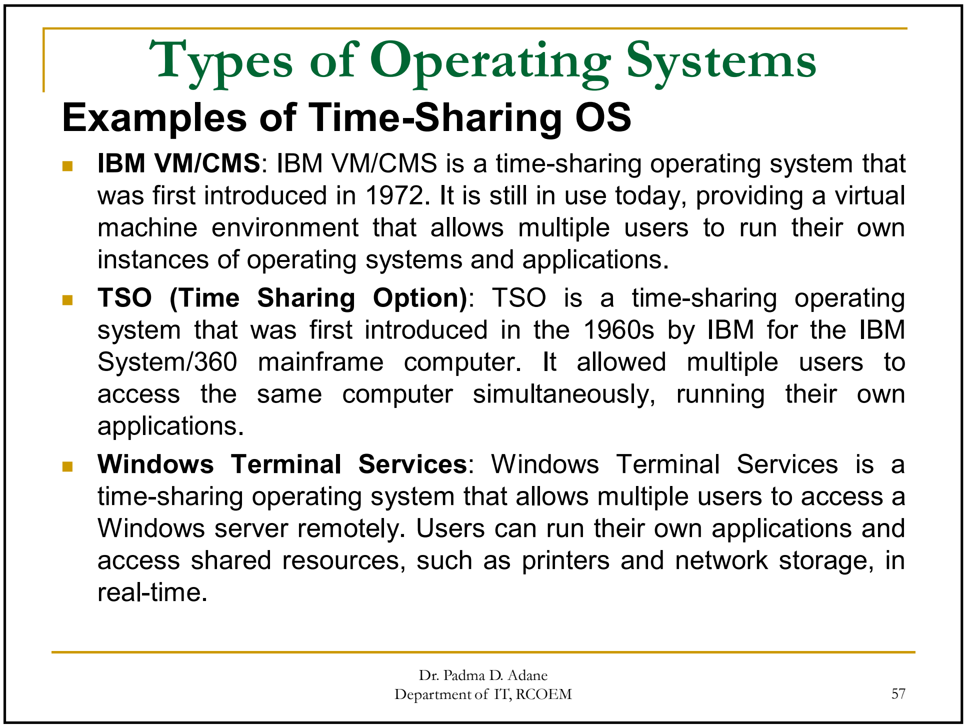
Advantages of Time Sharing Systems

- Each task gets an equal opportunity.
- CPU idle time can be reduced.
- Allow multiple users to share hardware resources such as the CPU, memory, and peripherals, reducing the cost of hardware and increasing efficiency.
- Allows users to work concurrently, thereby reducing the waiting time for their turn to use the computer. This increased productivity translates to more work getting done in less time.
- Provides an interactive environment that allows users to communicate with the computer in real time, providing a better user experience than batch processing.

Types of Operating Systems

Disadvantages of Time Sharing Systems

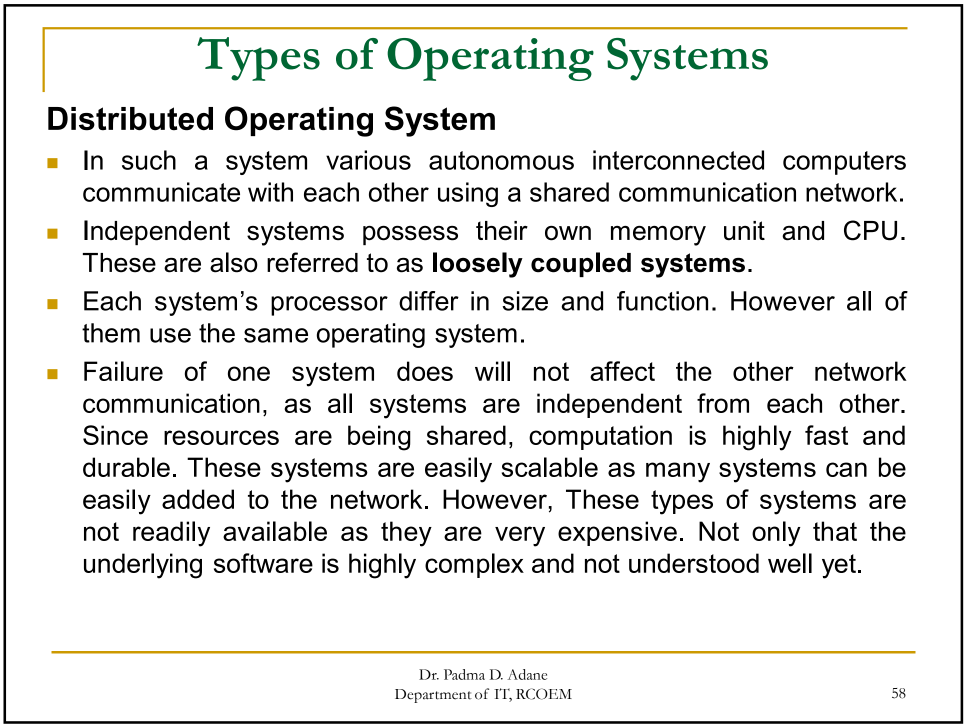
- Higher overhead than other operating systems due to the need for frequent context switching
- With multiple users sharing resources, the risk of security breaches increases. Time-sharing systems require careful management of user access, authentication, and authorization to ensure the security of data and software.
- More complex than other OS.



Types of Operating Systems

Examples of Time-Sharing OS

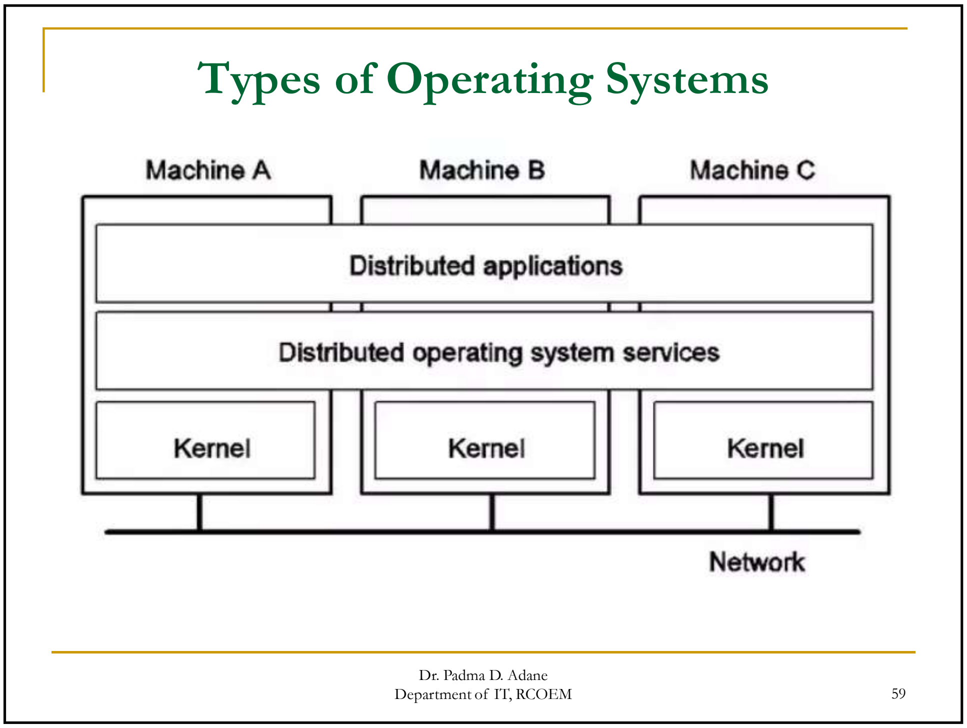
- **IBM VM/CMS**: IBM VM/CMS is a time-sharing operating system that was first introduced in 1972. It is still in use today, providing a virtual machine environment that allows multiple users to run their own instances of operating systems and applications.
- **TSO (Time Sharing Option)**: TSO is a time-sharing operating system that was first introduced in the 1960s by IBM for the IBM System/360 mainframe computer. It allowed multiple users to access the same computer simultaneously, running their own applications.
- **Windows Terminal Services**: Windows Terminal Services is a time-sharing operating system that allows multiple users to access a Windows server remotely. Users can run their own applications and access shared resources, such as printers and network storage, in real-time.



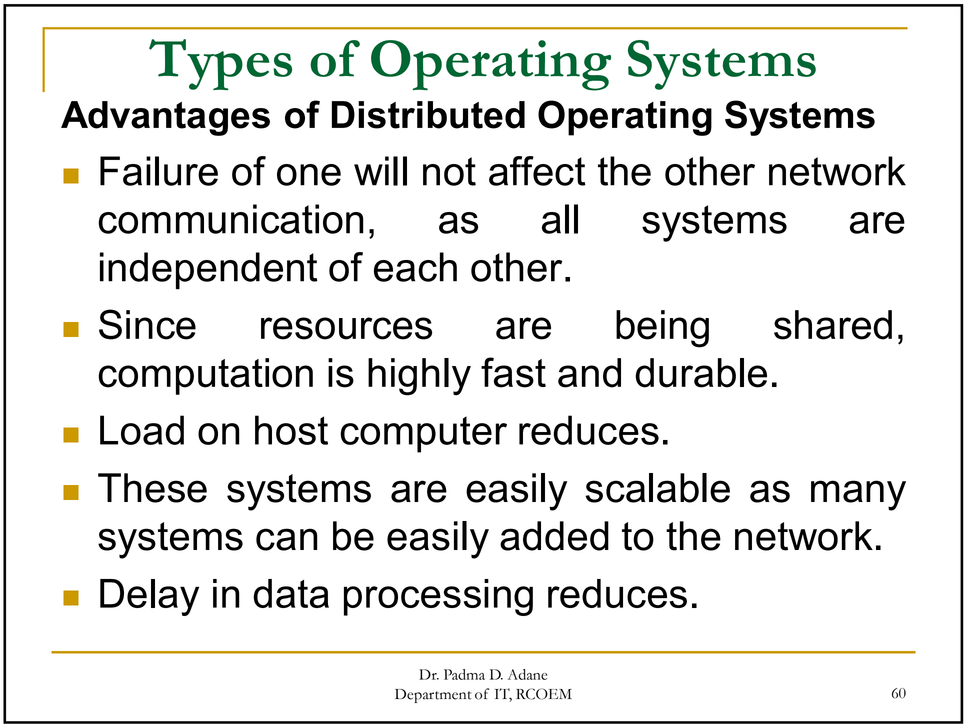
Types of Operating Systems

Distributed Operating System

- In such a system various autonomous interconnected computers communicate with each other using a shared communication network.
- Independent systems possess their own memory unit and CPU. These are also referred to as **loosely coupled systems**.
- Each system's processor differ in size and function. However all of them use the same operating system.
- Failure of one system does will not affect the other network communication, as all systems are independent from each other. Since resources are being shared, computation is highly fast and durable. These systems are easily scalable as many systems can be easily added to the network. However, These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet.



Types of Operating Systems



Types of Operating Systems

Advantages of Distributed Operating Systems

- Failure of one will not affect the other network communication, as all systems are independent of each other.
- Since resources are being shared, computation is highly fast and durable.
- Load on host computer reduces.
- These systems are easily scalable as many systems can be easily added to the network.
- Delay in data processing reduces.

Types of Operating Systems

Disadvantages of Distributed Operating Systems

- Failure of the main network will stop the entire communication.
- To establish distributed systems the language is used not well-defined yet.
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet.

Example of Distributed Operating Systems

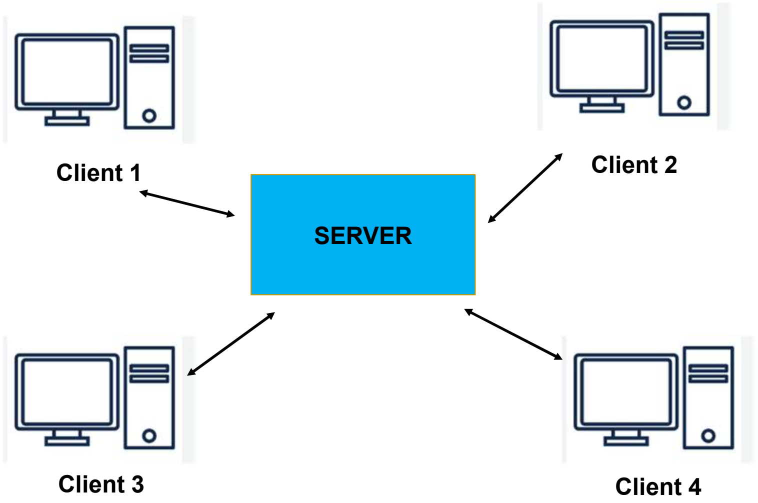
LOCUS

Types of Operating Systems

Network Operating system

- A Network OS runs on a server and provides the server the capability to manage data, users, groups, security, applications, and other networking functions.
- The primary purpose of the network operating system is to allow shared file and printer access among multiple computers in a network, typically a local area network (LAN), a private network or to other networks.
- In such a system, each node can have a different OS.
- All the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that's why these computers are popularly known as **tightly coupled systems.**

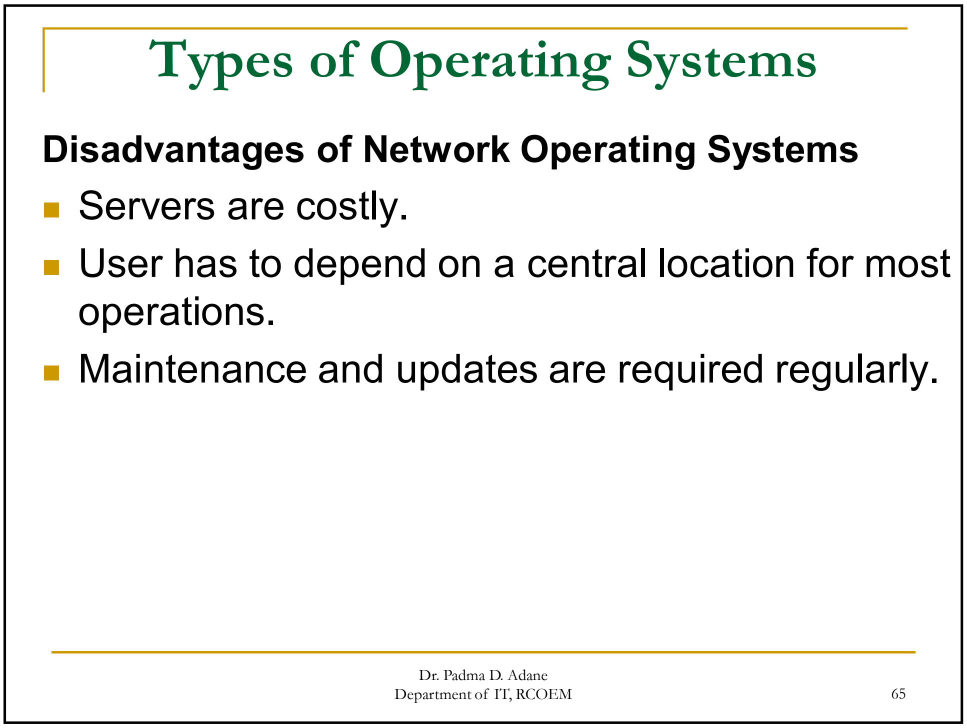
Types of Operating Systems



Types of Operating Systems

Advantages of Network Operating Systems

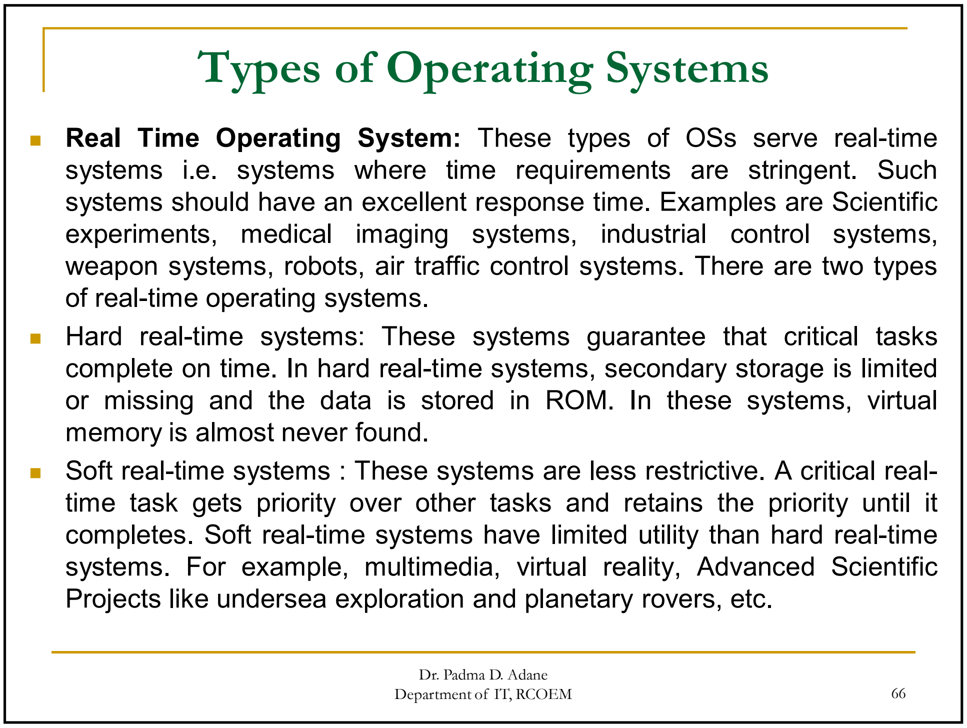
- Highly stable centralized servers.
- Security concerns are handled through servers.
- New technologies and hardware up-gradation are easily integrated into the system.
- Server access is possible remotely from different locations and types of systems.



Types of Operating Systems

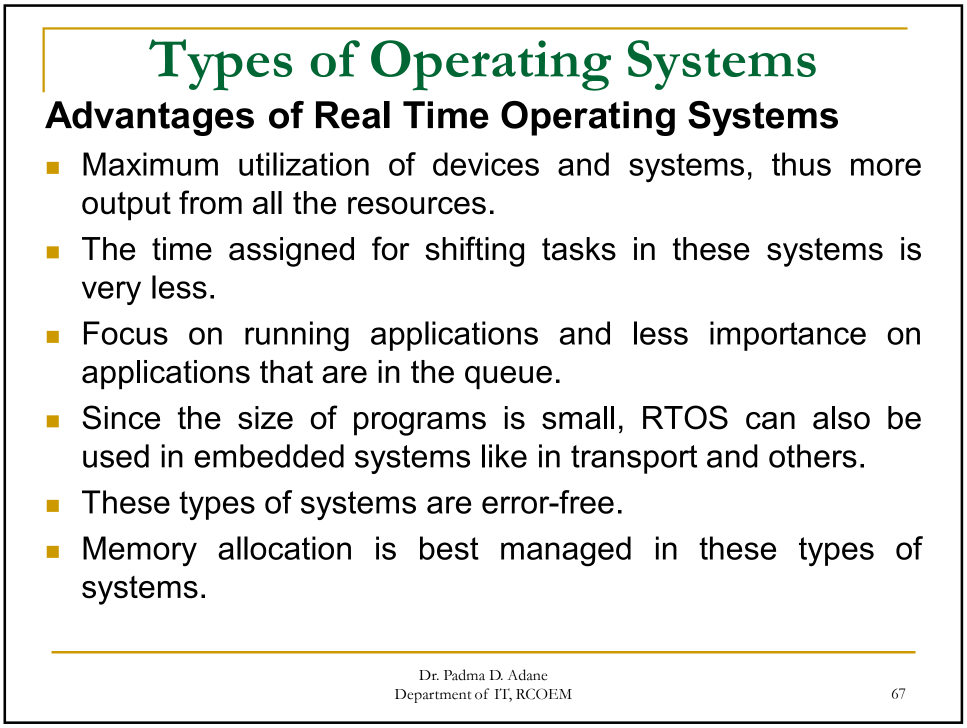
Disadvantages of Network Operating Systems

- Servers are costly.
- User has to depend on a central location for most operations.
- Maintenance and updates are required regularly.



Types of Operating Systems

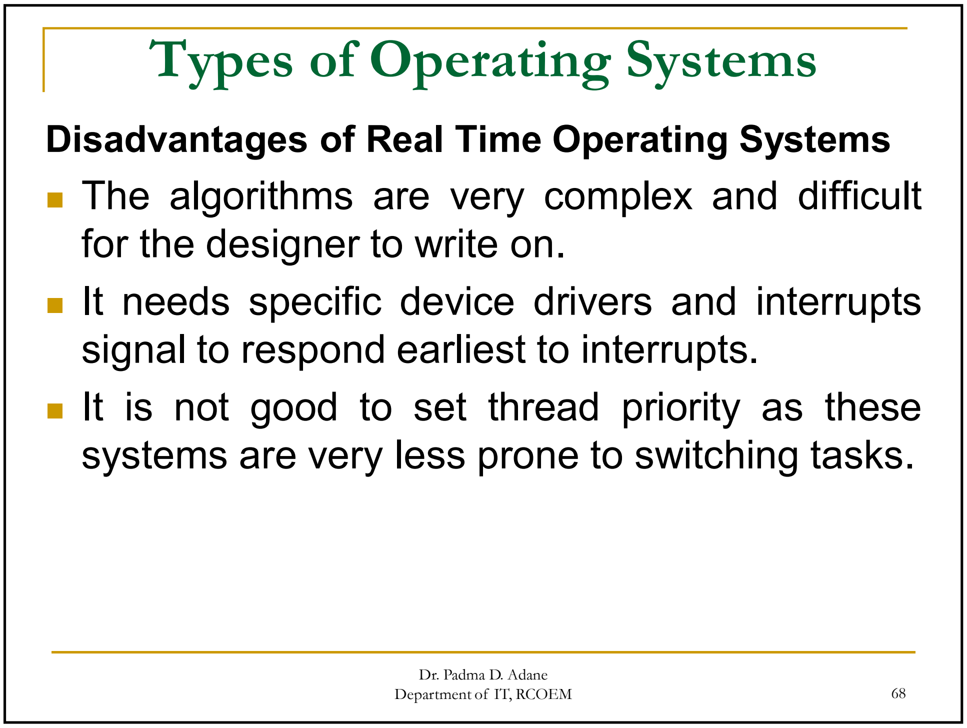
- **Real Time Operating System:** These types of OSs serve real-time systems i.e. systems where time requirements are stringent. Such systems should have an excellent response time. Examples are Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems. There are two types of real-time operating systems.
- Hard real-time systems: These systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.
- Soft real-time systems : These systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.



Types of Operating Systems

Advantages of Real Time Operating Systems

- Maximum utilization of devices and systems, thus more output from all the resources.
- The time assigned for shifting tasks in these systems is very less.
- Focus on running applications and less importance on applications that are in the queue.
- Since the size of programs is small, RTOS can also be used in embedded systems like in transport and others.
- These types of systems are error-free.
- Memory allocation is best managed in these types of systems.



Types of Operating Systems

Disadvantages of Real Time Operating Systems

- The algorithms are very complex and difficult for the designer to write on.
- It needs specific device drivers and interrupts signal to respond earliest to interrupts.
- It is not good to set thread priority as these systems are very less prone to switching tasks.