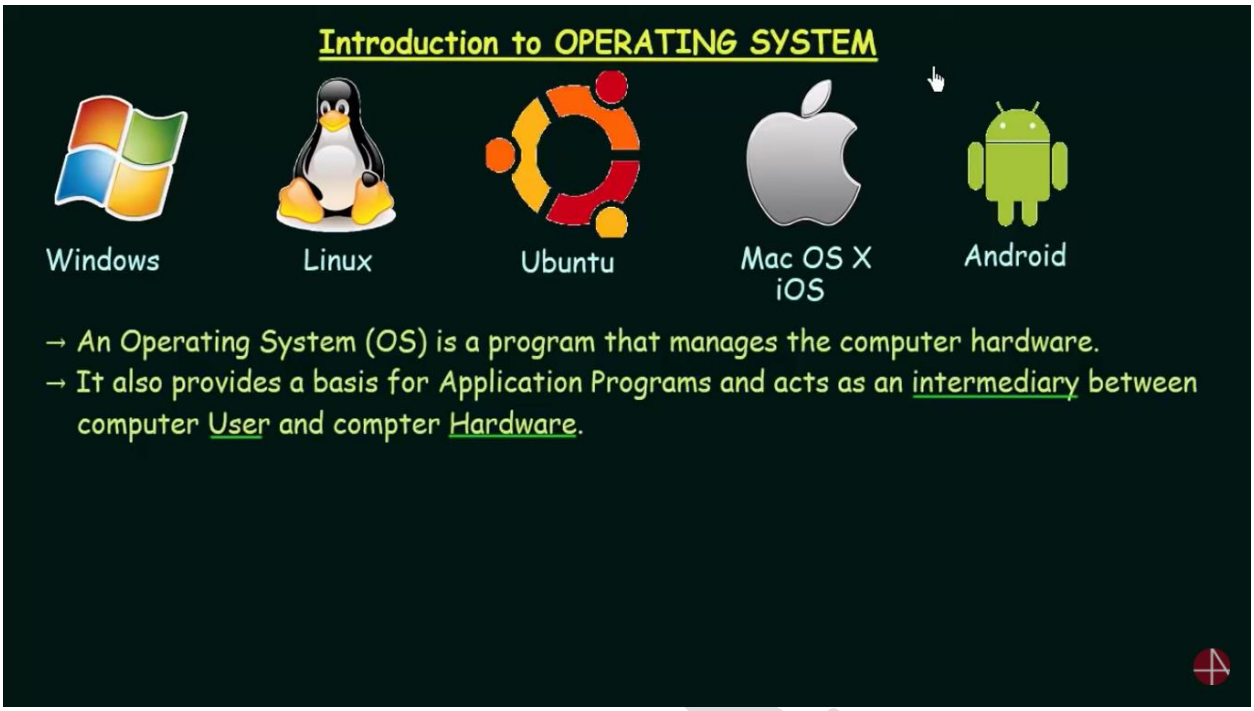
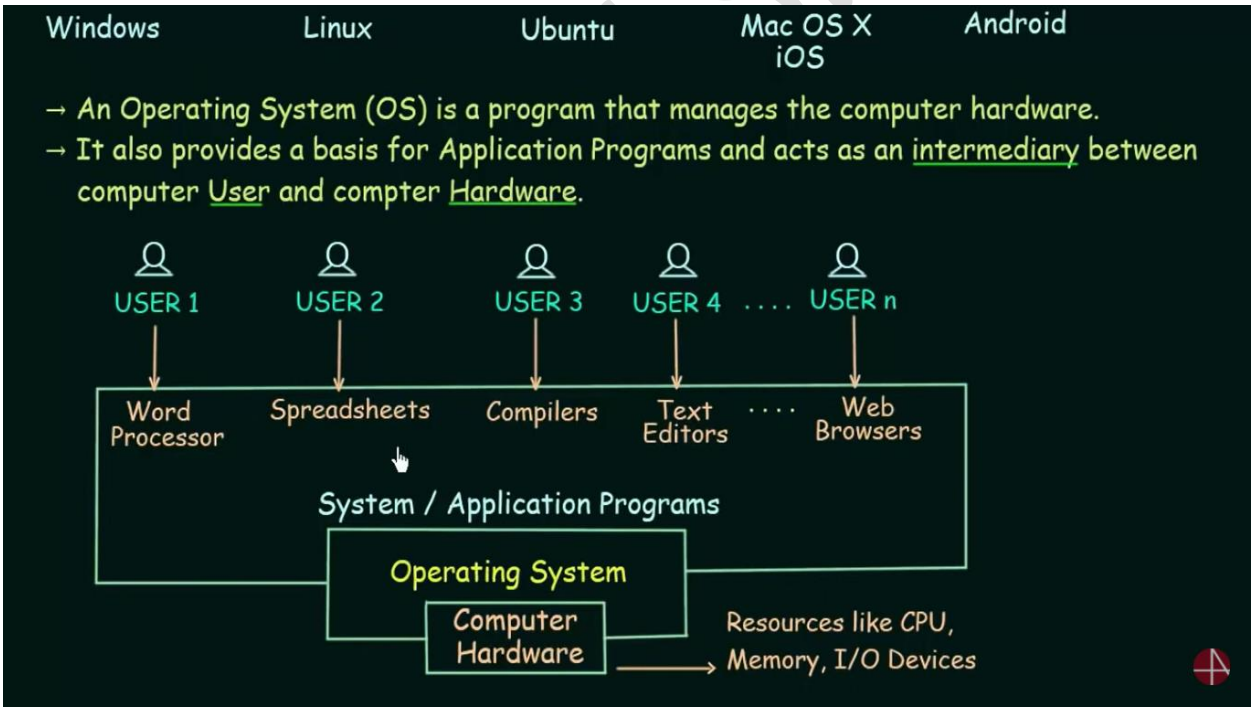
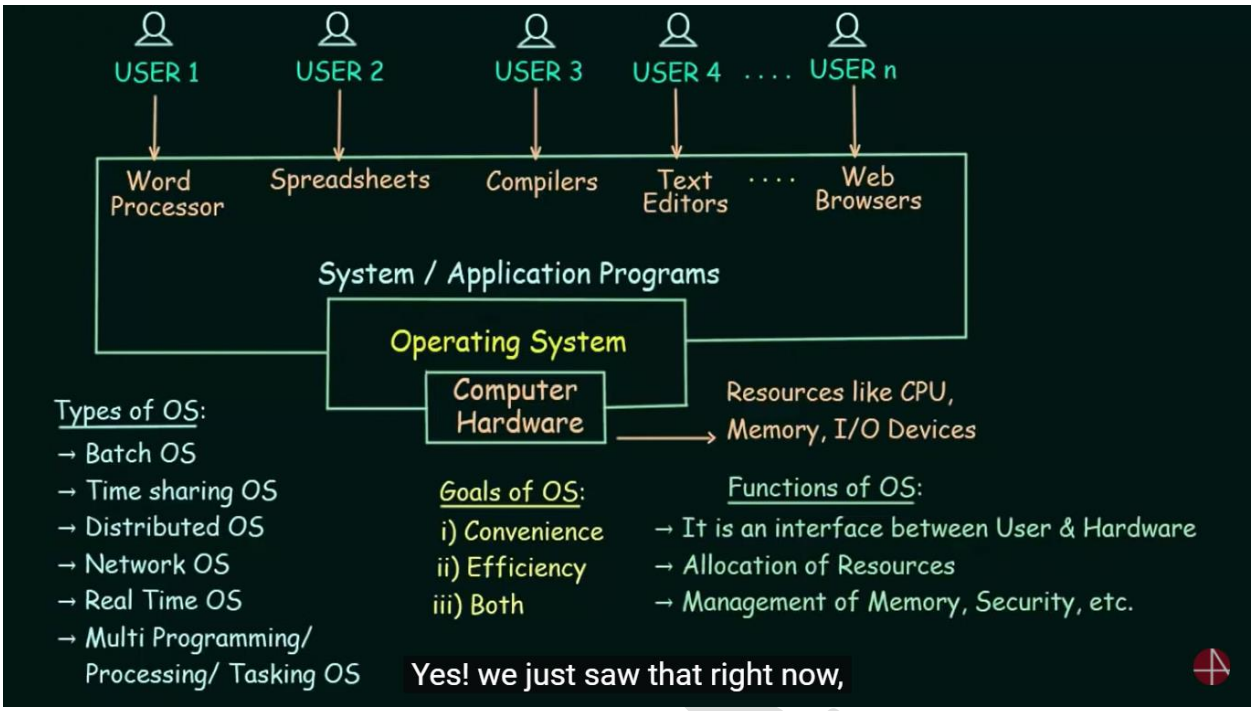
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

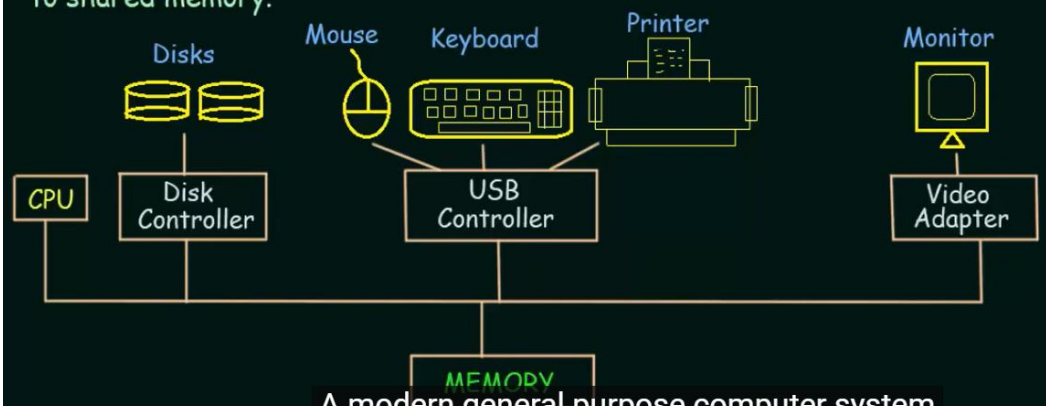


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Basics of Operating System (Computer System Operation)

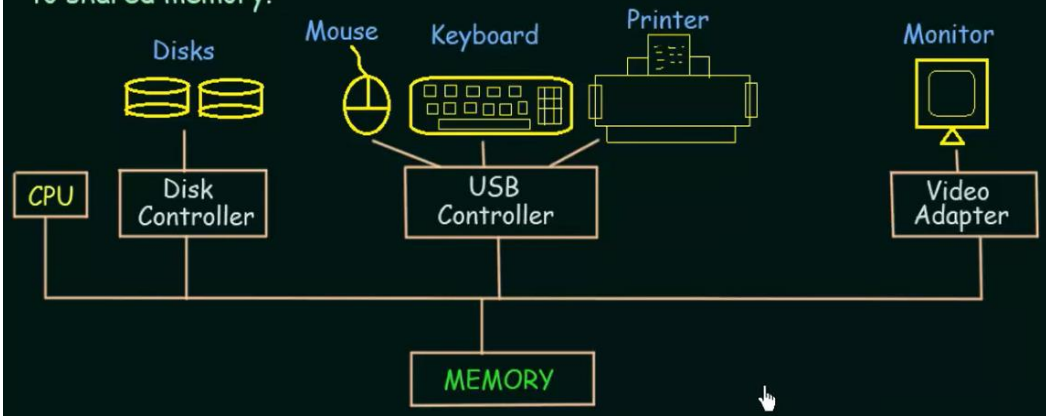
Some basic knowledge of the structure of Computer System is required to understand how Operating Systems work.

→ A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory.



A modern general purpose computer system

a number of device controllers connected through a common bus that provides access to shared memory.



→ Each device controller is in charge of a specific type of device

→ The CPU and the device controllers can execute concurrently, competing for memory cycles

→ To ensure orderly access to the shared memory, a memory controller is provided whose function is to synchronize acc

how things are arranged.

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

→To ensure orderly access to the shared memory, a memory controller is provided whose function is to synchronize access to the memory

Some important terms:

1) Bootstrap Program:→The initial program that runs when a computer is powered up or rebooted.
→It is stored in the ROM.
→It must know how to load the OS and start executing that system.
→It must locate and load into memory the OS Kernel.

2) Interrupt: → The occurence of an event is usually signalled by an Interrupt from Hardware or Software.
→ Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by the way of the system bus.

3) System Call (Monitor call): → Software may trigger an interrupt by executing a special operation called System Call.

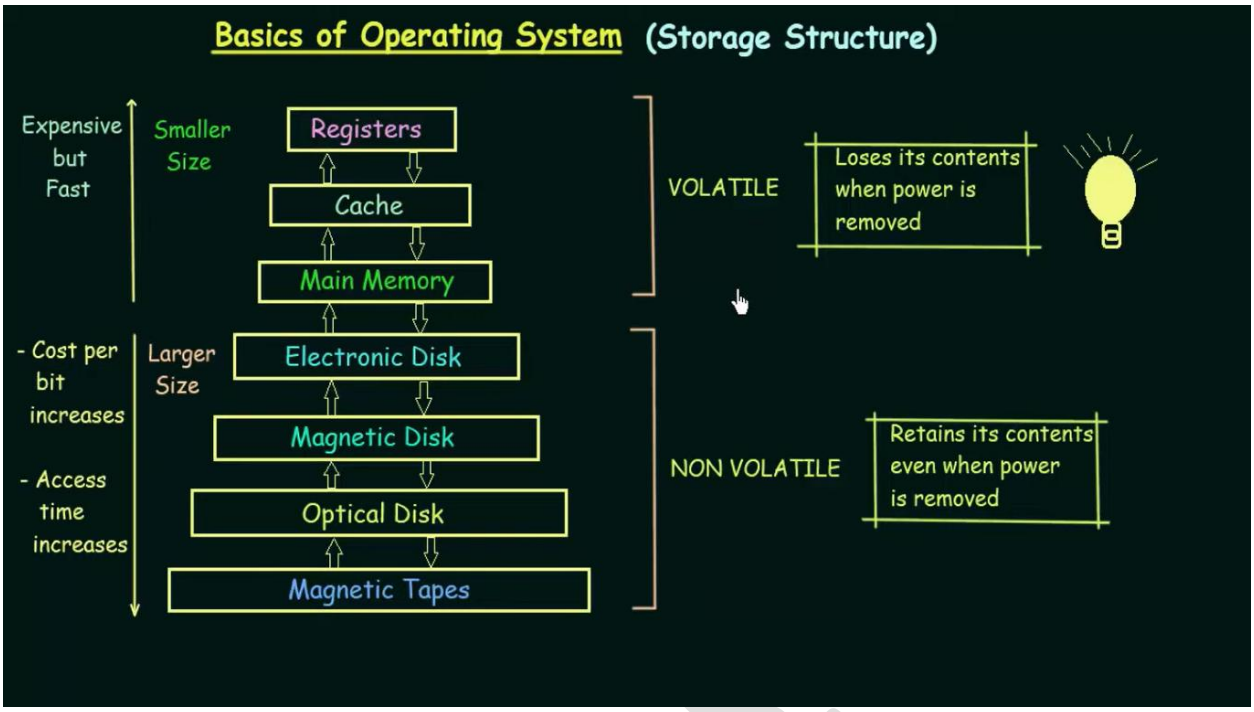
When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location.

→ The fixed location usually contains the starting address where the Service Routine of the interrupt is located.

The Interrupt Service Routine executes.

On completion, the CPU resumes the interrupted computation.

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Basics of Operating System (I/O Structure)

-> Storage is only one of many types of I/O devices within a computer

-> A large portion of operating system code is dedicated to managing I/O, both because of its importance to the reliability and performance of a system and because of the varying nature of the devices

-> A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus

-> Each device controller is in charge of a specific type of device

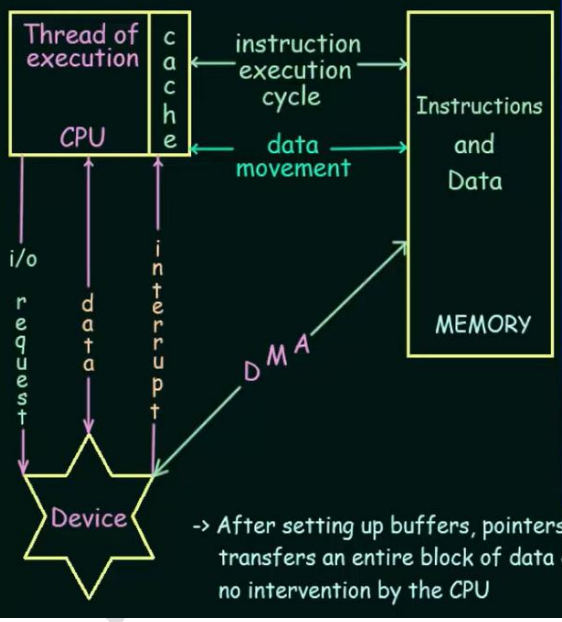
maintains

- Local Buffer Storage
- Set of Special Purpose Registers

-> Typically, operating systems have a device driver for each device controller

-> This device driver understands the device controller and presents a uniform interface to the device to the rest of the operating system

Working of an I/O Operation:



-> To start an I/O operation, the device driver loads the appropriate registers within the device controller

-> The device controller, in turn, examines the contents of these registers to determine what action to take

-> The controller starts the transfer of data from the device to its local buffer

-> Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation

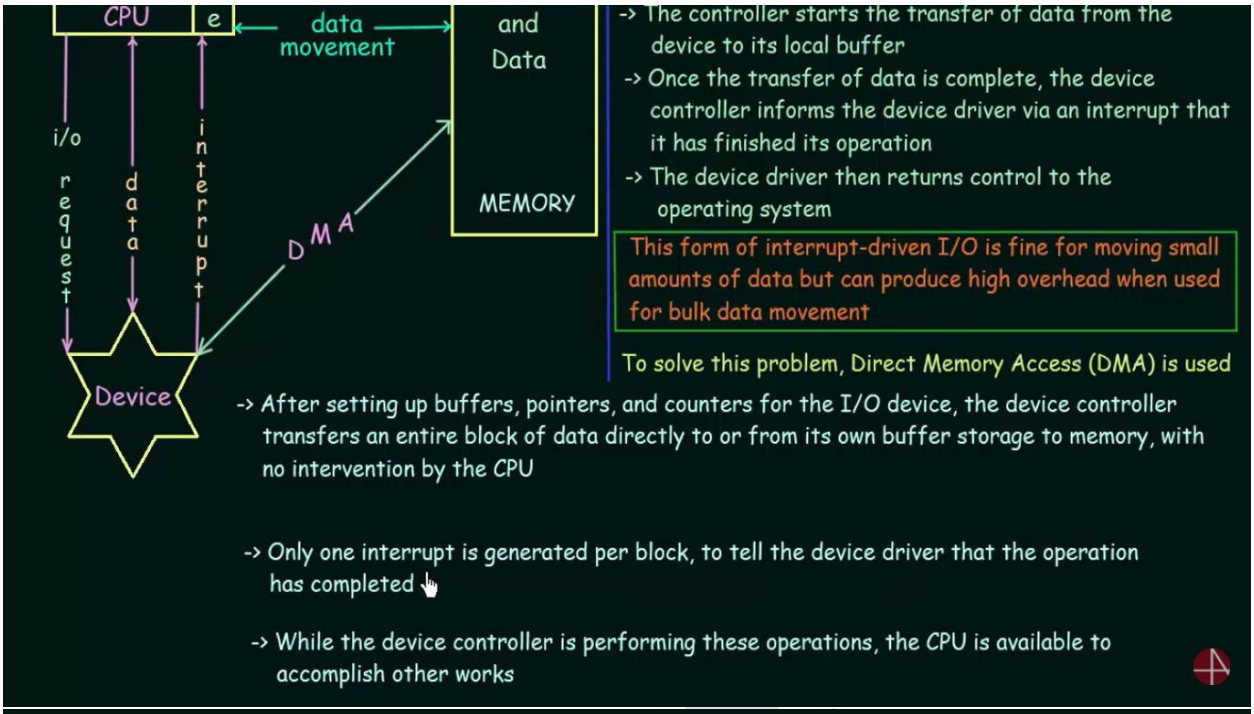
-> The device driver then returns control to the operating system

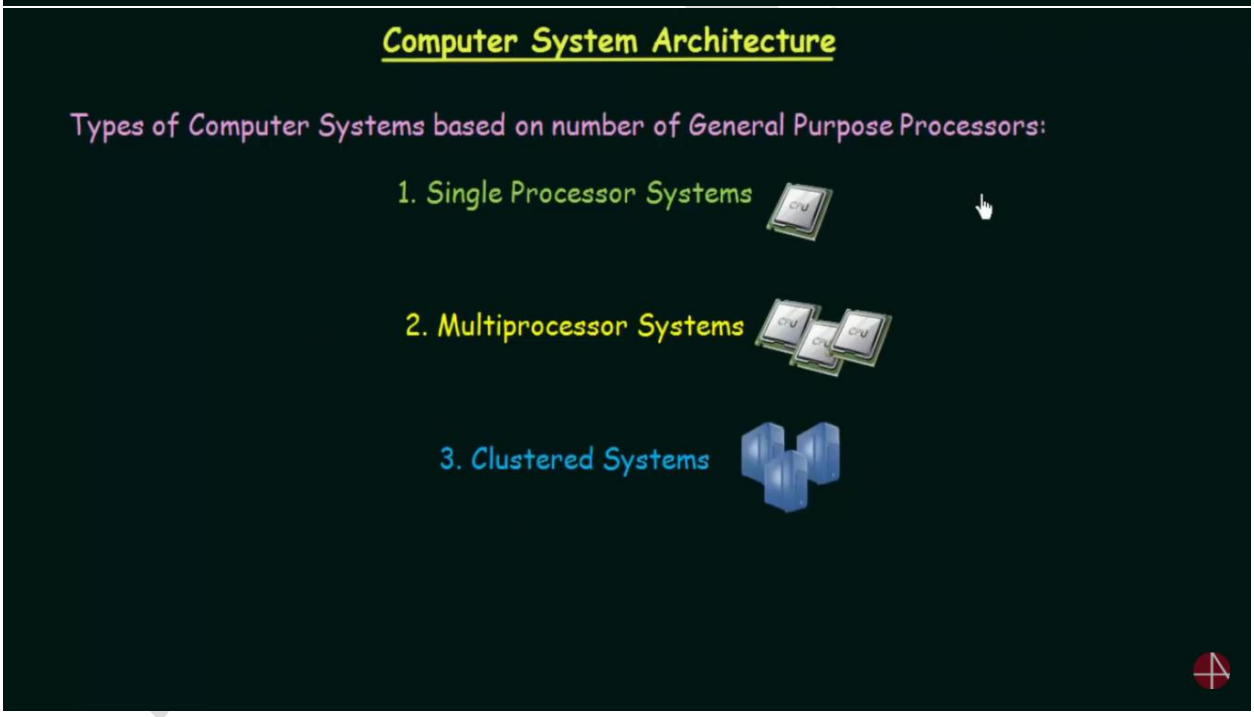
This form of interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement

To solve this problem, Direct Memory Access (DMA) is used

-> After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



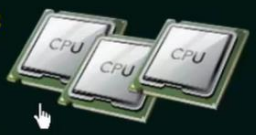


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

1. Single Processor Systems:

- One main CPU capable of executing a general purpose instruction set including instructions from user processes.
- Other special purpose processors are also present which perform device specific tasks

2. Multiprocessor Systems:

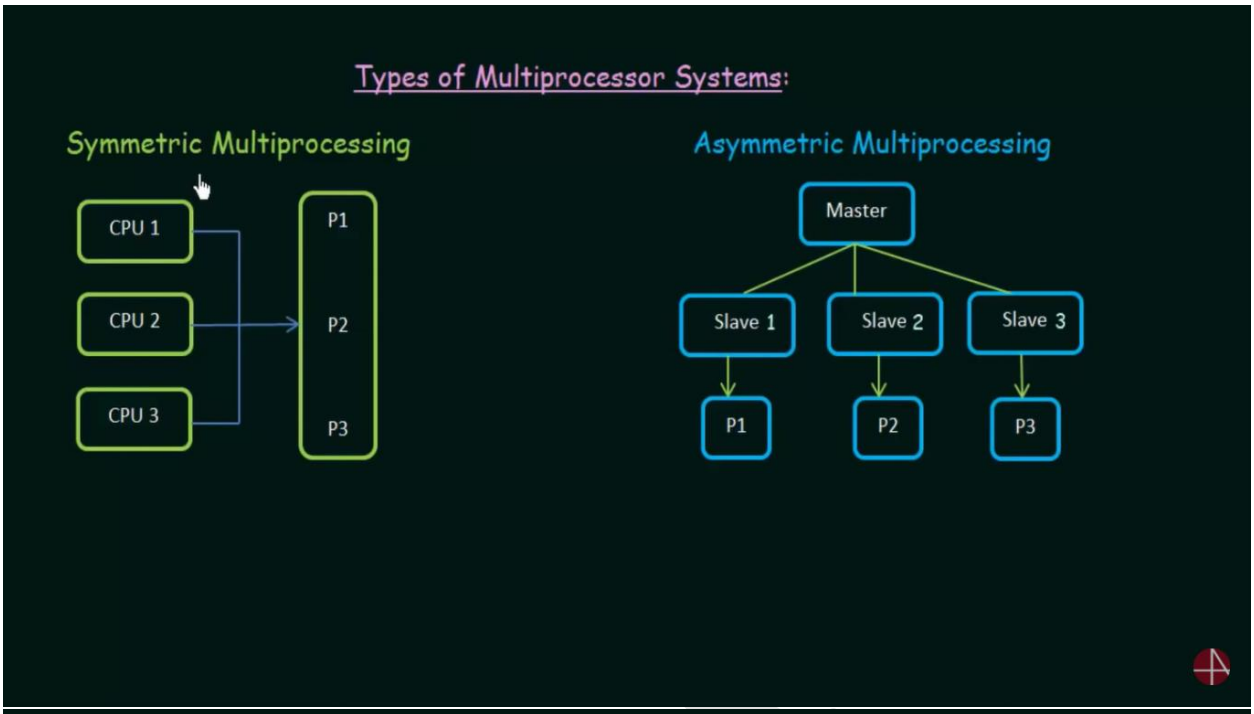


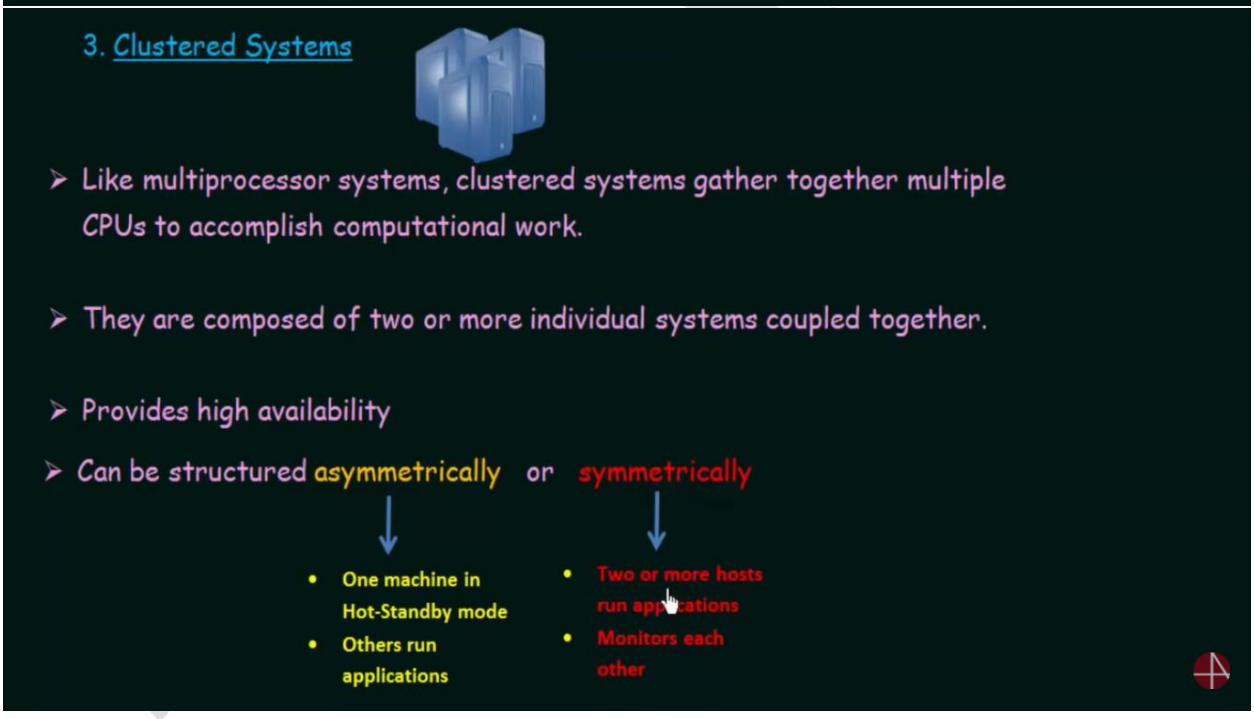
- Also known as parallel systems or tightly coupled systems.
- Has two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices

Advantages:

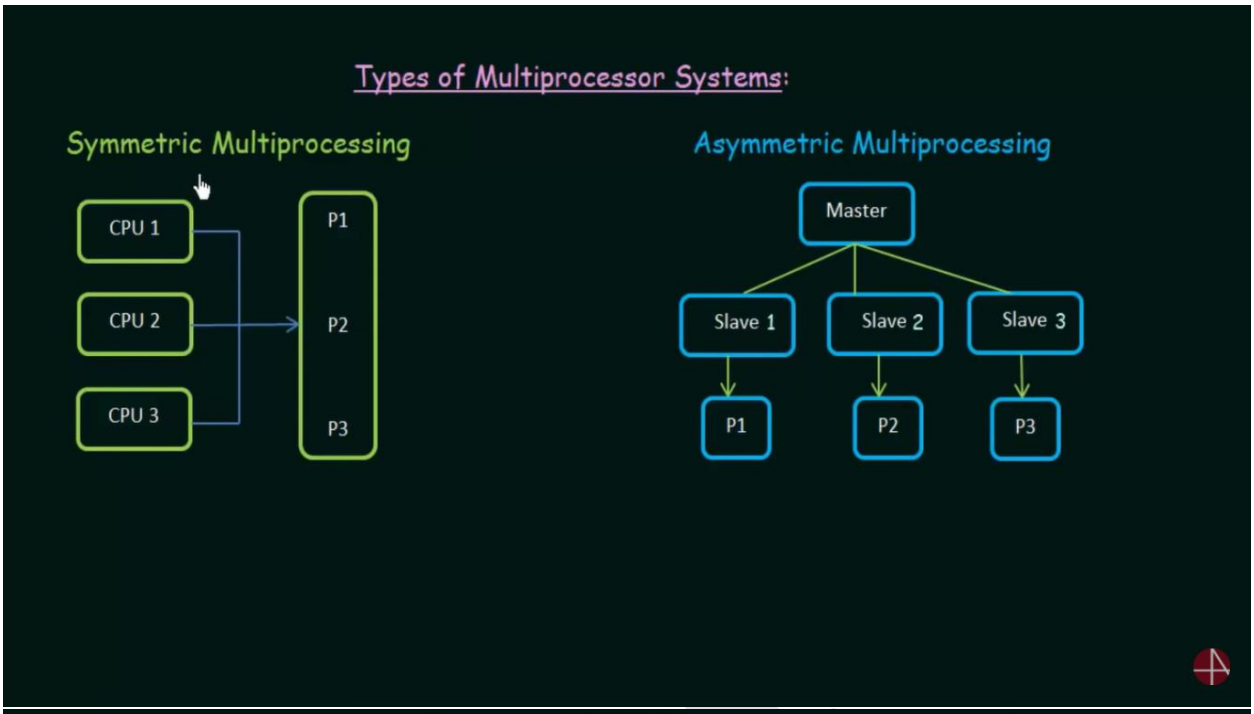
- Increased throughput
- Economy of scale
- Increased reliability

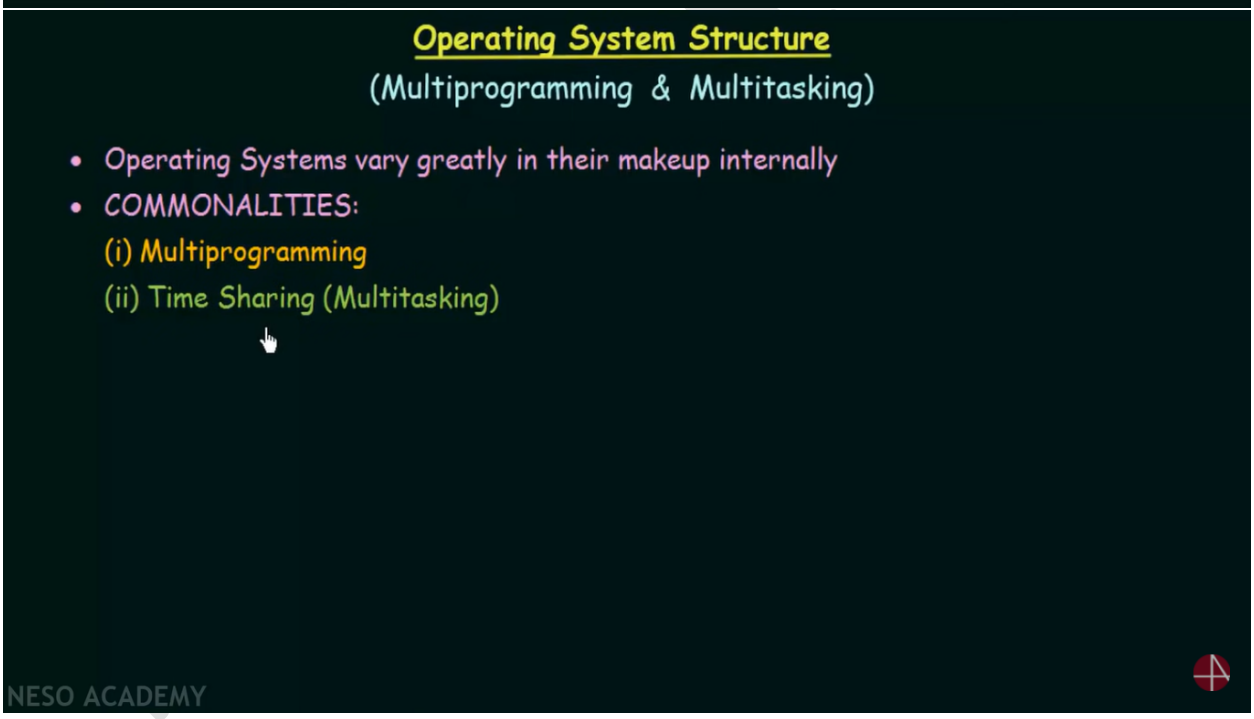
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

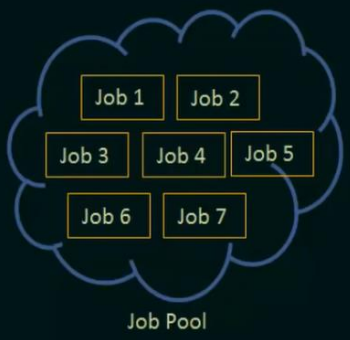


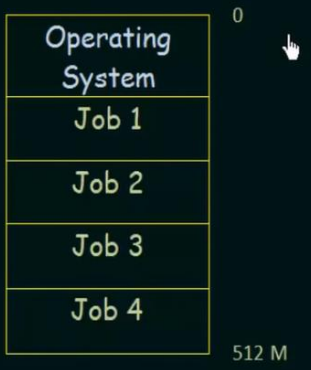


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

(i) Multiprogramming

- A single user cannot, in general, keep either the CPU or the I/O devices busy at all times
- Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.





Memory layout for a multiprogramming system

Multiprogrammed systems provide an environment in which the various system resources (for example, CPU, memory, and peripheral devices) are utilized effectively, but they do not provide for user interaction with the Computer system.

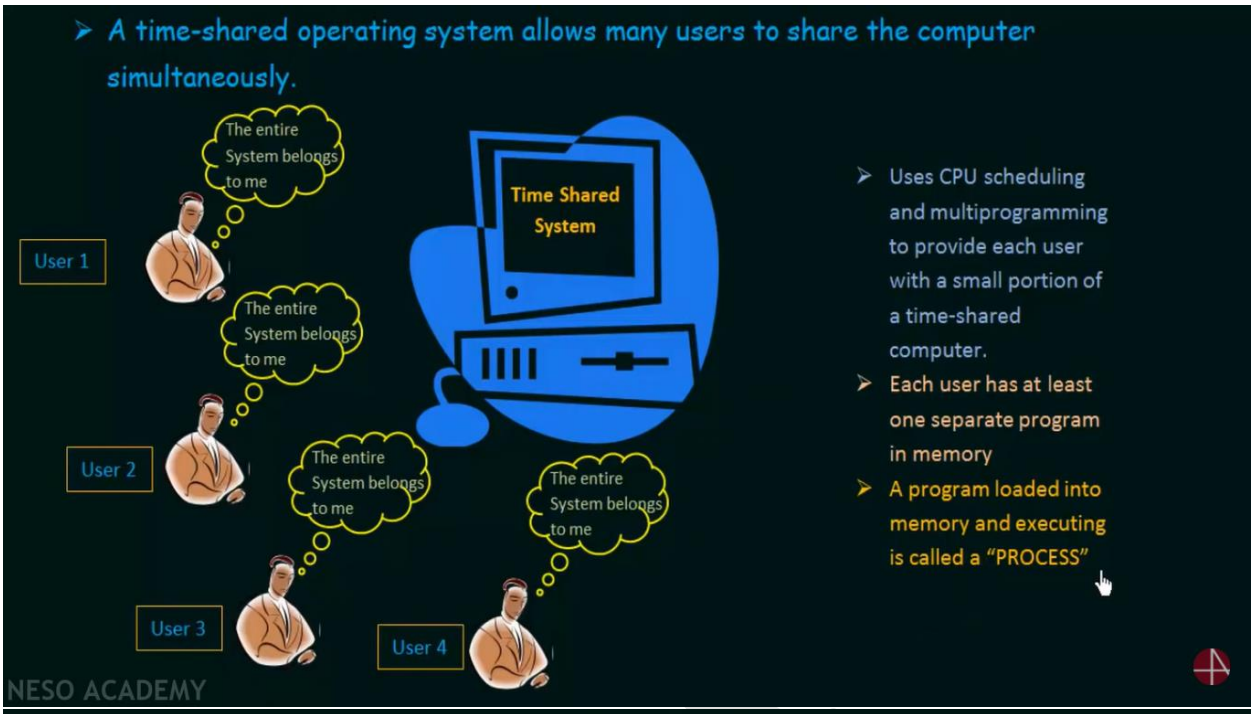
NESO ACADEMY

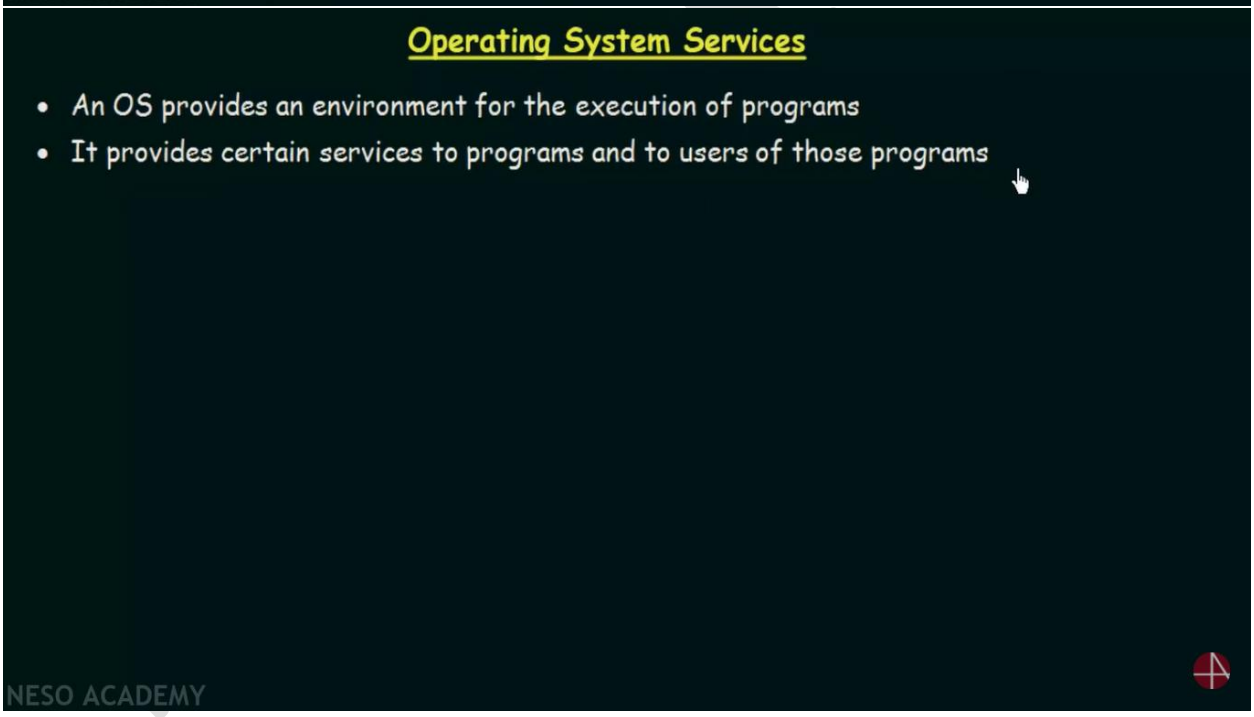
(ii) Time Sharing (Multitasking)

- CPU executes multiple jobs by switching among them
- Switches occur so frequently that the users can interact with each program while it is running
- Time sharing requires an interactive (or hands-on) computer system, which provides direct communication between the user and the system.
- A time-shared operating system allows many users to share the computer simultaneously.

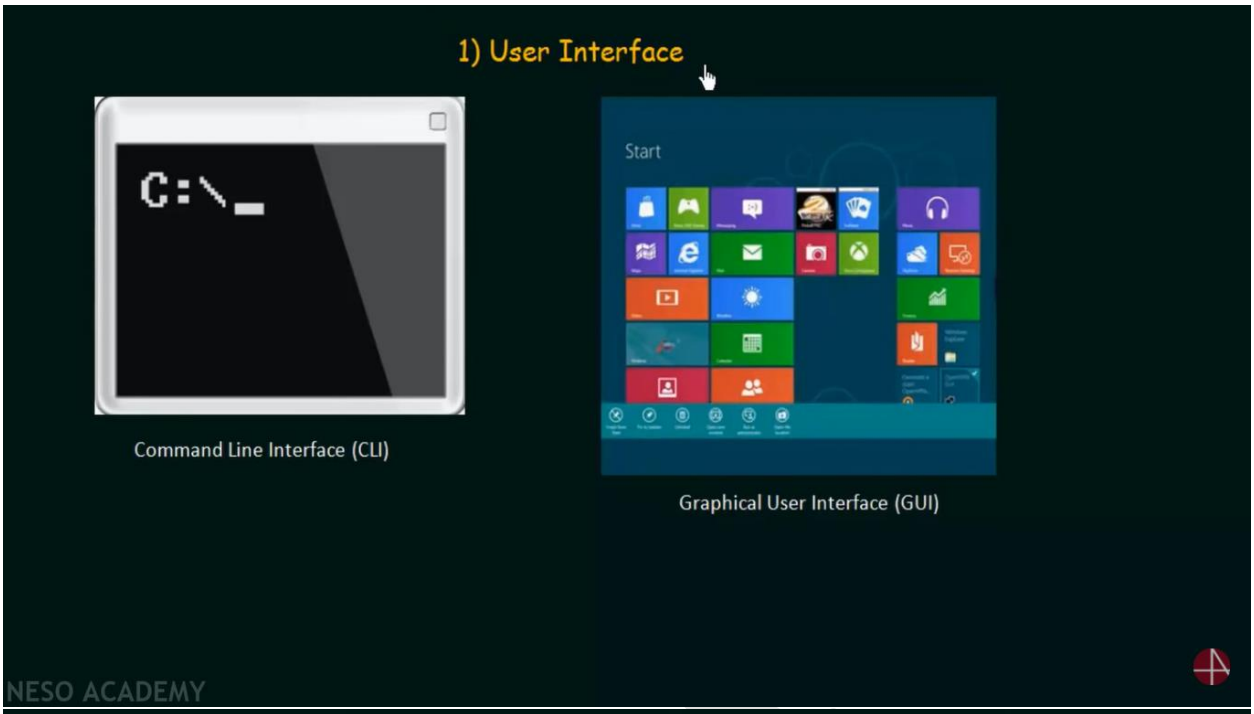
NESO ACADEMY

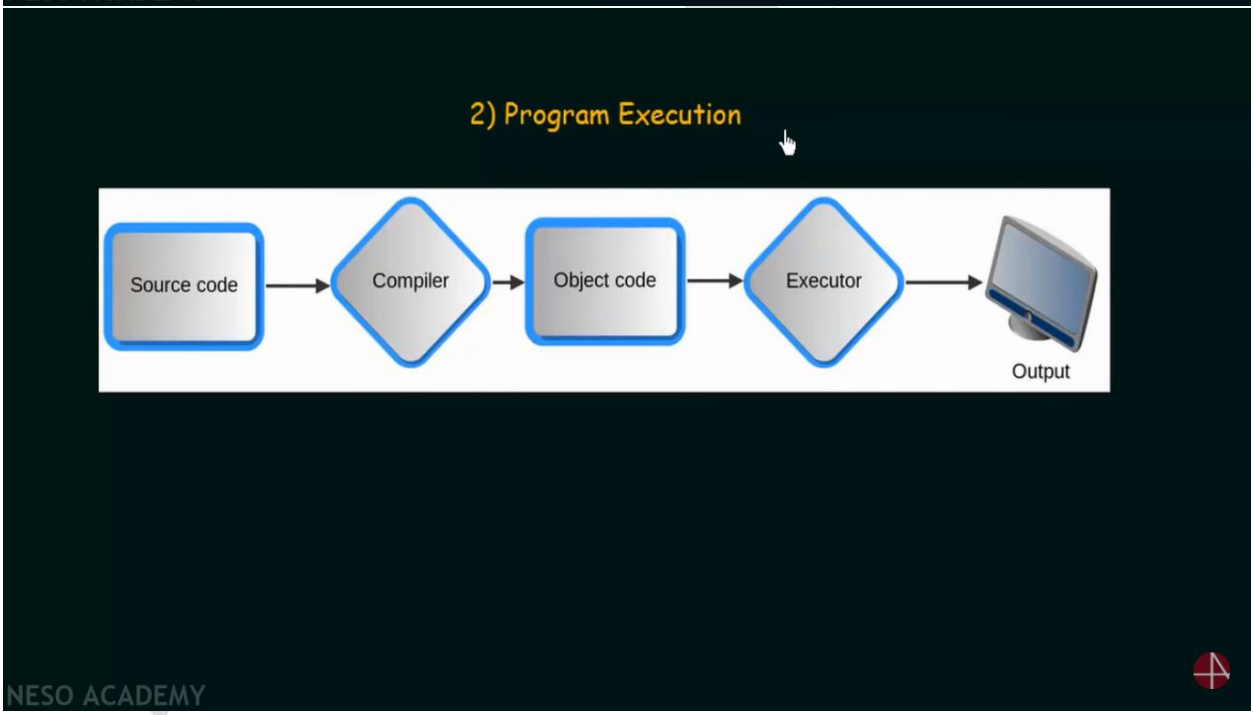
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



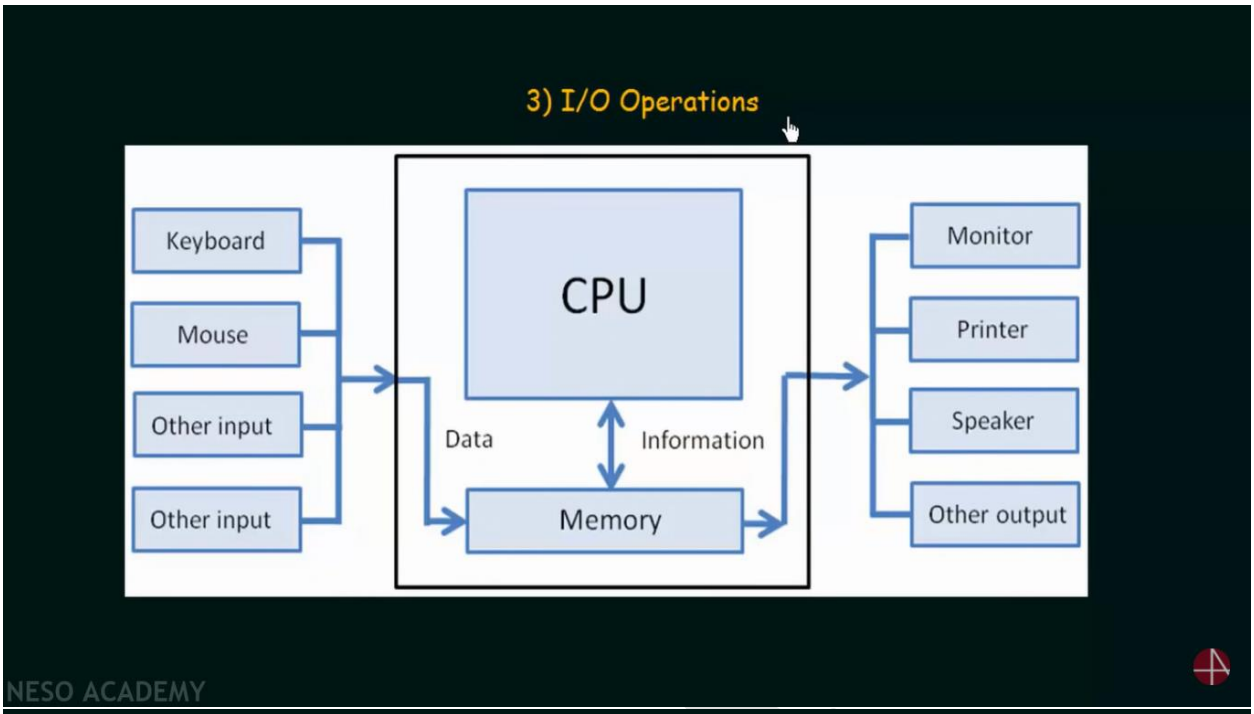


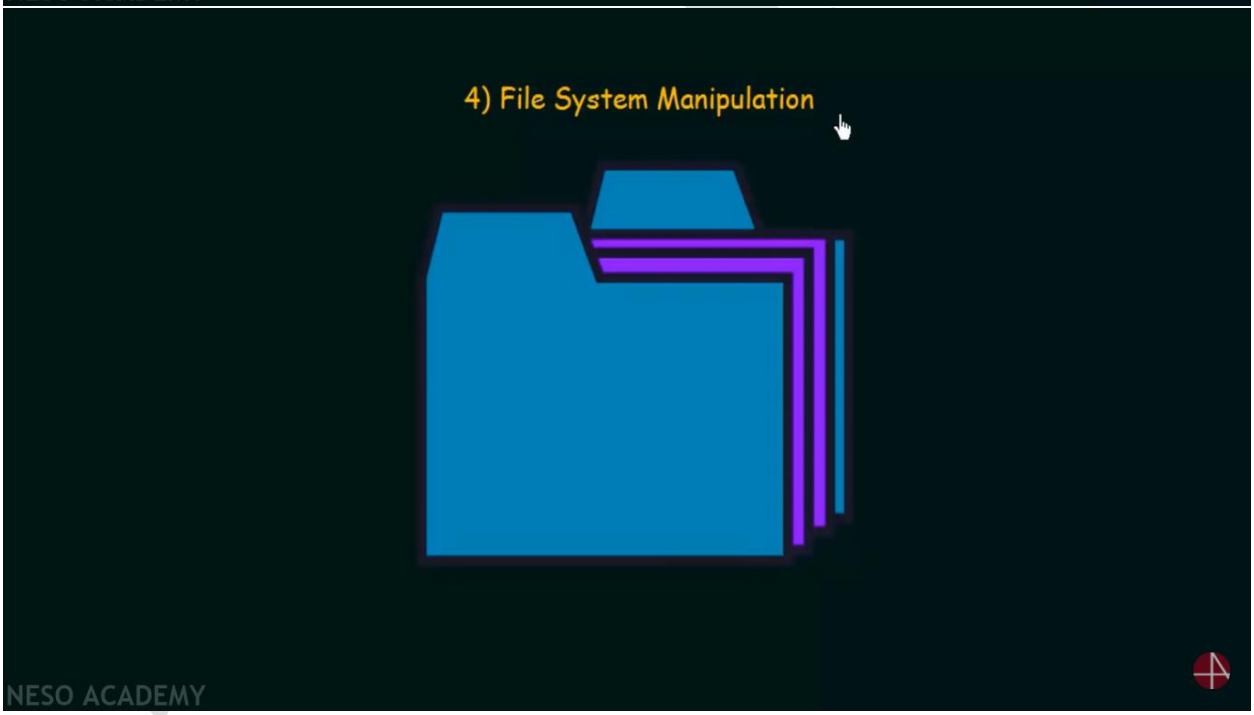
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



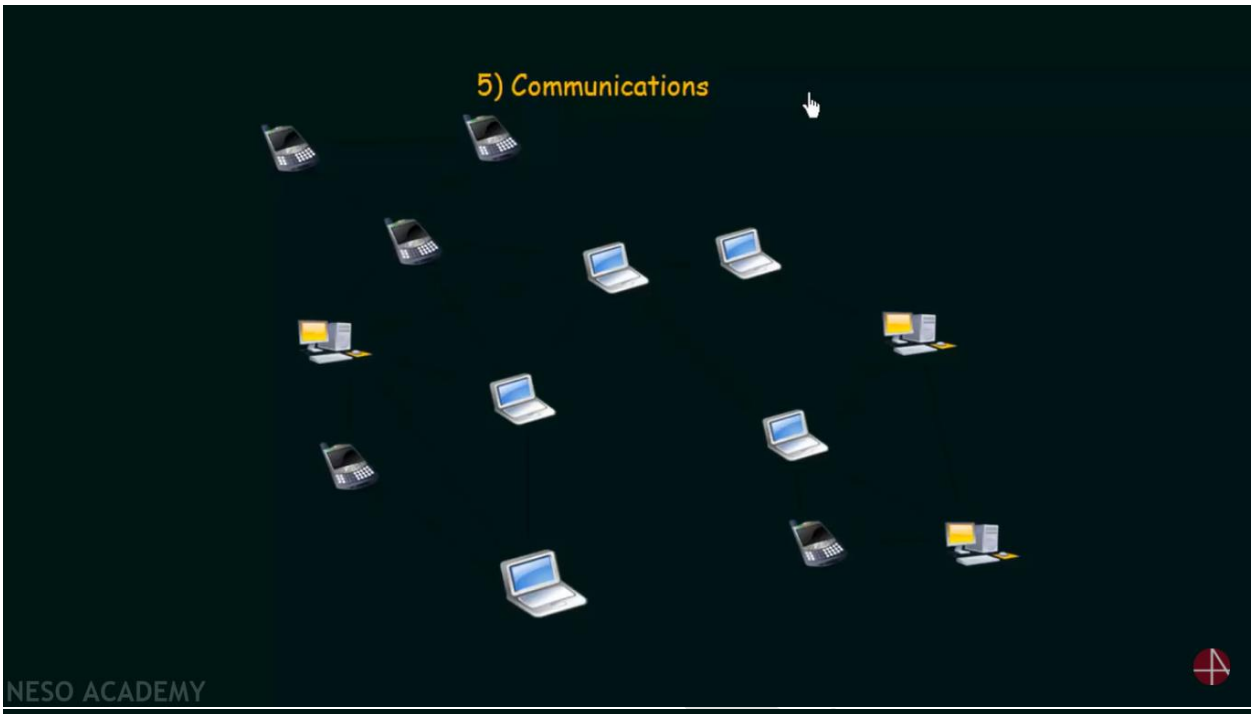


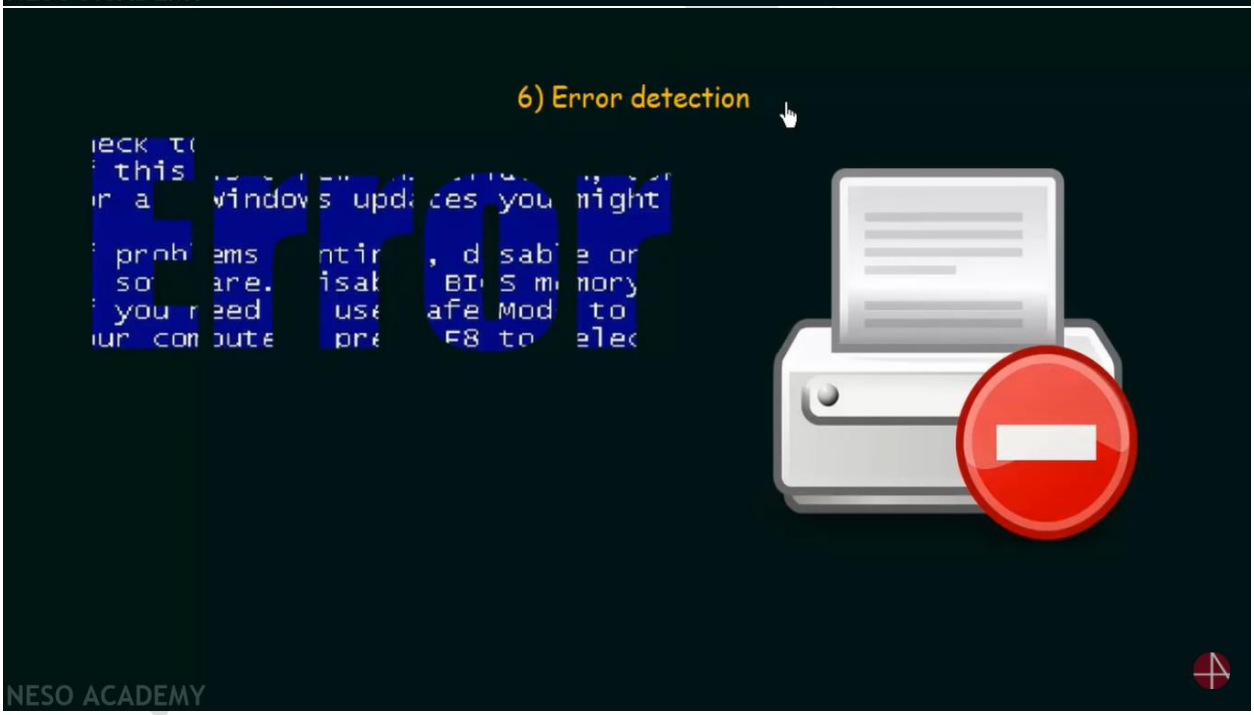
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



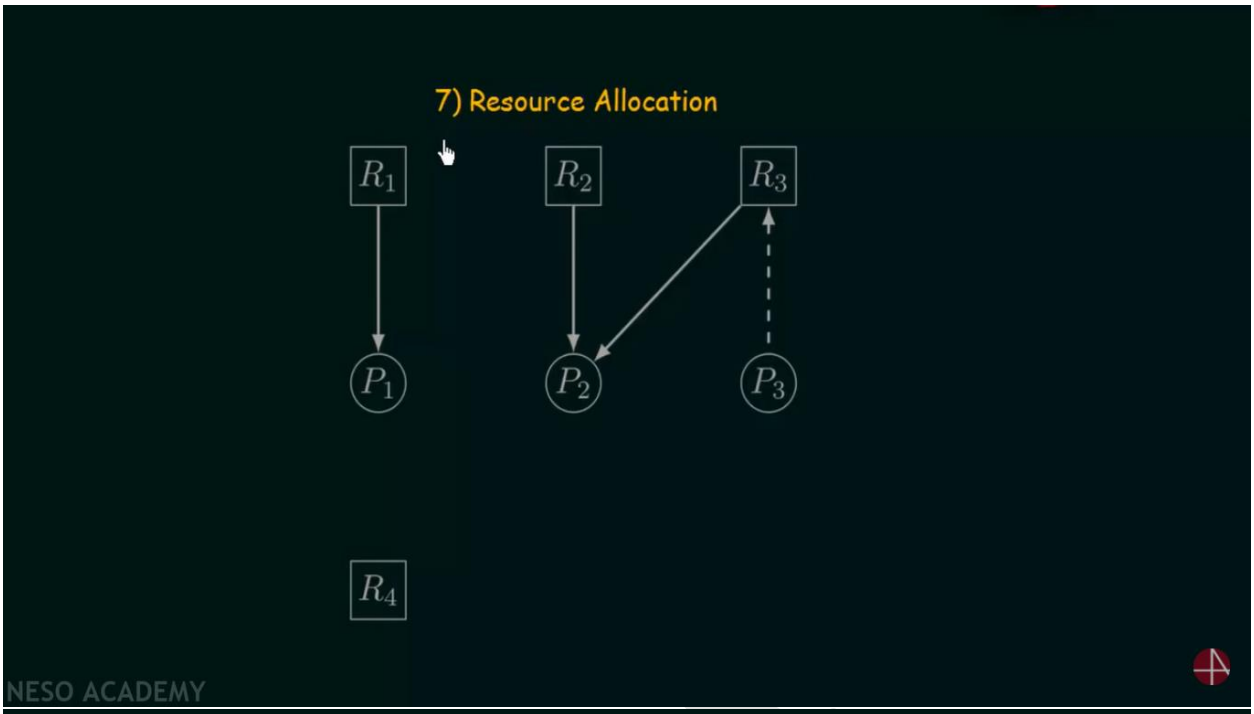


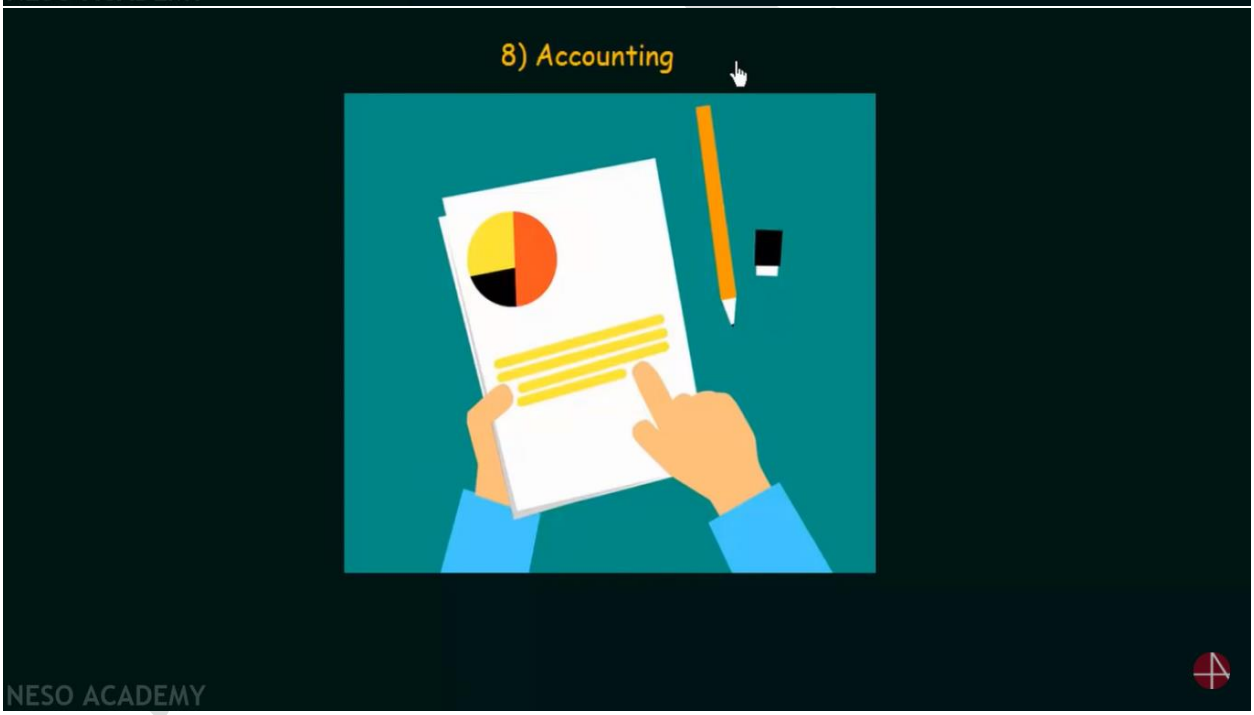
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



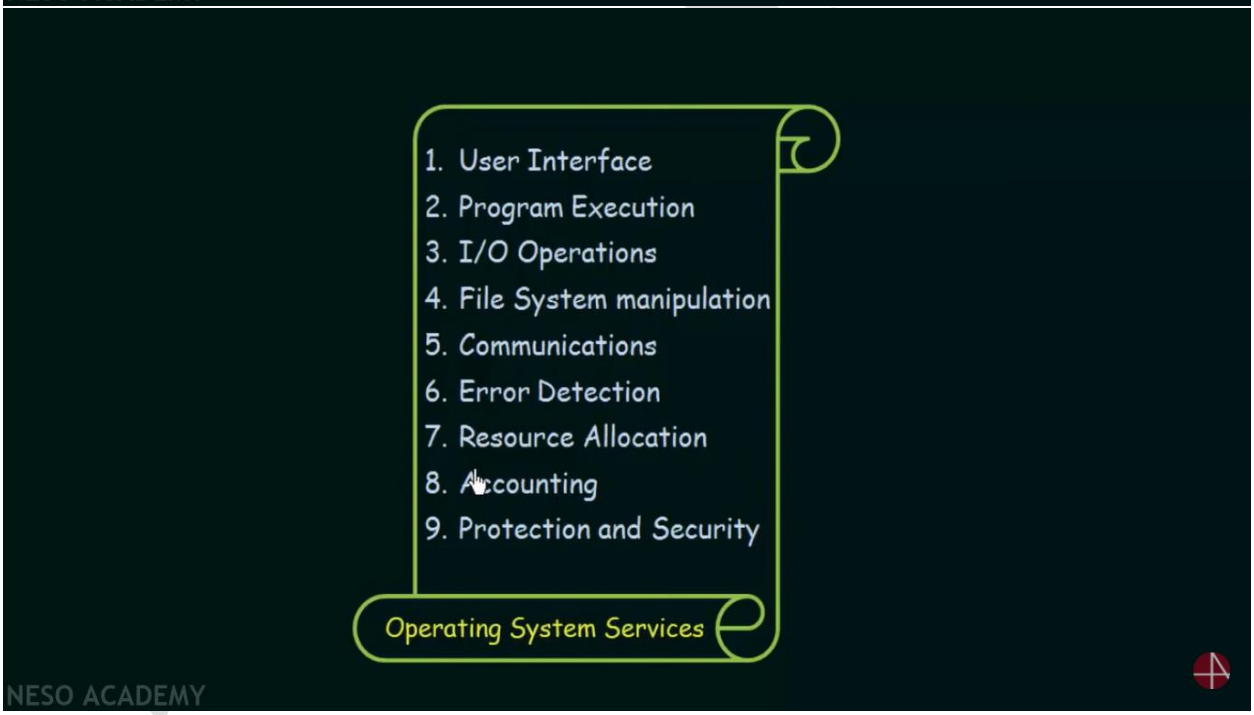


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

User Operating System Interface

There are two fundamental approaches for users to interface with the operating system:

1) Provide a Command-Line Interface (CLI) or Command Interpreter that allows users to directly enter commands that are to be performed by the operating system.

2) Allows the user to interface with the operating system via a Graphical User Interface or GUI.

NESO ACADEMY

Command Interpreter

- Some operating systems include the command interpreter in the kernel.
- Others, such as Windows XP and UNIX, treat the command interpreter as a special program.
- On systems with multiple command interpreters to choose from, the interpreters are known as shells.
- E.g.
  - Bourne shell,
  - C shell
  - Bourne-Again shell (BASH)
  - Korn shell, etc.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

User Operating System Interface

There are two fundamental approaches for users to interface with the operating system:

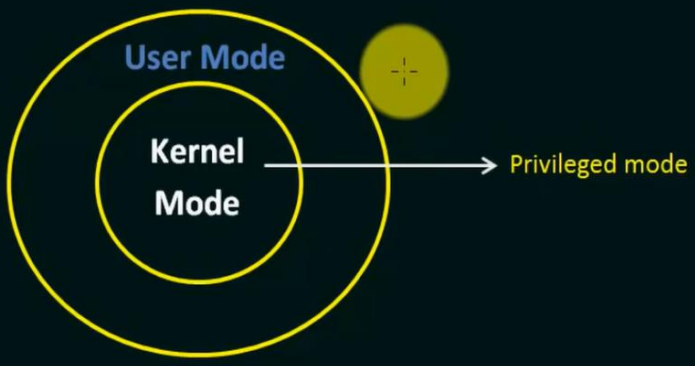
1) Provide a Command-Line Interface (CLI) or Command Interpreter that allows users to directly enter commands that are to be performed by the operating system.

2) Allows the user to interface with the operating system via a Graphical User Interface or GUI.

NESO ACADEMY

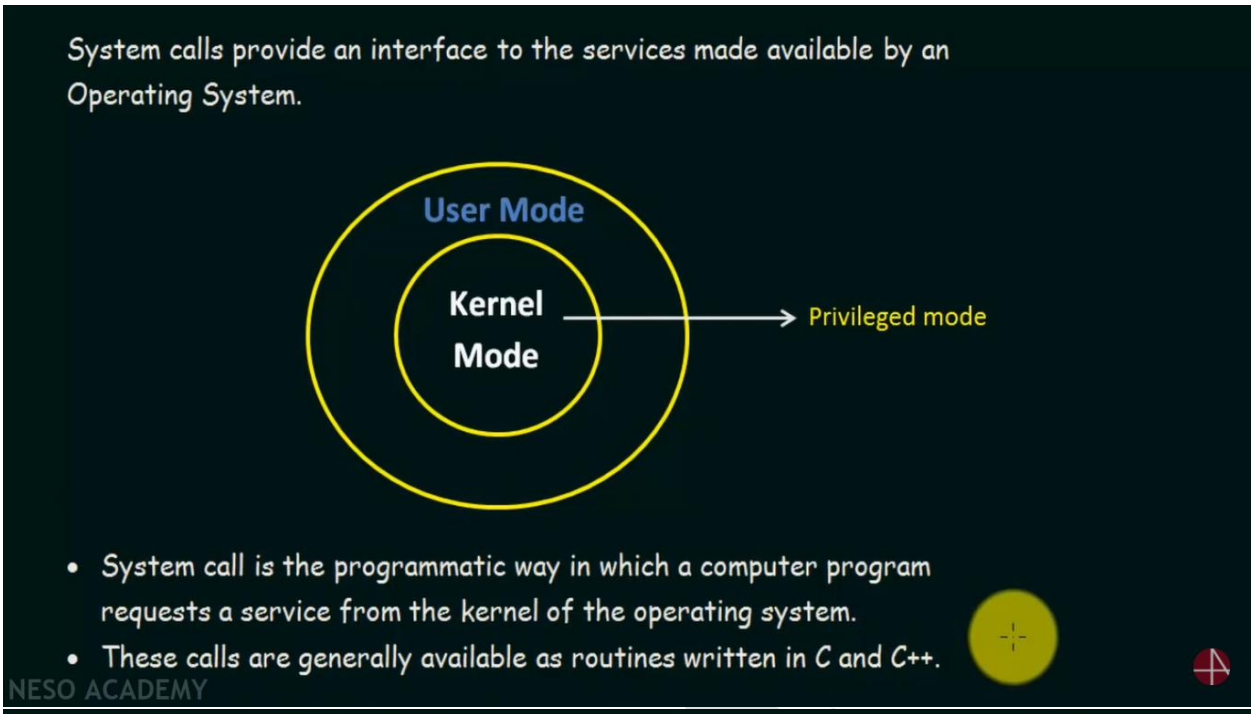
System Calls

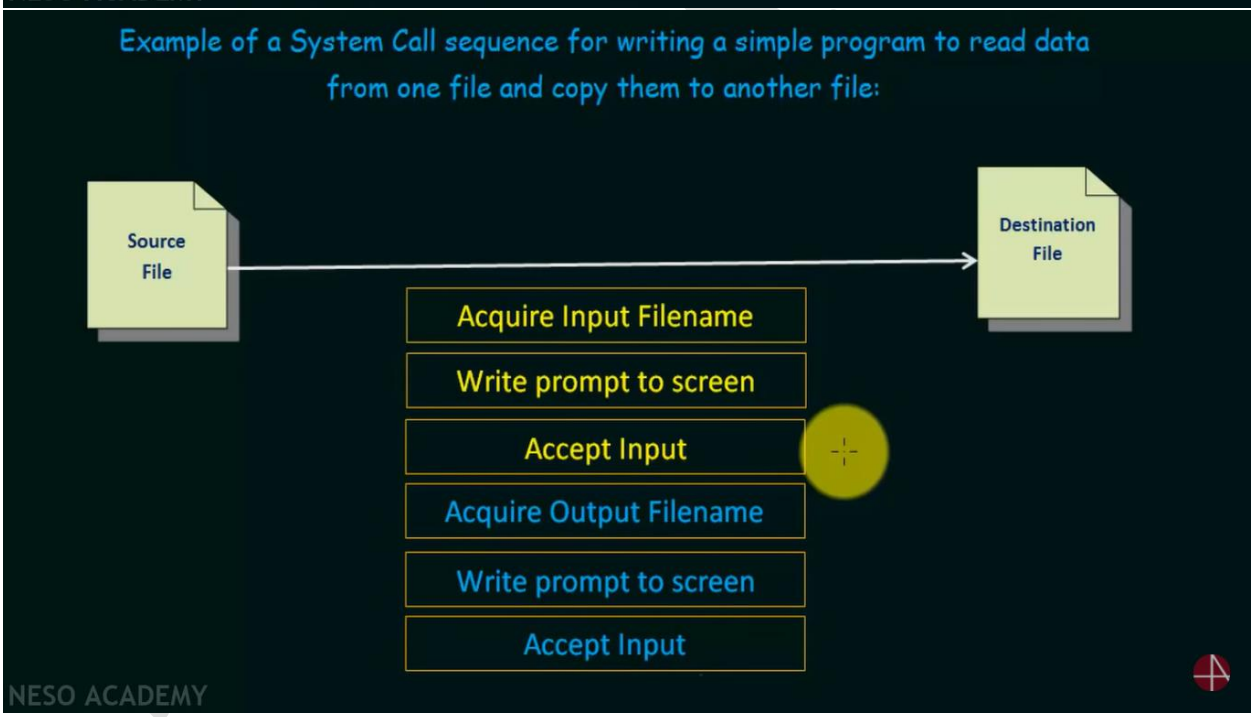
System calls provide an interface to the services made available by an Operating System.



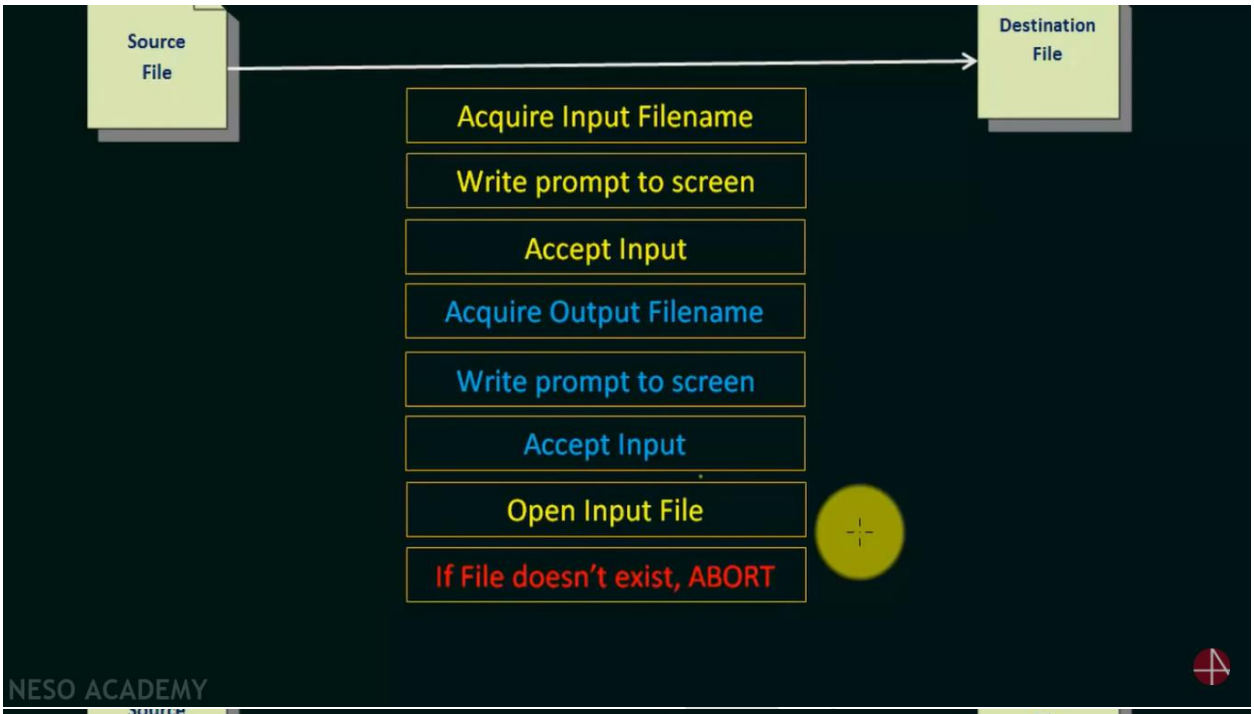
NESO ACADEMY

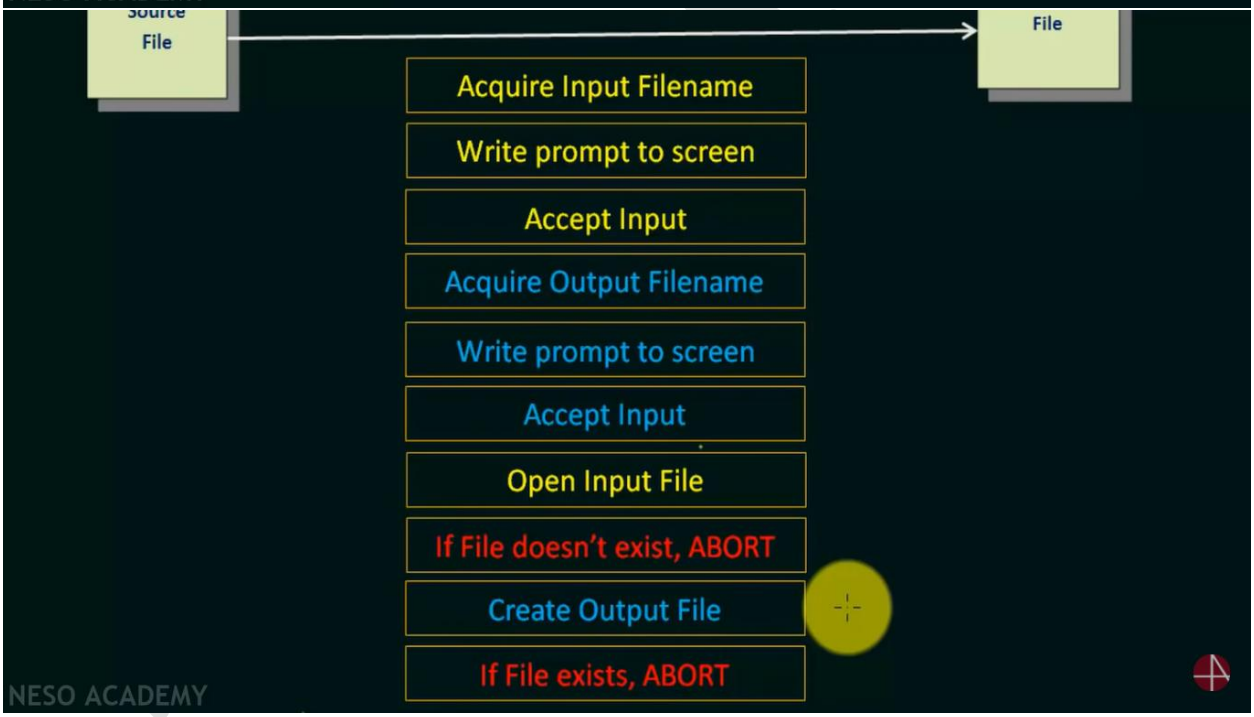
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





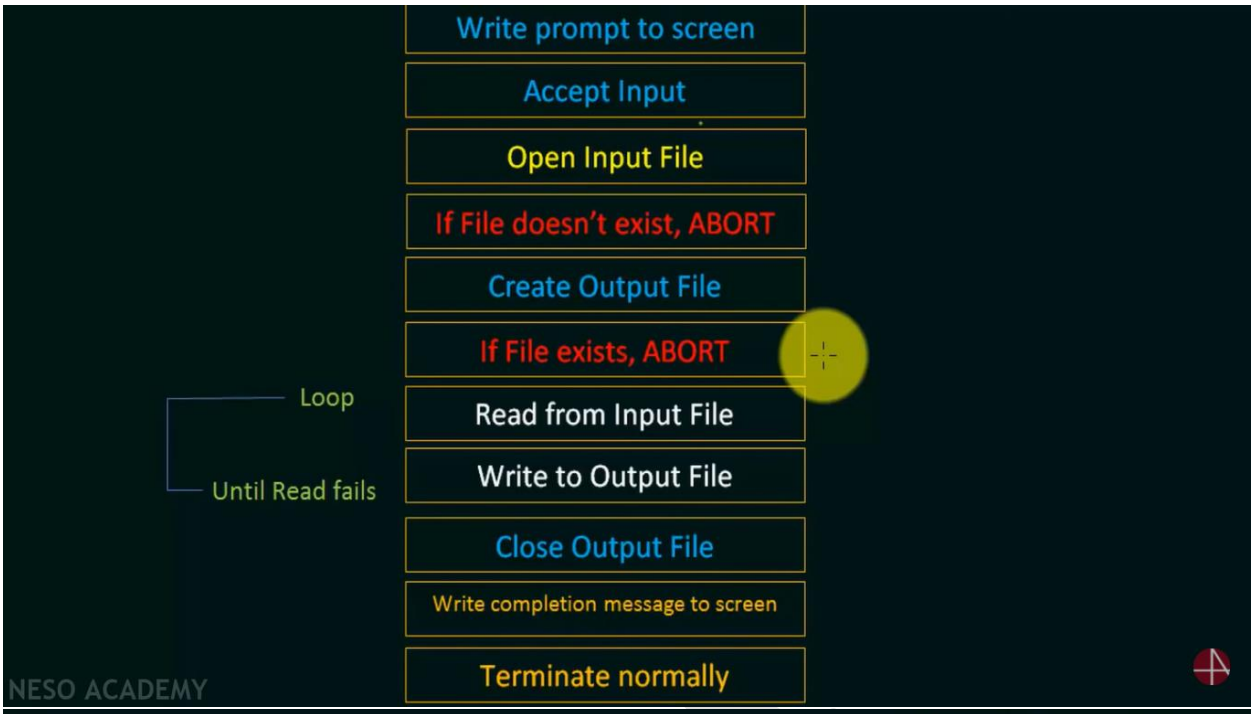
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

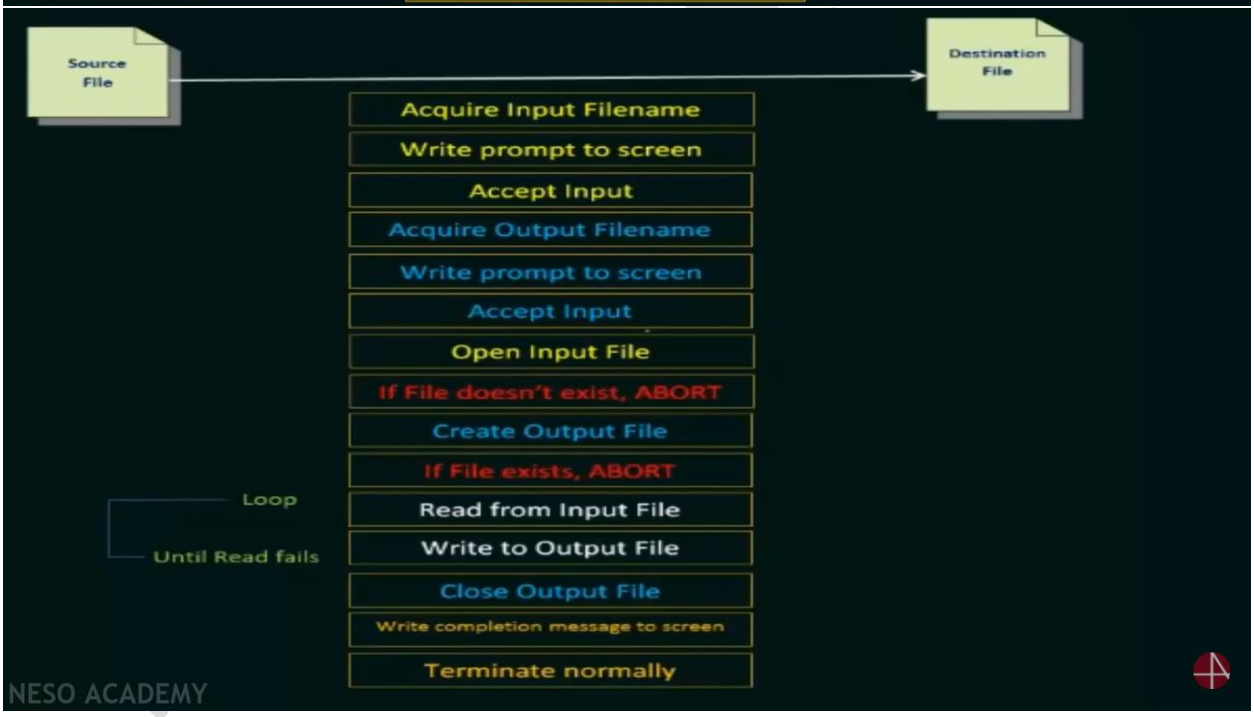




NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

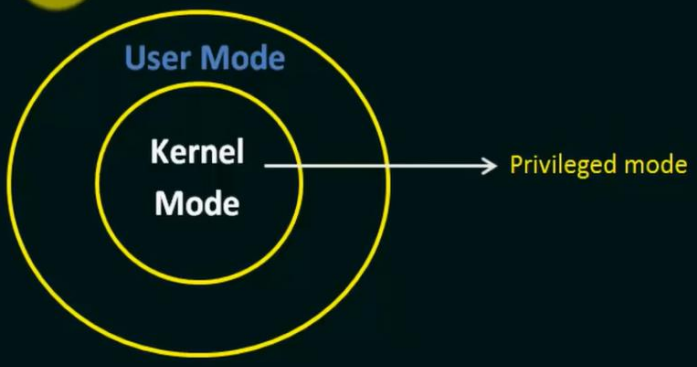
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

System calls provide an interface to the services made available by an Operating System.



- System call is the programmatic way in which a computer program requests a service from the kernel of the operating system.
- These calls are generally available as routines written in C and C++.

NESO ACADEMY

Types of System Calls

System calls can be grouped roughly into five major categories:

1. Process Control
2. File Manipulation
3. Device Management
4. Information Maintenance
5. Communications

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

1. 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

NESO ACADEMY

2. File Manipulation

- create file, delete file
- open, close
- read, write, reposition
- get file attributes, set file attributes

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

3. Device Manipulation

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices

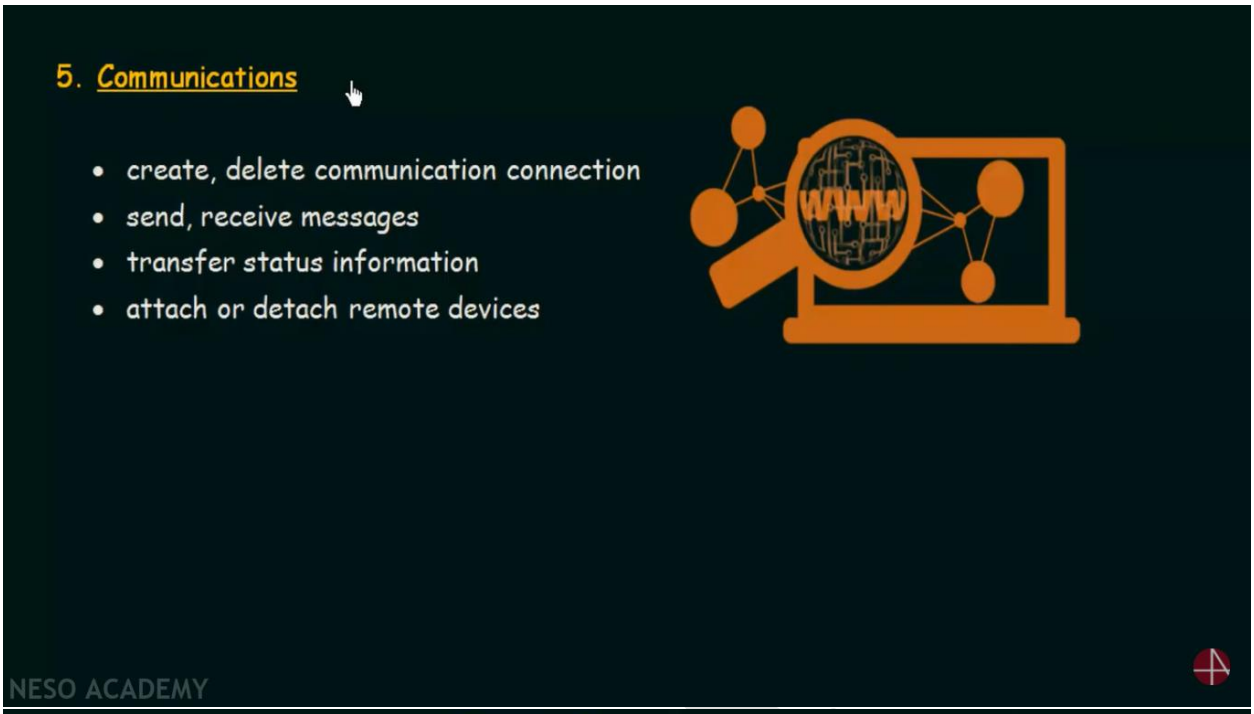
NESO ACADEMY

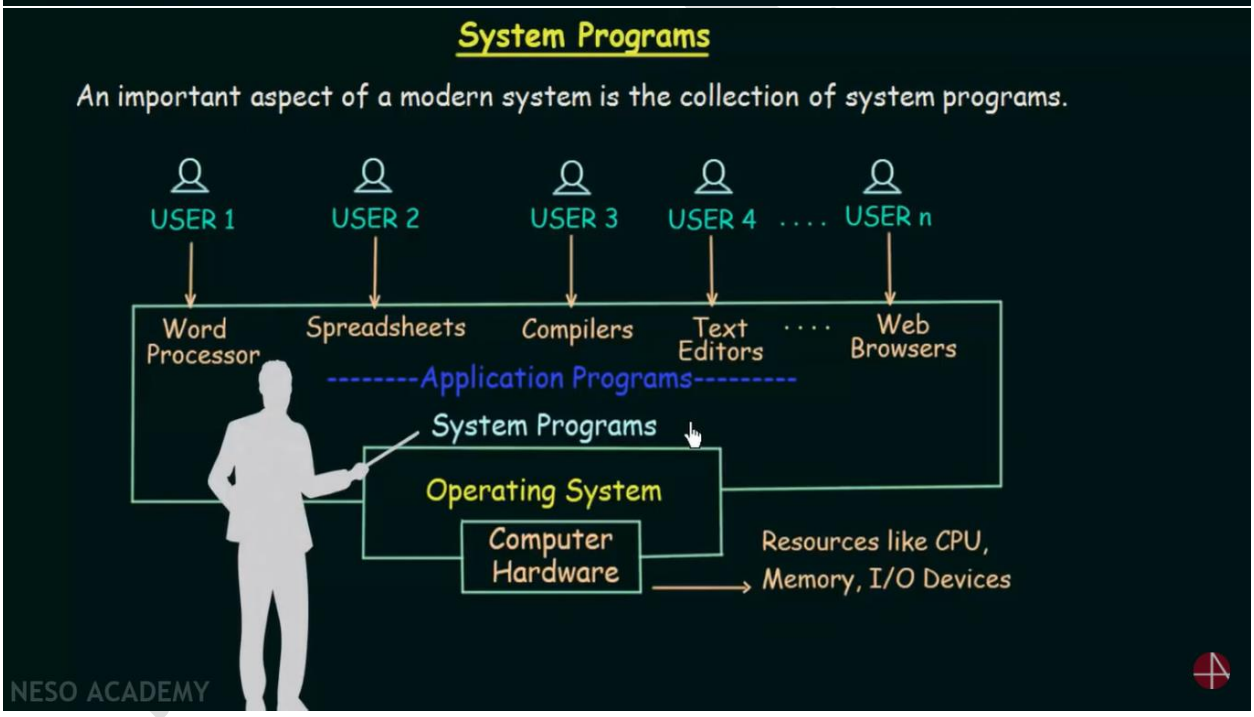
4. Information Maintenance

- 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

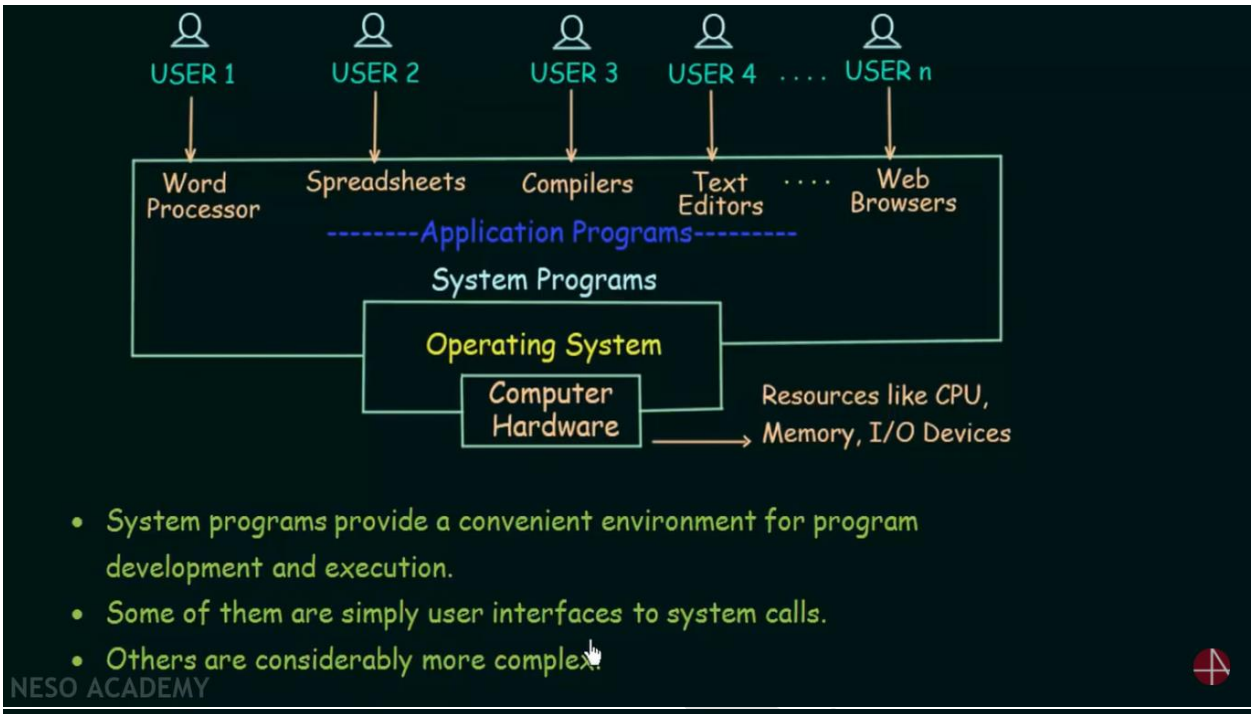
NESO ACADEMY

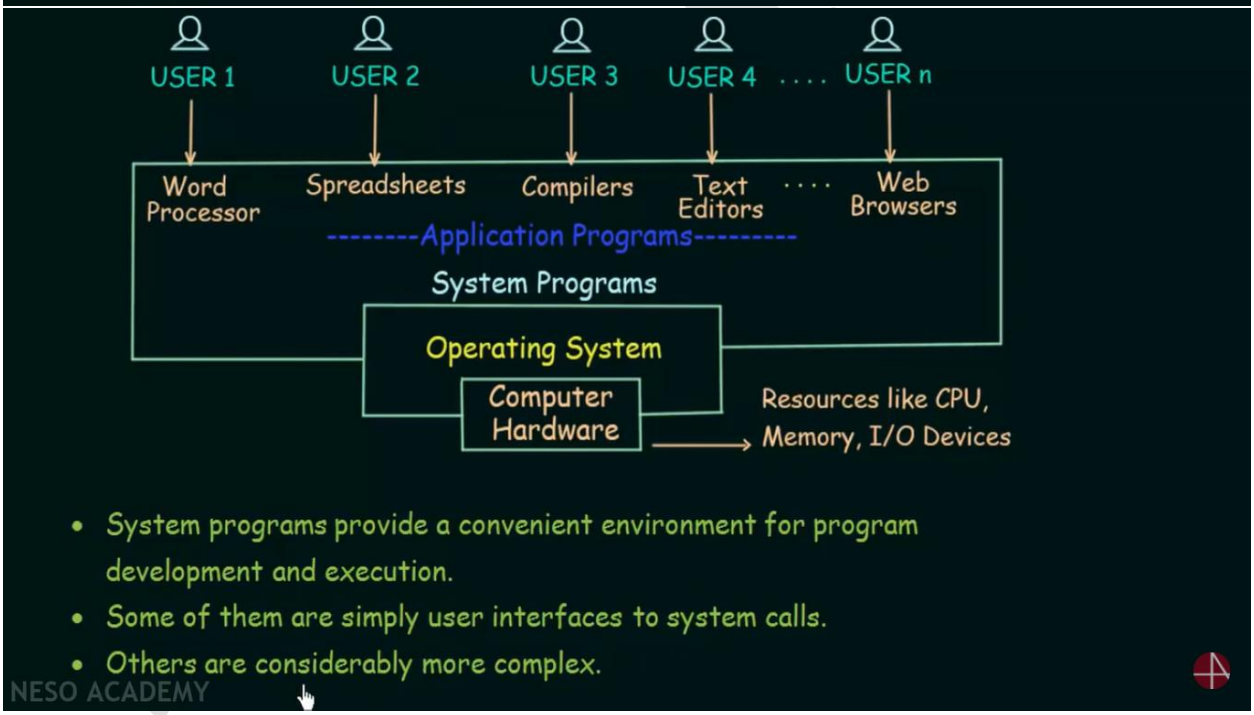
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

System Programs can be divided into the following categories:

File Management

- Create
- Delete
- Copy
- Rename
- Print
- Dump
- List, and generally manipulate files and directories.

NESO ACADEMY

Status Information

Ask the system for:

- Date, Time
- Amount of available memory or disk space
- Number of users
- Detailed performance
- Logging, and debugging information etc.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

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.

NESO ACADEMY

Programming-language support

- Compilers
- Assemblers
- Debuggers and
- Interpreters

for common programming languages
(such as C, C++, Java, Visual Basic, and PERL)
are often provided to the user with the operating system.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Program loading and execution

Once a program is assembled or compiled, it must be loaded into memory to be executed.

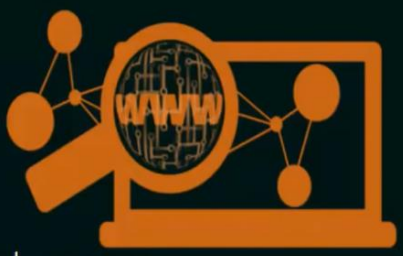
The system may provide:

- Absolute loaders
- Relocatable loaders
- Linkage editors and
- Overlay loaders

Debugging systems for either higher-level languages or machine language are needed as well.

NESO ACADEMY

Communications

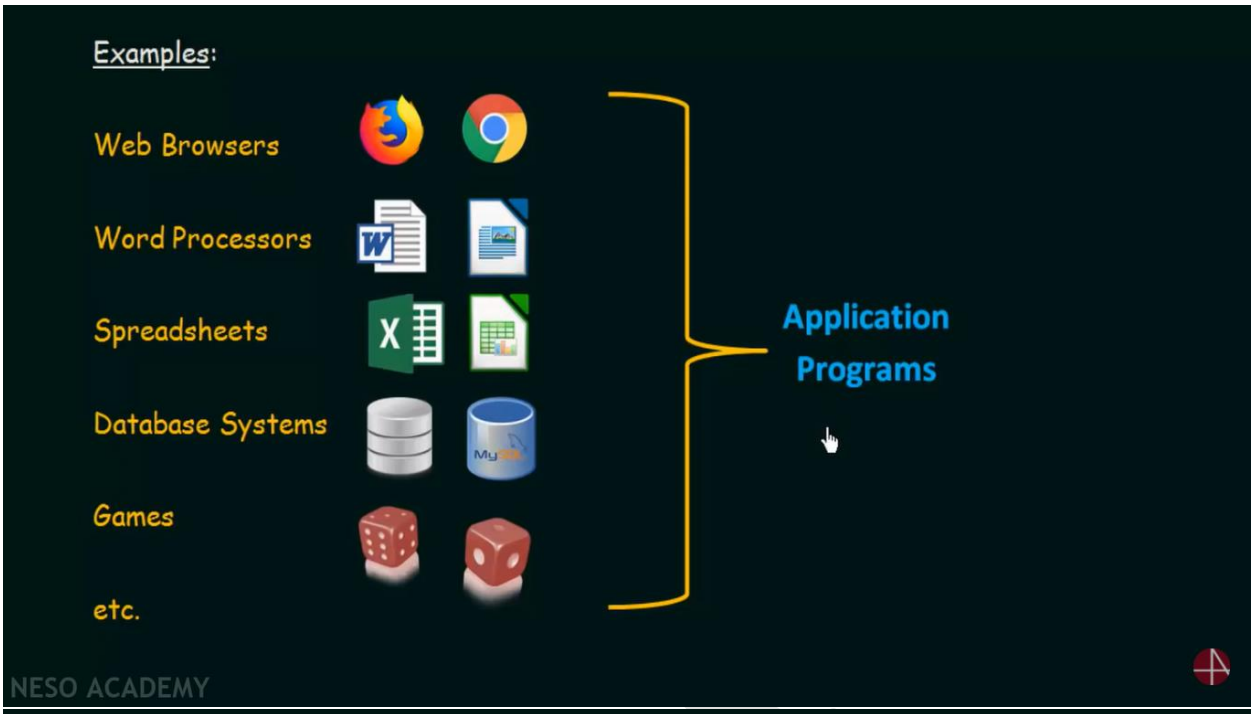


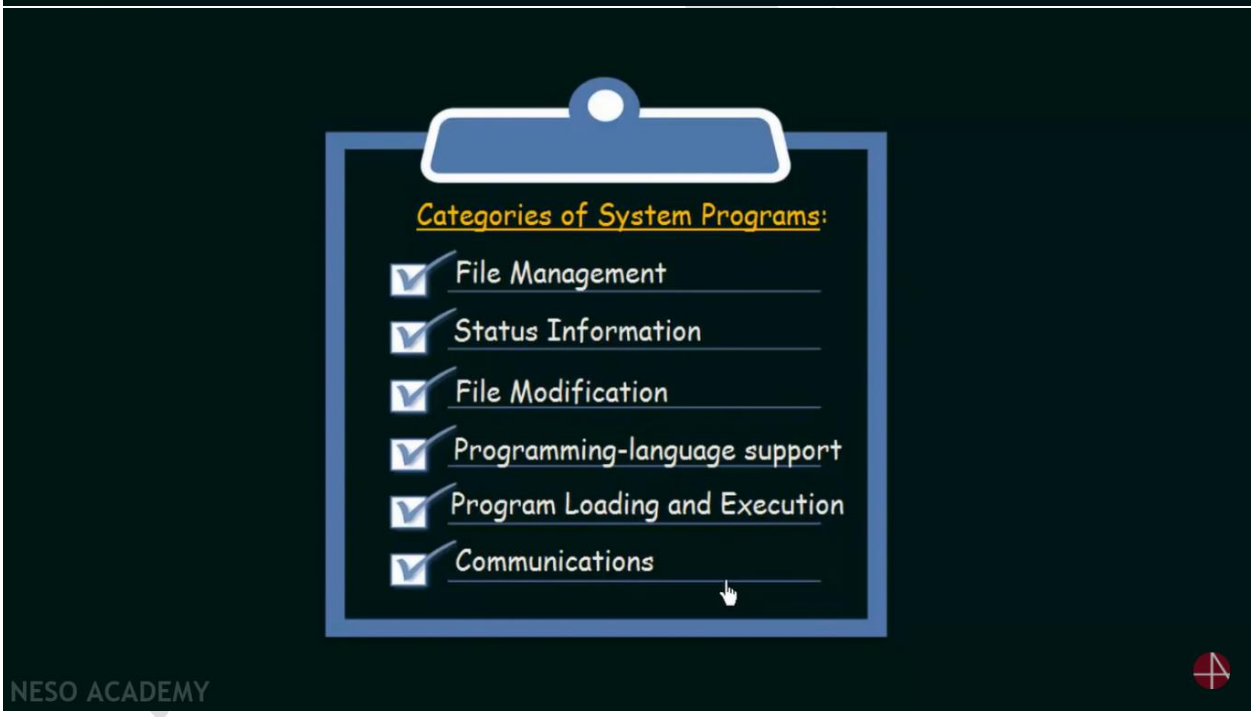
These programs provide the mechanism for:

- Creating virtual connections among processes, users, and computer systems.
- Allowing users to send messages to one another's screens
- To browse webpages
- To send electronic-mail messages
- To log in remotely or to transfer files from one machine to another.

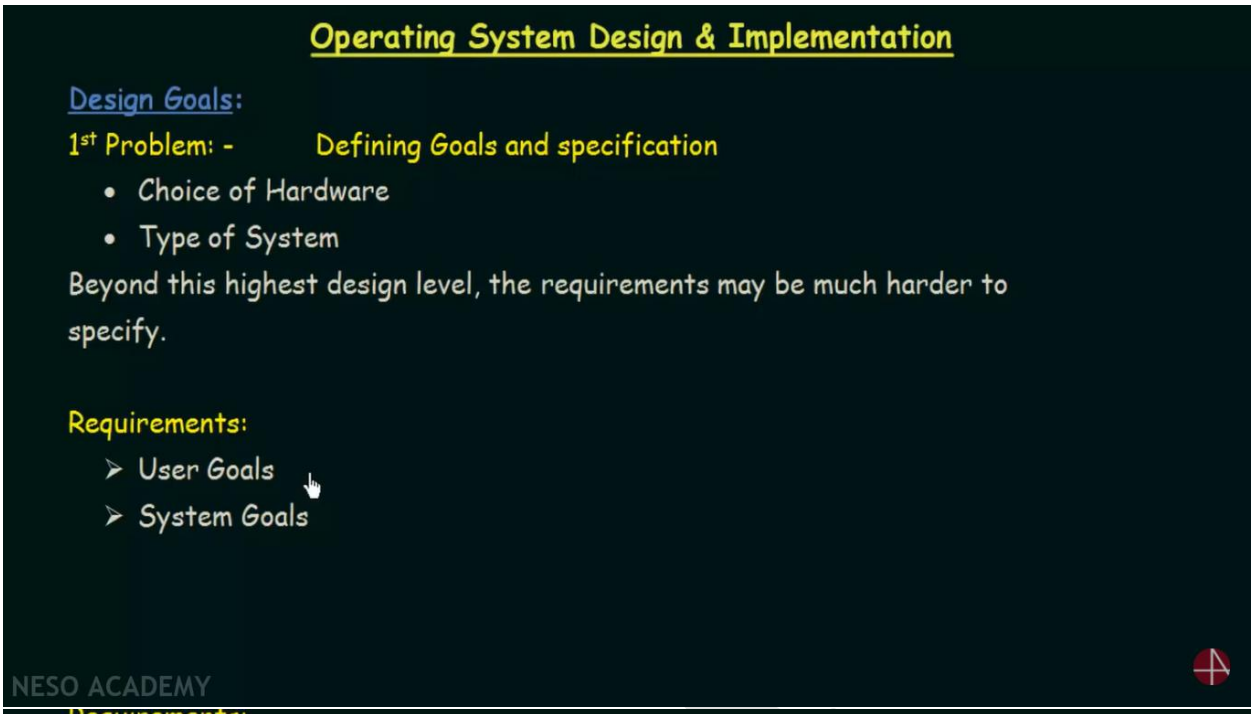
NESO ACADEMY

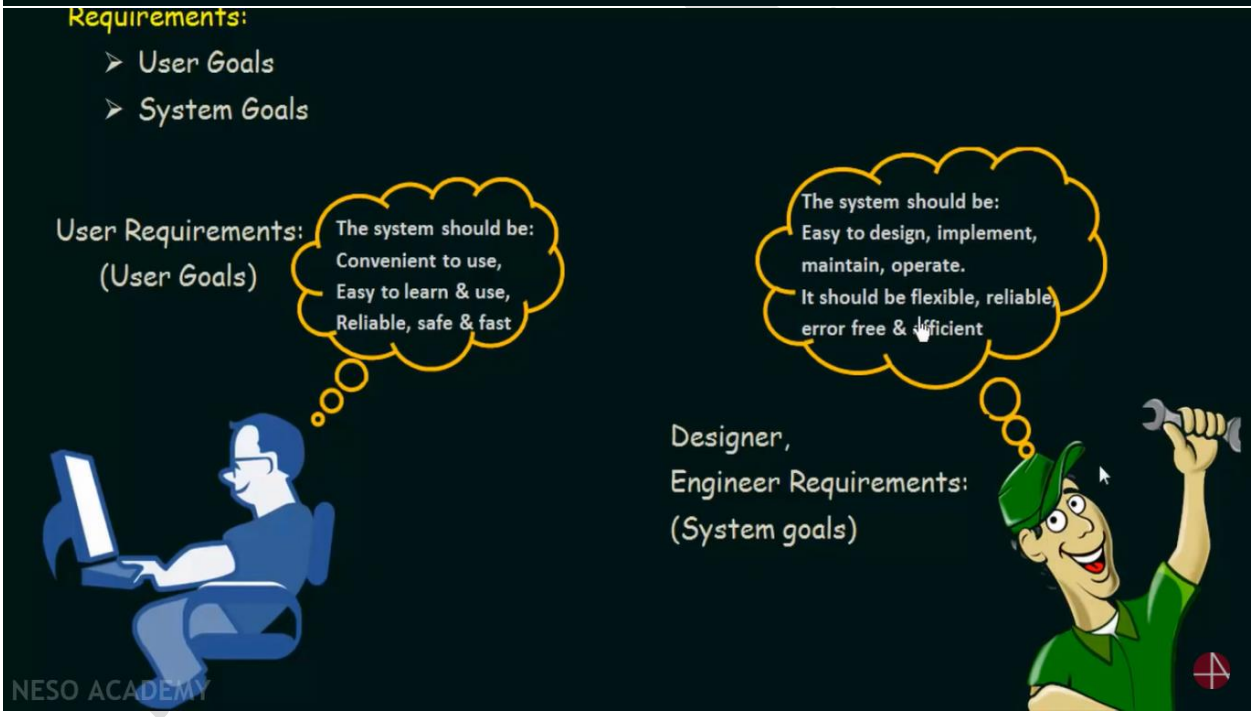
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Mechanisms and Policies:

Mechanisms determine how to do something.

Policies determine what will be done.

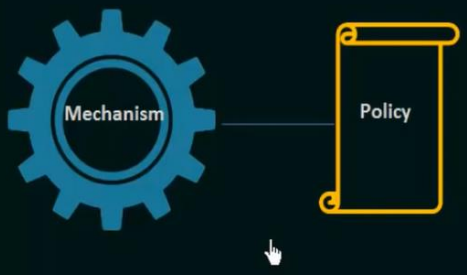
NESO ACADEMY

Mechanisms and Policies:

Mechanisms determine how to do something.

Policies determine what will be done.

One important principle is the separation of policy from mechanism.

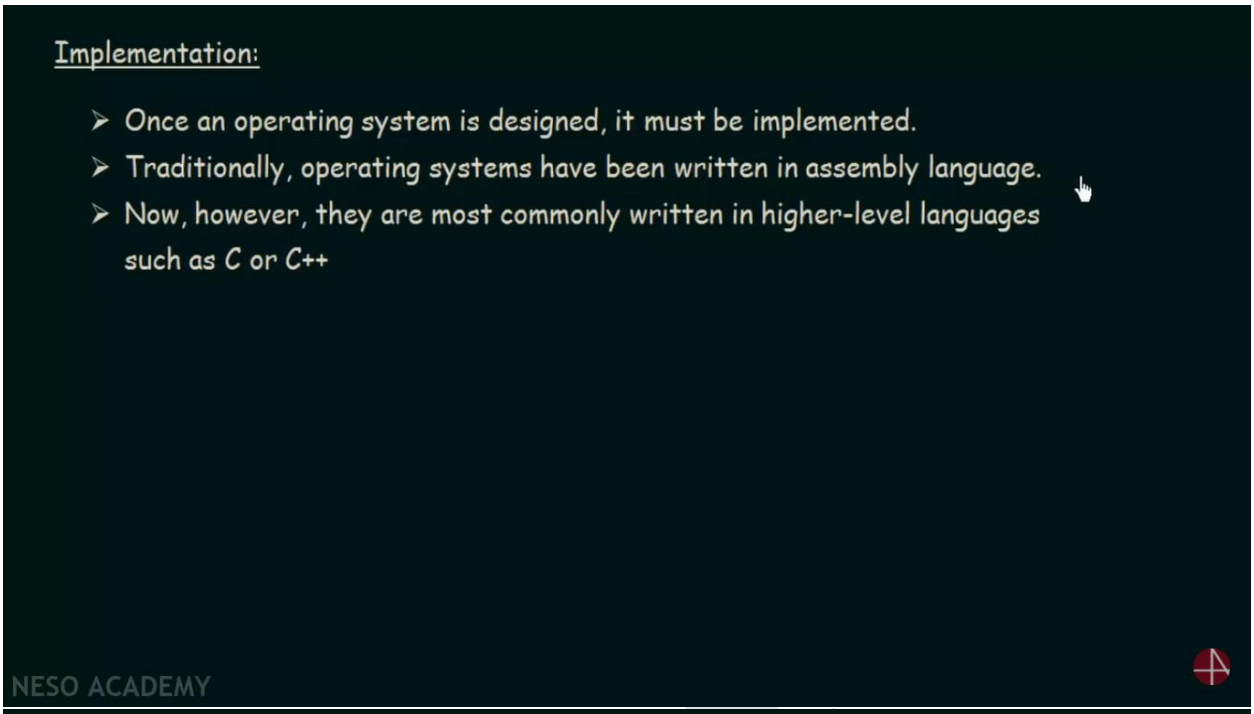


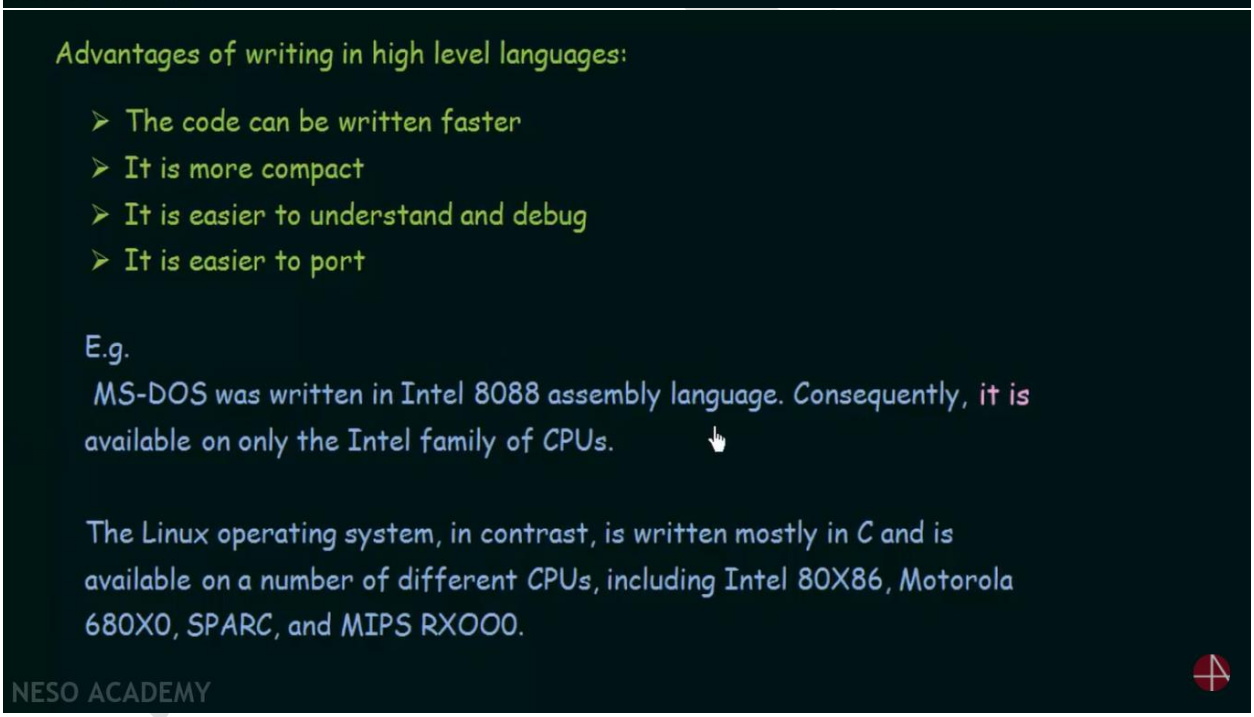
Good and Flexible

Not Good

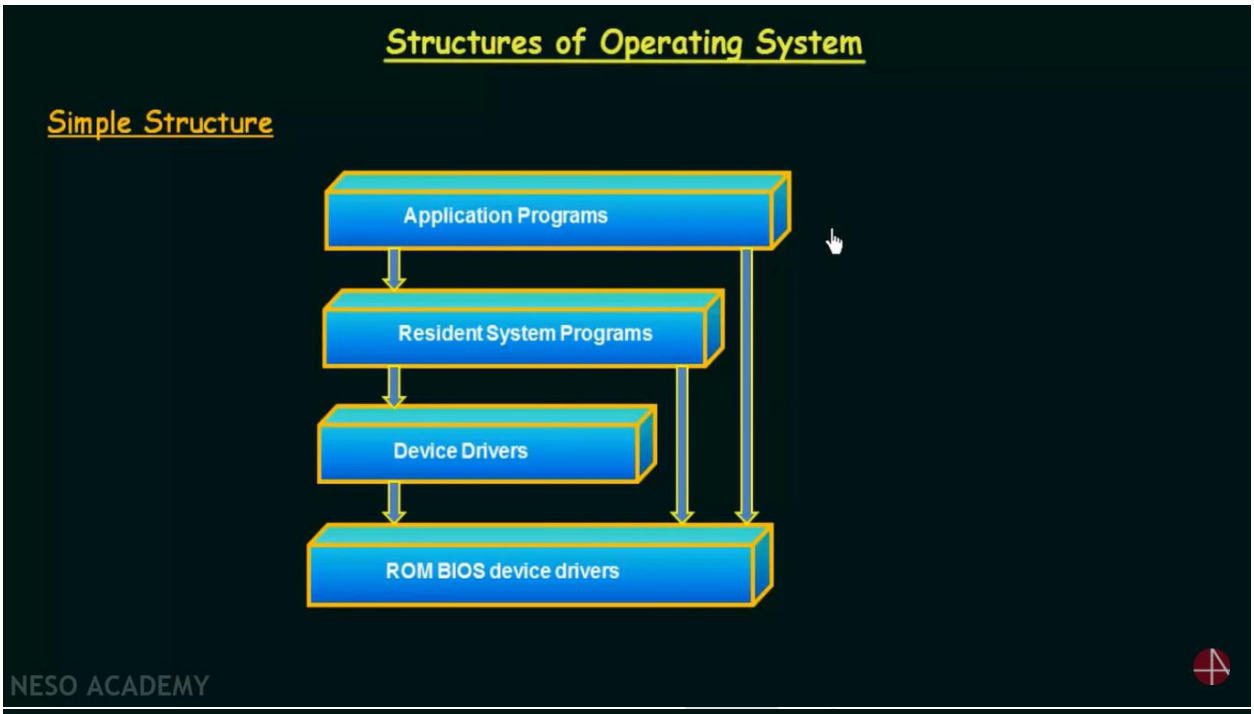
NESO ACADEMY

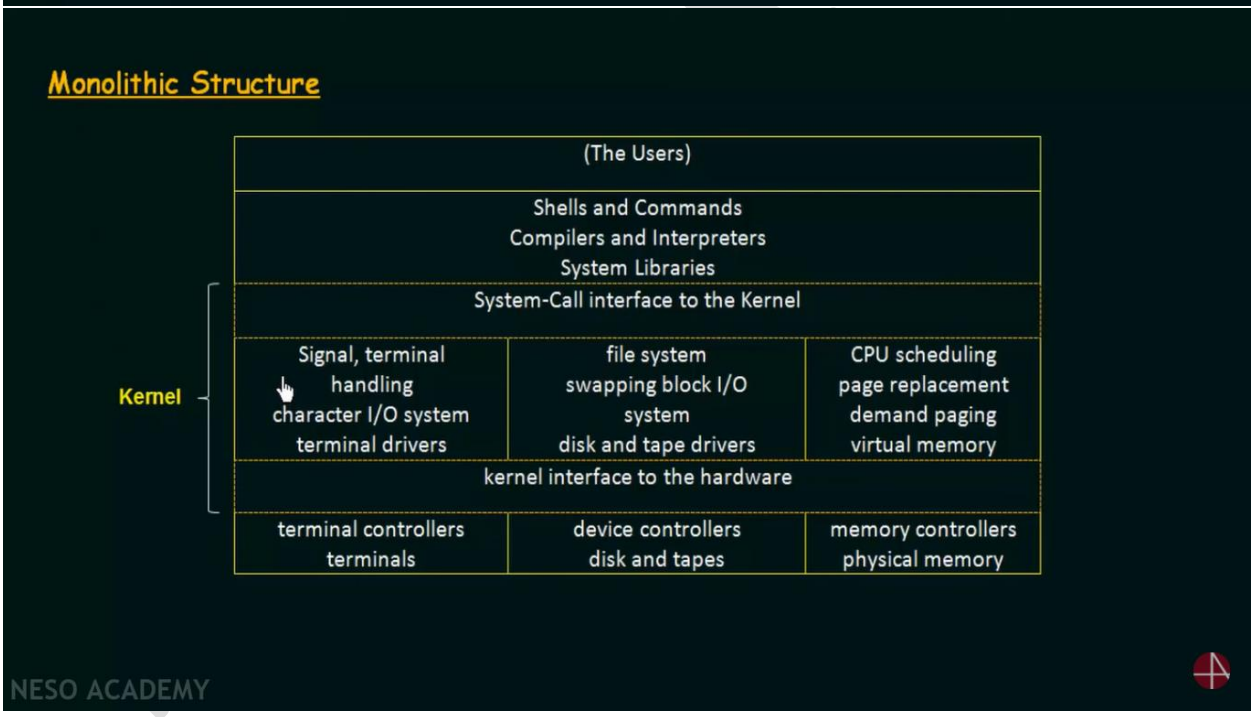
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



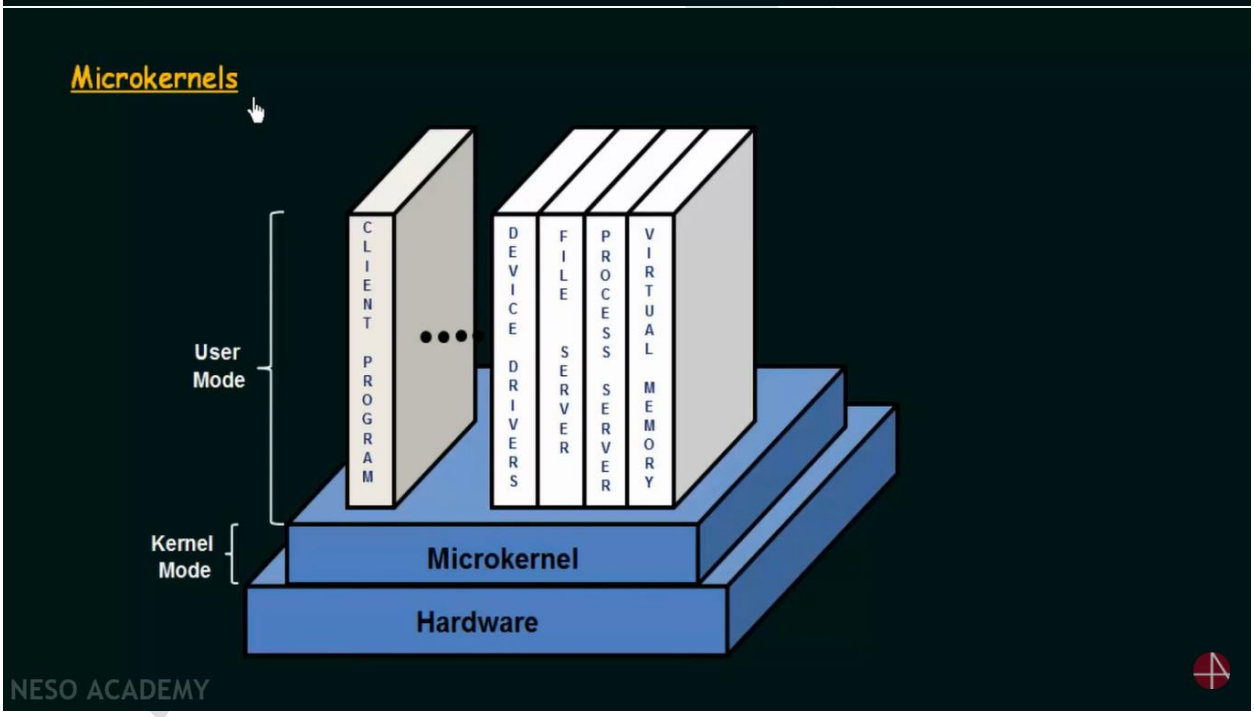


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

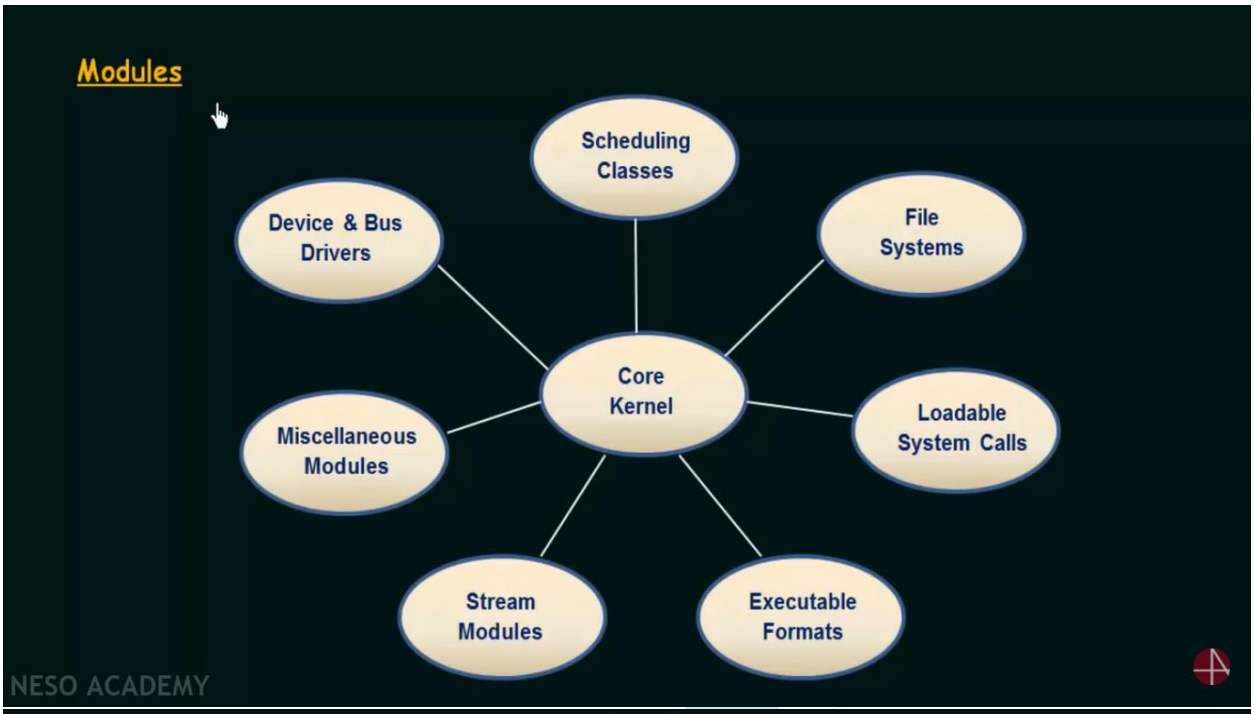


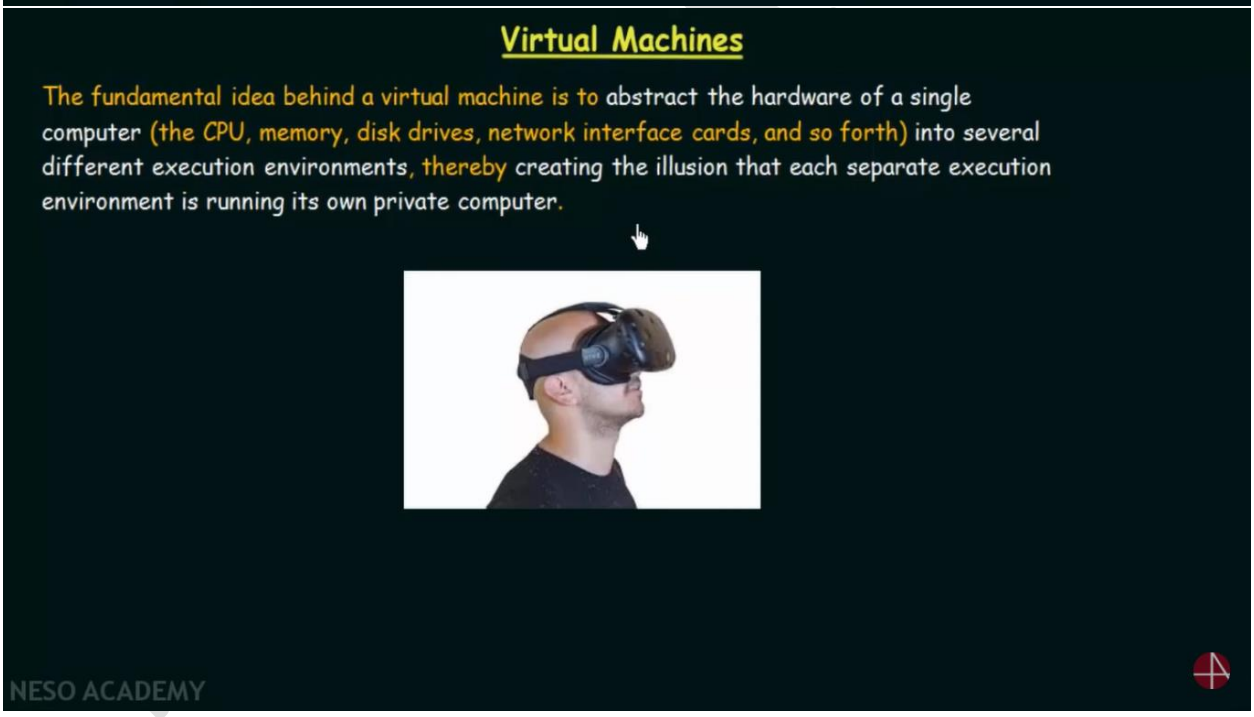


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

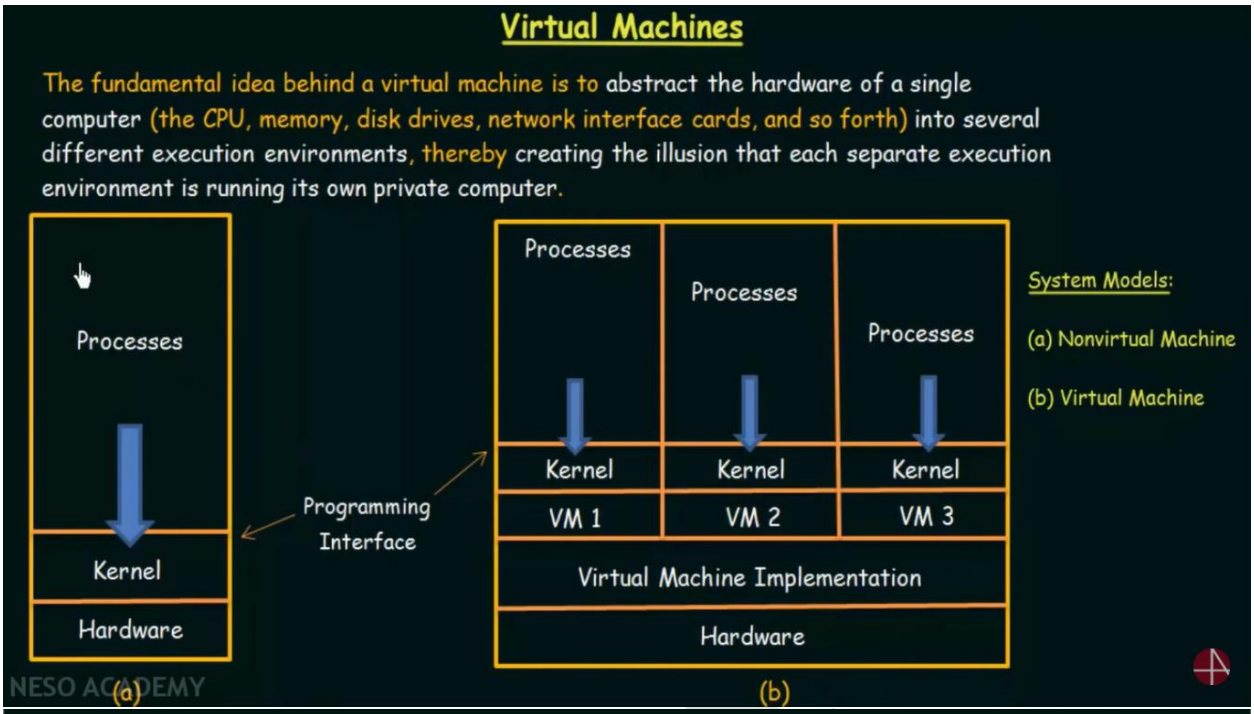


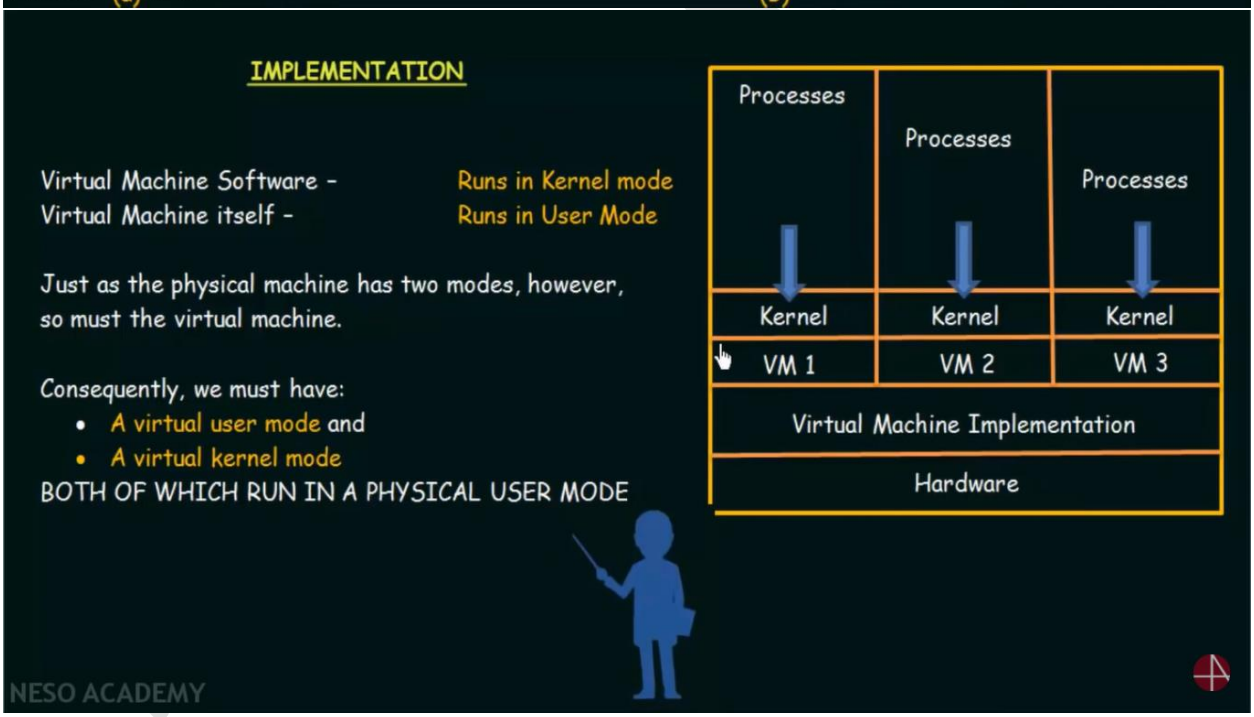
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Operating System Generation & System Boot

- Design, code, and implement an operating system specifically for one machine at one site
- Operating systems are designed to run on any of a class of machines at a variety of sites with a variety of peripheral configurations
- The system must then be configured or generated for each specific computer site, a process sometimes known as system generation (SYSGEN) is used for this

The following kinds of information must be determined by the SYSGEN Program:

- What CPU is to be used?
- How much memory is available?
- What devices are available?
- What operating-system options are desired?

NESO ACADEMY

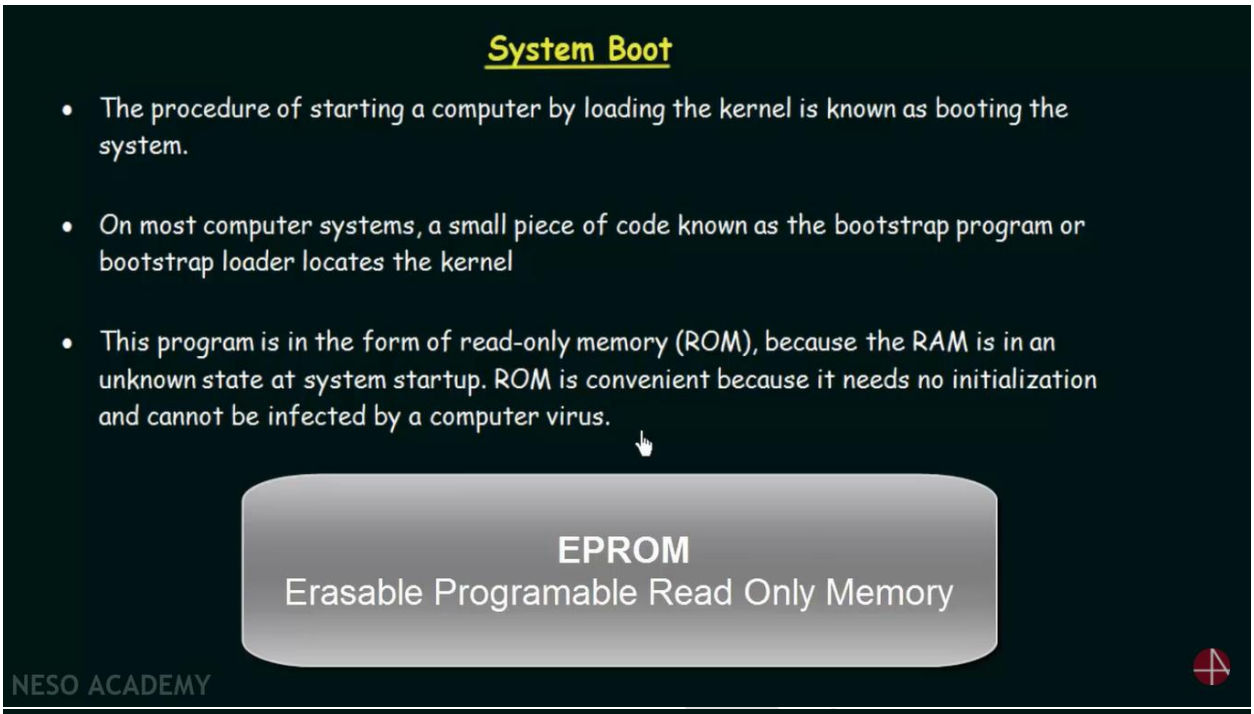
System Boot

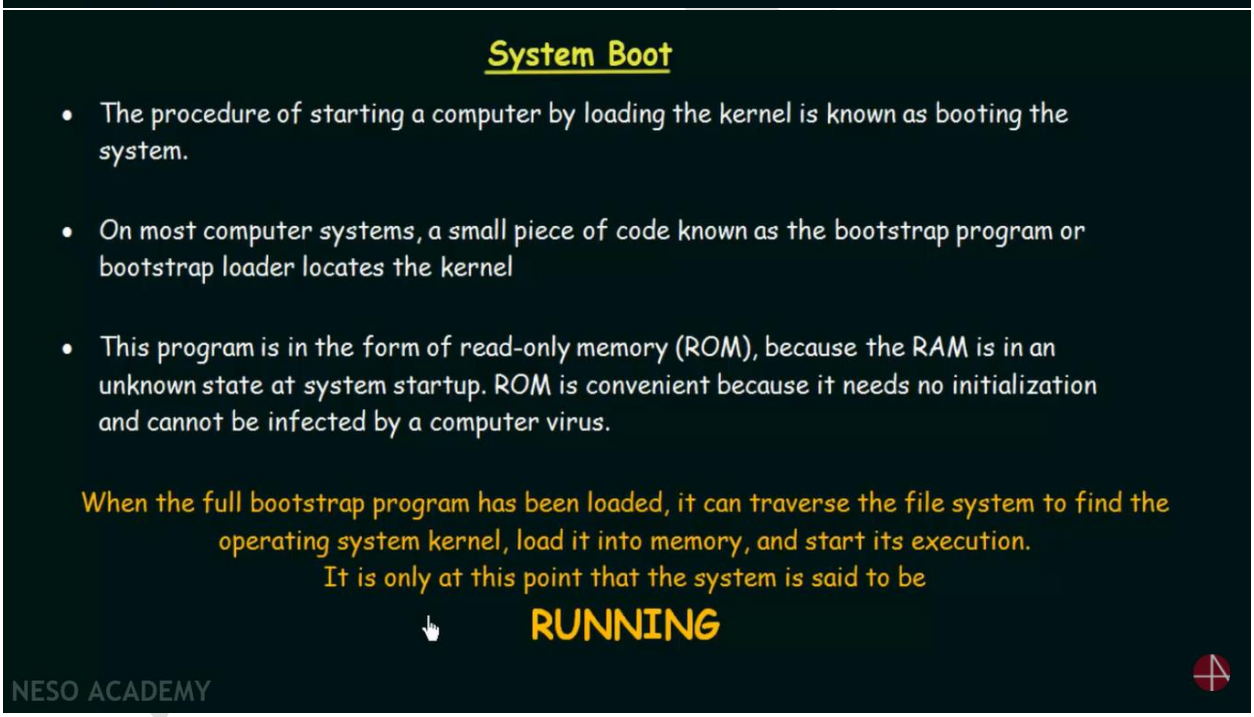
- The procedure of starting a computer by loading the kernel is known as booting the system.
- On most computer systems, a small piece of code known as the bootstrap program or bootstrap loader locates the kernel
- This program is in the form of read-only memory (ROM), because the RAM is in an unknown state at system startup. ROM is convenient because it needs no initialization and cannot be infected by a computer virus.

Firmware

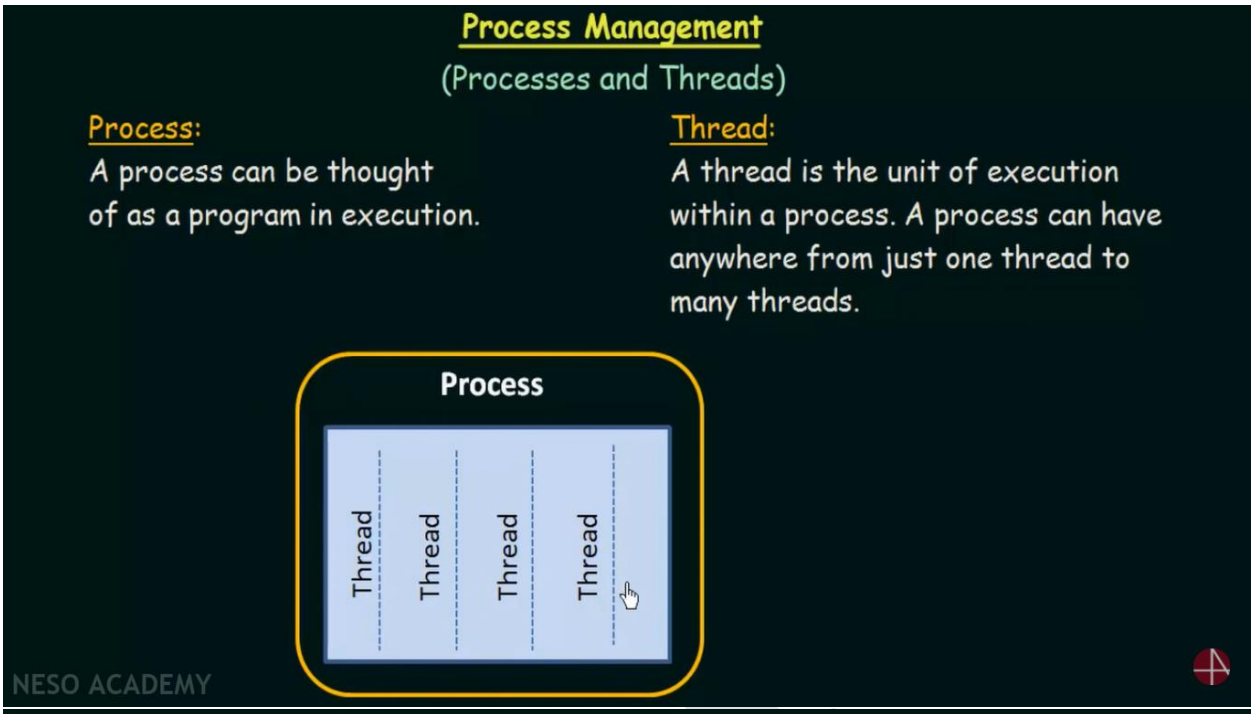
NESO ACADEMY

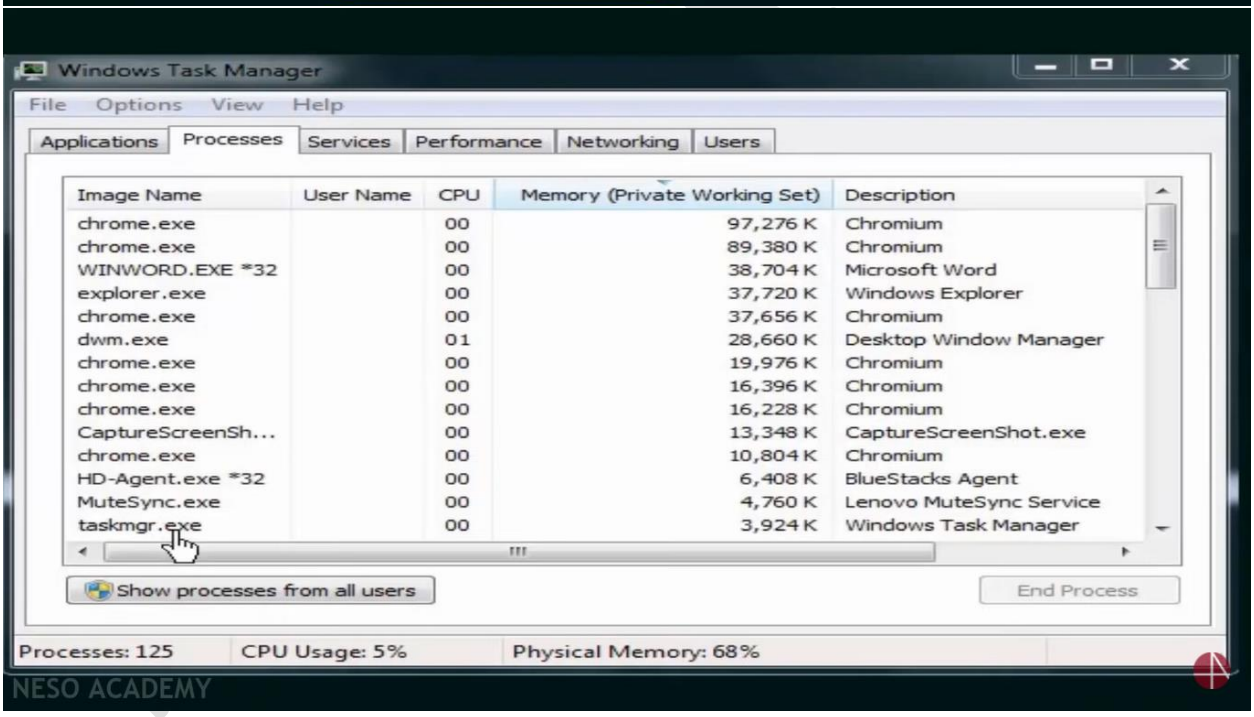
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



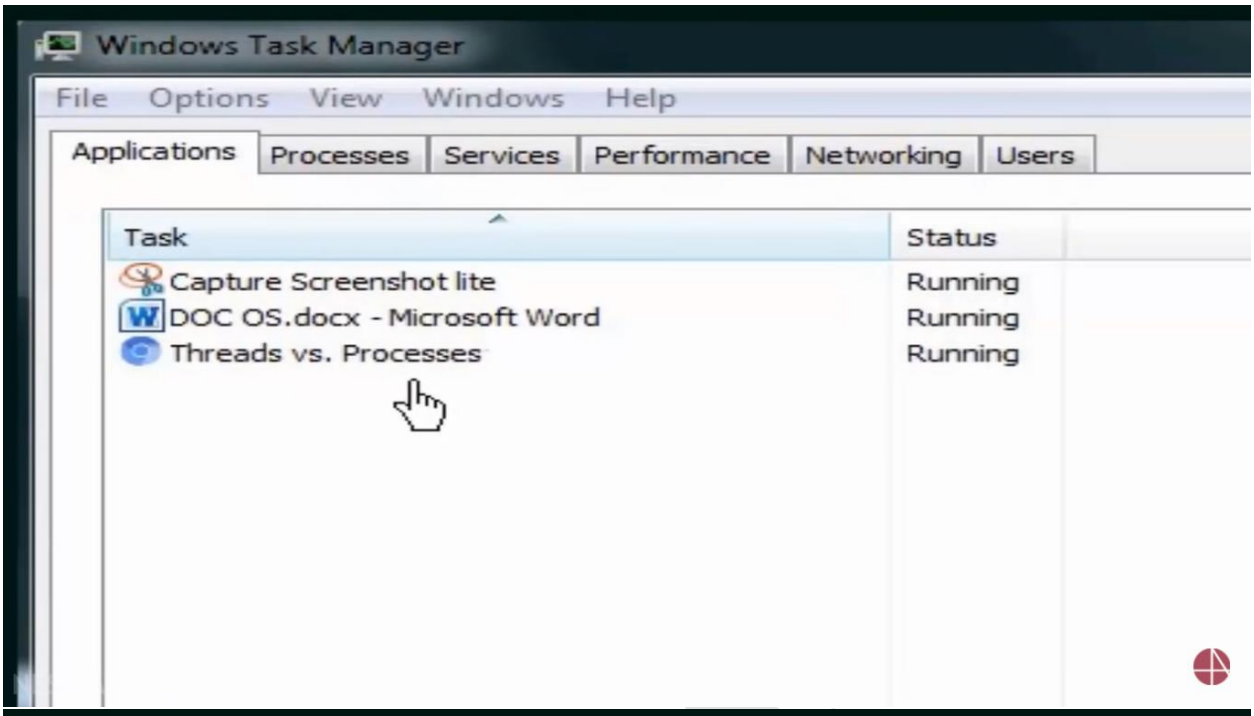


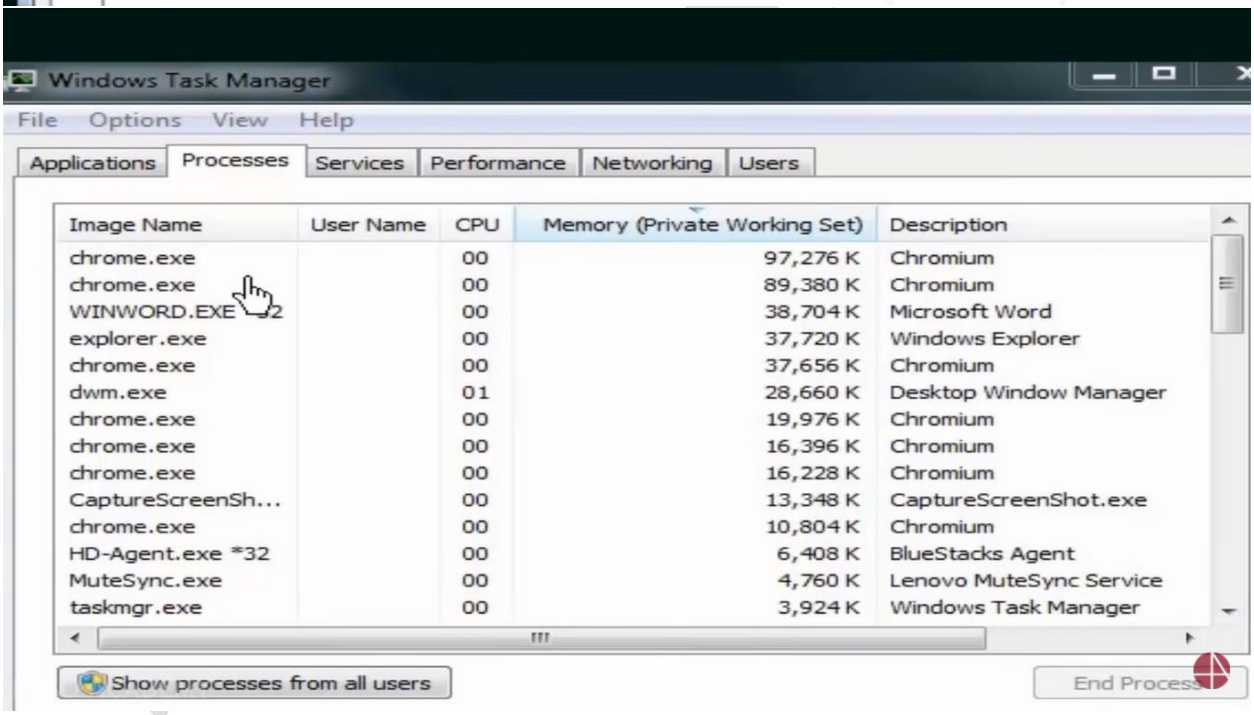
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



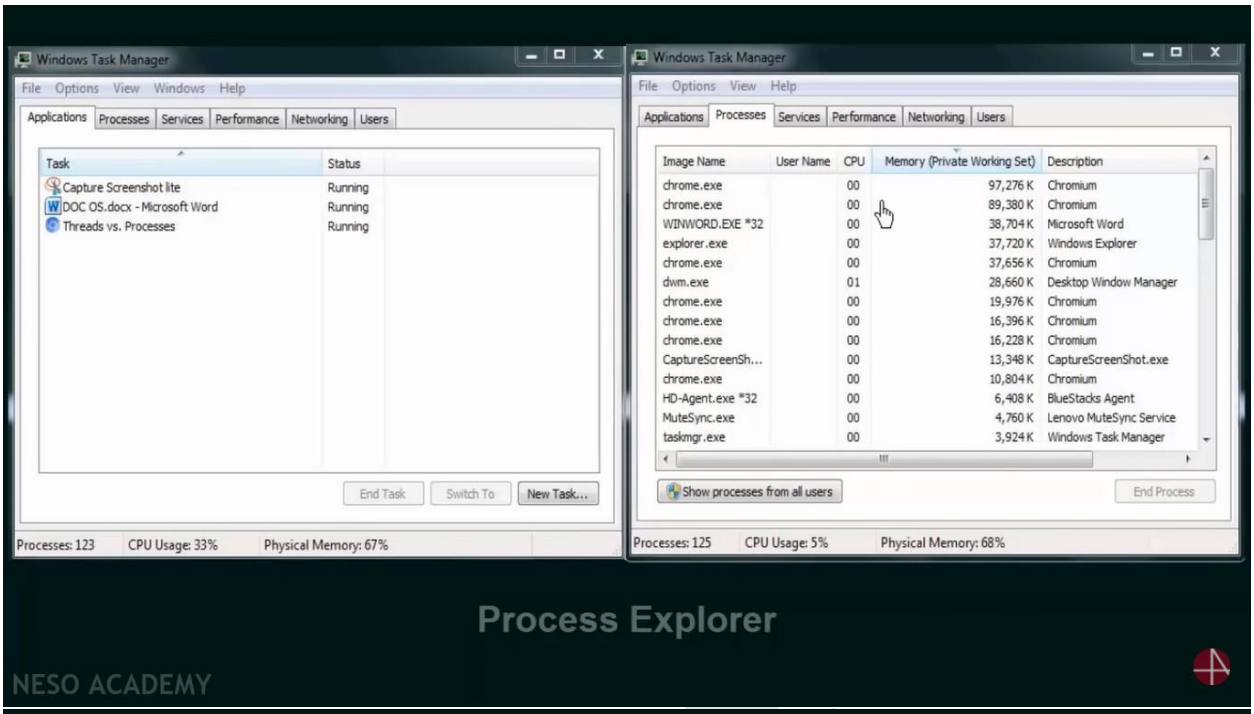


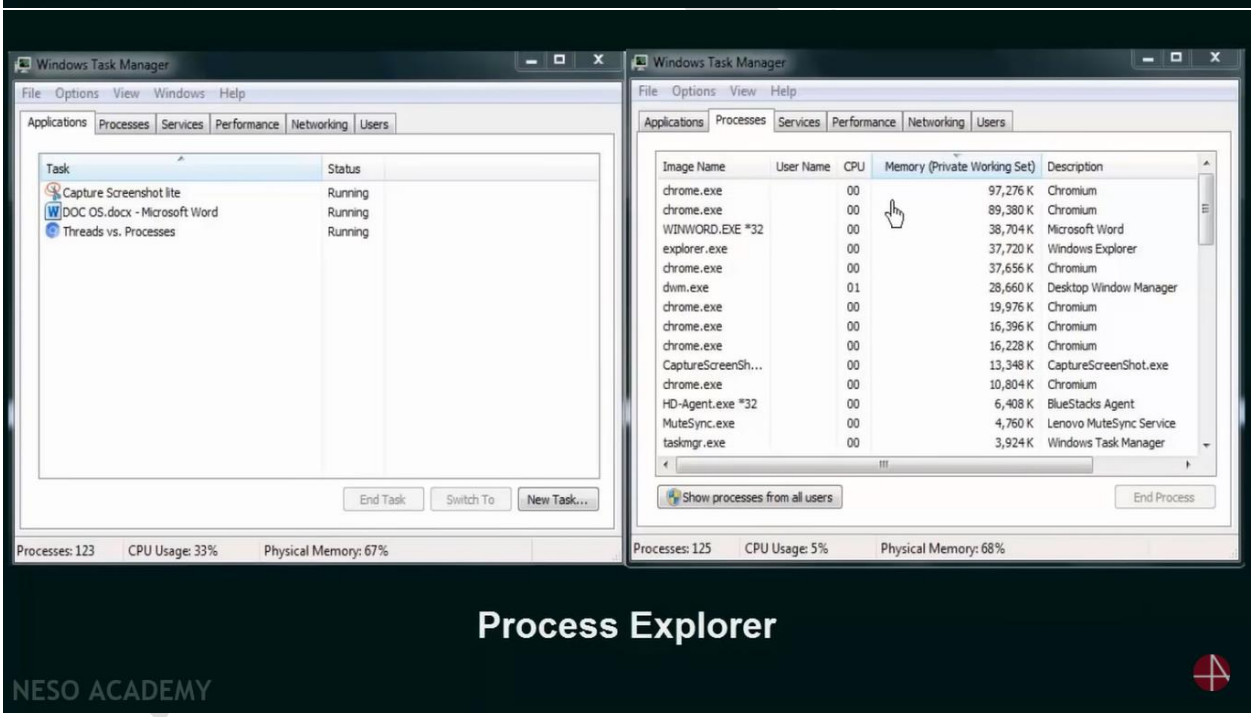
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



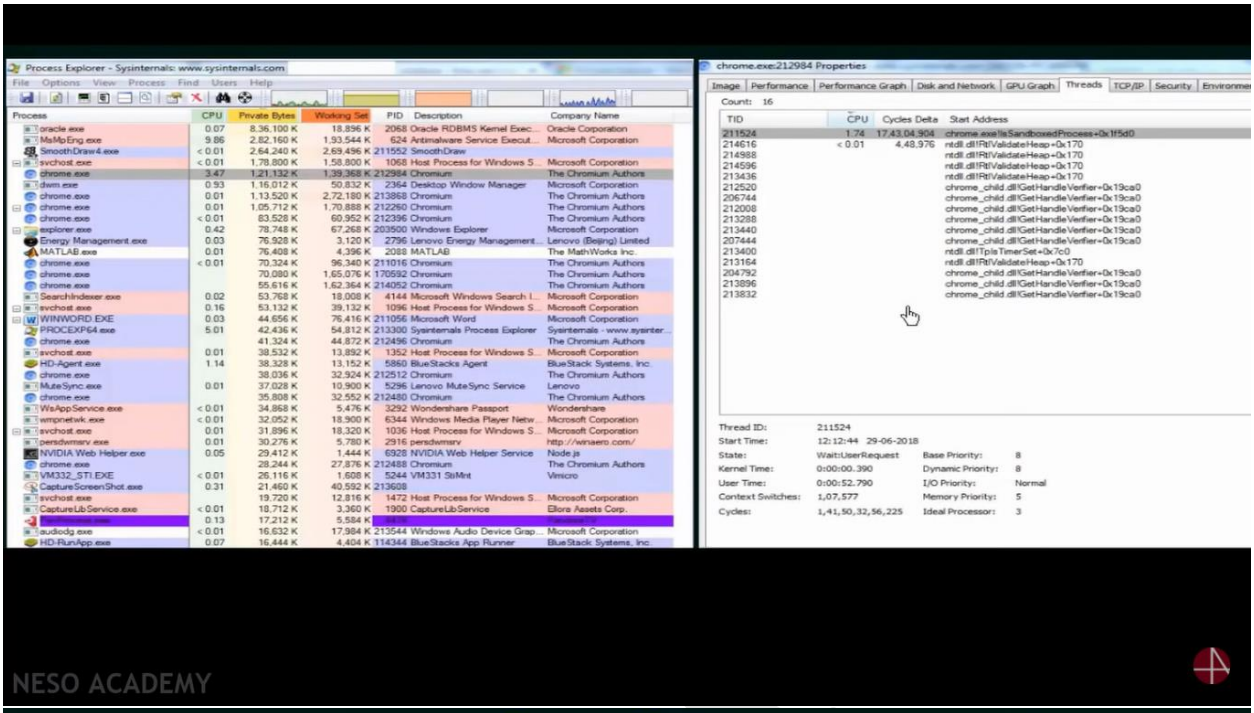


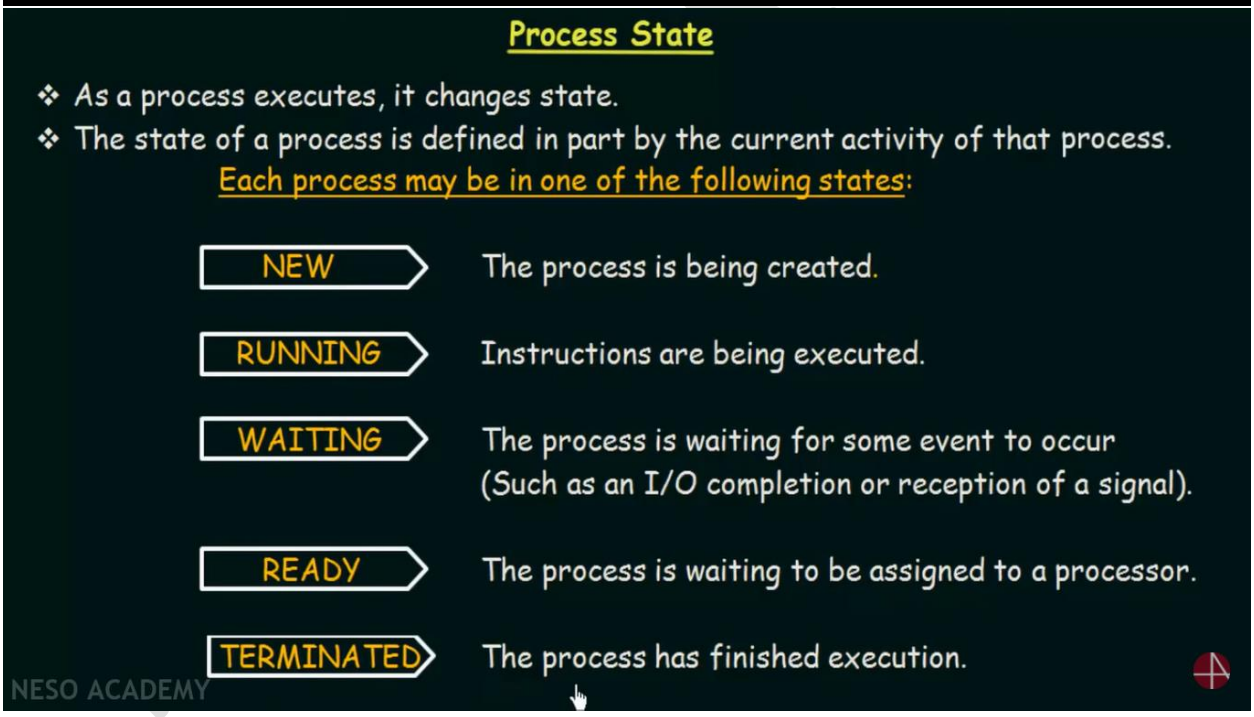
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

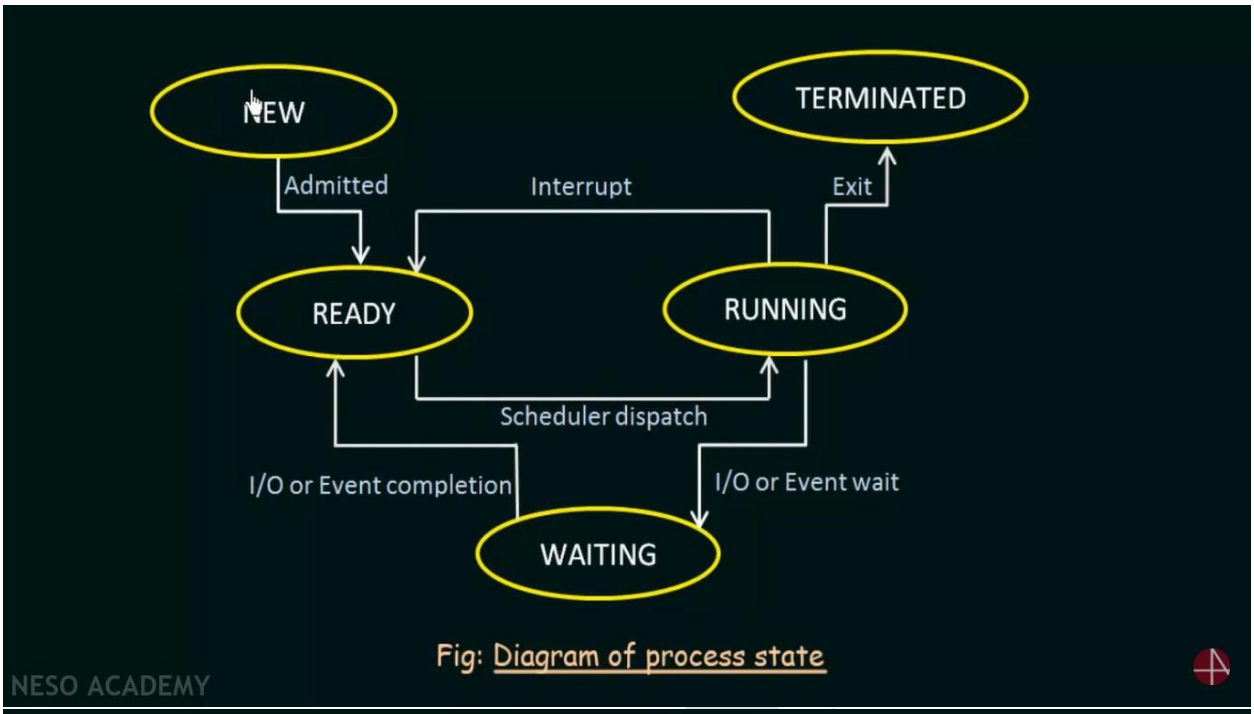
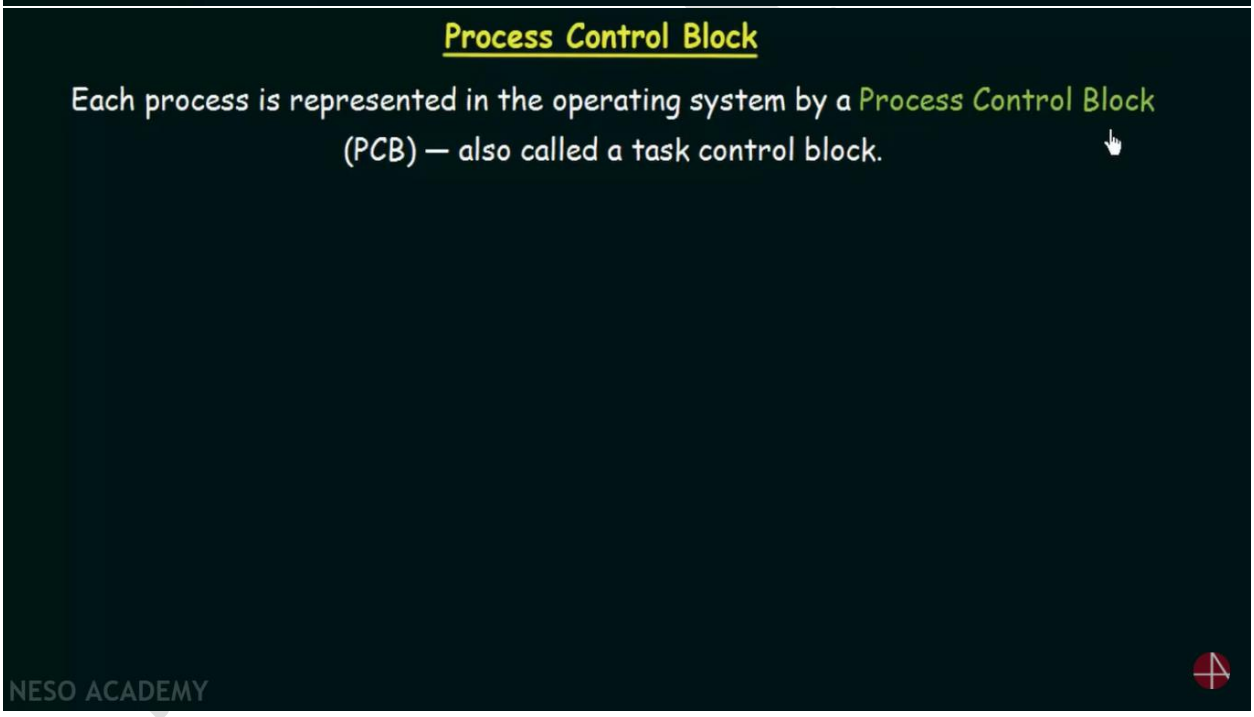
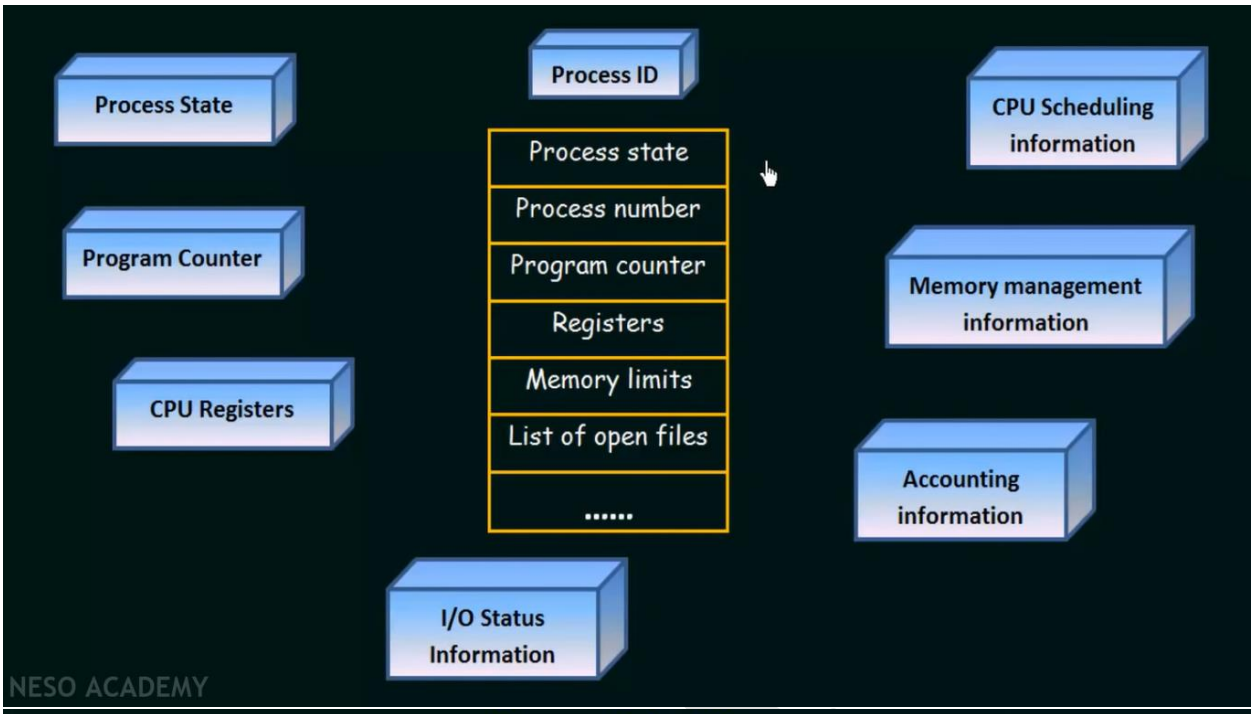


Fig: Diagram of process state



NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

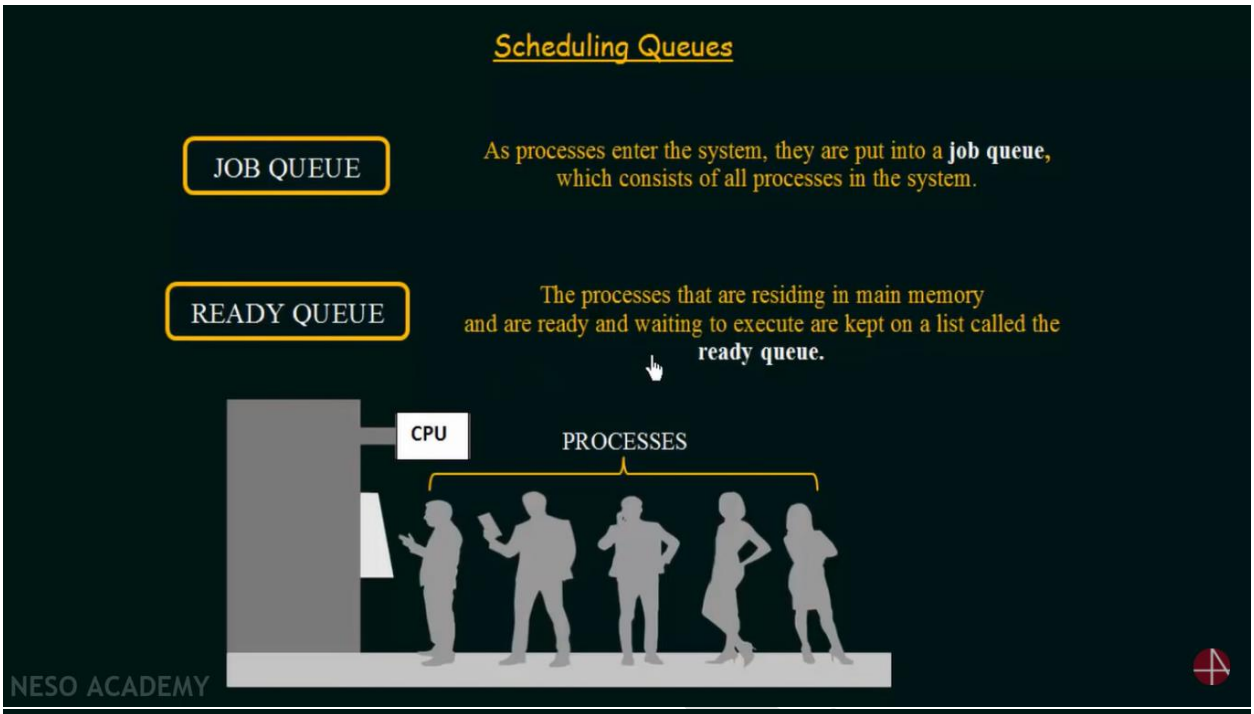


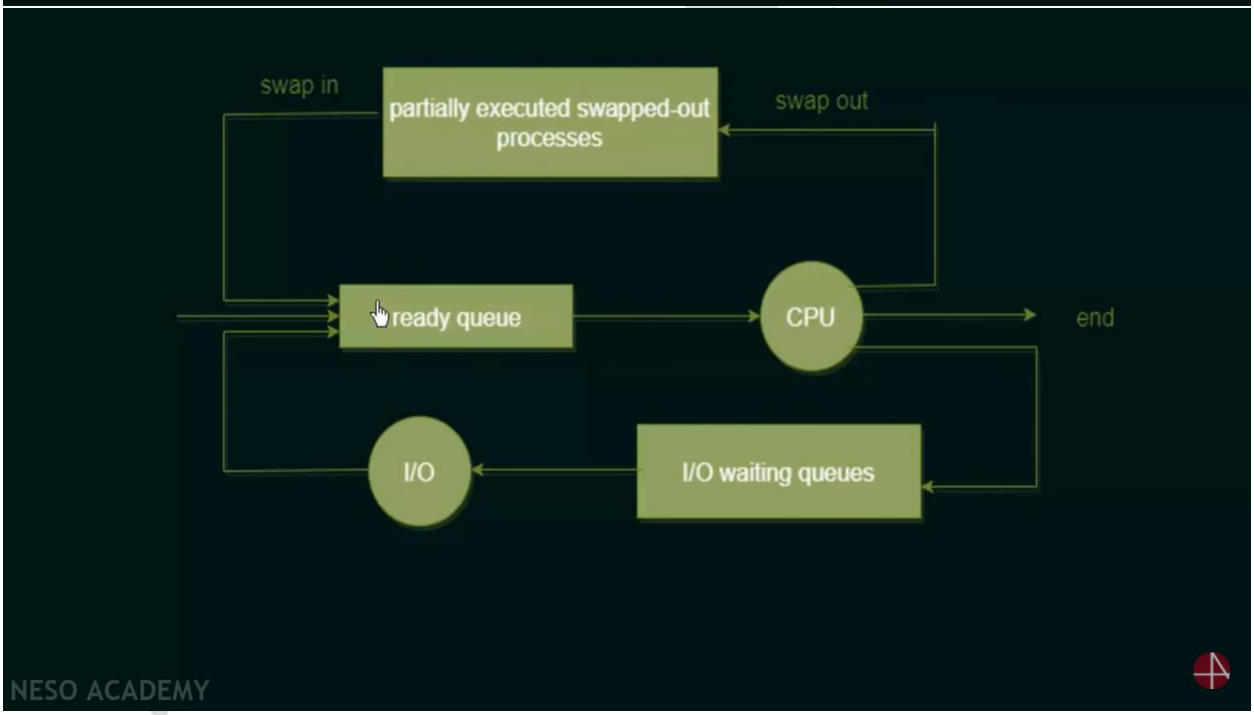
Process Scheduling

- The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
- The objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running.
- To meet these objectives, the process scheduler selects an available process (possibly from a set of several available processes) for program execution on the CPU.
  - For a single-processor system, there will never be more than one running process.
  - If there are more processes, the rest will have to wait until the CPU is free and can be rescheduled.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Context Switch

- Interrupts cause the operating system to change a CPU from its current task and to run a kernel routine.
- Such operations happen frequently on general-purpose systems.

When an interrupt occurs, the system needs to save the current **context** of the process currently running on the CPU so that it can restore that context when its processing is done, essentially suspending the process and then resuming it.

- The context is represented in the PCB of the process

Process Control Block

NESO ACADEMY

Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process.

State Save

State Restore

This task is known as a **context switch**.

- Context-switch time is pure overhead, because the system does no useful work while switching.
- Its speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions (such as a single instruction to load or store all registers).
- Typical speeds are a few milliseconds.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Operations on Processes
(Process Creation)

- A process may create several new processes, via a create-process system call, during the course of execution.
- The creating process is called a parent process, and the new processes are called the children of that process.
- Each of these new processes may in turn create other processes, forming a tree of processes.

NESO ACADEMY

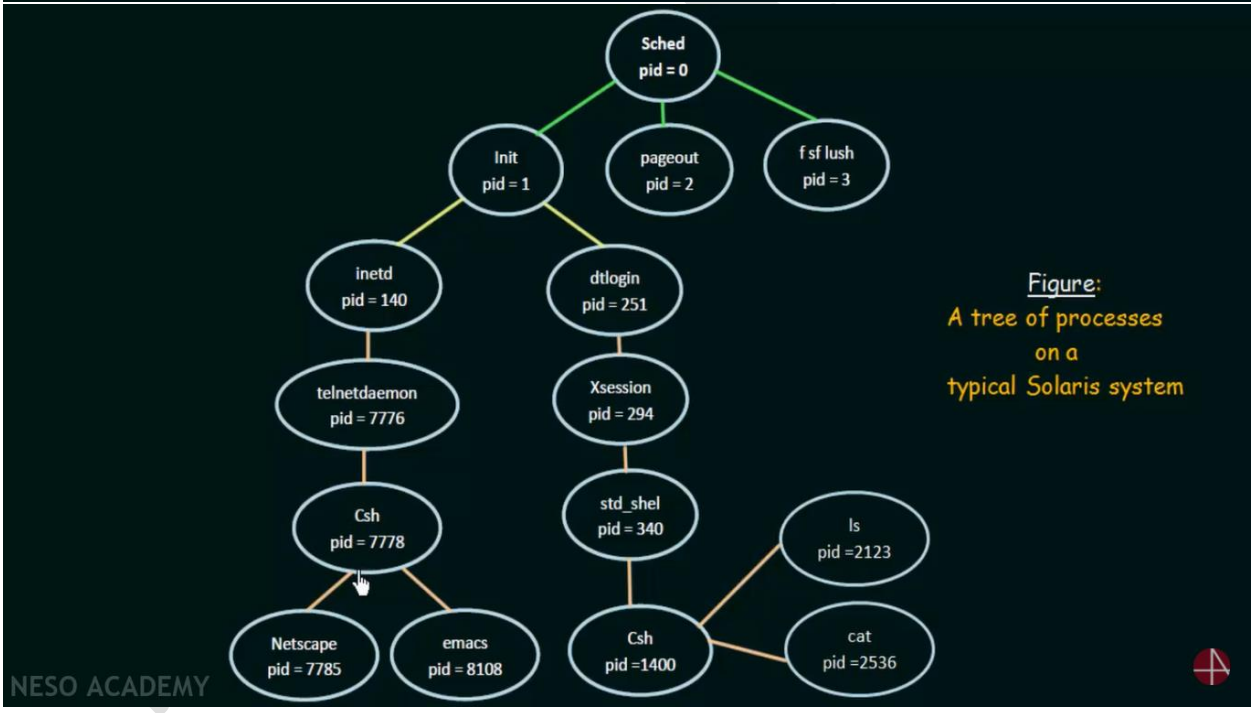


Figure:
A tree of processes
on a
typical Solaris system

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

When a process creates a new process, two possibilities exist in terms of execution:

1. The parent continues to execute concurrently with its children.
2. The parent waits until some or all of its children have terminated.

There are also two possibilities in terms of the address space of the new process:

1. The child process is a duplicate of the parent process (it has the same program and data as the parent).
2. The child process has a new program loaded into it.

NESO ACADEMY

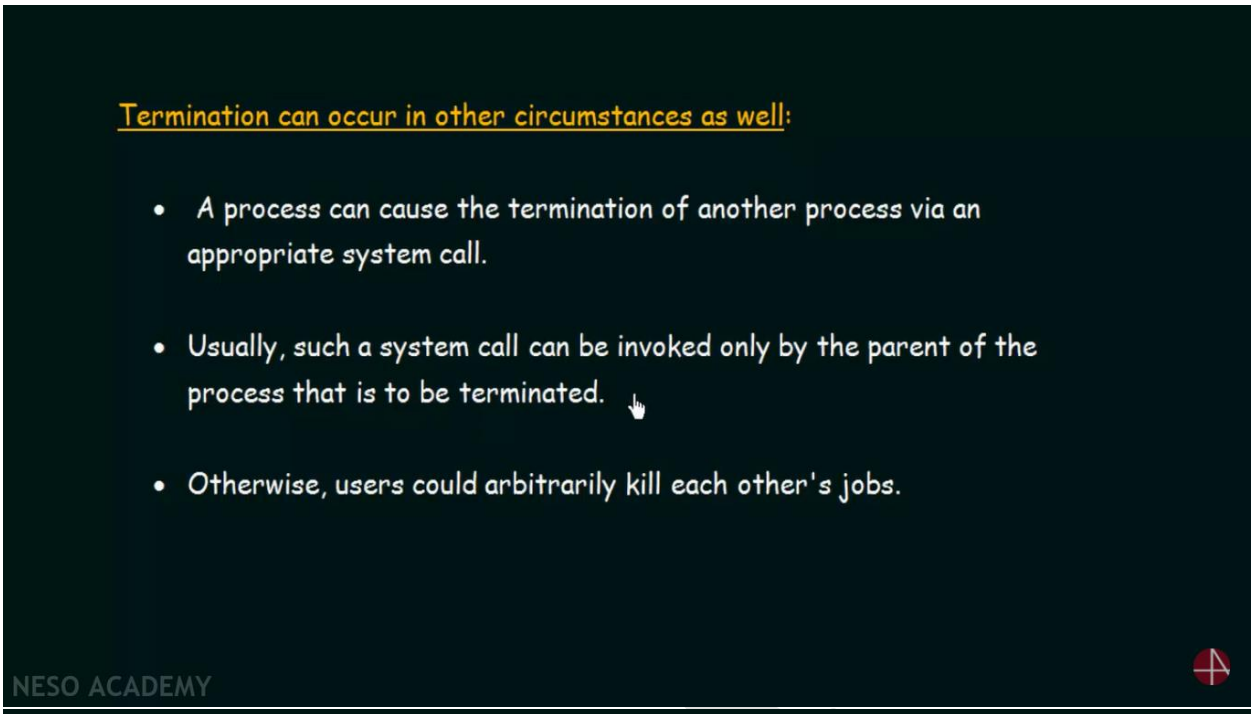
Operations on Processes

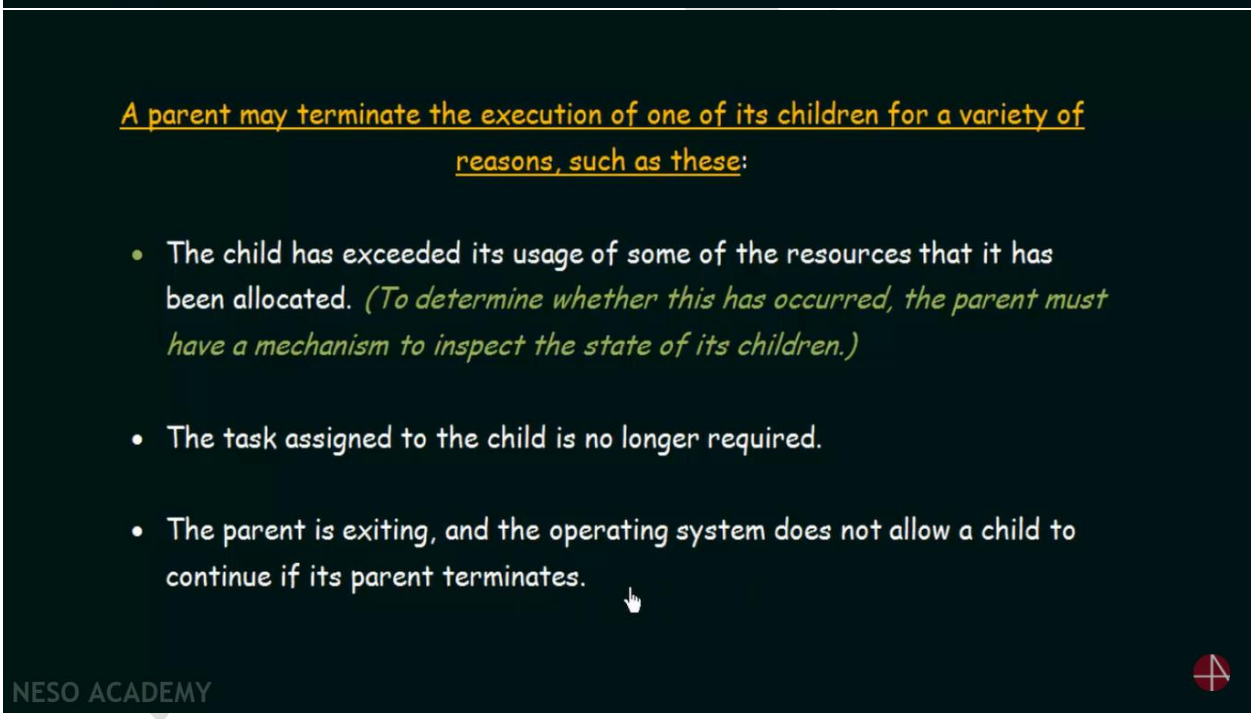
(Process Termination)

- A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the exit () system call.
- At that point, the process may return a status value (typically an integer) to its parent process (via the wait() system call).
- All the resources of the process—including physical and virtual memory, open files, and I/O buffers—are deallocated by the operating system.

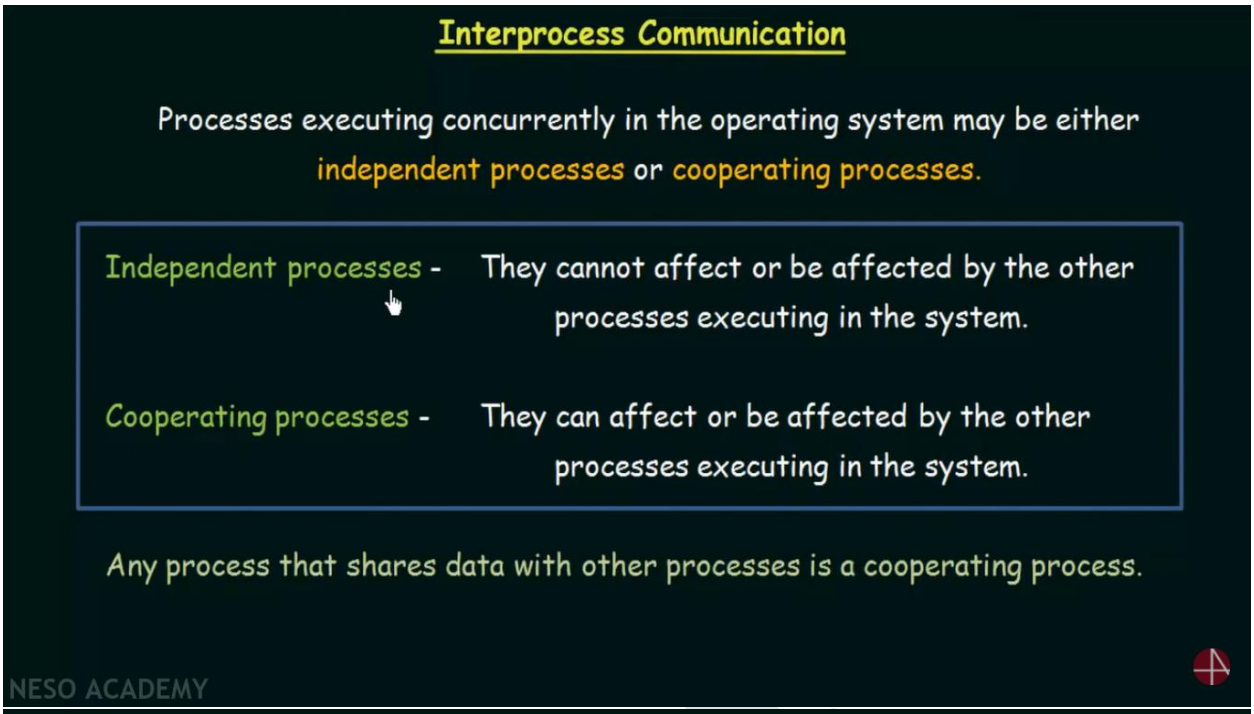
NESO ACADEMY

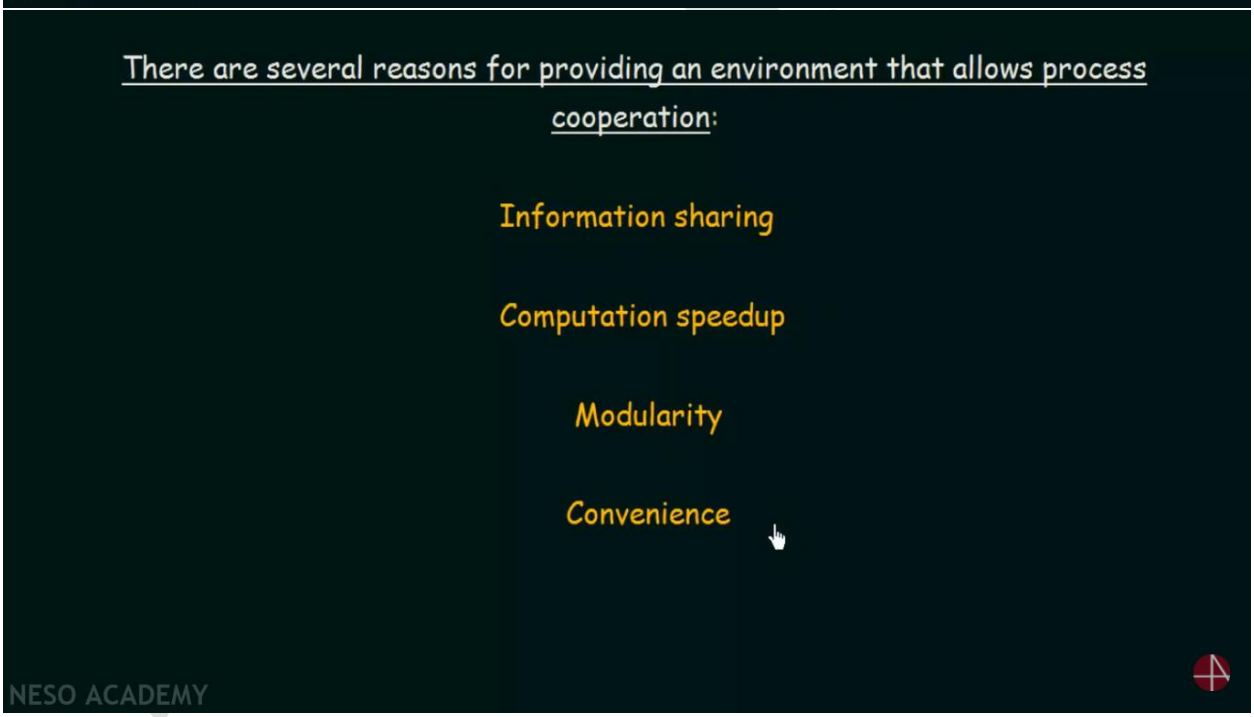
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Cooperating processes require an interprocess communication (IPC) mechanism that will allow them to exchange data and information.

There are two fundamental models of interprocess communication:

(1) Shared memory

(2) Message passing

- In the shared-memory model, a region of memory that is shared by cooperating processes is established.
  Processes can then exchange information by reading and writing data to the shared region.
- In the message passing model, communication takes place by means of messages exchanged between the cooperating processes.

NESO ACADEMY

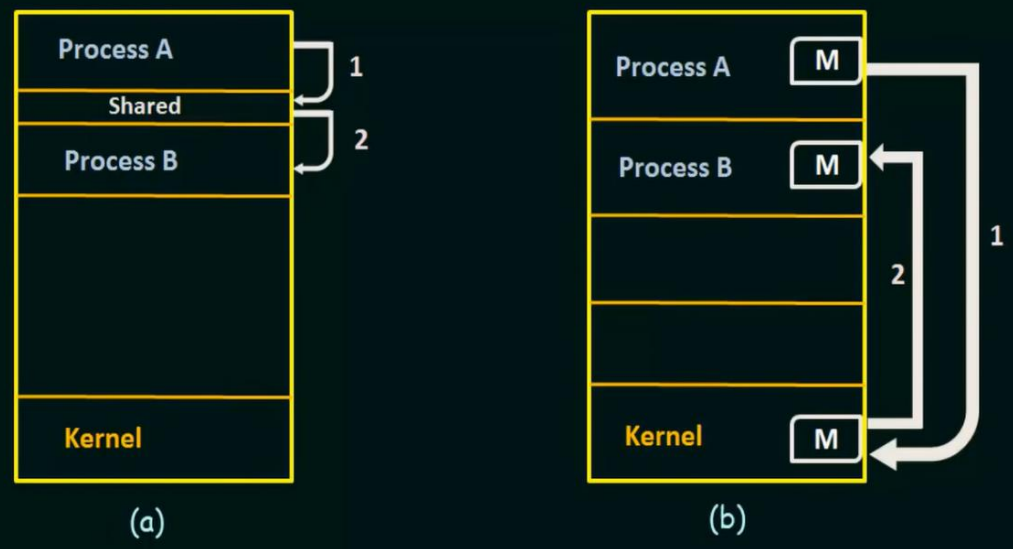


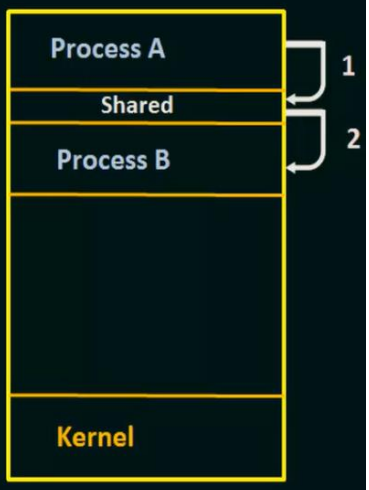
Fig: Communications models, (a) Shared memory, (b) Message Passing.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Shared Memory Systems

- Interprocess communication using shared memory requires communicating processes to establish a region of shared memory.
- Typically, a shared-memory region resides in the address space of the process creating the shared-memory segment.
- Other processes that wish to communicate using this shared-memory segment must attach it to their address space.
- Normally, the operating system tries to prevent one process from accessing another process's memory.
- Shared memory requires that two or more processes agree to remove this restriction.



NESO ACADEMY

Producer Consumer Problem

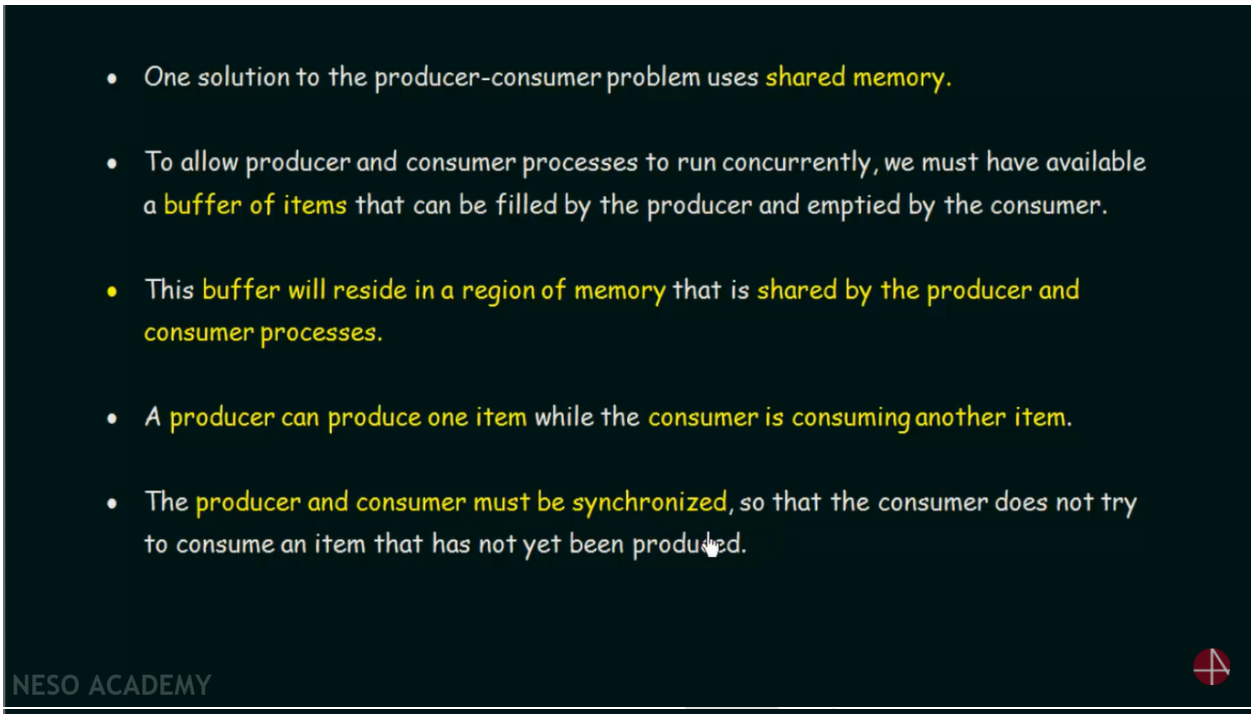
A producer process produces information that is consumed by a consumer process.

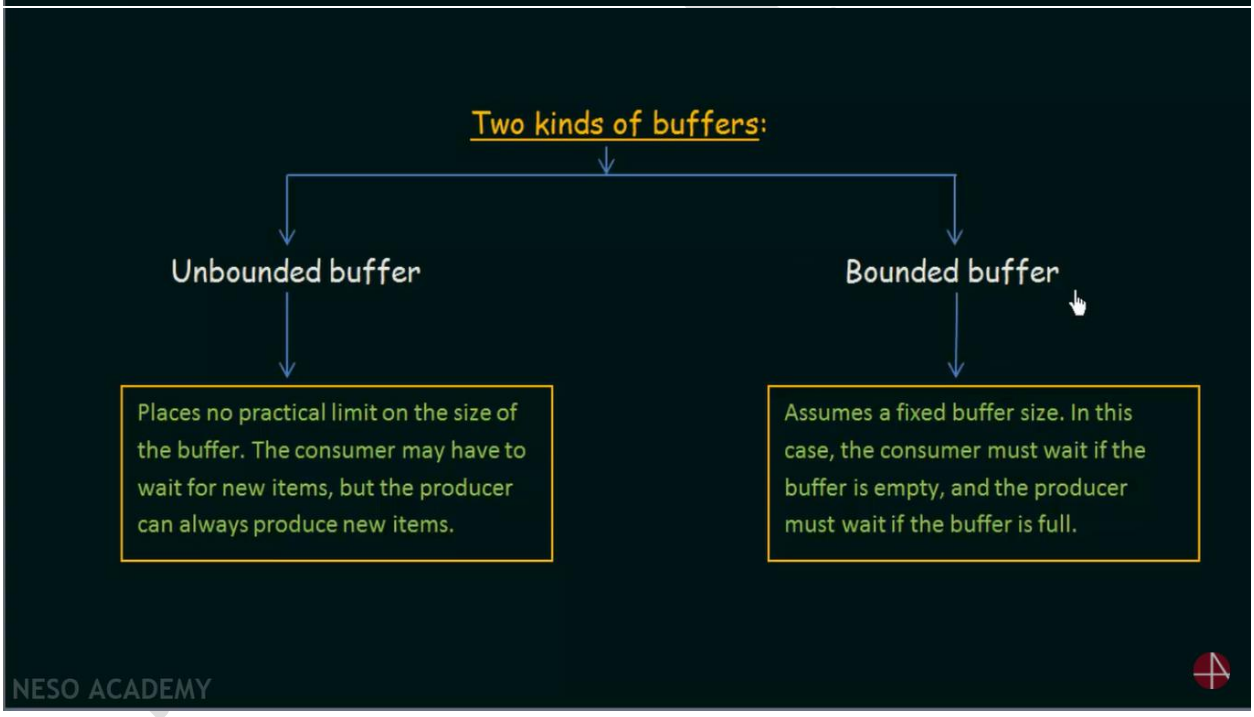
For example, a compiler may produce assembly code, which is consumed by an assembler. The assembler, in turn, may produce object modules, which are consumed by the loader.

- One solution to the producer-consumer problem uses shared memory.
- To allow producer and consumer processes to run concurrently, we must have available a buffer of items that can be filled by the producer and emptied by the consumer.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

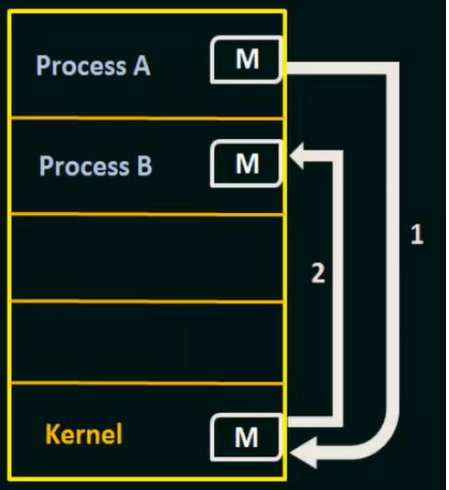




NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Message-Passing Systems (Part-1)

Message passing provides a mechanism to allow processes to communicate and to synchronize their actions without sharing the same address space and is particularly useful in a distributed environment, where the communicating processes may reside on different computers connected by a network.



NESO ACADEMY

A message-passing facility provides at least two operations:

- send (message)

and

- receive (message)

Messages sent by a process can be of either fixed or variable size.

FIXED SIZE: The system-level implementation is straightforward.
But makes the task of programming more difficult.

VARIABLE SIZE: Requires a more complex system-level implementation.
But the programming task becomes simpler.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

If processes P and Q want to communicate, they must send messages to and receive messages from each other.

A communication link must exist between them.

This link can be implemented in a variety of ways. There are several methods for logically implementing a link and the **send( ) /receive( )** operations, like:

- Direct or indirect communication
- Synchronous or asynchronous communication
- Automatic or explicit buffering

There are several issues related with features like:

- **Naming**
- **Synchronization**
- **Buffering**

NESO ACADEMY

Message-Passing Systems (Part-2)

If processes P and Q want to communicate, they must send messages to and receive messages from each other.

A communication link must exist between them.

This link can be implemented in a variety of ways. There are several methods for logically implementing a link and the **send( ) /receive( )** operations, like:

- Direct or indirect communication
- Synchronous or asynchronous communication
- Automatic or explicit buffering

There are several issues related with features like:

- **Naming**
- **Synchronization**
- **Buffering**

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Naming

Processes that want to communicate must have a way to refer to each other.
They can use either **direct** or **indirect** communication.

Under direct communication- Each process that wants to communicate must explicitly name the recipient or sender of the communication.

- send (P, message) - Send a message to process P.
- receive (Q, message) - Receive a message from process Q.

A communication link in this scheme has the following properties:

- A link is established automatically between every pair of processes that want to communicate. The processes need to know only each other's identity to communicate.
- A link is associated with exactly two processes.
- Between each pair of processes, there exists exactly one link.

This scheme exhibits **symmetry in addressing**; that is, both the sender process and the receiver process must name the other to communicate.

NESO ACADEMY

Another variant of Direct Communication- Here, only the sender names the recipient; the recipient is not required to name the sender.

- send (P, message) - Send a message to process P.
- receive (id, message) - Receive a message from any process; *the variable id is set to the name of the process with which communication has taken place.*

This scheme employs **asymmetry in addressing**.

The disadvantage in both of these schemes (symmetric and asymmetric) is the limited modularity of the resulting process definitions.
Changing the identifier of a process may necessitate examining all other process definitions.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

With indirect communication:

The messages are sent to and received from **mailboxes**, or ports.

- A mailbox can be viewed abstractly as an object into which messages can be placed by processes and from which messages can be removed.
- Each mailbox has a unique identification.
- Two processes can communicate only if the processes have a shared mailbox
  - send (A, message) — Send a message to mailbox A.
  - receive (A, message) — Receive a message from mailbox A.

NESO ACADEMY

A communication link in this scheme has the following properties:

- A link is established between a pair of processes only if both members of the pair have a shared mailbox.
- A link may be associated with more than two processes.
- Between each pair of communicating processes, there may be a number of different links, with each link corresponding to one mailbox.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Now suppose that processes P1, P2, and P3 all share mailbox A

Process P1 sends a message to A, while both P2 and P3 execute a receive() from A. **Which process will receive the message sent by P1?**

NESO ACADEMY

Process P1 sends a message to A, while both P2 and P3 execute a receive() from A. **Which process will receive the message sent by P1?**

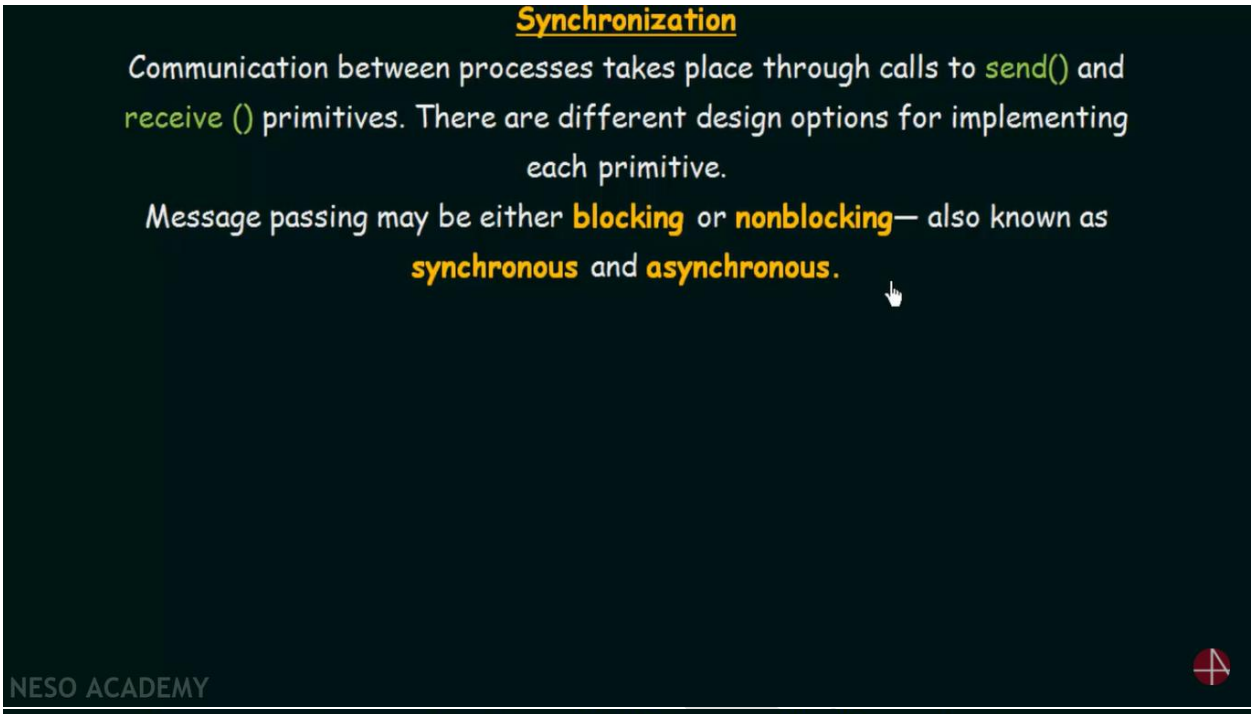
The answer depends on which of the following methods we choose:

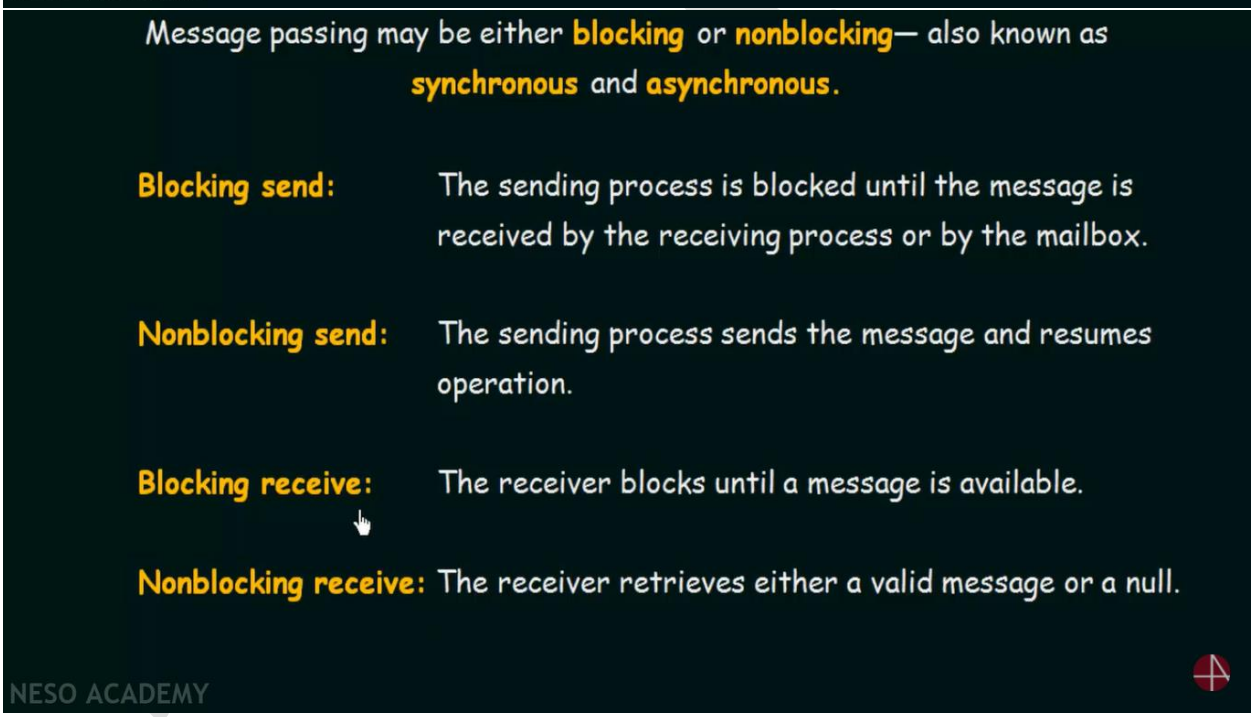
- Allow a link to be associated with two processes at most.
- Allow at most one process at a time to execute a receive () operation.
- Allow the system to select arbitrarily which process will receive the message (that is, either P2 or P3, but not both, will receive the message). The system also may define an algorithm for selecting which process will receive the message (that is, round robin where processes take turns receiving messages). The system may identify the receiver to the sender.

A **mailbox** may be **owned** either by a **process** or by the **operating system.**

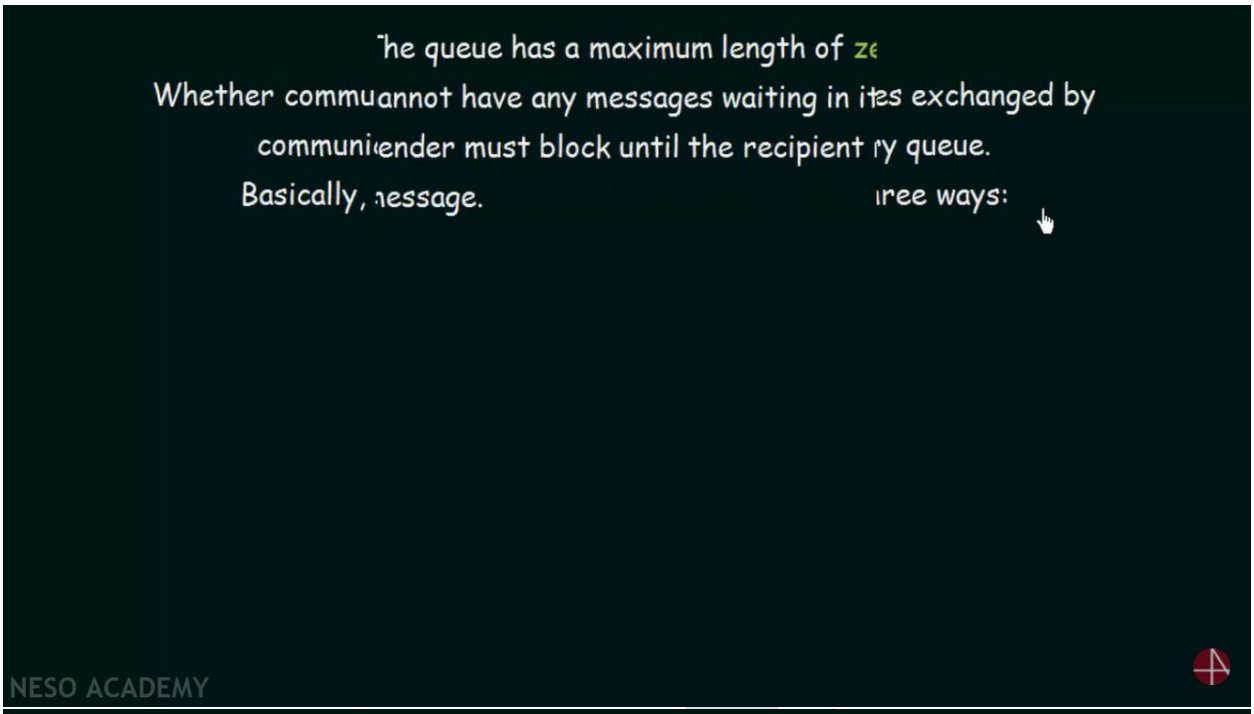
NESO ACADEMY

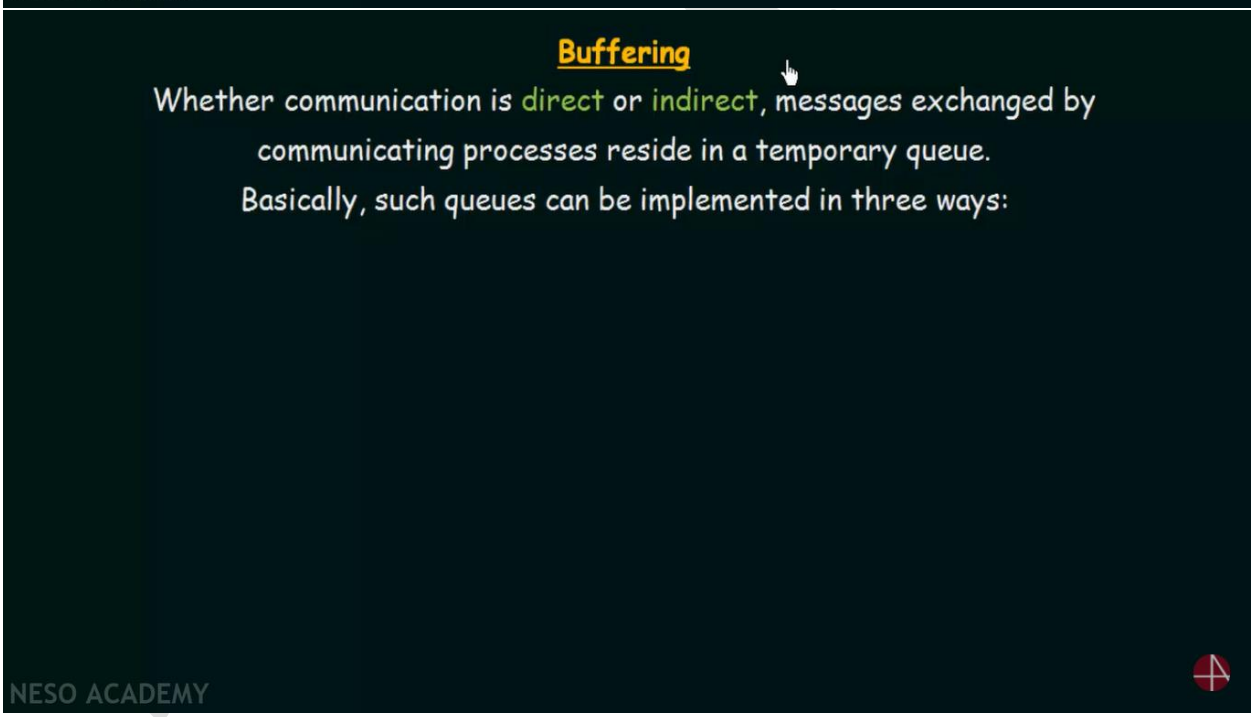
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



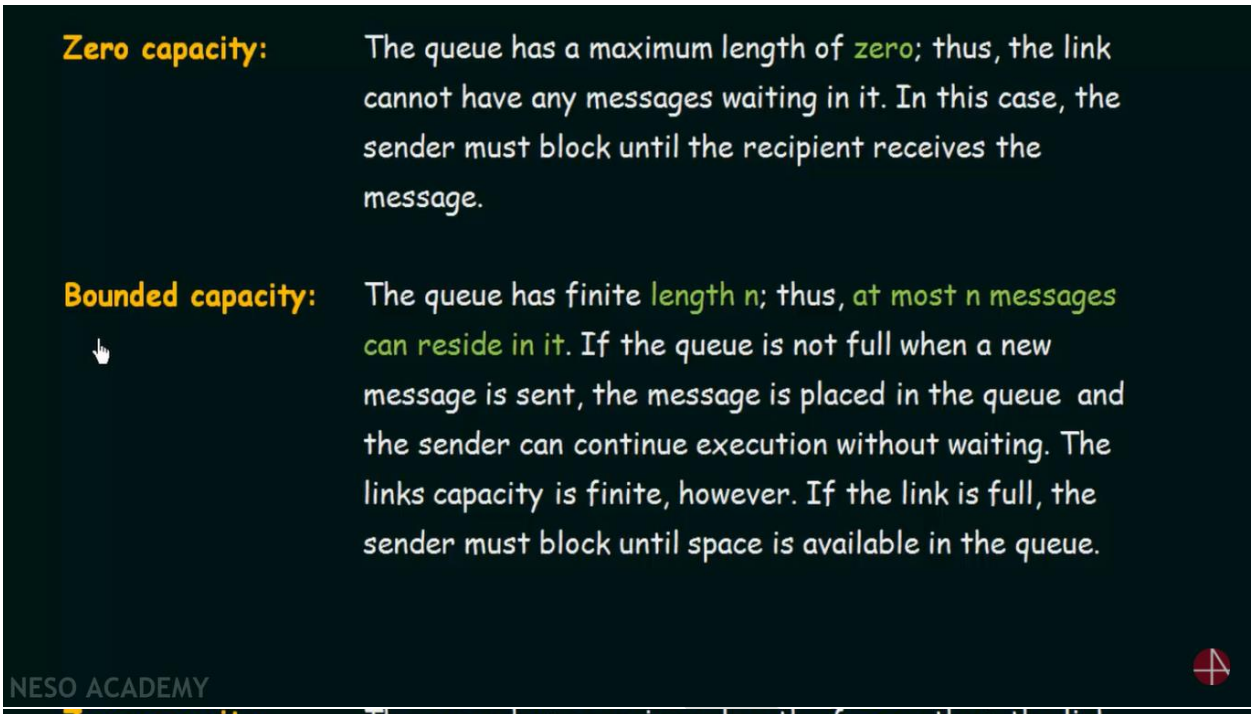


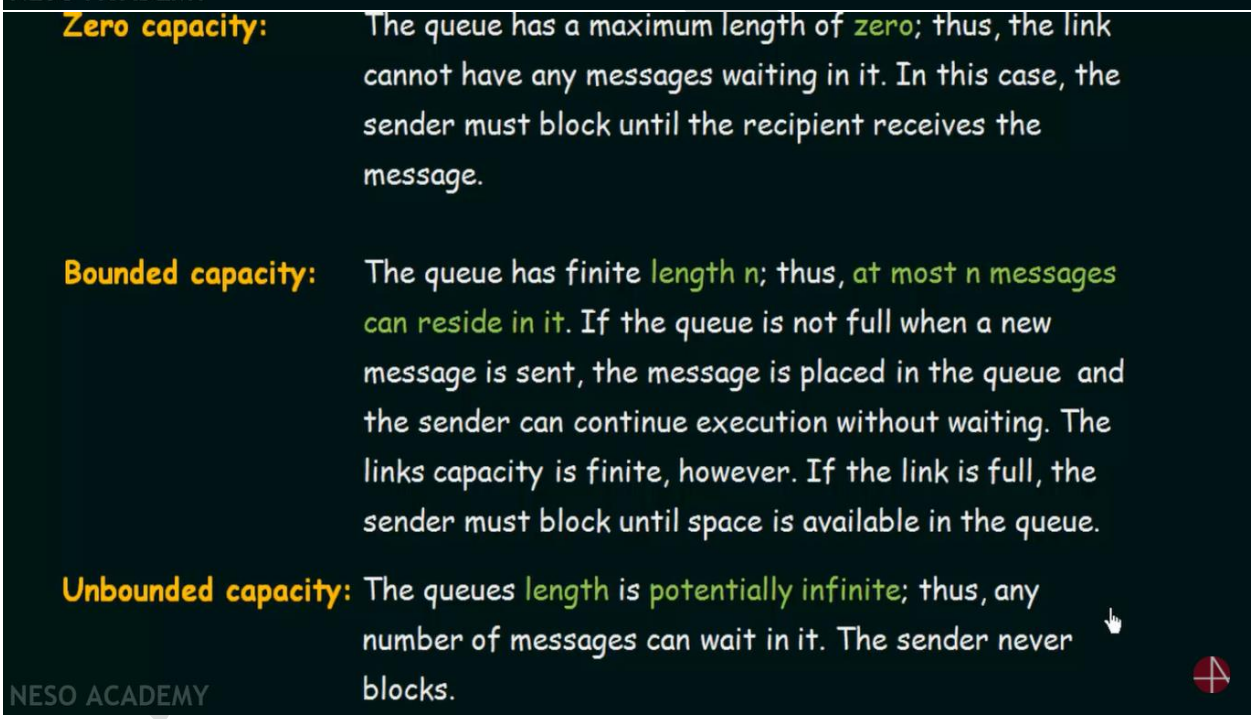
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



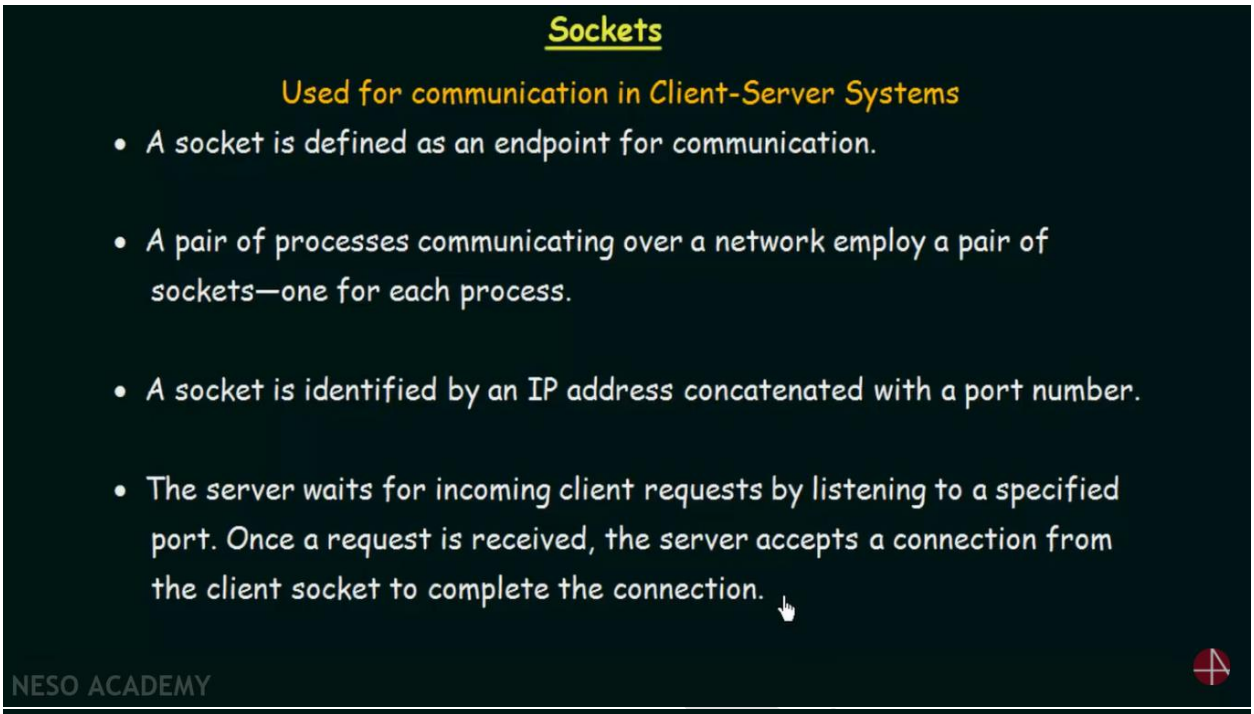


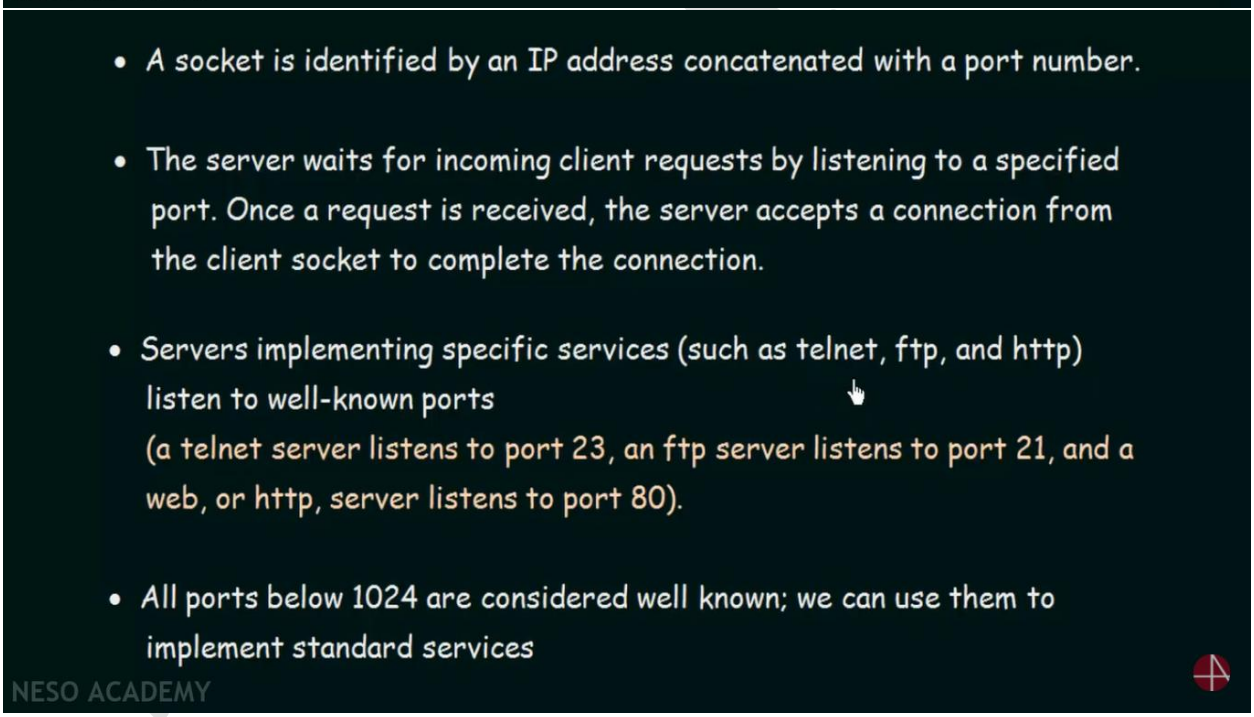
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



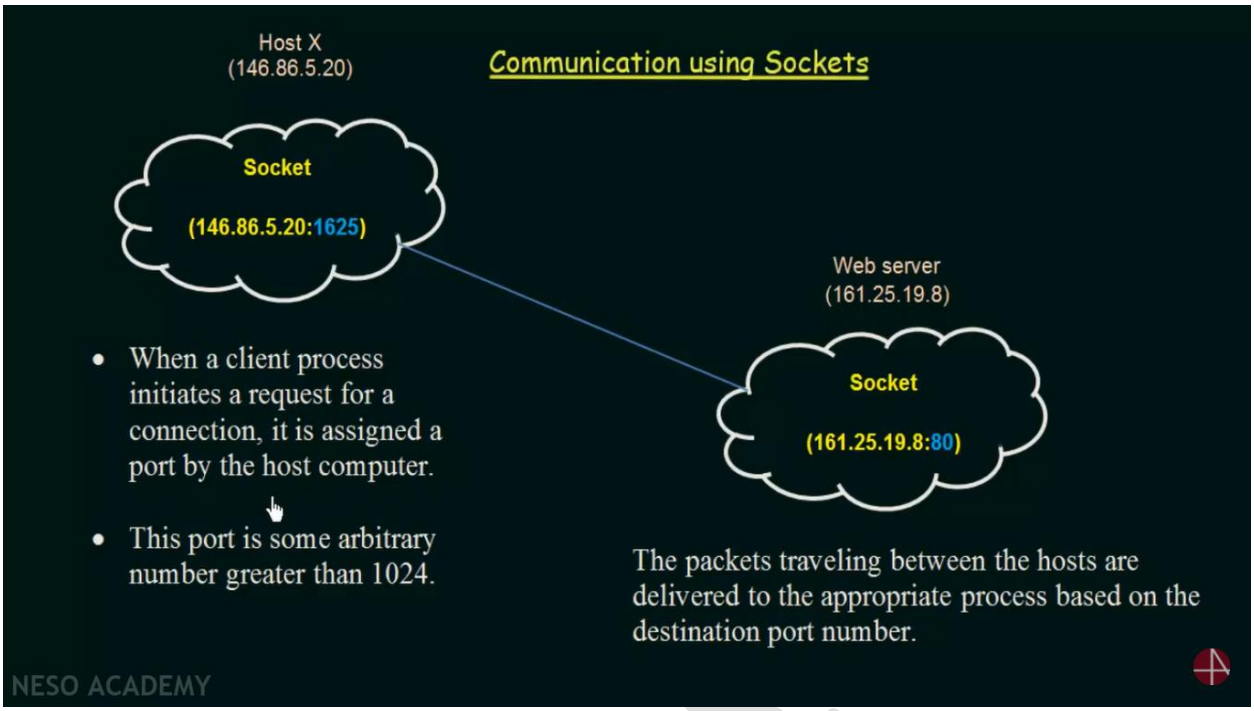


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

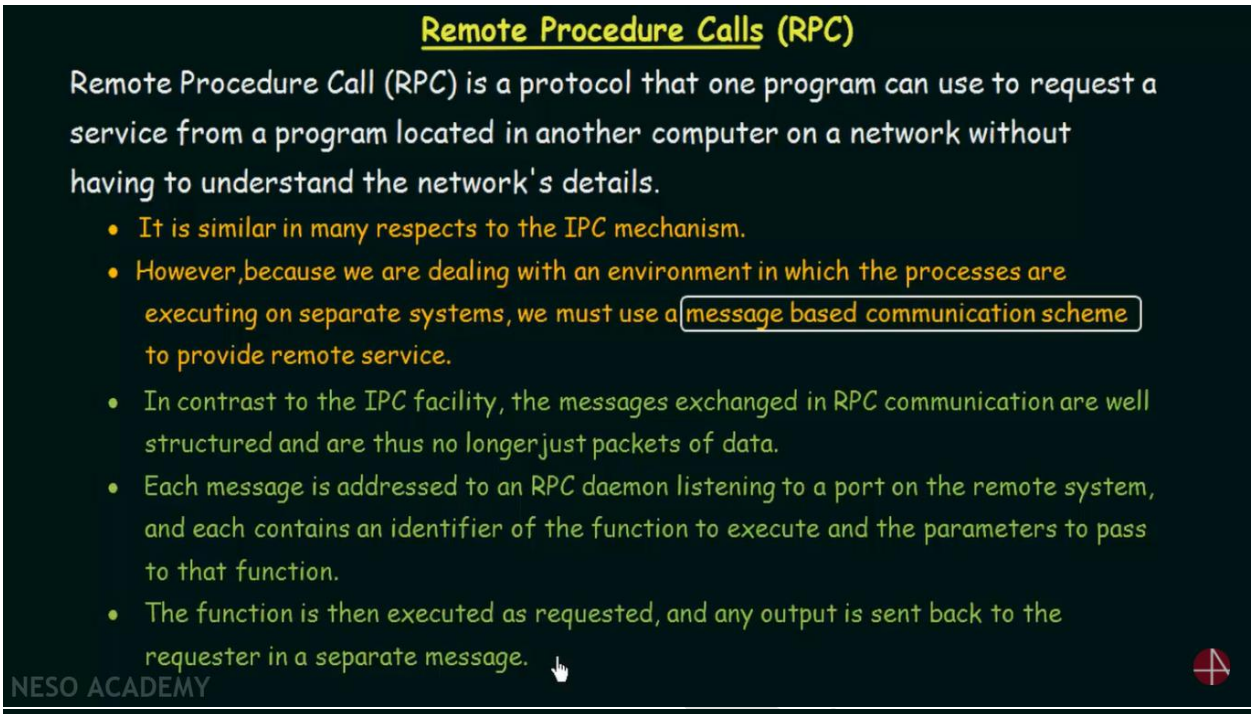


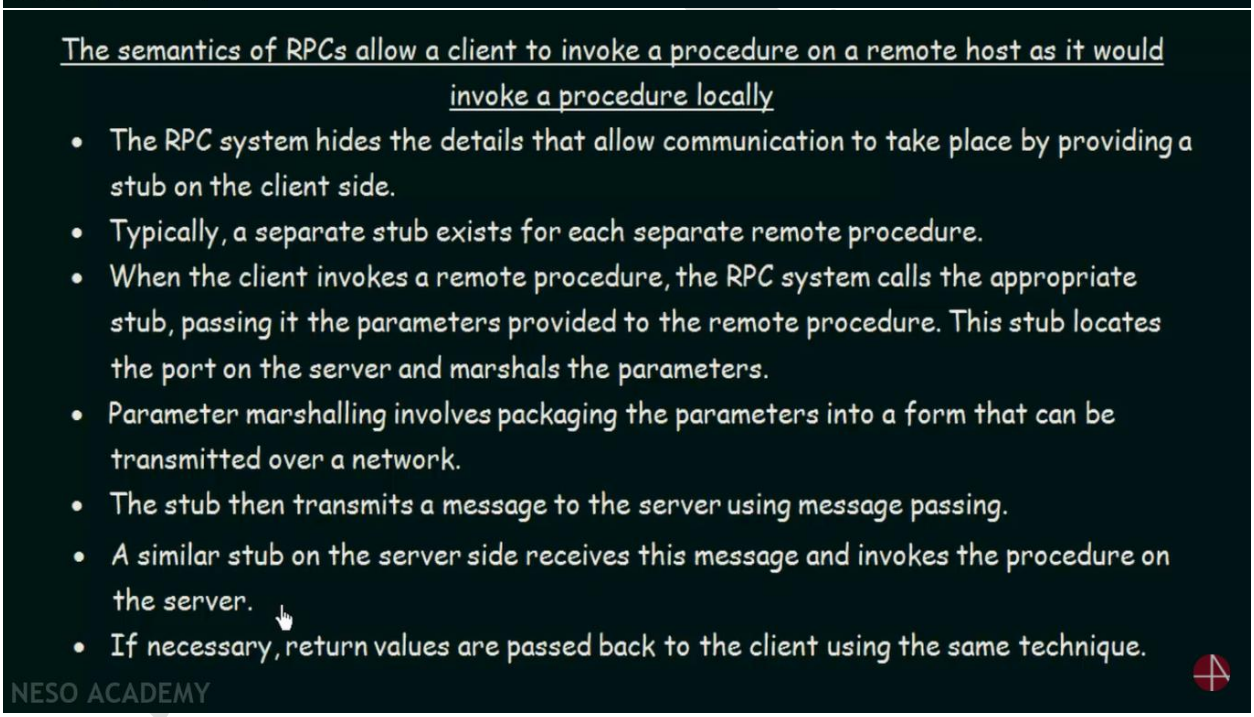


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

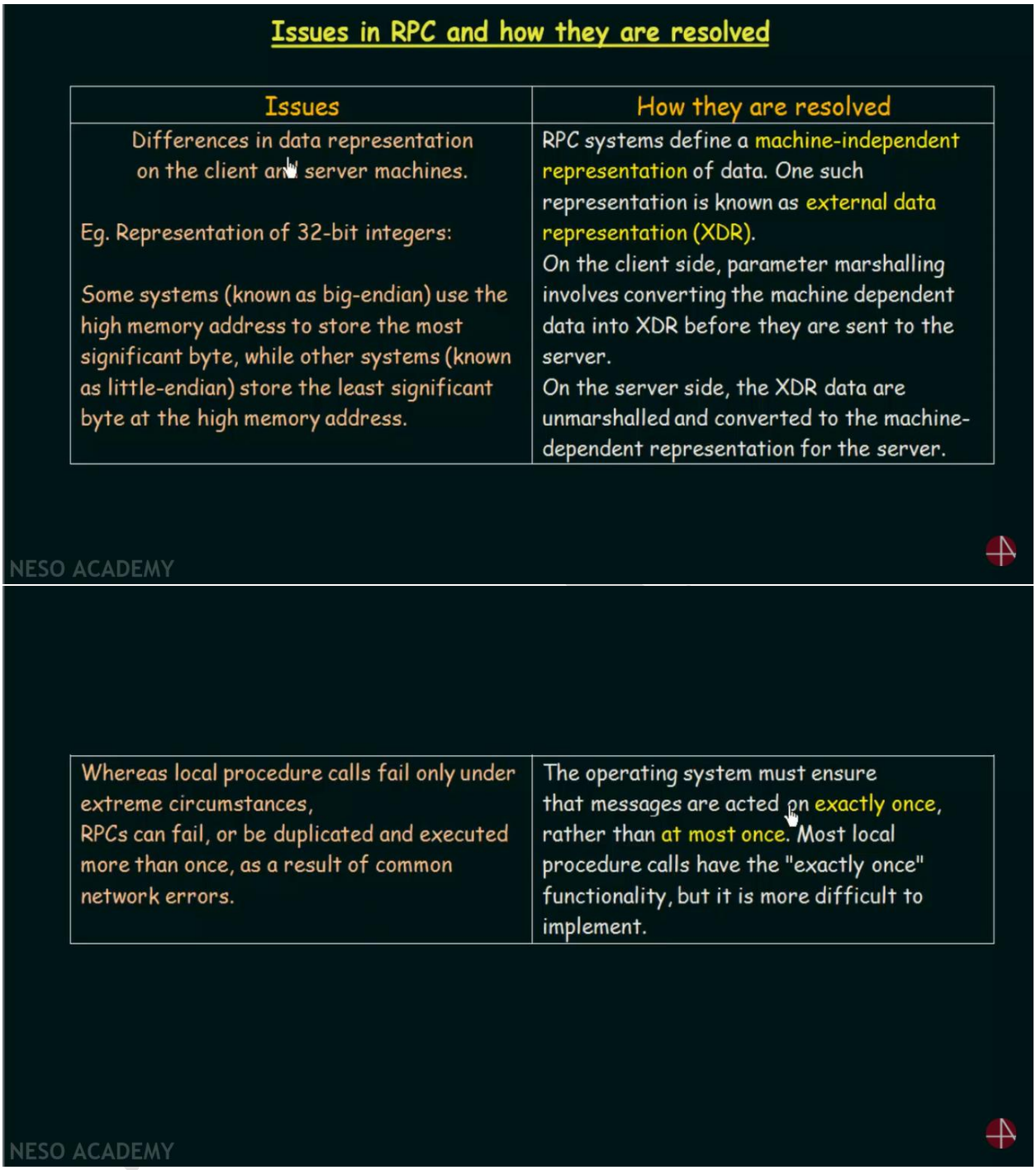


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



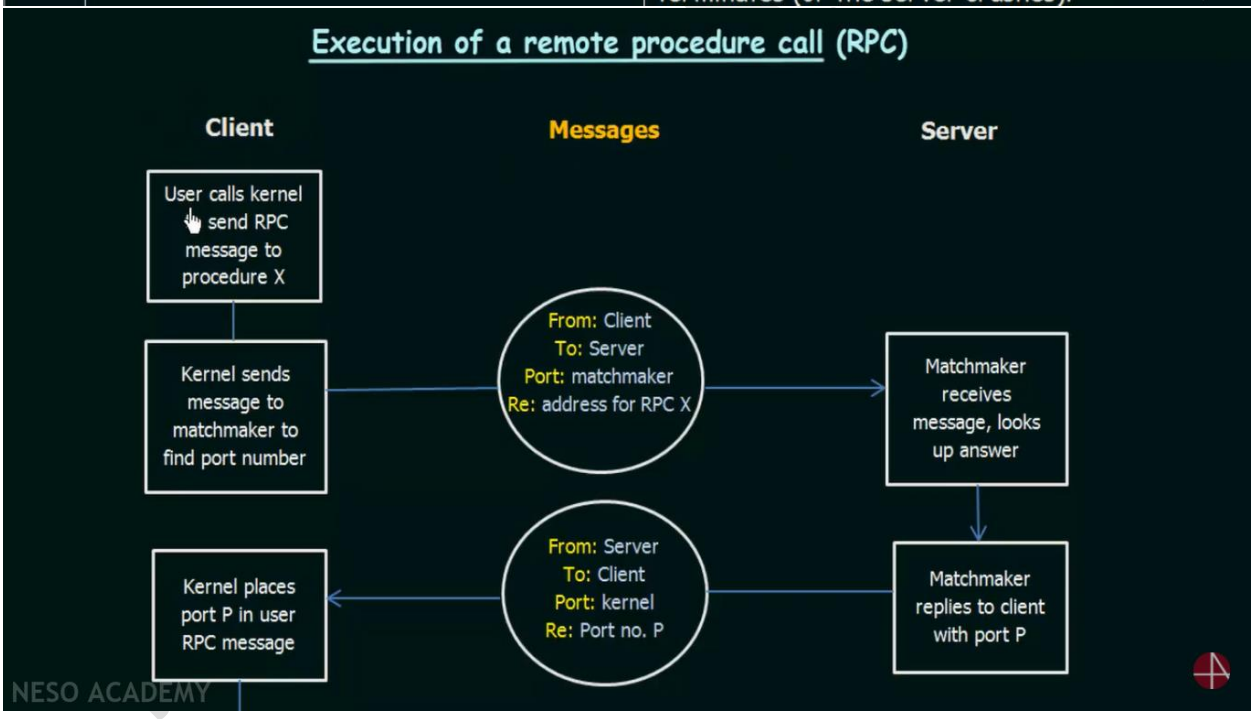
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

With standard procedure calls, some form of binding takes place during link, load, or execution time so that a procedure call's name is replaced by the memory address of the procedure call. The RPC scheme requires a similar binding of the client and the server port, but how does a client know the port numbers on the server? Neither system has full information about the other because they do not share memory.

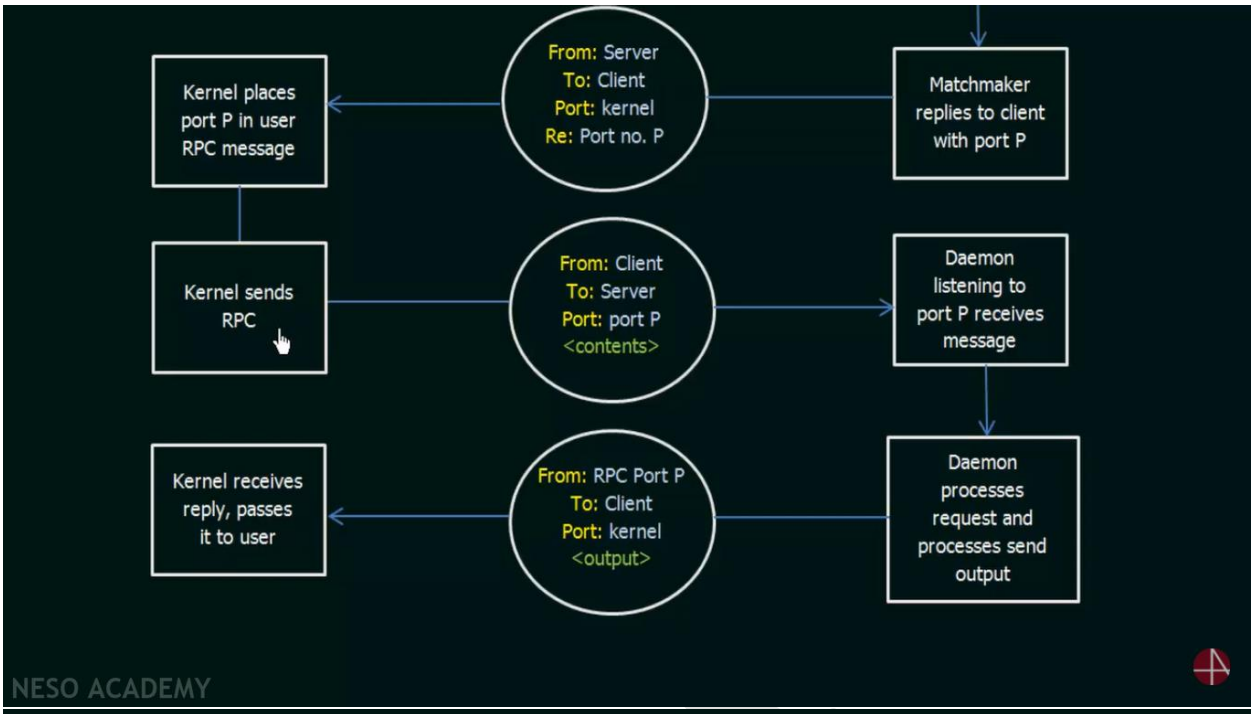
1) The binding information may be predetermined, in the form of fixed port addresses. At compile time, an RPC call has a fixed port number associated with it. Once a program is compiled, the server cannot change the port number of the requested service.

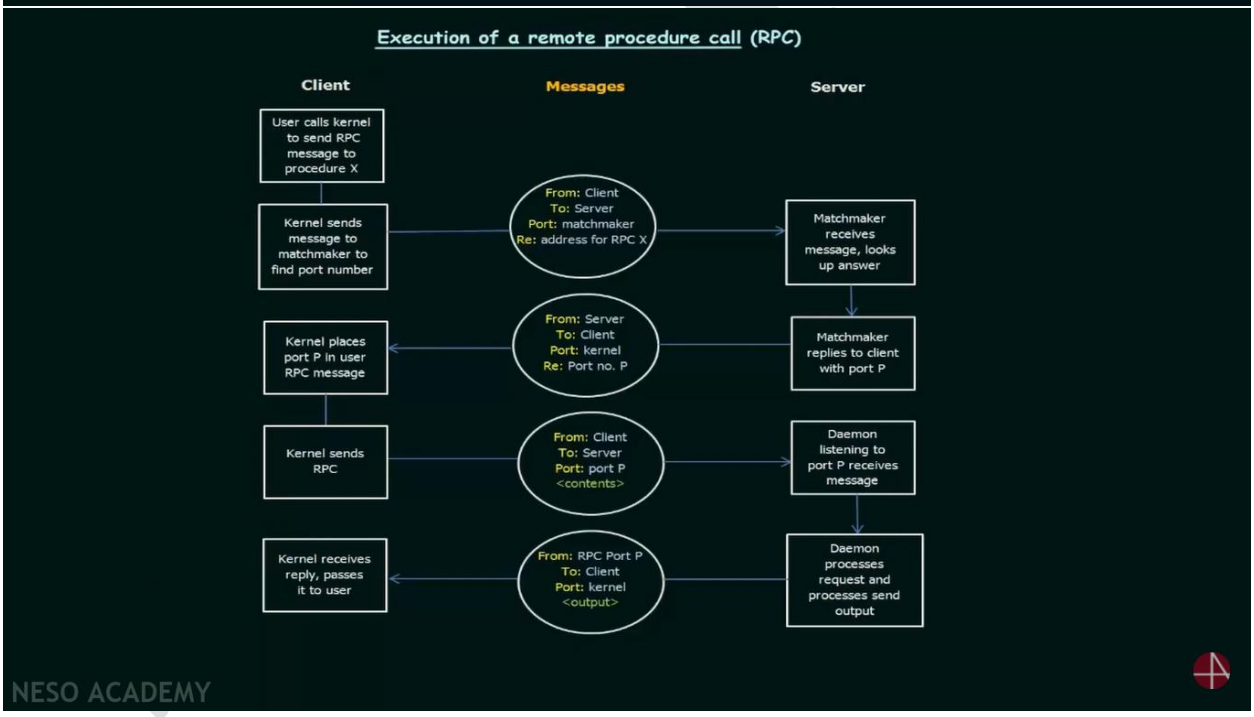
2) Binding can be done dynamically by a rendezvous mechanism. Typically, an operating system provides a rendezvous (also called a matchmaker) daemon on a fixed RPC port. A client then sends a message containing the name of the RPC to the rendezvous daemon requesting the port address of the RPC it needs to execute. The port number is returned, and the RPC calls can be sent to that port until the process terminates (or the server crashes).

NESO ACADEMY

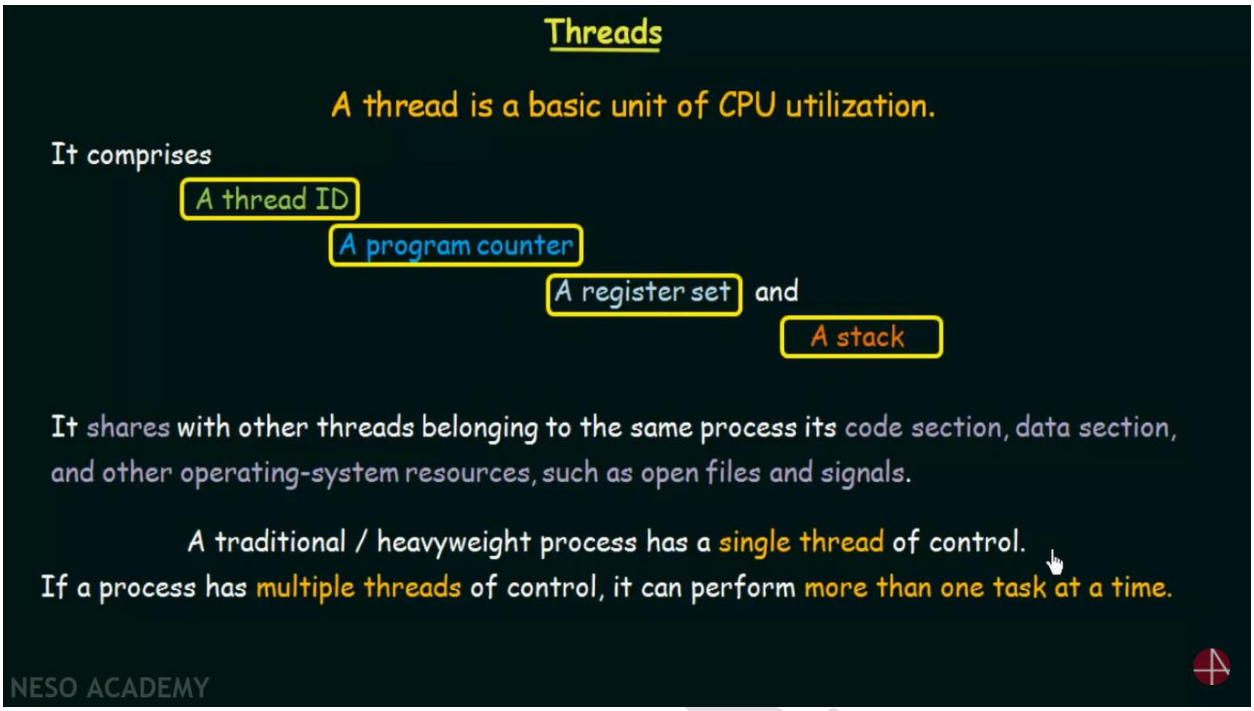


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

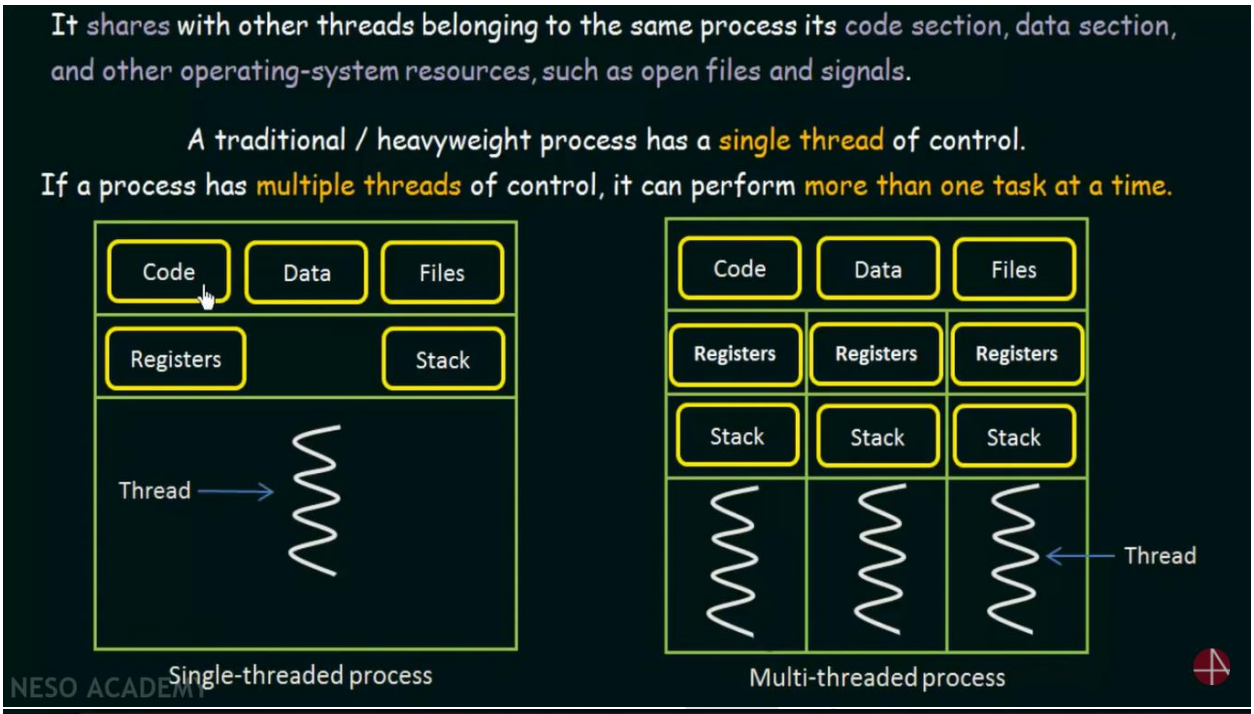


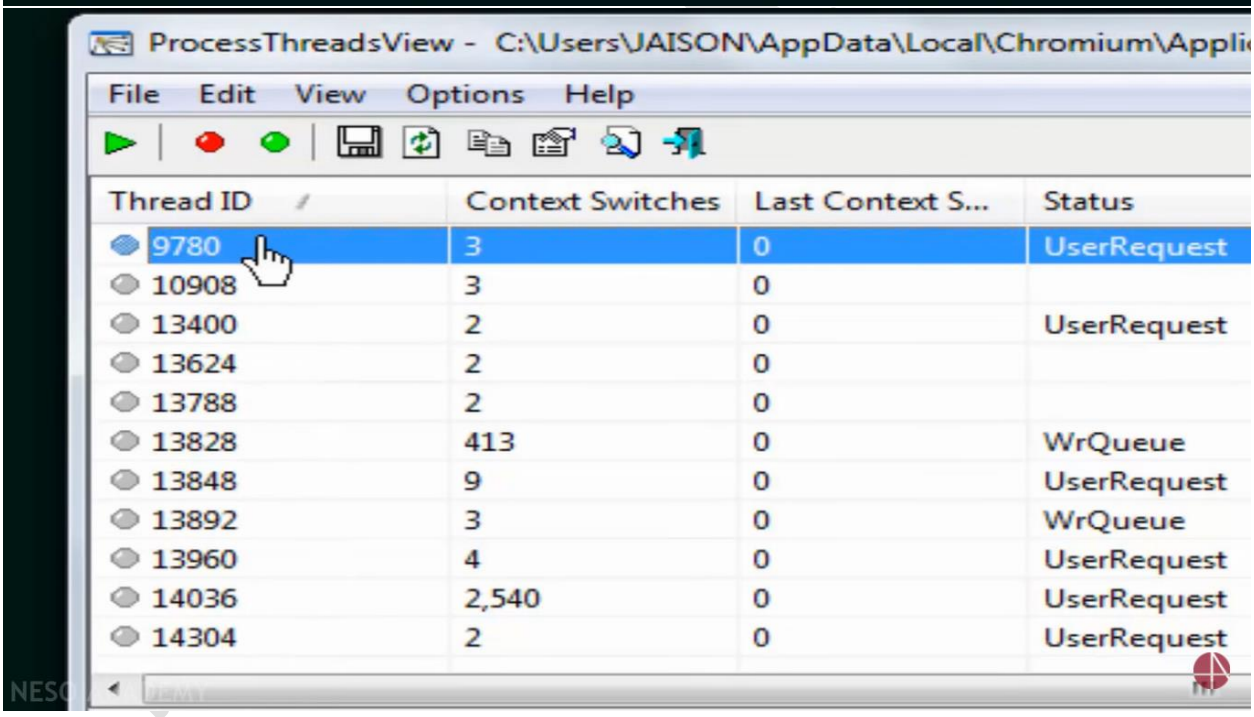


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

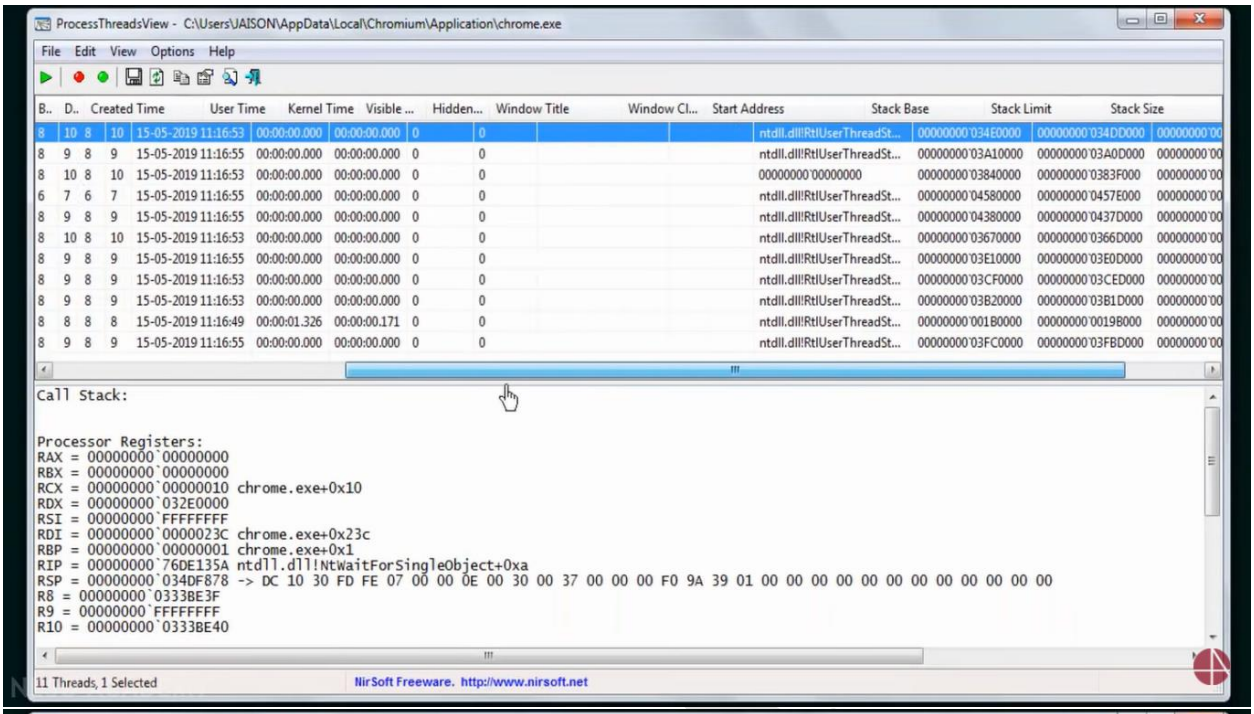


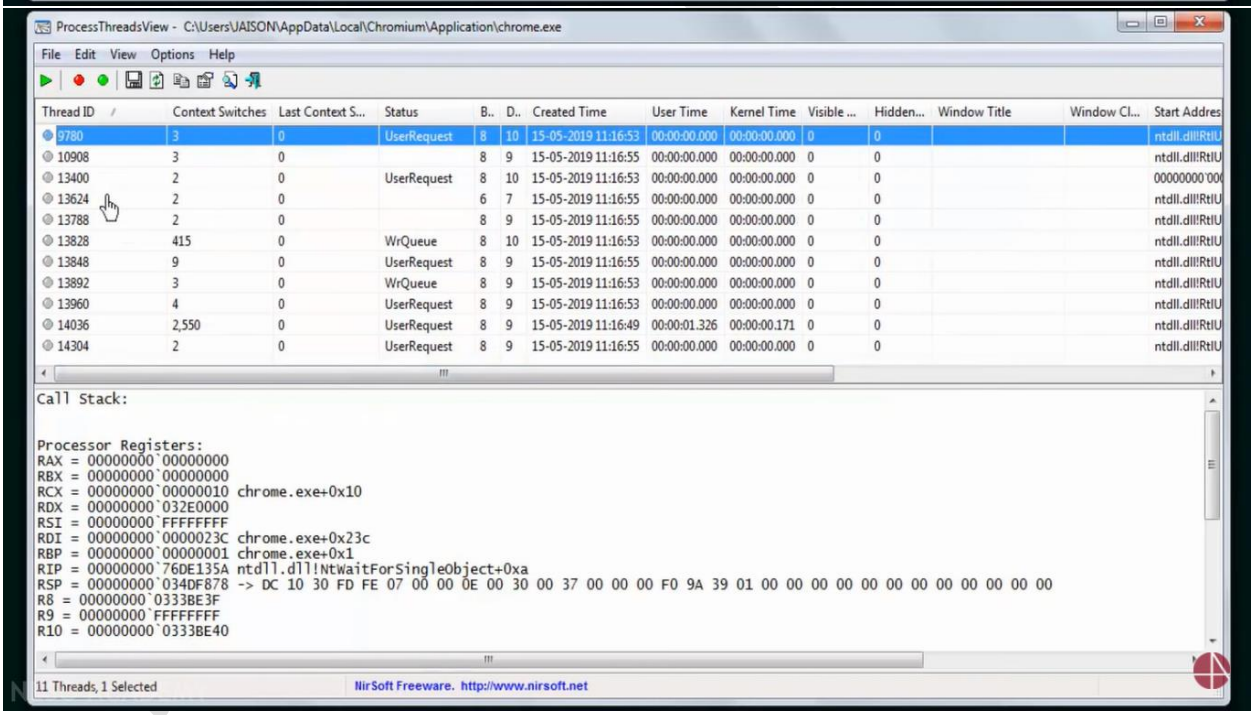
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



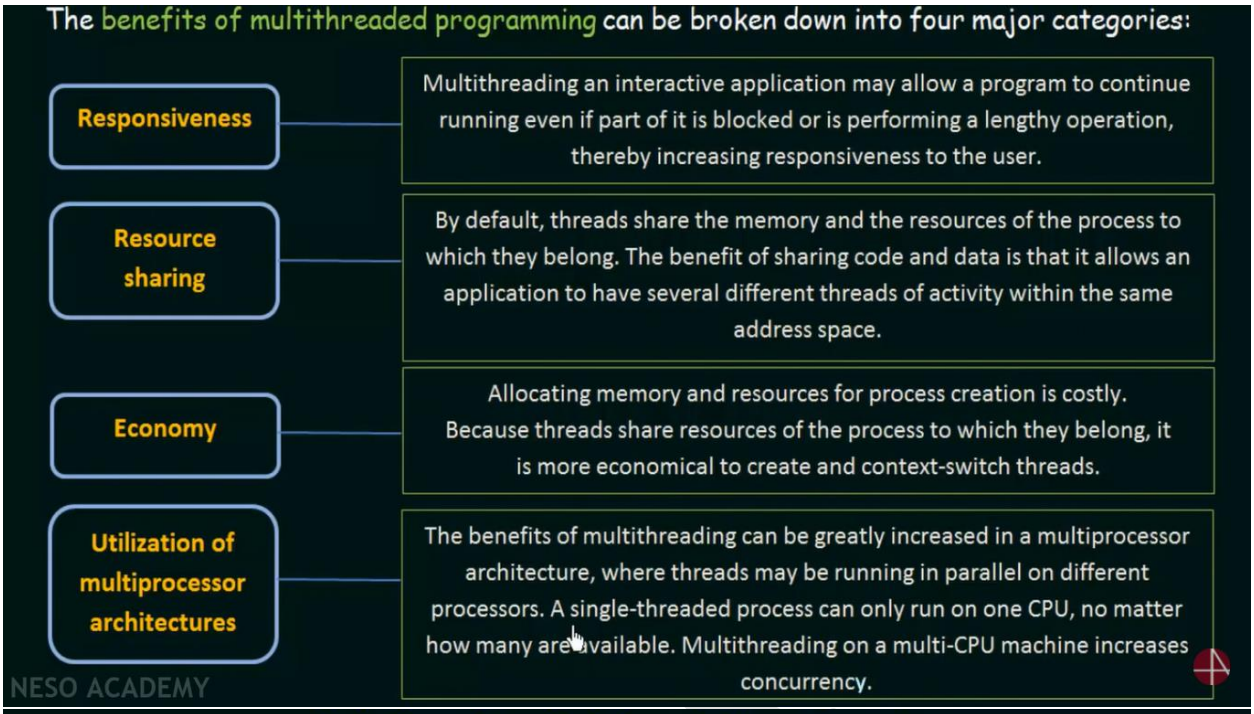


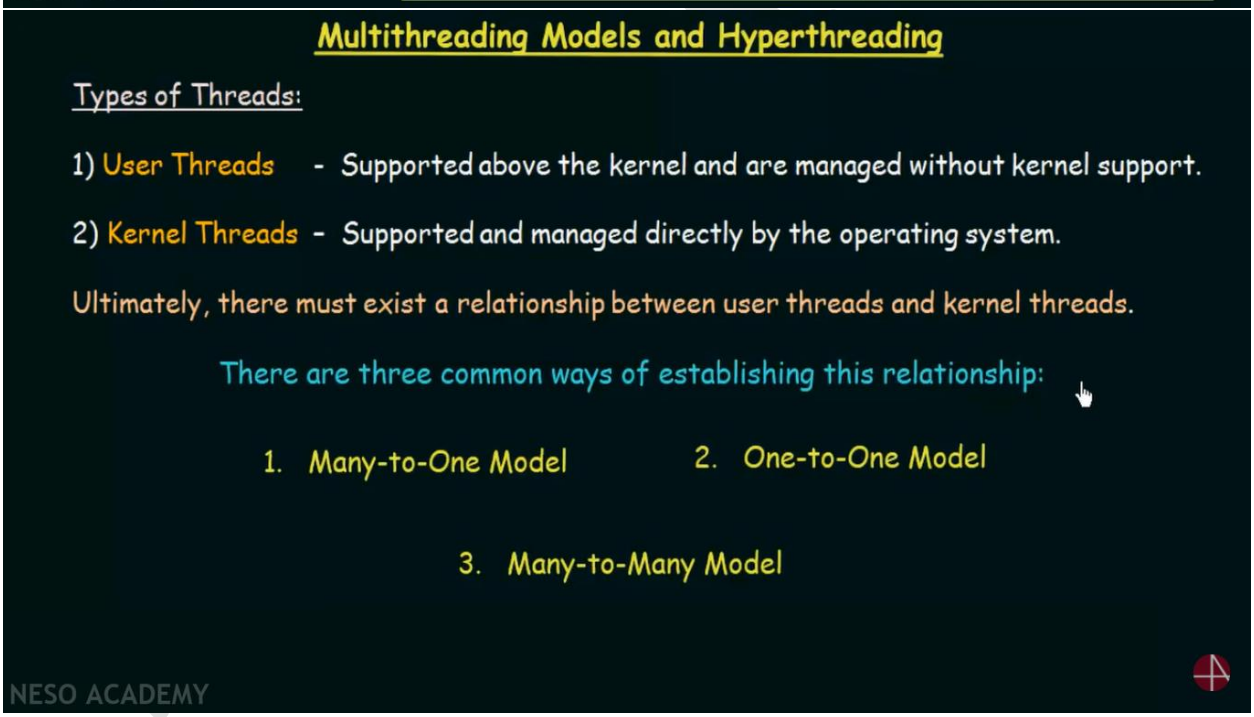
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



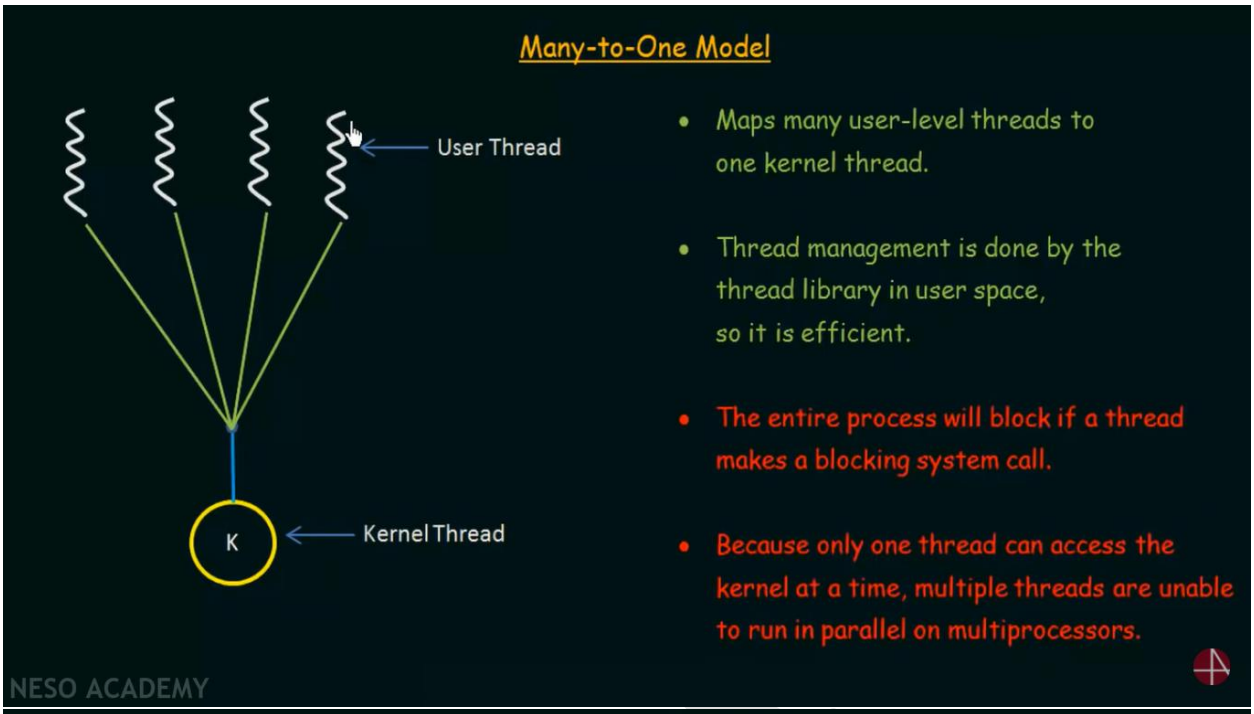


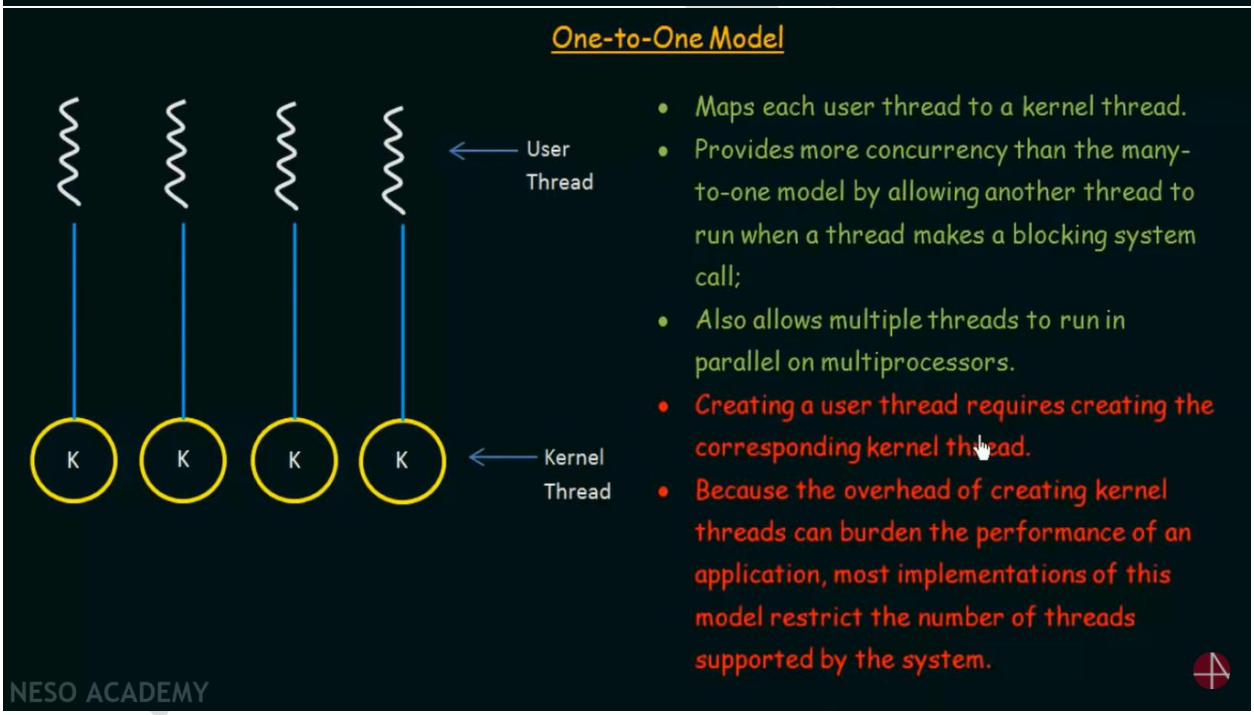
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





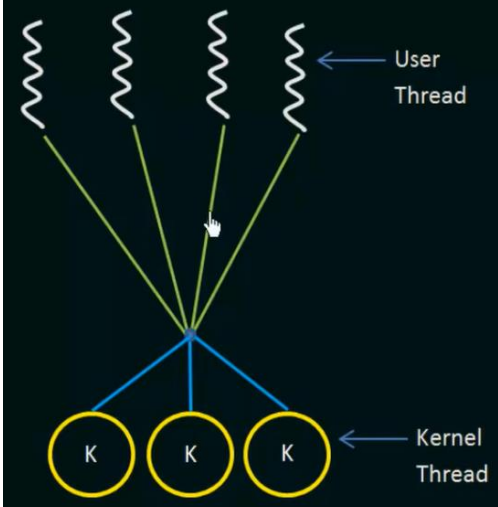
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Many-to-Many Model



- Multiplexes many user-level threads to a smaller or equal number of kernel threads.
- The number of kernel threads may be specific to either a particular application or a particular machine.
- Developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor.
- Also, when a thread performs a blocking system call, the kernel can schedule another thread for execution.

NESO ACADEMY

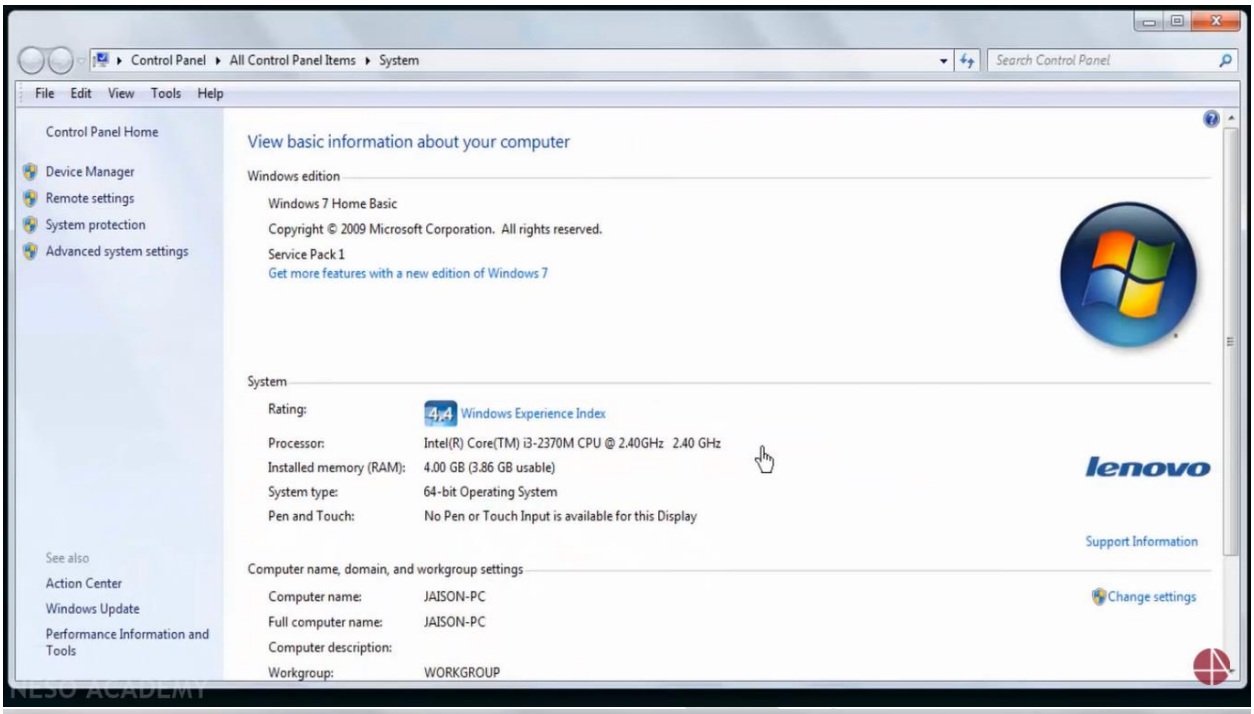
Hyperthreading
or
Simultaneous Multithreading (SMT)

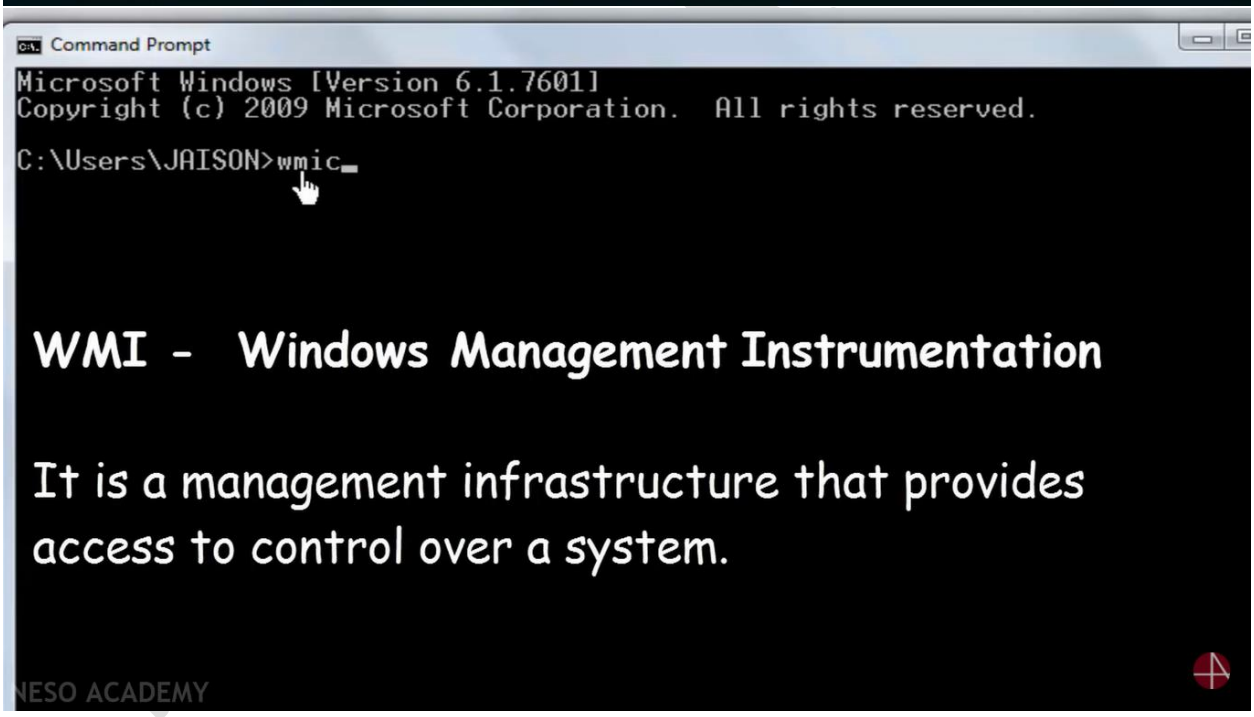
Hyperthreaded systems allow their processor cores' resources to become multiple logical processors for performance.

It enables the processor to execute two threads, or sets of instructions, at the same time. Since hyper-threading allows two streams to be executed in parallel, it is almost like having two separate processors working together.

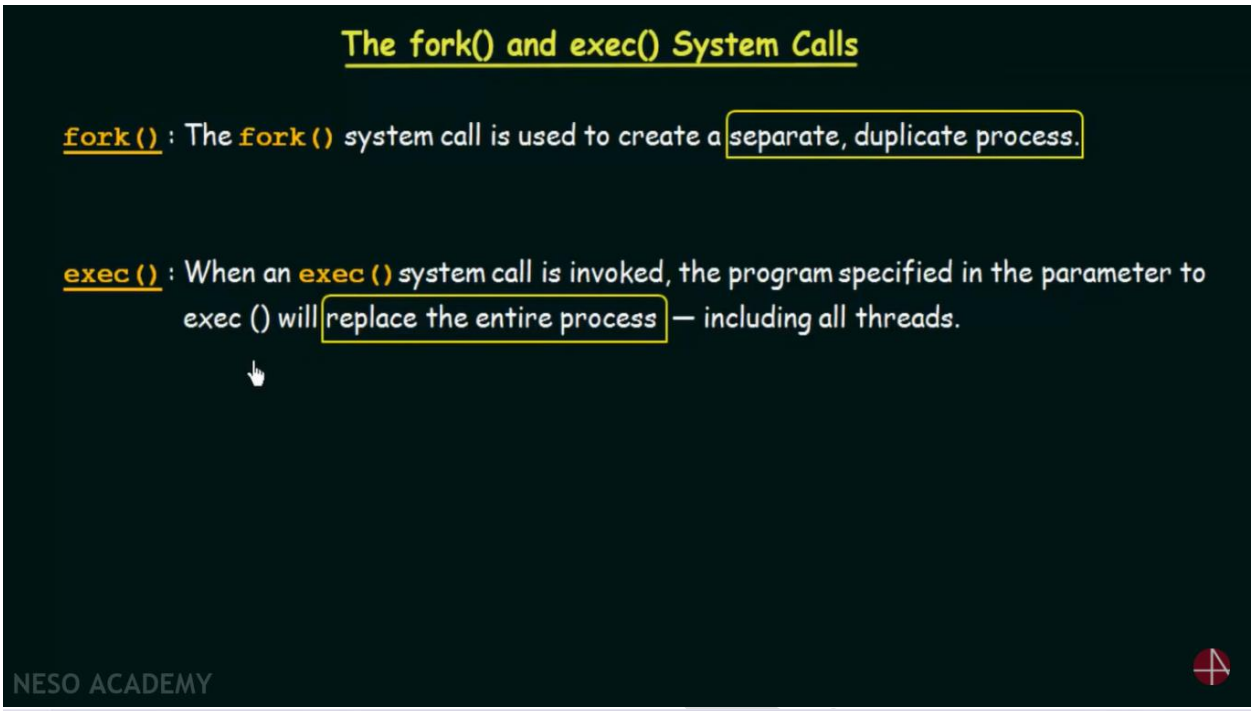
NESO ACADEMY

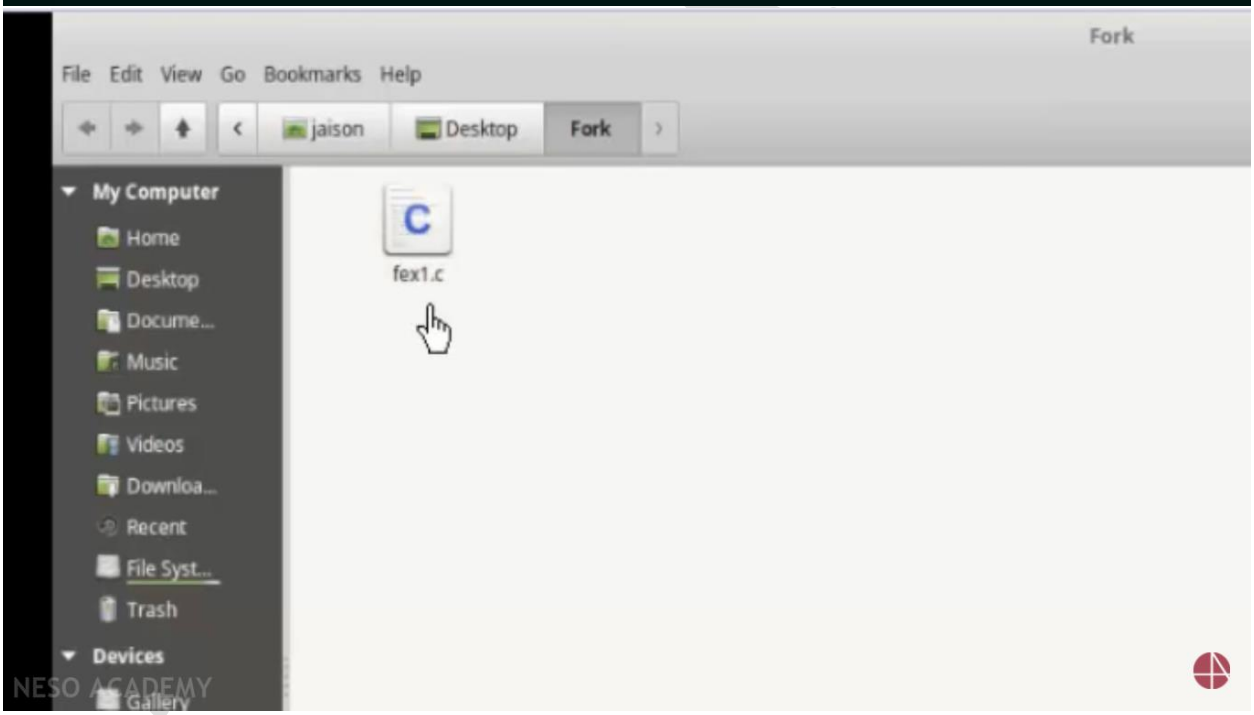
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



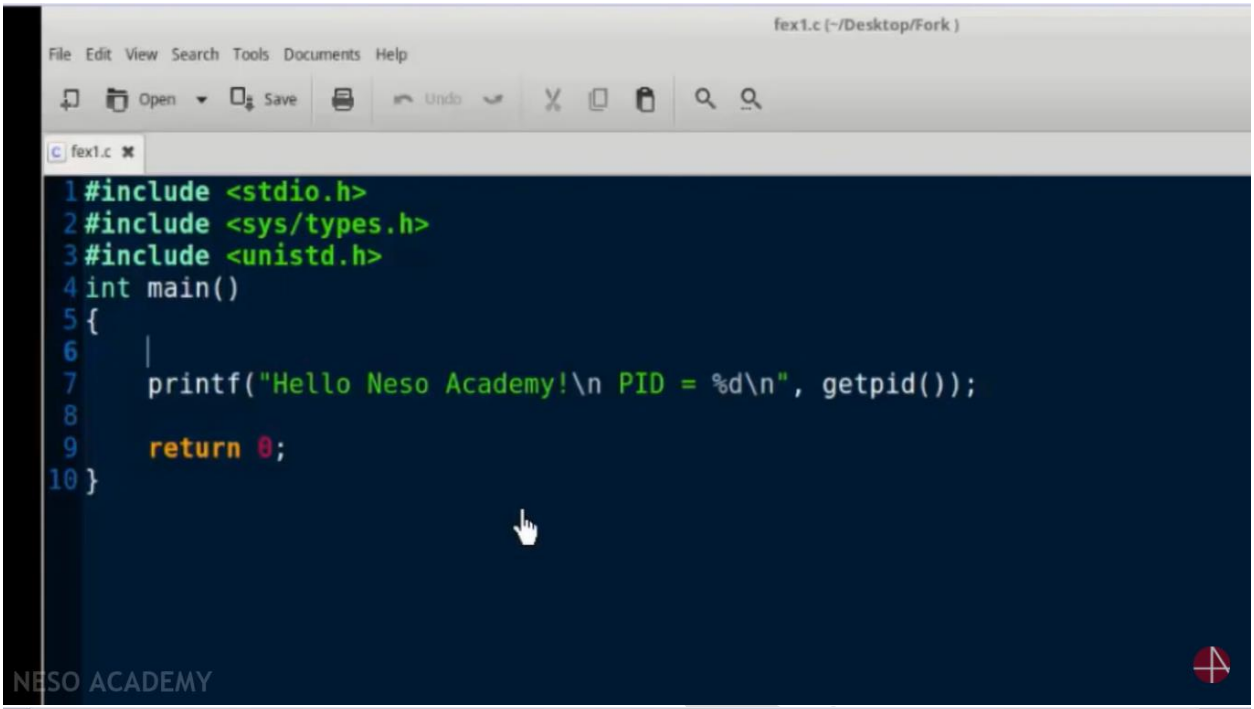


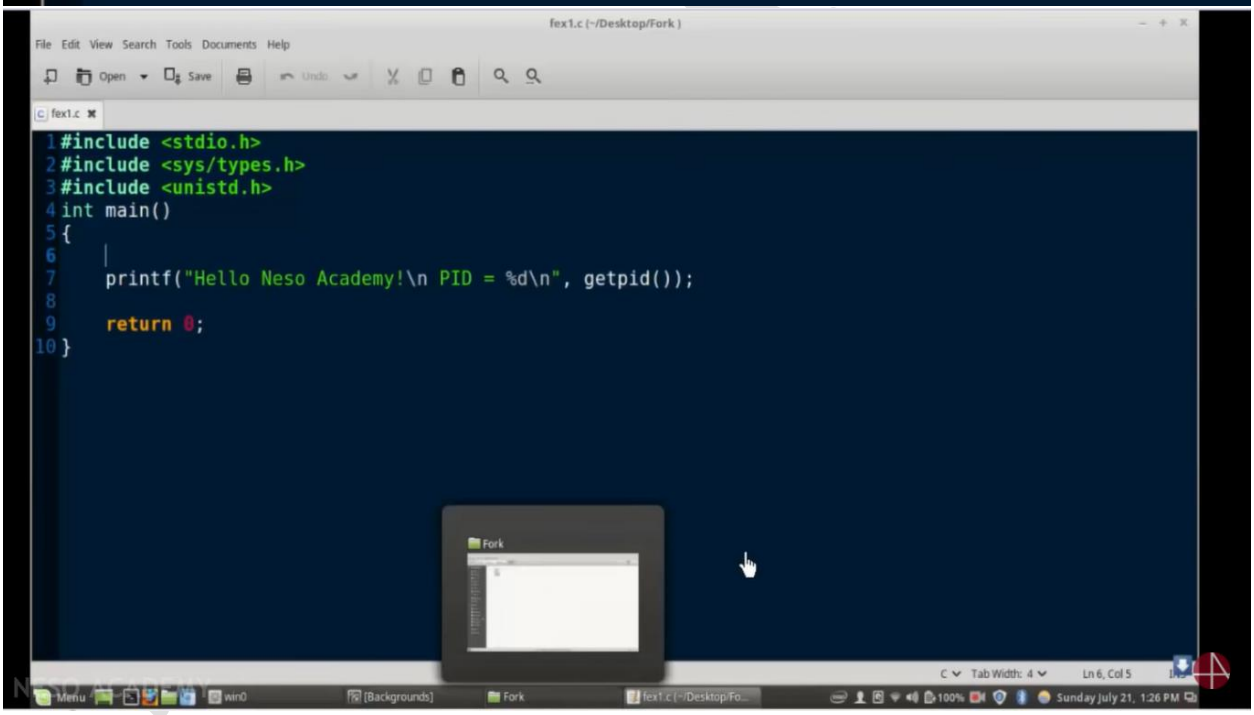
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



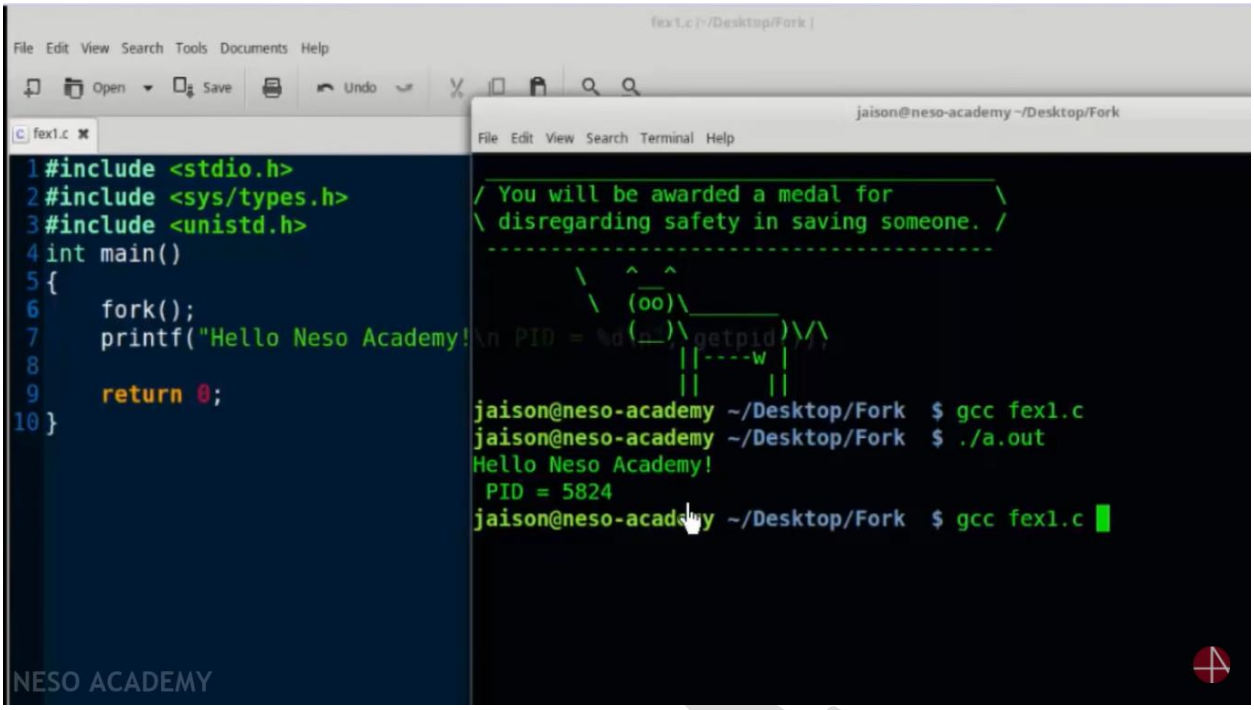


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

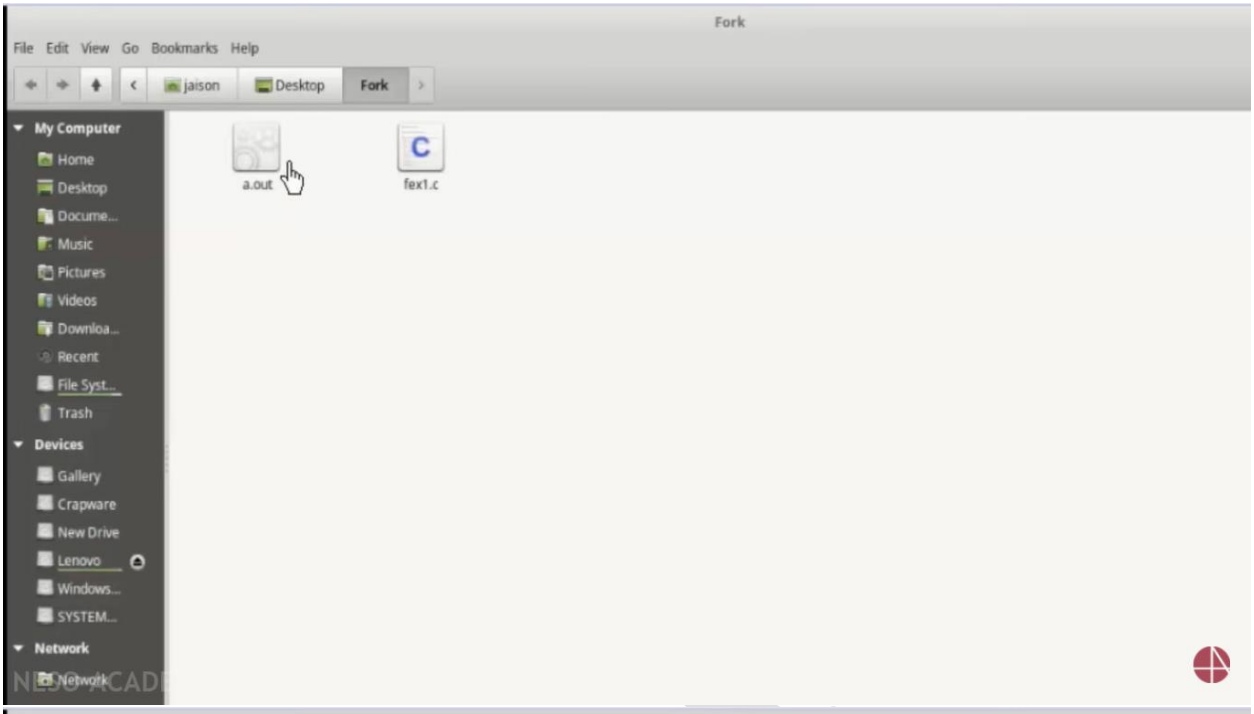


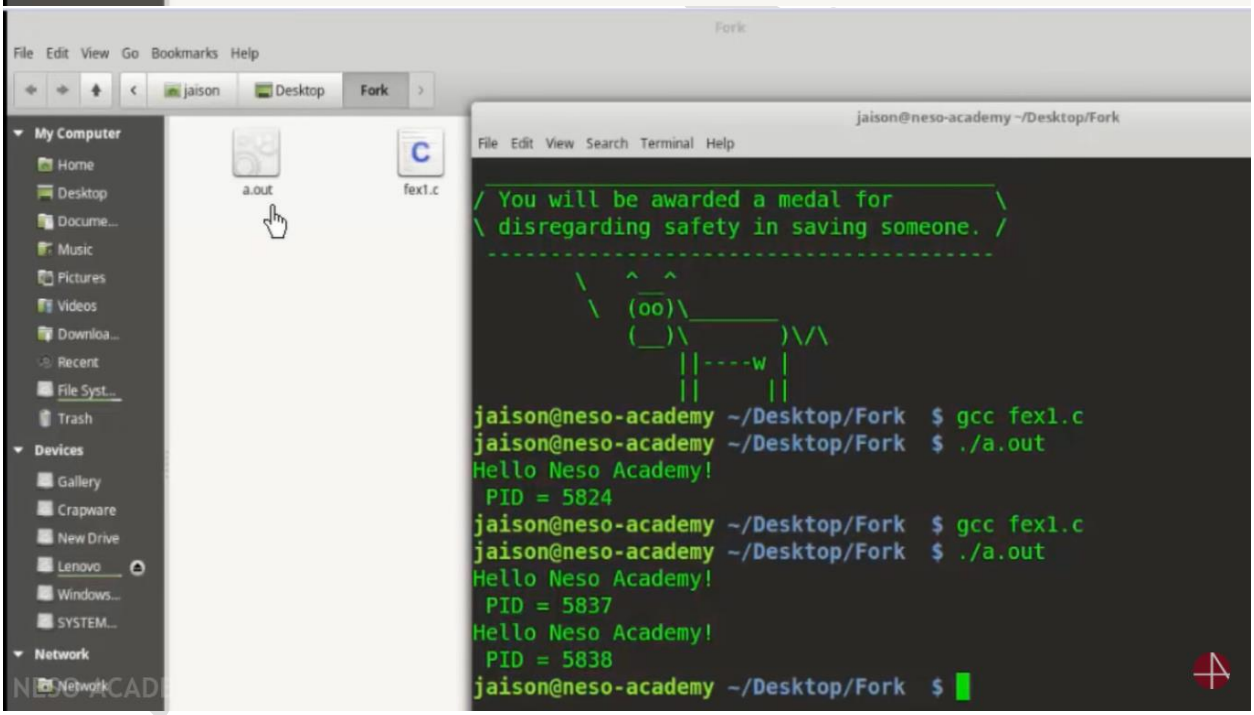


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

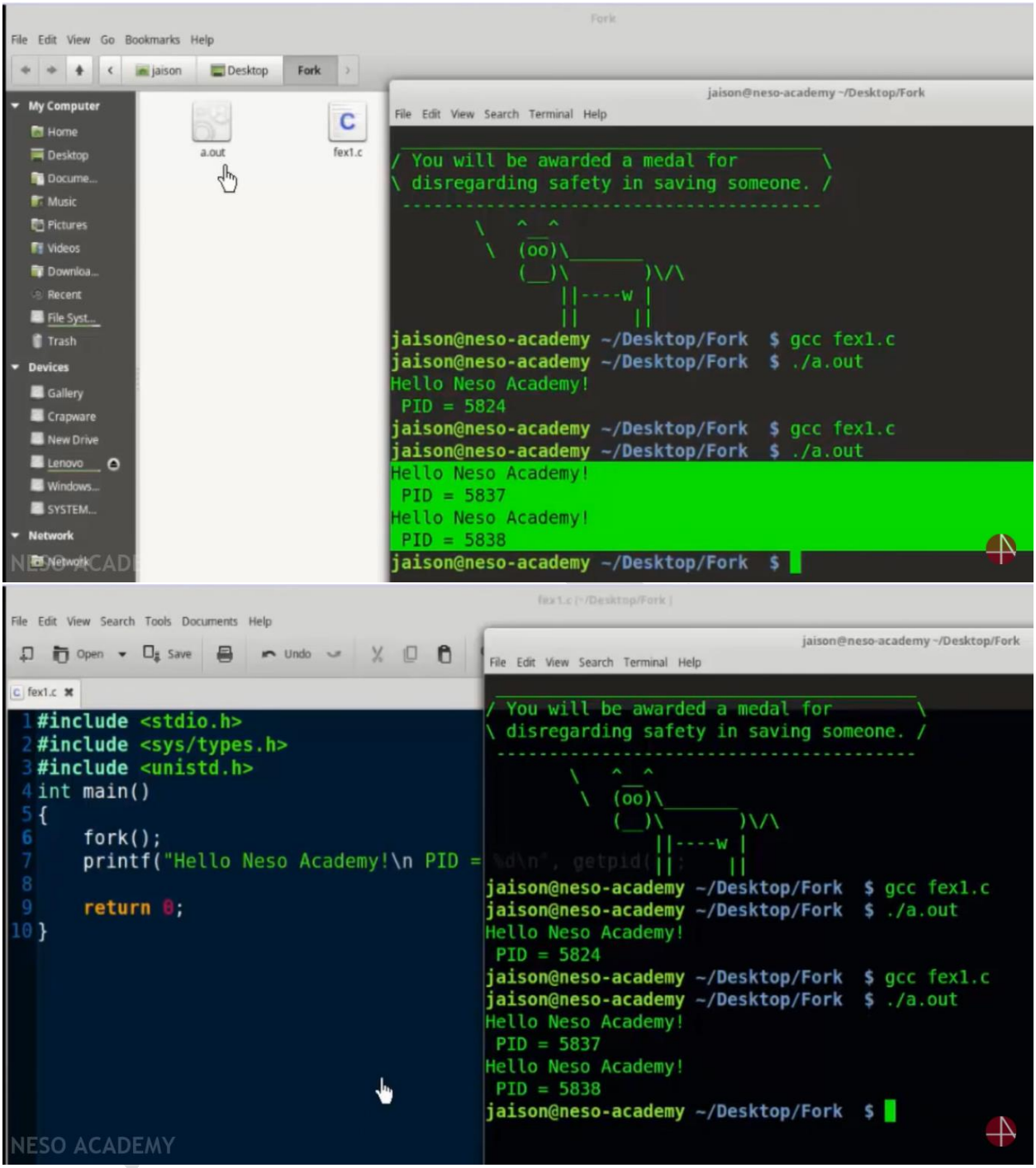


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

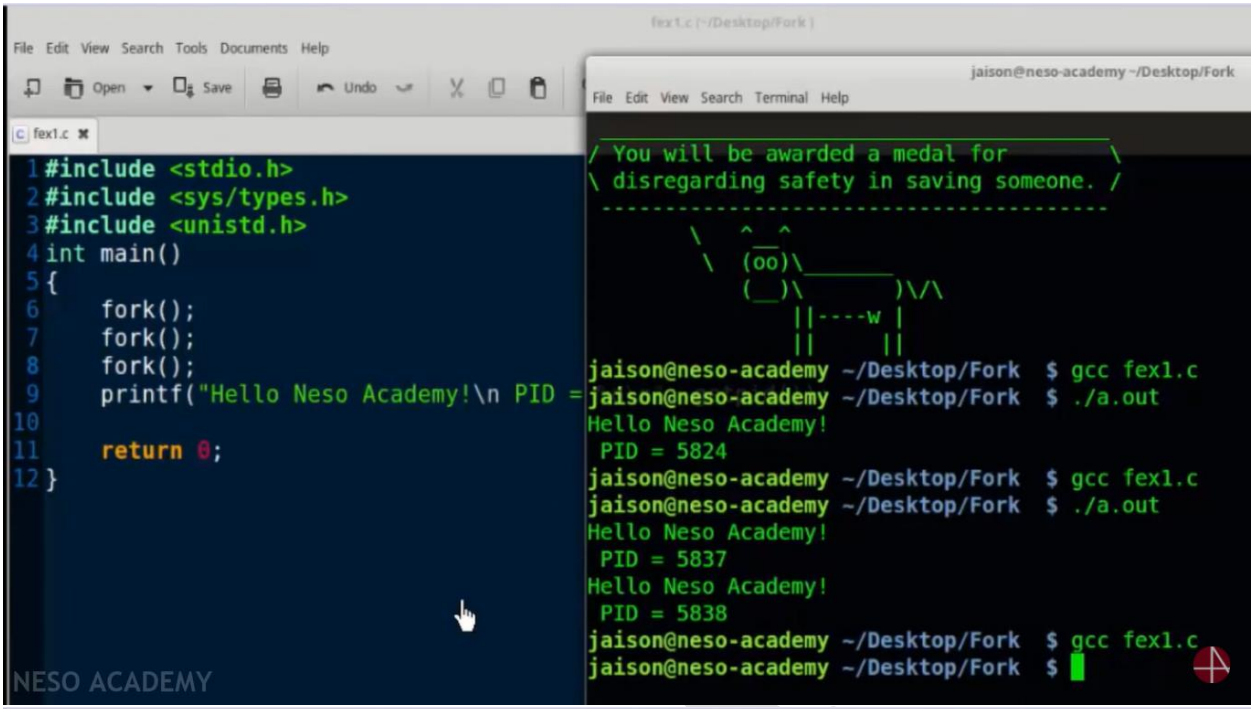


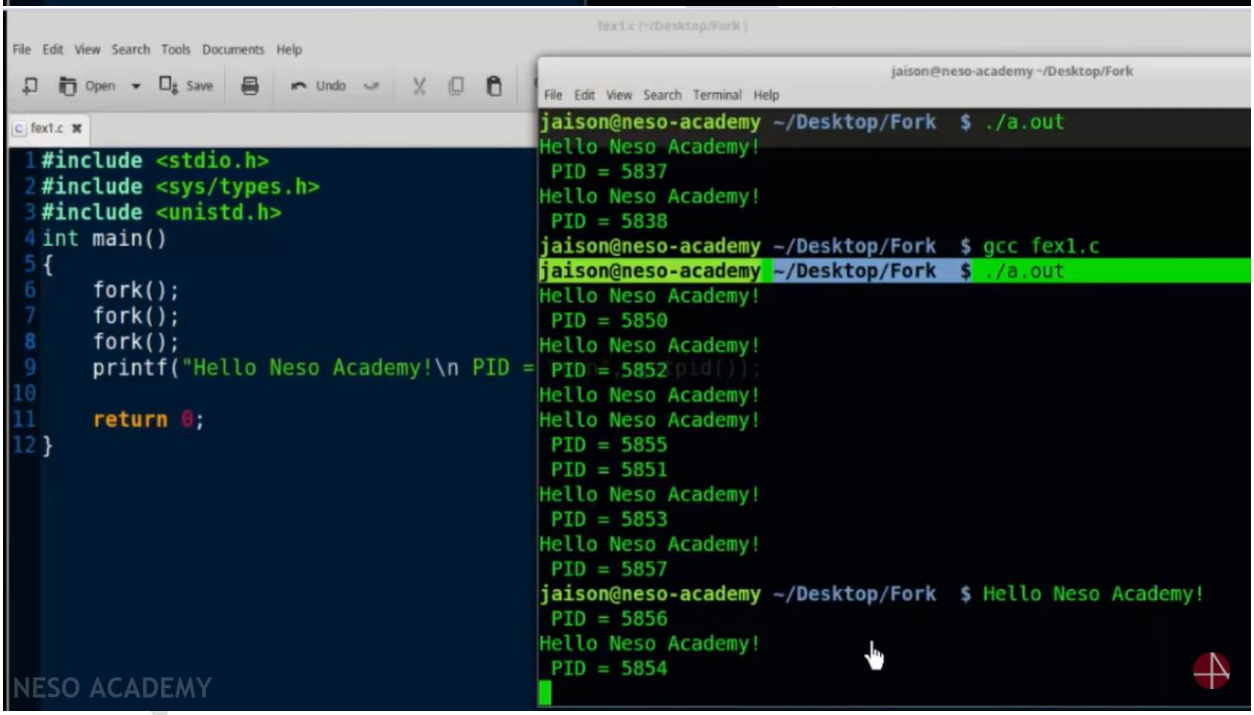


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

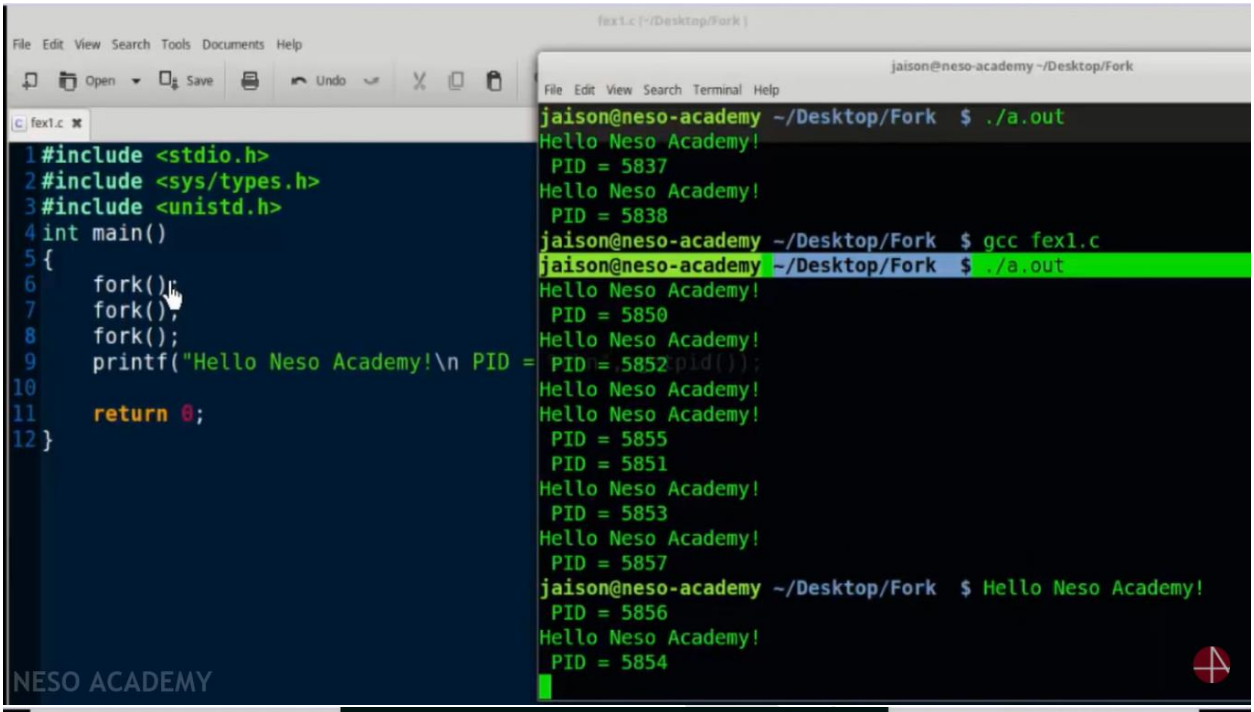


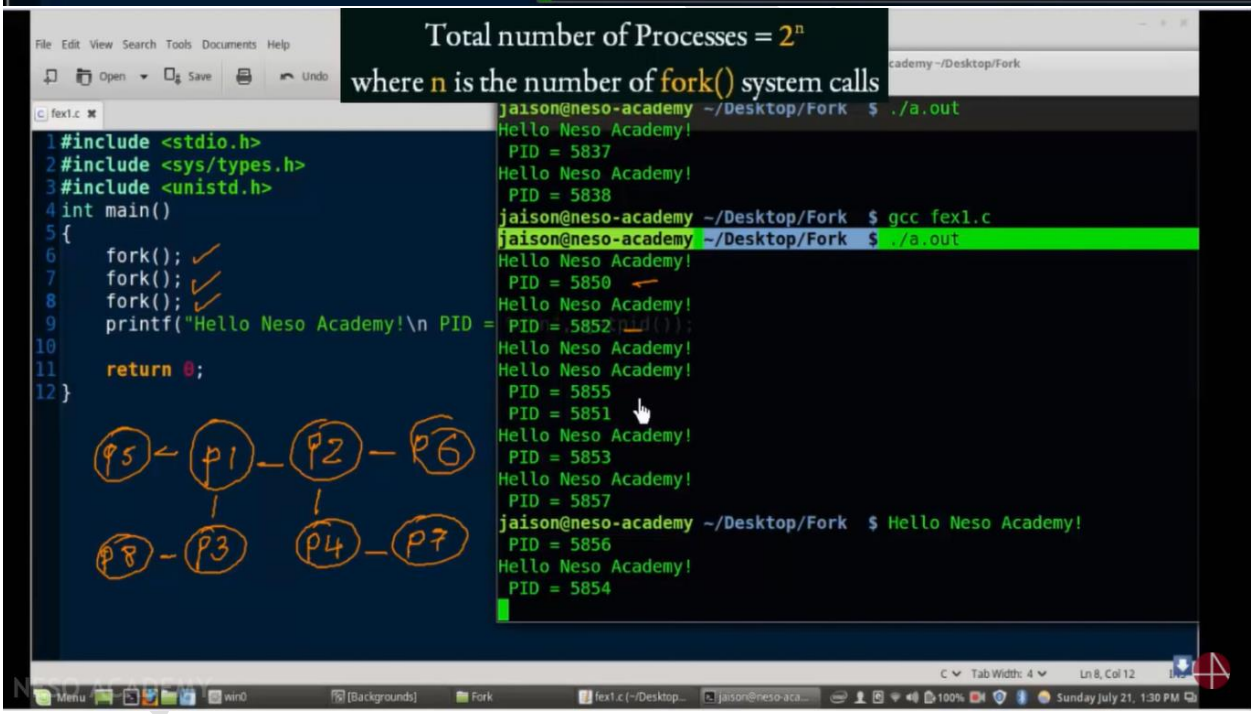
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



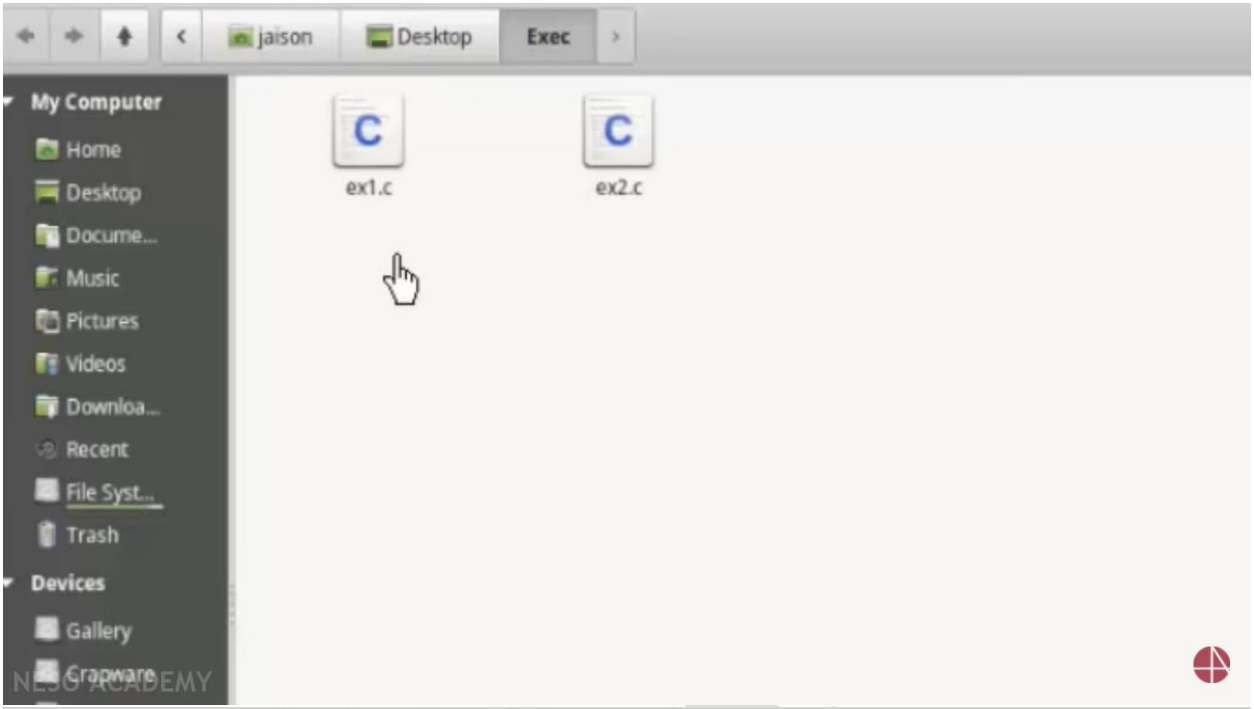


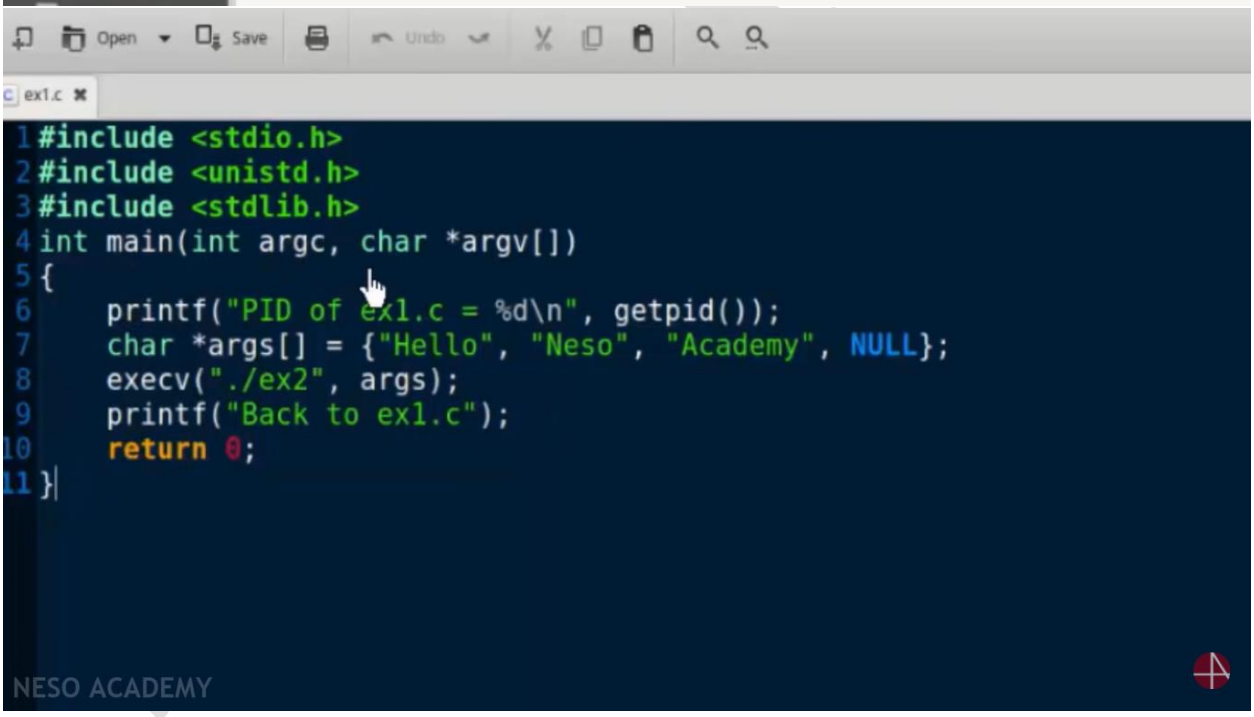
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



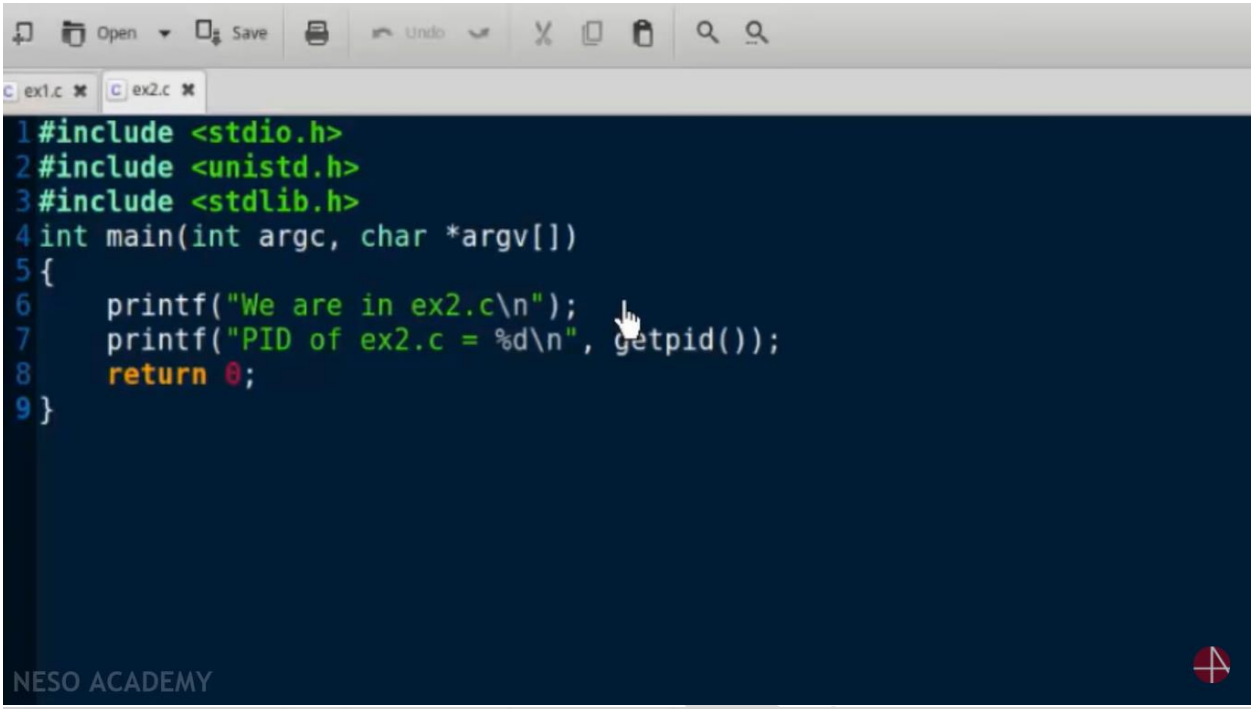


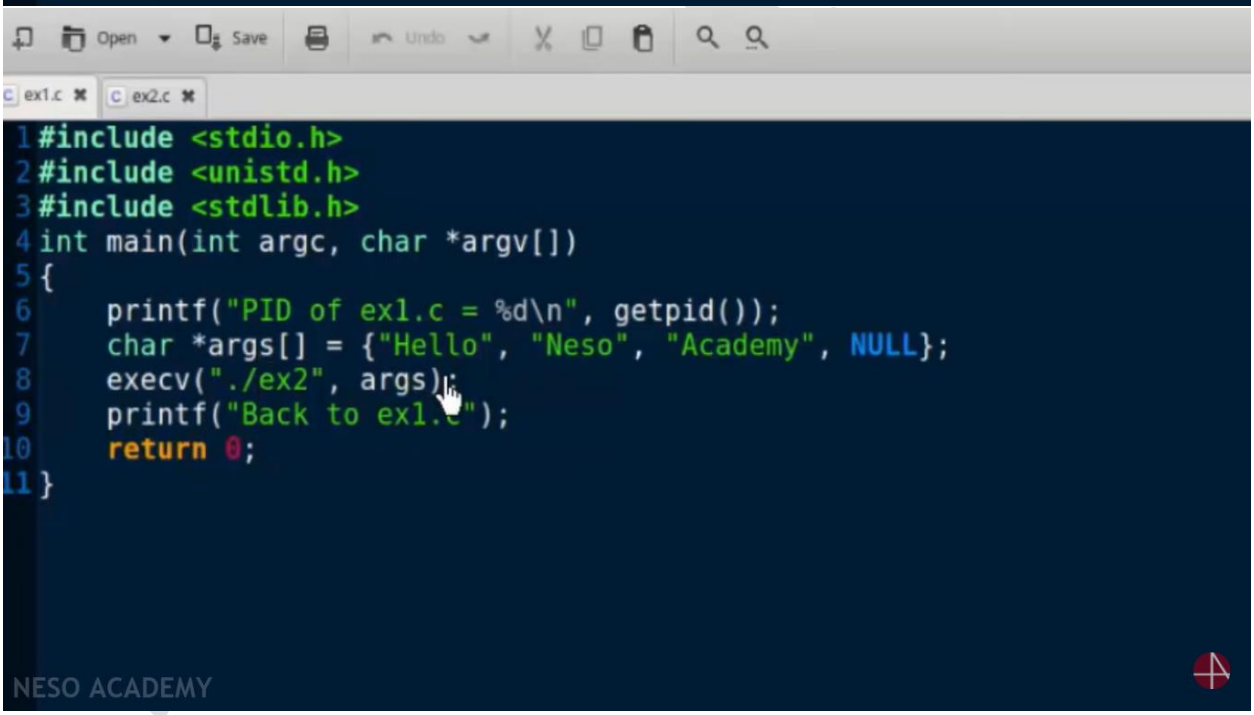
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



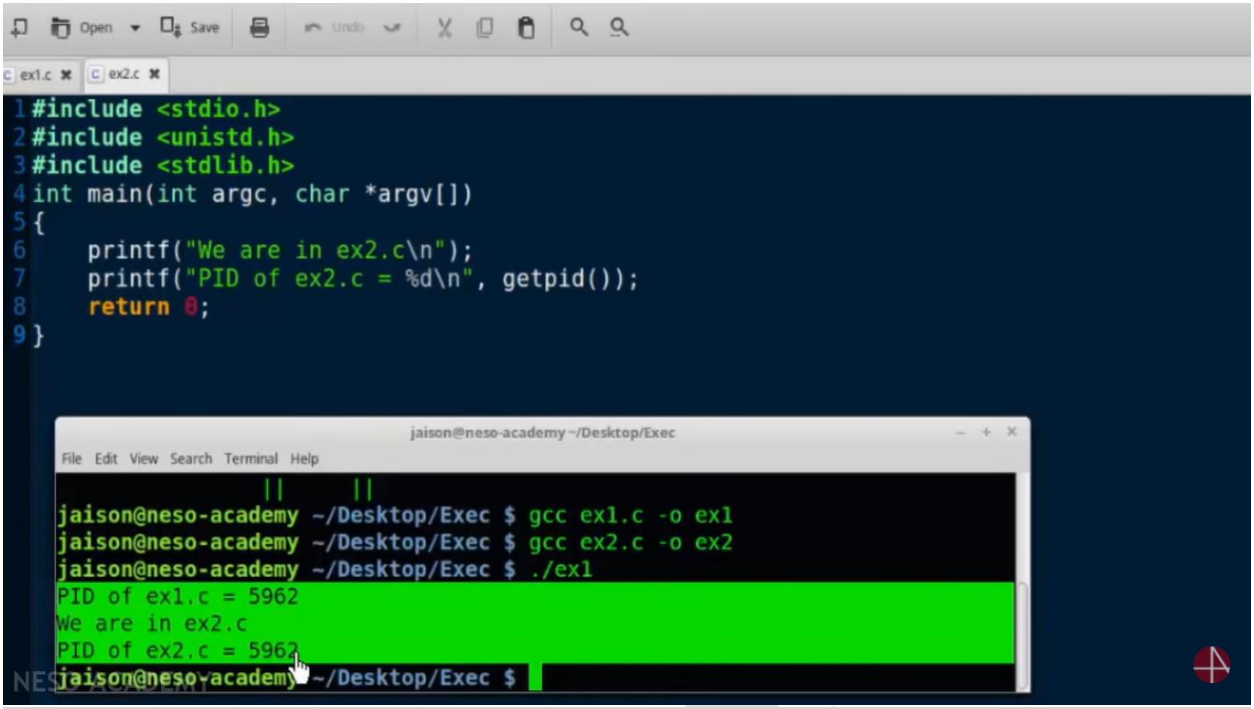


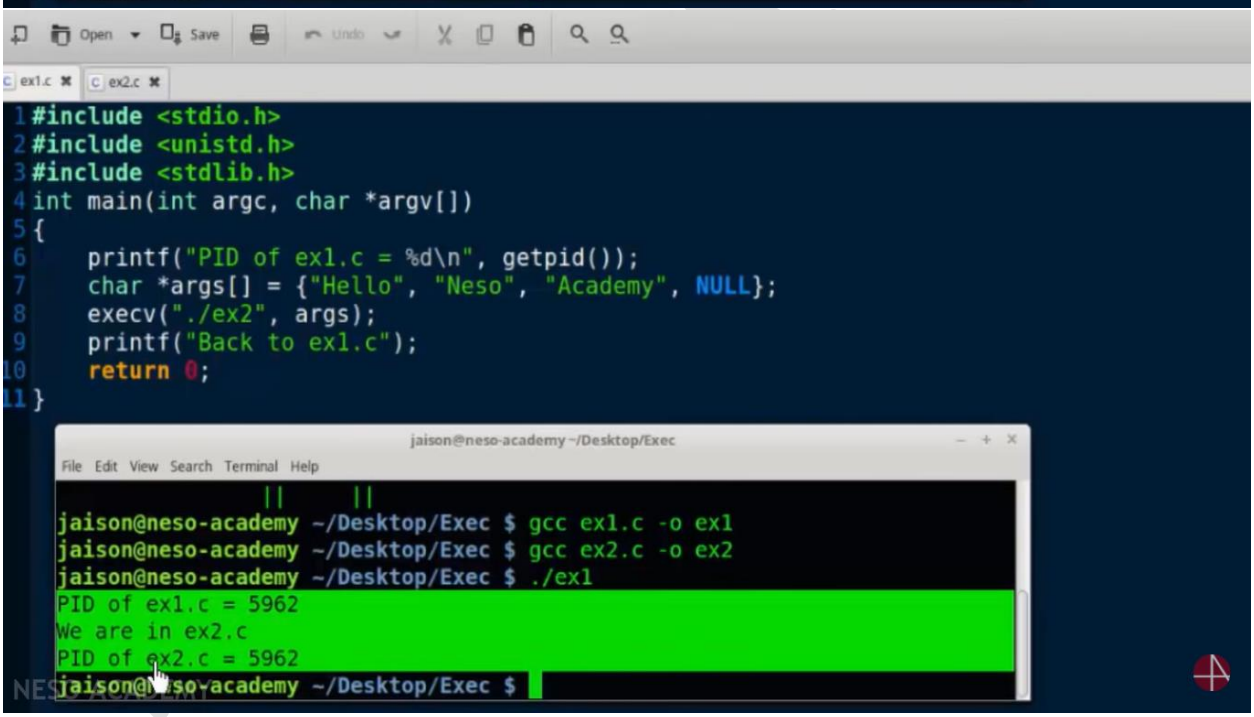
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Threading Issues (Part-1)

The fork() and exec() System Calls

The semantics of the fork() and exec() system calls change in a multithreaded program.

Issue

If one thread in a program calls fork(), does the new process duplicate all threads, or is the new process single-threaded?

Solution

Some UNIX systems have chosen to have two versions of fork (), one that duplicates all threads and another that duplicates only the thread that invoked the fork () system call.

But which version of fork () to use and when ?

Also, if a thread invokes the exec () system call, the program specified in the parameter to exec () will replace the entire process —including all threads.

But which version of fork () to use and when ?

Also, if a thread invokes the exec () system call, the program specified in the parameter to exec () will replace the entire process —including all threads.

Which of the two versions of fork () to use depends on the application.

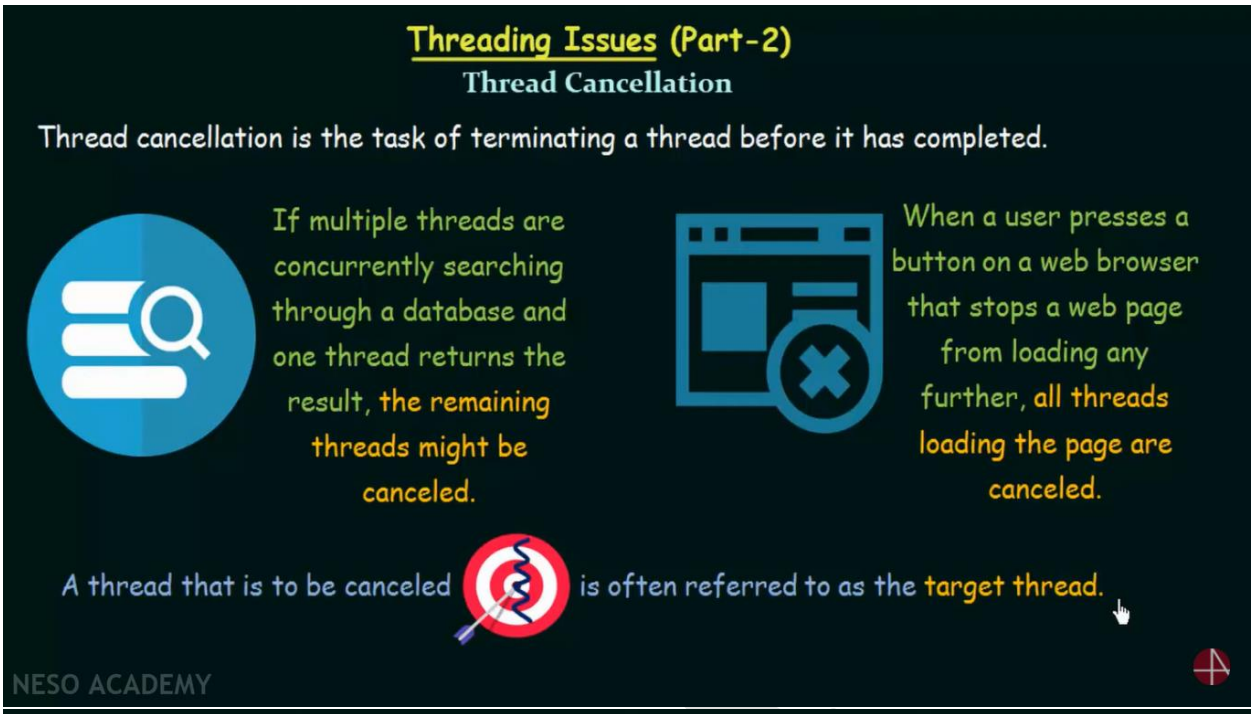
If exec() is called immediately after forking

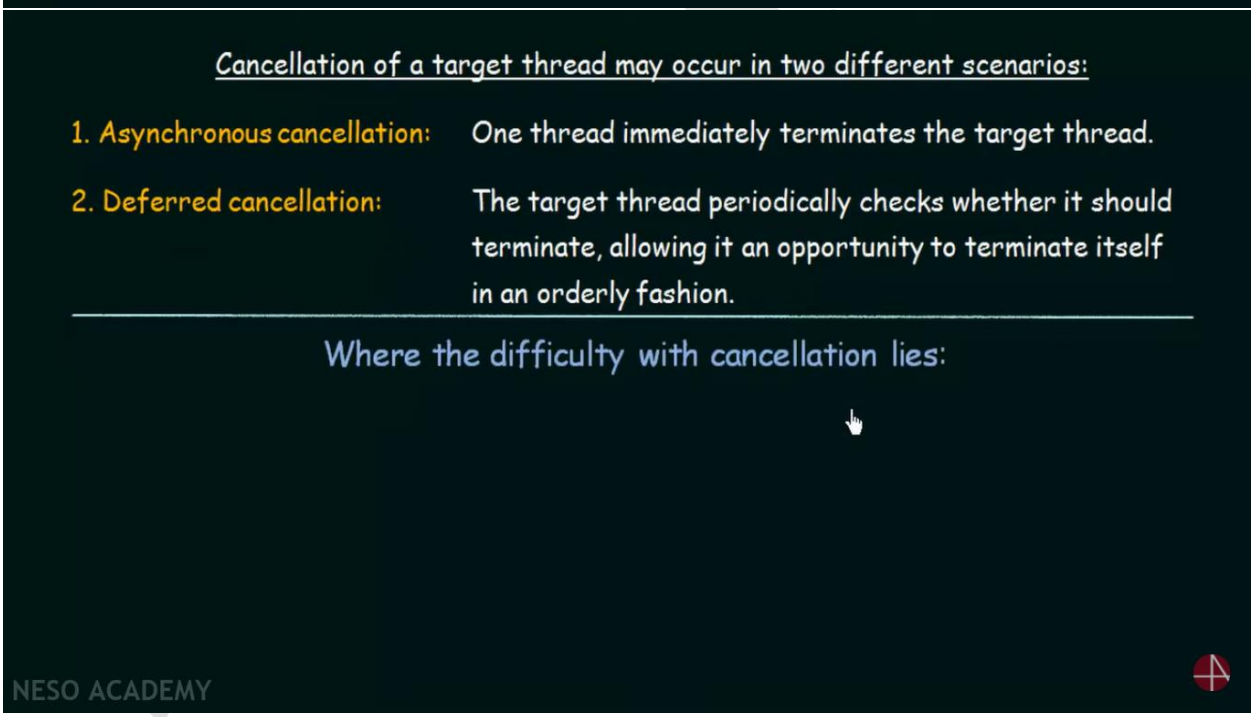
Then duplicating all threads is unnecessary, as the program specified in the parameters to exec () will replace the process.
In this instance, duplicating only the calling thread is appropriate.

If the separate process does not call exec () after forking

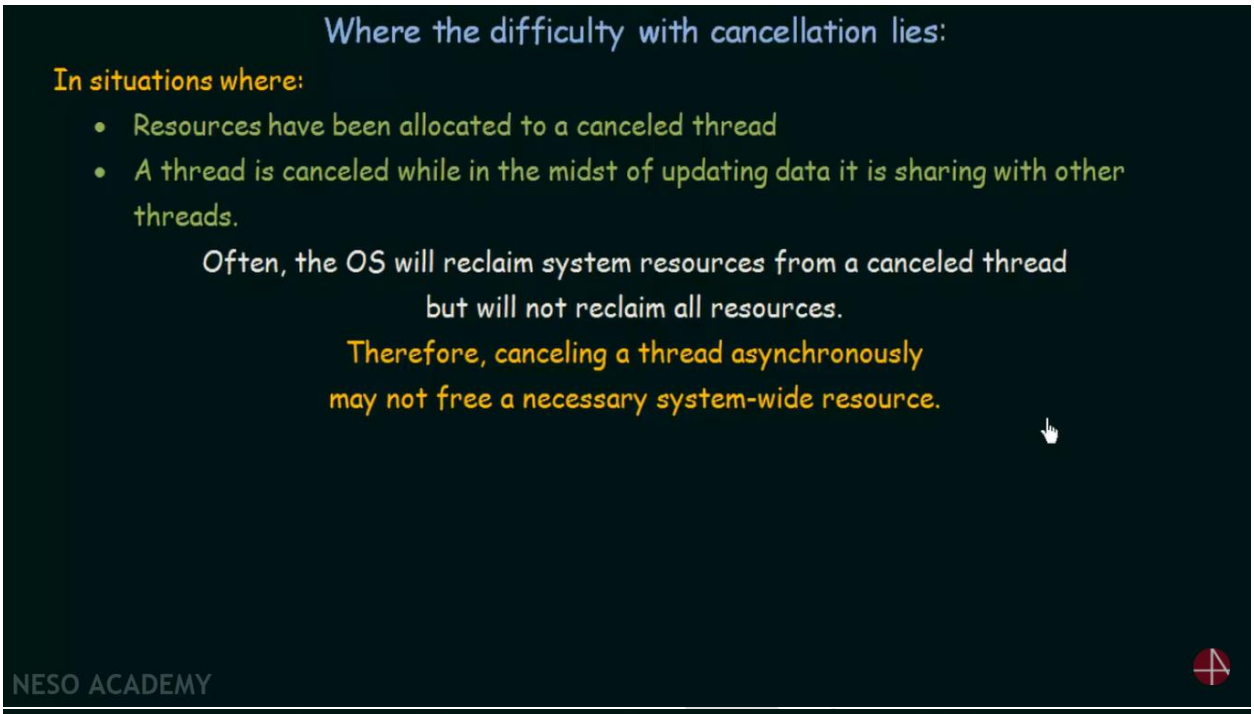
Then the separate process should duplicate all threads.

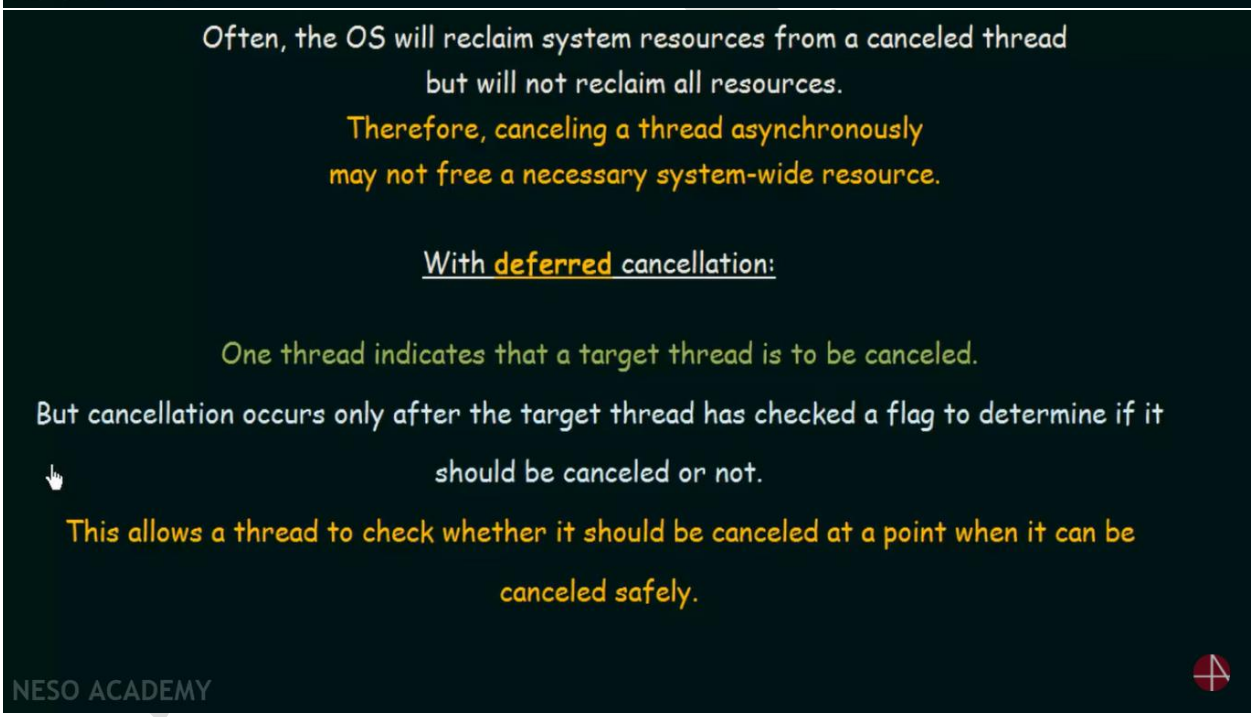
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



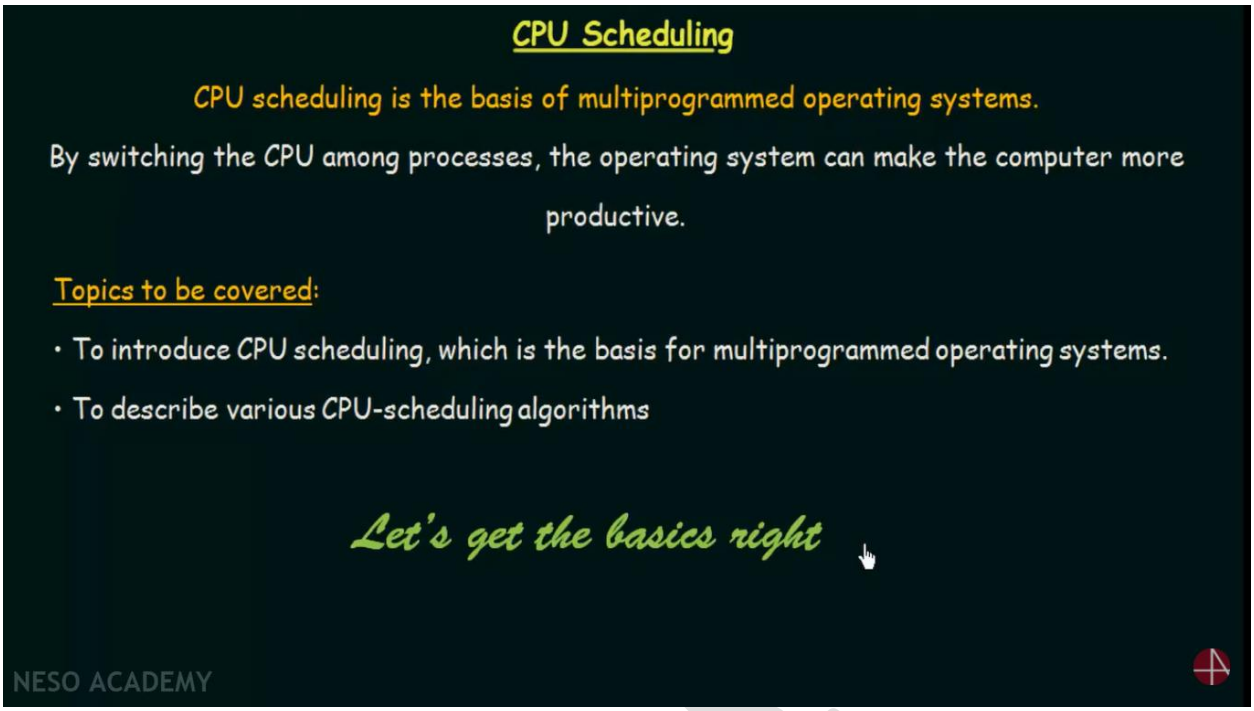


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Let's get the basics right

- In a single-processor system, only one process can run at a time.
- Any others must wait until the CPU is free and can be rescheduled.
    - The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization.
    - A process is executed until it must wait, typically for the completion of some I/O request.

In a simple computer system, the CPU then just sits idle.
All this waiting time is wasted; no useful work is accomplished.

NESO ACADEMY

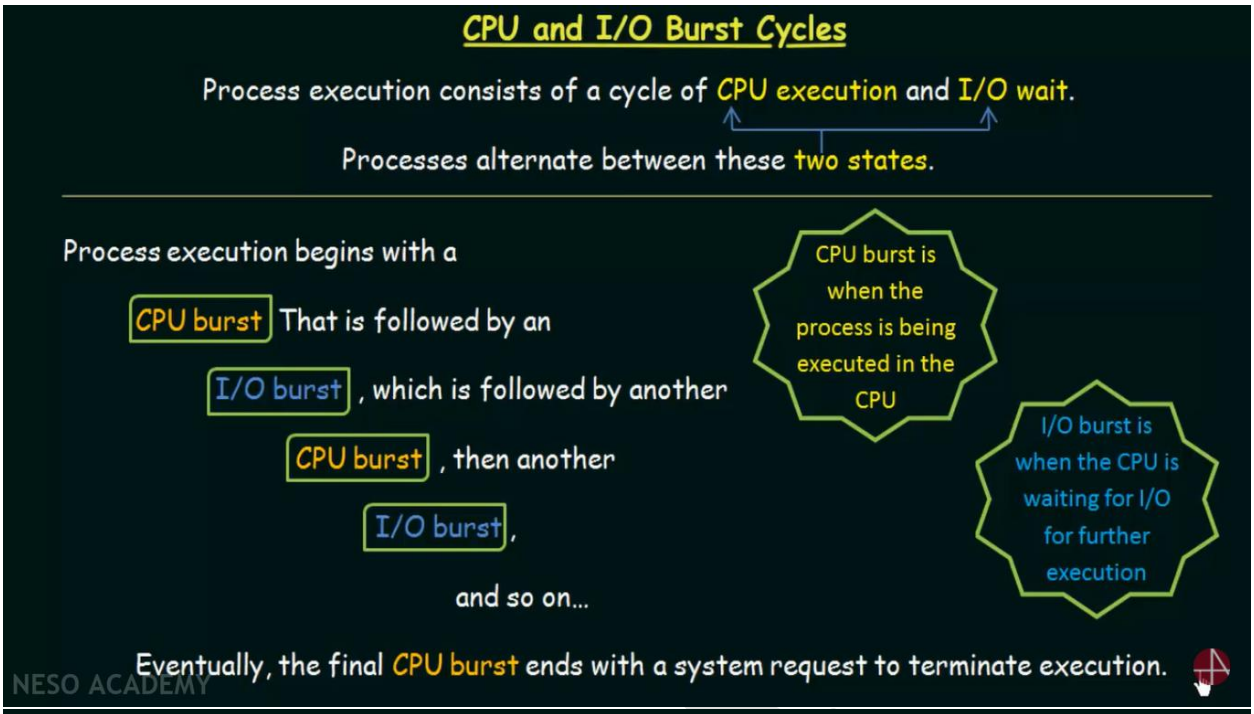
In a simple computer system, the CPU then just sits idle.
All this waiting time is wasted; no useful work is accomplished.

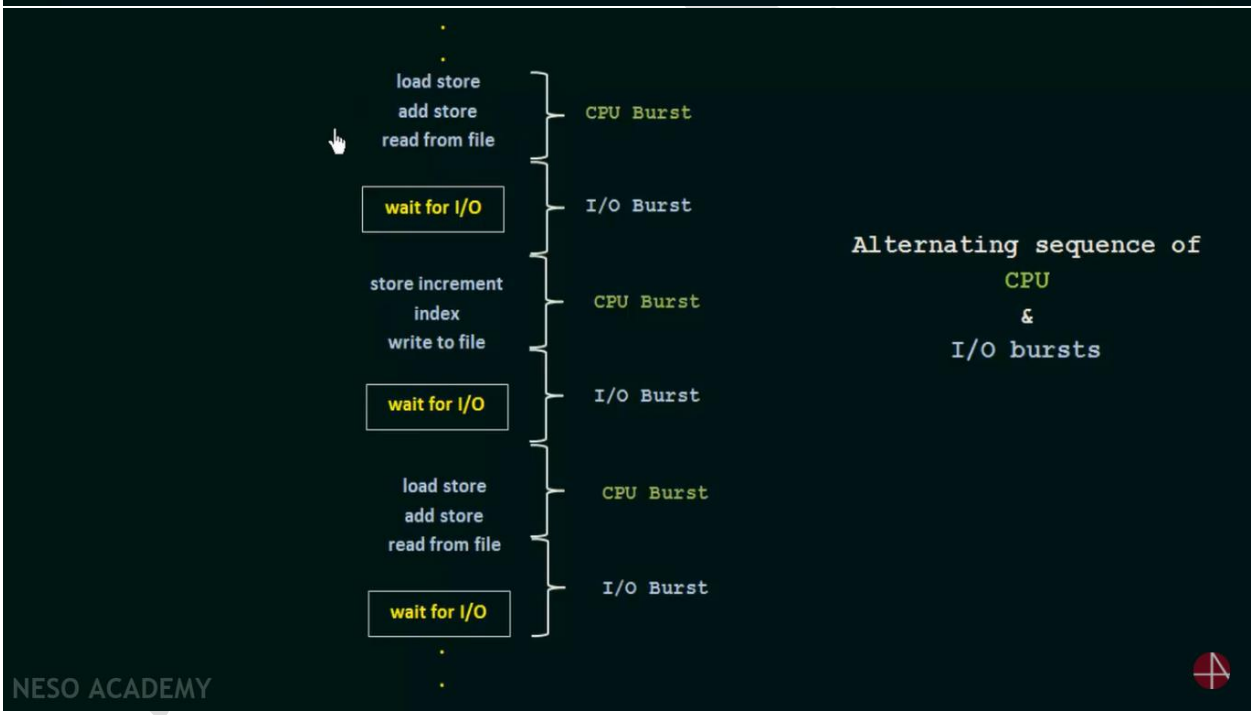
- With multiprogramming, we try to use this time productively.
    - Several processes are kept in memory at one time.

- When one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process and this pattern continues.

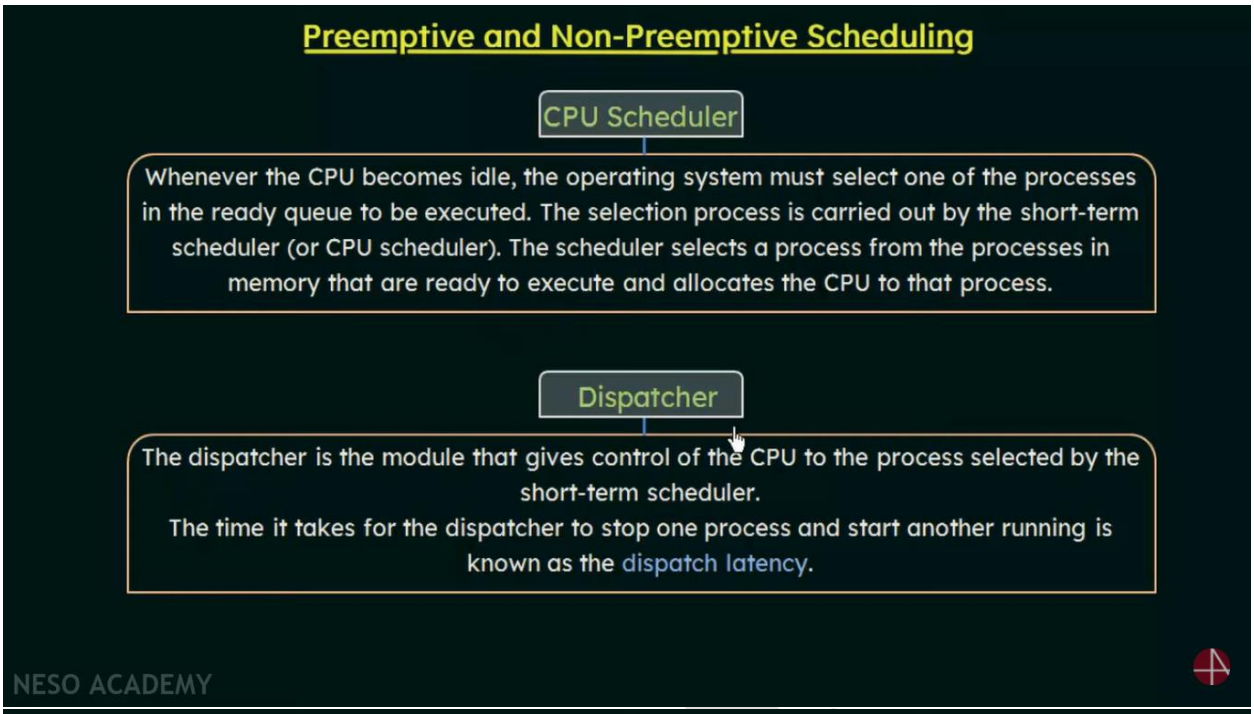
NESO ACADEMY

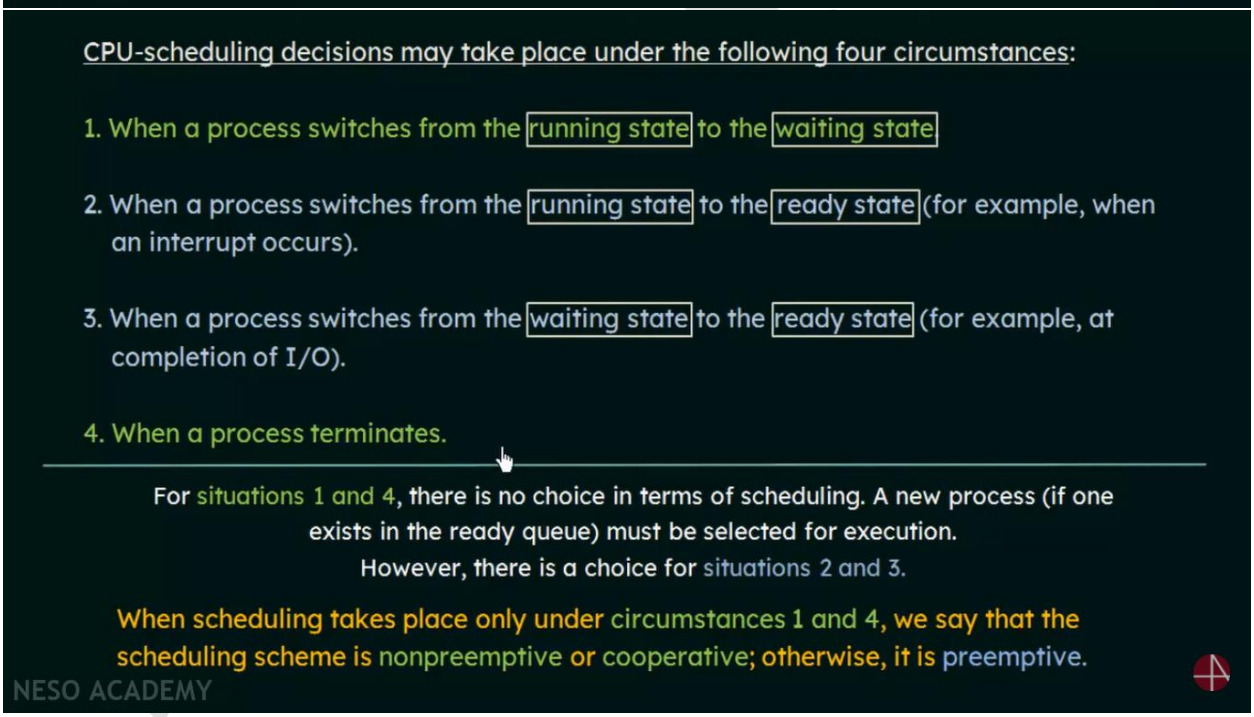
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



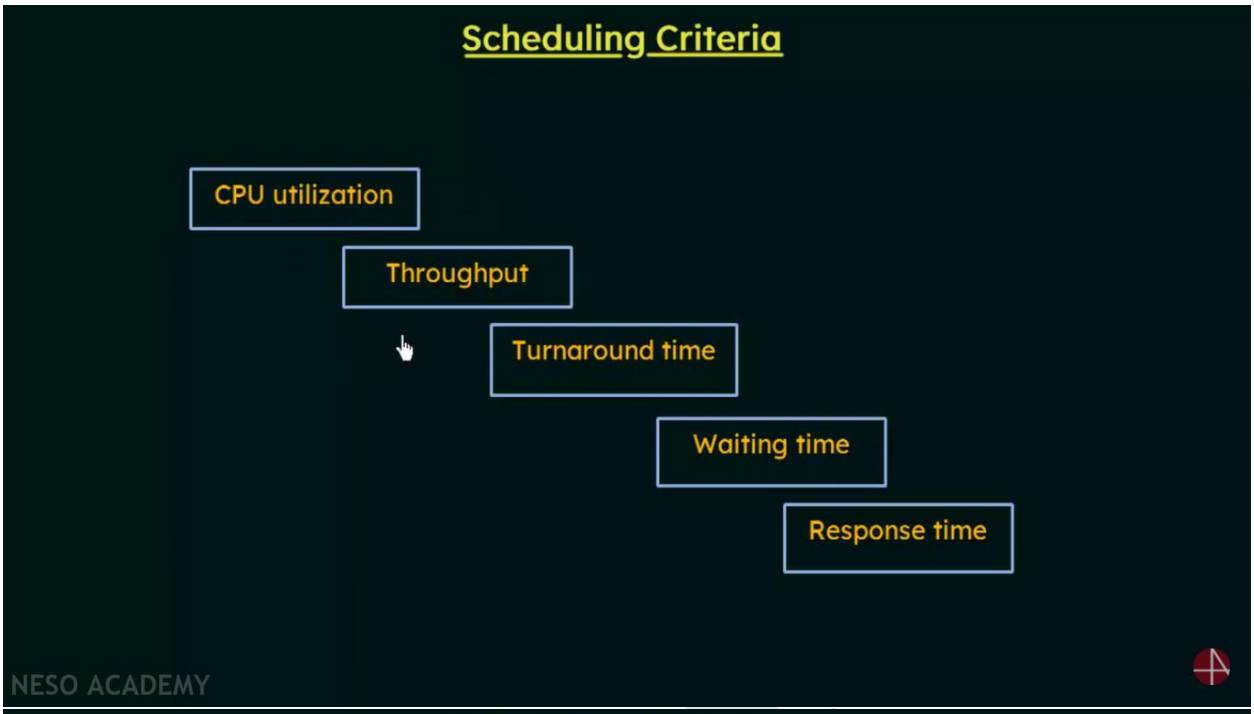


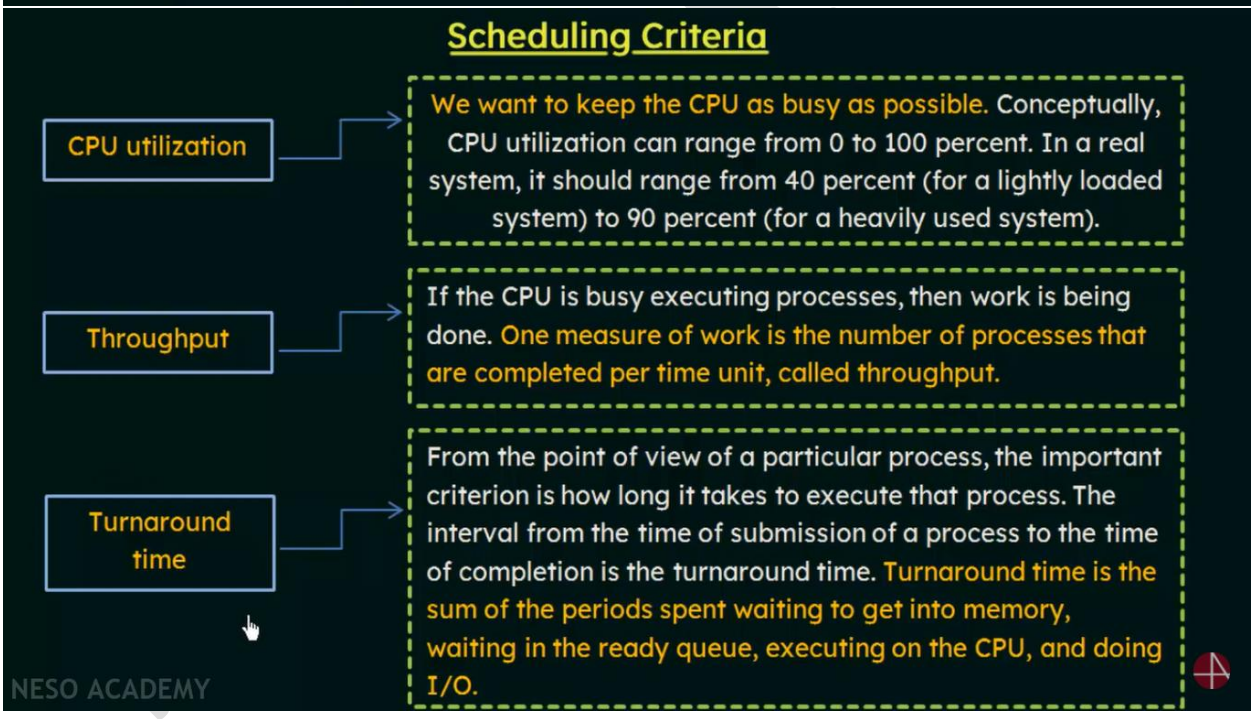
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Waiting time

The CPU scheduling algorithm does not affect the amount of time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

Response time

In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.

NESO ACADEMY

Scheduling Algorithms
(First-Come, First-Served Scheduling)

- By far the simplest CPU-scheduling algorithm.
- The process that requests the CPU first is allocated the CPU first.
- The implementation of the FCFS policy is easily managed with a FIFO queue.

- When a process enters the ready queue, its PCB is linked onto the tail of the queue.
- When the CPU is free, it is allocated to the process at the head of the queue.
- The running process is then removed from the queue.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Consider the following set of processes that arrive at time 0

Process	Burst Time (ms)
P1	24
P2	3
P3	3

If the processes arrive in the order P1, P2, P3, and are served in FCFS order, we get the result shown in the following Gantt chart:

Waiting Time for P1 = 0 ms
Waiting Time for P2 = 24 ms
Waiting Time for P3 = 27 ms

Average Waiting Time = (0 + 24 + 27)/3 = 17 ms

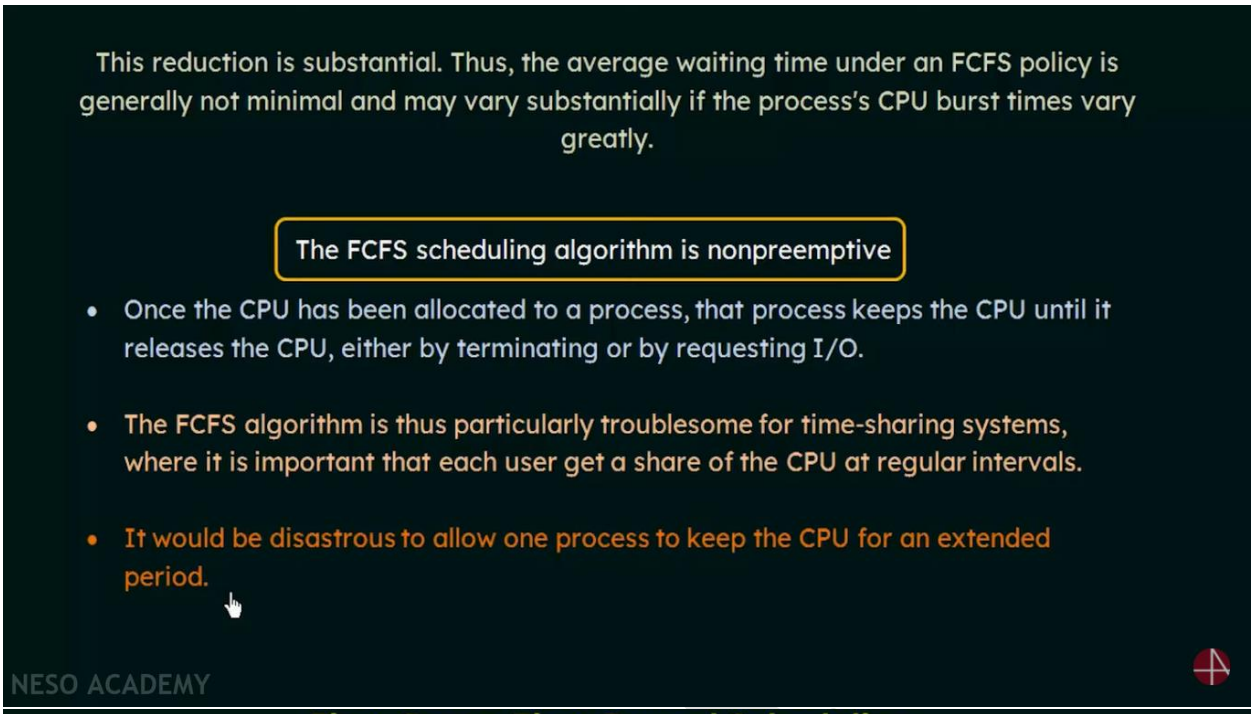
If the processes arrive in the order P2, P3, P1, however the result will be shown in the following Gantt chart:

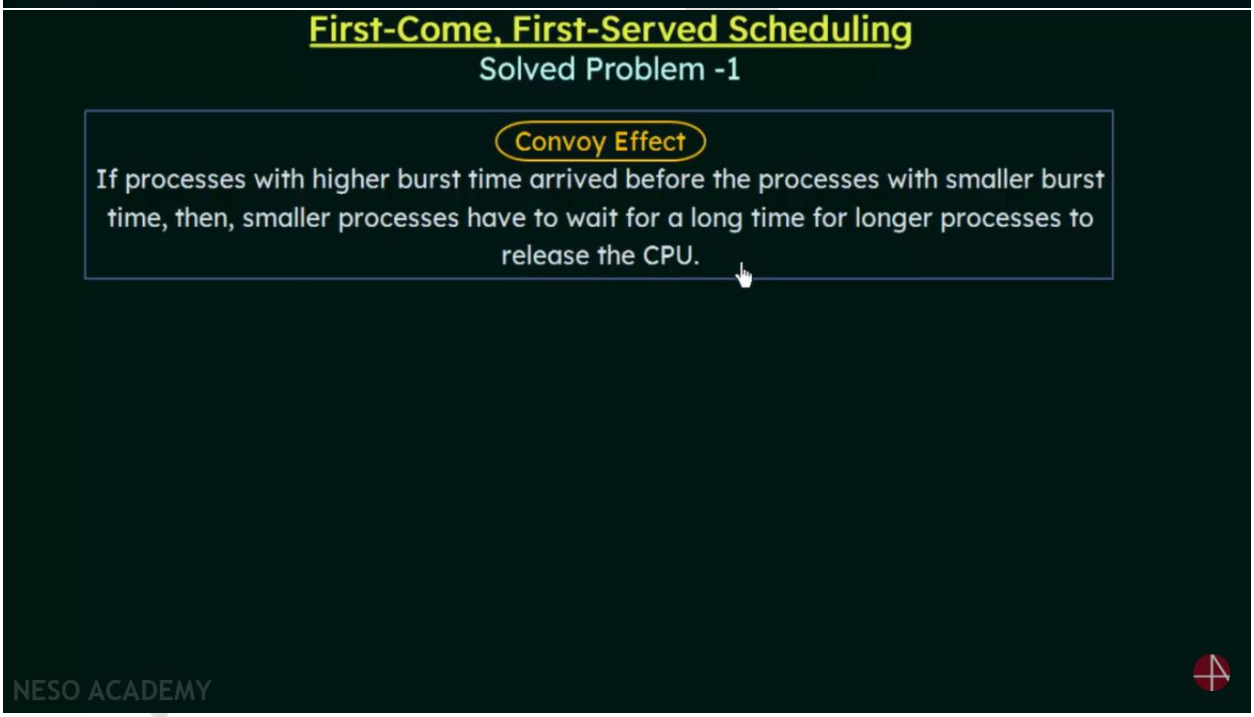
Waiting Time for P1 = 6 ms
Waiting Time for P2 = 0 ms
Waiting Time for P3 = 3 ms

Average Waiting Time = (6 + 0 + 3)/3 = 3 ms

This reduction is substantial. Thus, the average waiting time under an FCFS policy is generally not minimal and may vary substantially if the process's CPU burst times vary greatly.

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Consider the set of 5 processes whose arrival time and burst time are given below:

Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Calculate the average waiting time and average turnaround time, if FCFS Scheduling Algorithm is followed.

NESO ACADEMY

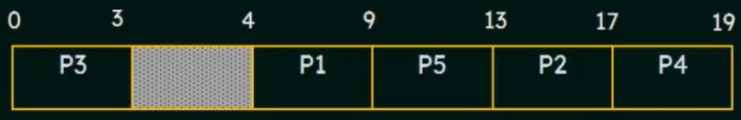
Consider the set of 5 processes whose arrival time and burst time are given below:

Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Calculate the average waiting time and average turnaround time, if FCFS Scheduling Algorithm is followed.

Solution:

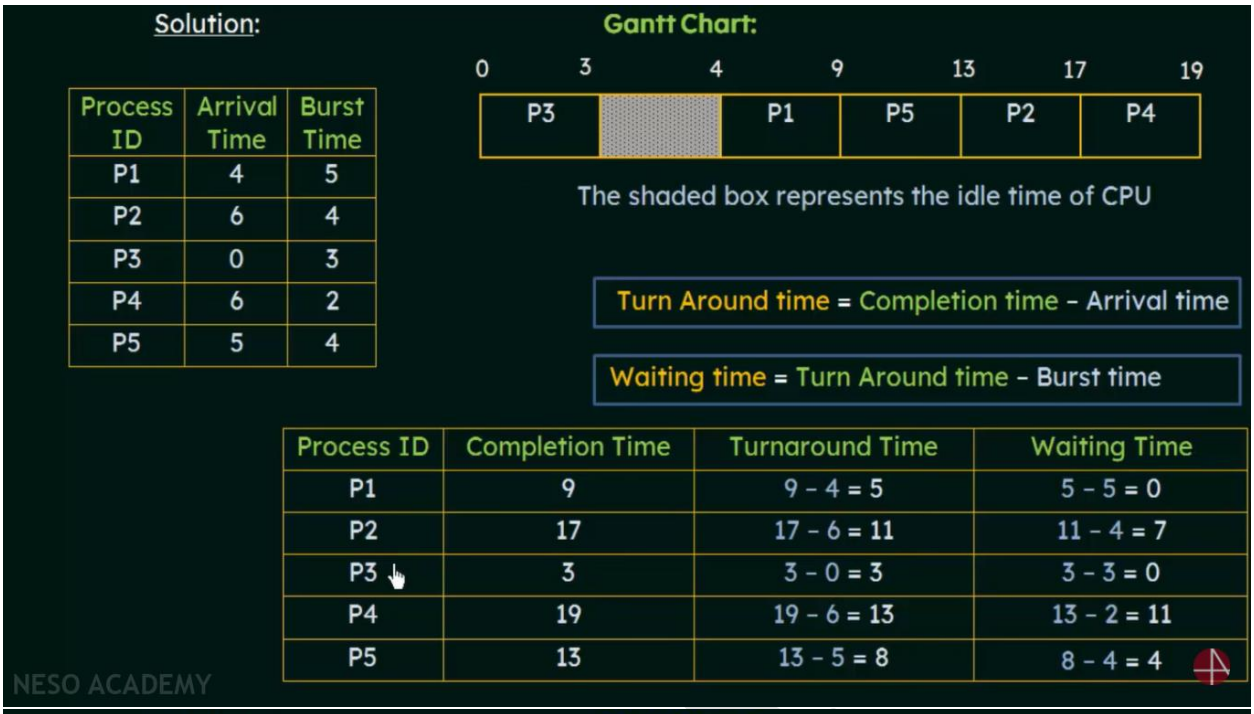
Gantt Chart:



The shaded box represents the idle time of CPU

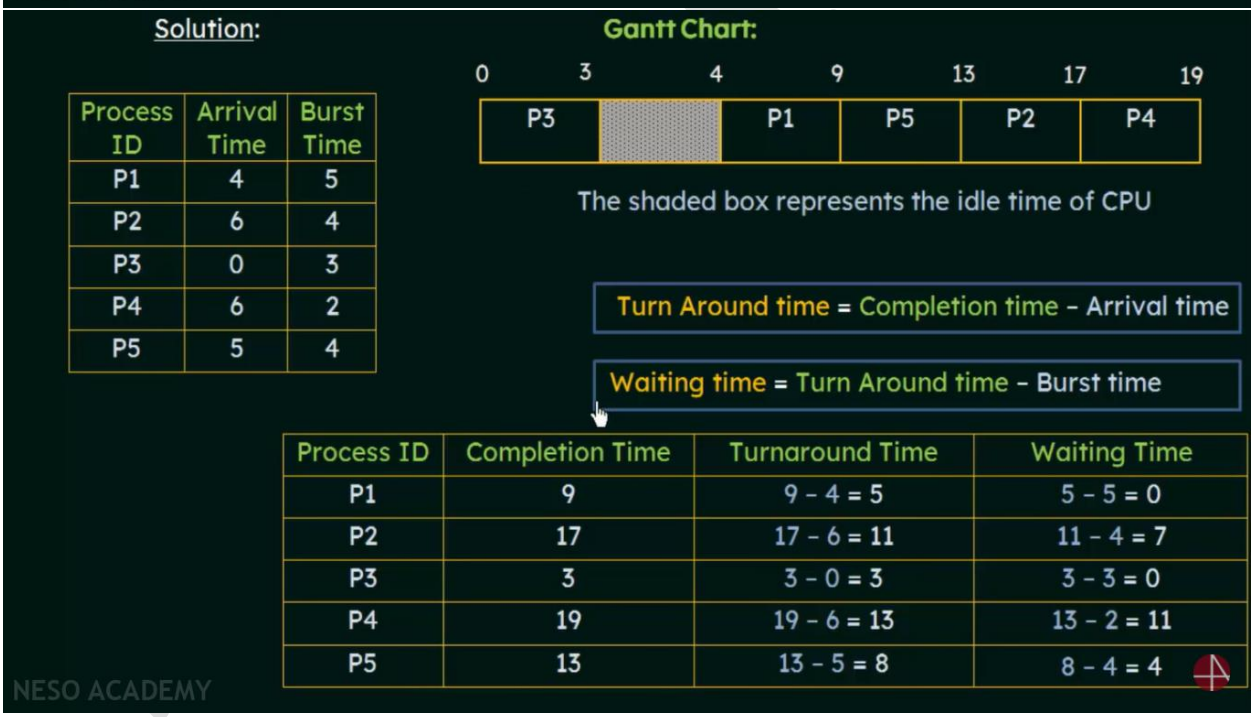
NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	9 – 4 = 5	5 – 5 = 0
P2	17	17 – 6 = 11	11 – 4 = 7
P3	3	3 – 0 = 3	3 – 3 = 0
P4	19	19 – 6 = 13	13 – 2 = 11
P5	13	13 – 5 = 8	8 – 4 = 4



Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	9 – 4 = 5	5 – 5 = 0
P2	17	17 – 6 = 11	11 – 4 = 7
P3	3	3 – 0 = 3	3 – 3 = 0
P4	19	19 – 6 = 13	13 – 2 = 11
P5	13	13 – 5 = 8	8 – 4 = 4

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	9 – 4 = 5	5 – 5 = 0
P2	17	17 – 6 = 11	11 – 4 = 7
P3	3	3 – 0 = 3	3 – 3 = 0
P4	19	19 – 6 = 13	13 – 2 = 11
P5	13	13 – 5 = 8	8 – 4 = 4

Now,

Average Turn Around time = (5 + 11 + 3 + 13 + 8) / 5
= 40 / 5
= 8 units

Average waiting time = (0 + 7 + 0 + 11 + 4) / 5
= 22 / 5
= 4.4 units

NESO ACADEMY

First-Come, First-Served Scheduling

Solved Problem - 2

The arrival times and burst times for a set of 6 processes are given in the table below:

Process ID	Arrival Time	Burst Time
P1	0	3
P2	1	2
P3	2	1
P4	3	4
P5	4	5
P6	5	2

If FCFS Scheduling Algorithm is followed and there is 1 unit of overhead in scheduling the processes, find the efficiency of the algorithm.

NESO ACADEMY

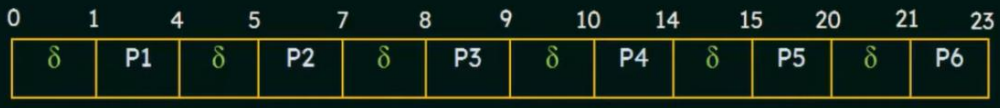
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Process ID	Arrival Time	Burst Time
P1	0	3
P2	1	2
P3	2	1
P4	3	4
P5	4	5
P6	5	2

If FCFS Scheduling Algorithm is followed and there is 1 unit of overhead in scheduling the processes, find the efficiency of the algorithm.

Solution:

Gantt Chart:



Here, δ denotes the unit overhead in scheduling the processes

NESO ACADEMY

Solution:

Gantt Chart:

0 1 4 5 7 8 9 10 14 15 20 21 23
δ P1 δ P2 δ P3 δ P4 δ P5 δ P6

Here, δ denotes the unit overhead in scheduling the processes

Now,

Useless time or Wasted time = 6 x δ = 6 x 1
= 6 units

Total time = 23 units

Useful time = 23 units – 6 units
= 17 units

Efficiency (η) = Useful time / Total Time
= 17 units / 23 units
= 0.7391
= 73.91%

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Scheduling Algorithms
(Shortest-Job-First Scheduling)

- This algorithm associates with each process the length of the process's next CPU burst.
- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.
- If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

The SJF algorithm can be either preemptive or nonpreemptive

A more appropriate term for this scheduling method would be the **Shortest-Next-CPU-Burst Algorithm** because scheduling depends on the length of the next CPU burst of a process, rather than its total length.

NESO ACADEMY

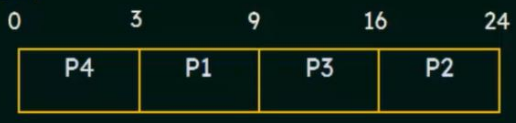
Example of SJF Scheduling (Non-Premptive)

Consider the following set of processes,with the length of the CPU burst given in milliseconds:

Process ID	Burst Time
P1	6
P2	8
P3	7
P4	3

Waiting Time for P1 = 3 ms
Waiting Time for P2 = 16 ms
Waiting Time for P3 = 9 ms
Waiting Time for P4 = 0 ms

Gantt Chart:



Average Waiting Time
= (3 + 16 + 9 + 0)/4 = 7 ms

By comparison, if we were using the FCFS scheduling scheme, the average waiting time would be 10.25 milliseconds.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Example of SJF Scheduling (Premptive)

Consider the following four processes, with the length of the CPU burst given in milliseconds and the processes arrive at the ready queue at the times shown:

Process ID	Arrival Time	Burst Time
P1	0	~~8~~ 7
P2	1	4
P3	2	9
P4	3	5

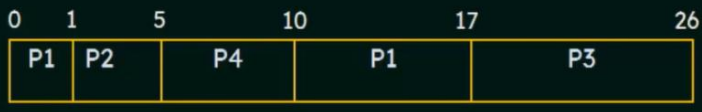
Waiting Time for P1 = (10 – 1 - 0) = 9 ms

Waiting Time for P2 = (1 – 0 - 1) = 0 ms

Waiting Time for P3 = (17 – 0 - 2) = 15 ms

Waiting Time for P4 = (5 – 0 - 3) = 2 ms

Gantt Chart:



Average Waiting Time

= (9 + 0 + 15 + 2)/4 = 6.5 ms

Waiting Time =Total waiting Time – No. of milliseconds Process executed – Arrival Time

Preemptive SJF scheduling is sometimes called Shortest-Remaining-Time-First Scheduling.

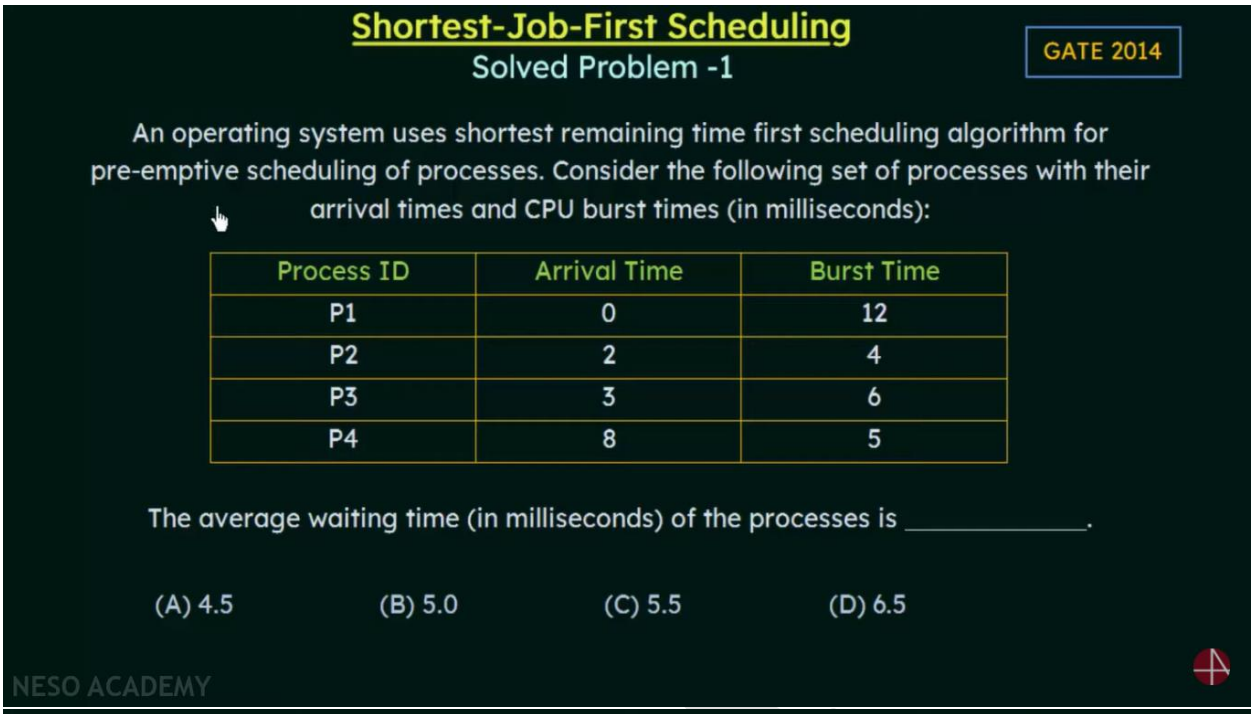
Problems with SJF Scheduling:

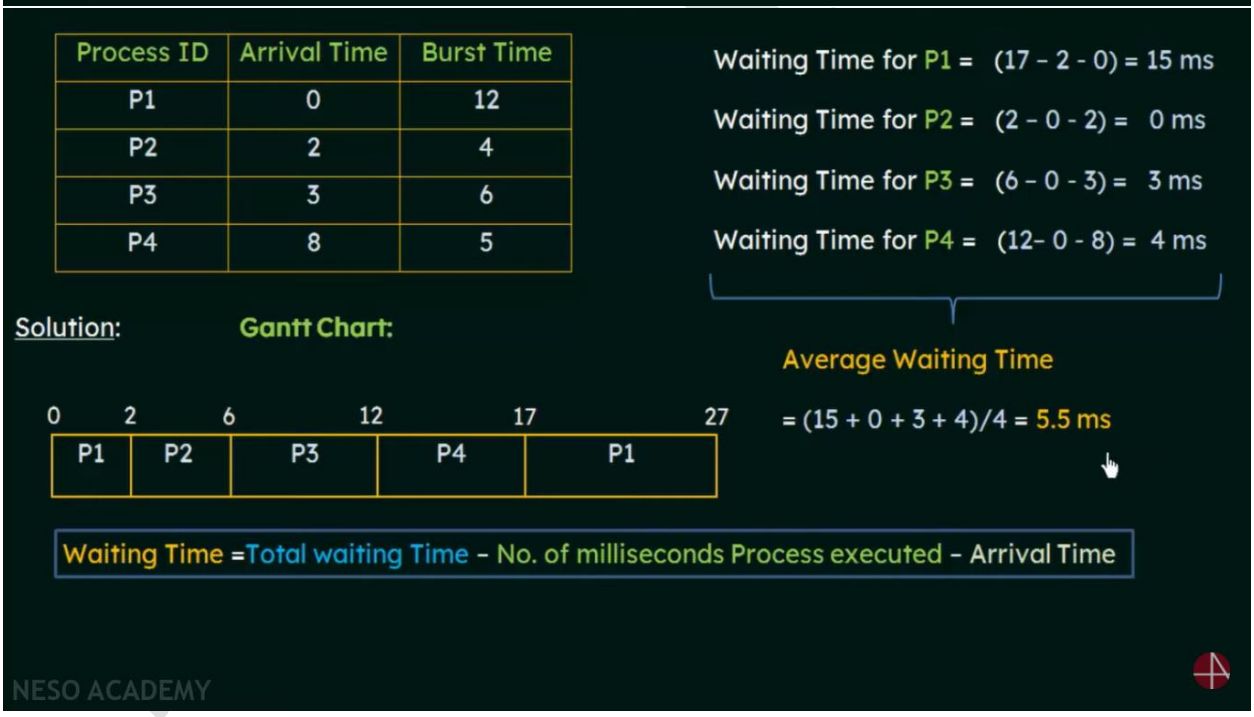
- The real difficulty with the SJF algorithm is knowing the length of the next CPU request.
- Although the SJF algorithm is optimal, it cannot be implemented at the level of short-term CPU scheduling.
- There is no way to know the length of the next CPU burst.

One approach is:

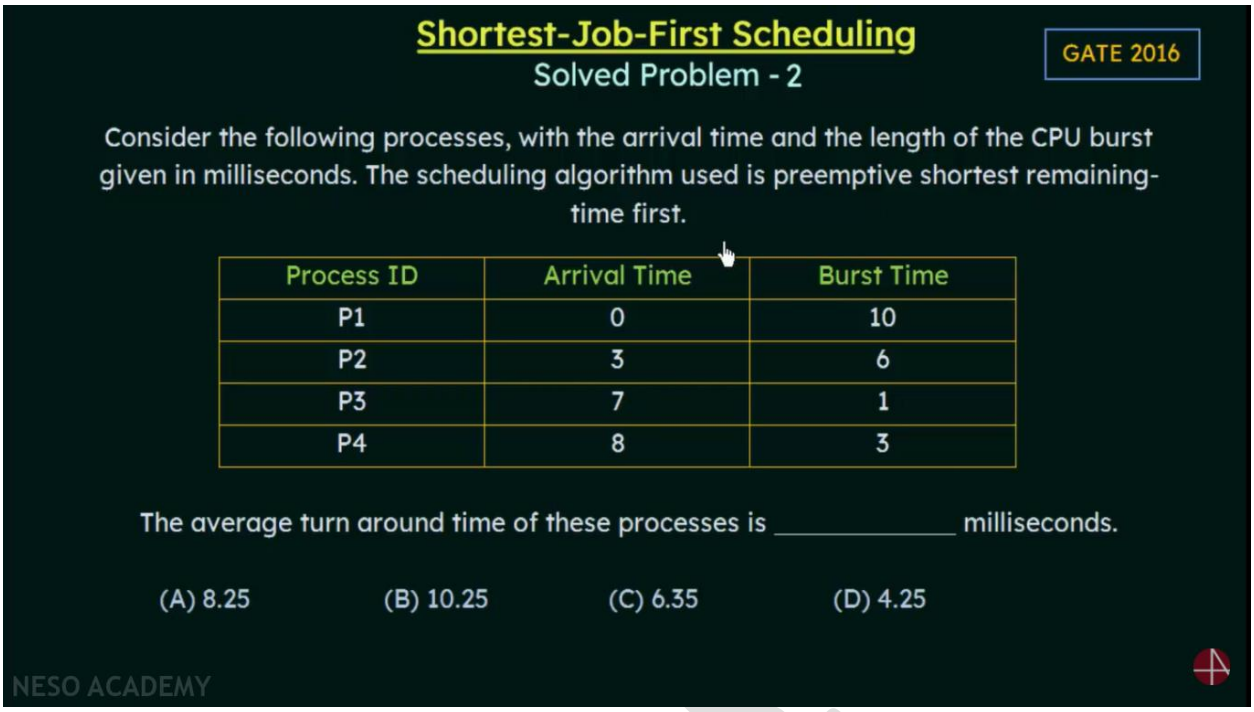
- To try to approximate SJF scheduling.
- We may not know the length of the next CPU burst, but we may be able to predict its value.
- We expect that the next CPU burst will be similar in length to the previous ones.
- Thus, by computing an approximation of the length of the next CPU burst, we can pick the process with the shortest predicted CPU burst.

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Process ID	Arrival Time	Burst Time
P1	0	~~10~~ 7
P2	3	~~6~~ 2
P3	7	1
P4	8	3

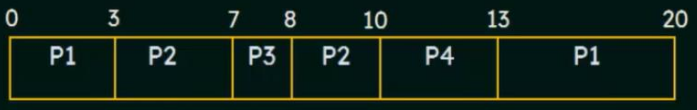
Turnaround Time for P1 = (20 – 0) = 20 ms

Turnaround Time for P2 = (10 – 3) = 7 ms

Turnaround Time for P3 = (8 – 7) = 1 ms

Turnaround Time for P4 = (13– 8) = 5 ms

Solution: Gantt Chart:



Average Turnaround Time

= (20 + 7 + 1 + 5)/4

= 33/4

= 8.25 ms

Turn Around time = Completion time – Arrival time

NESO ACADEMY

Scheduling Algorithms
(Priority Scheduling)

- A priority is associated with each process, and the CPU is allocated to the process with the highest priority.
- Equal-priority processes are scheduled in FCFS order.
- An SJF algorithm is simply a priority algorithm where the priority is the inverse of the (predicted) next CPU burst.
  The larger the CPU burst, the lower the priority, and vice versa.

Priority scheduling can be either preemptive or nonpreemptive.

A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process.

A nonpreemptive priority scheduling algorithm will simply put the new process at the head of the ready queue.

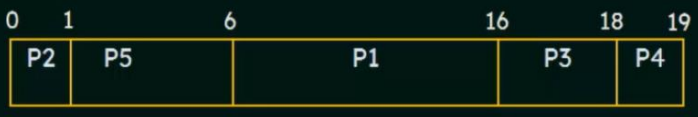
NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Consider the following set of processes, assumed to have arrived at time 0, in the order P1, P2, P3, P4, P5, with the length of the CPU burst given in milliseconds:

Process ID	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Using Priority Scheduling, we would schedule these processes according to the following Gantt Chart:



Waiting Time for P1 = 6 ms

Waiting Time for P2 = 0 ms

Waiting Time for P3 = 16 ms

Waiting Time for P4 = 18 ms

Waiting Time for P5 = 1 ms

Average Waiting Time

= (6 + 0 + 16 + 18 + 1) / 5

= 41 / 5 ms

= 8.2 ms

NESO ACADEMY

Problem with Priority Scheduling

- A major problem with priority scheduling algorithms is indefinite blocking, or starvation.
- A process that is ready to run but waiting for the CPU can be considered blocked.
- A priority scheduling algorithm can leave some low priority processes waiting indefinitely.
- In a heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU.

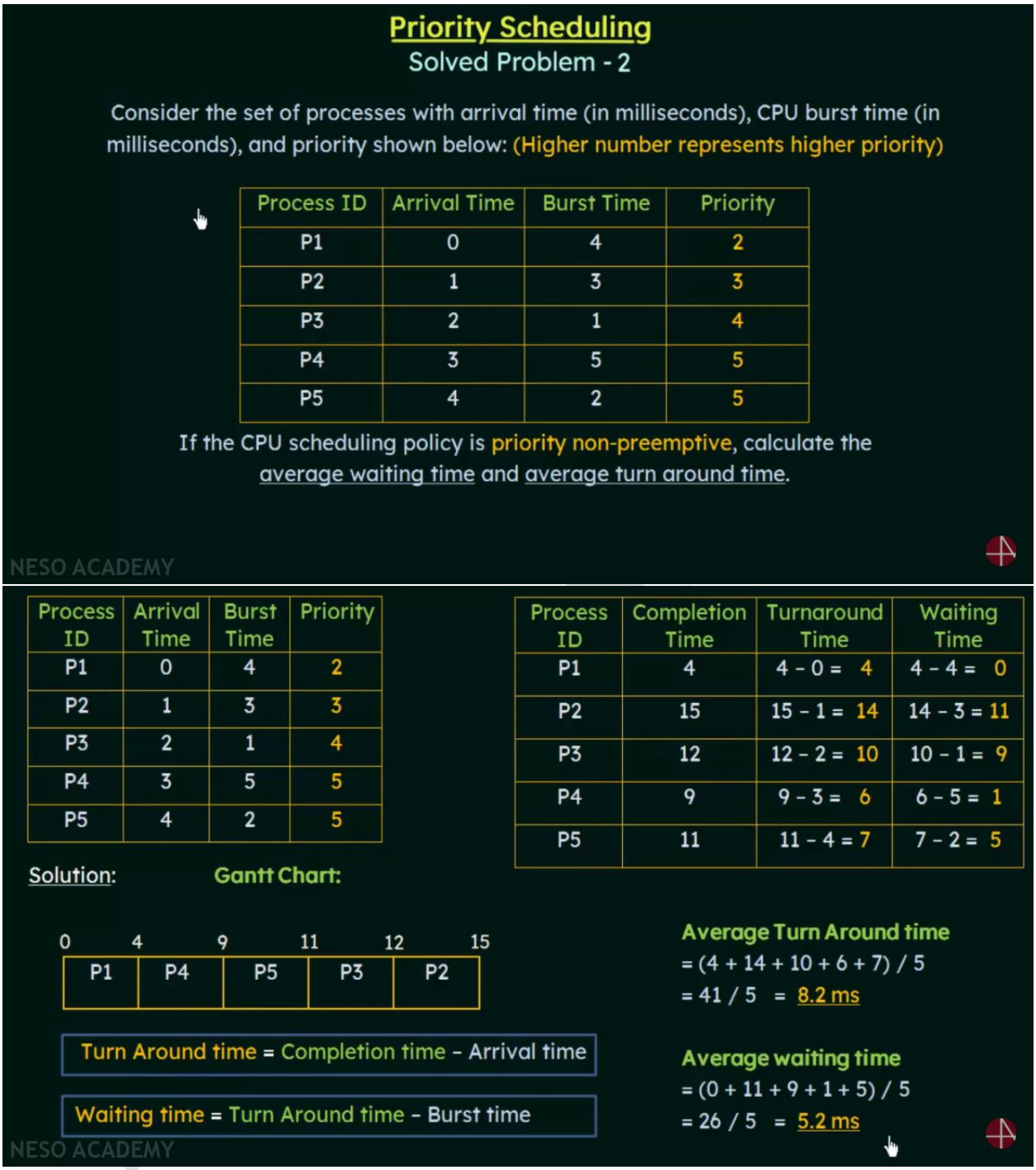
Solution to the Problem

- A solution to the problem of indefinite blockage of low-priority processes is **aging**.
- Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time.
- For example,
  If priorities range from 127 (low) to 0 (high), we could increase the priority of a waiting process by 1 every 15 minutes.
- Eventually, even a process with an initial priority of 127 would have the highest priority in the system and would be executed.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

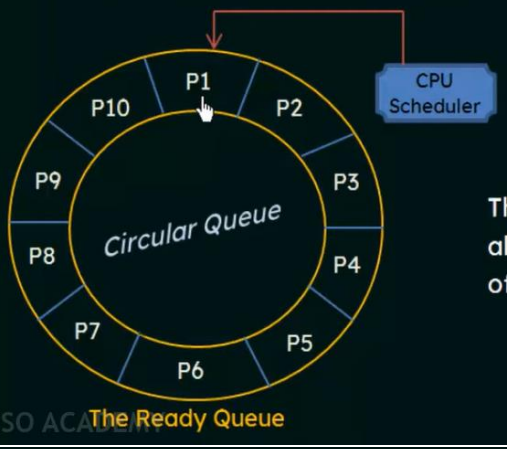
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Scheduling Algorithms
(Round-Robin Scheduling)

- The round-robin (RR) scheduling algorithm is designed especially for timesharing systems.
- It is similar to FCFS scheduling, but preemption is added to switch between processes.
- A small unit of time, called a time quantum or time slice, is defined (generally from 10 to 100 milliseconds)



- The ready queue is treated as a circular queue.

The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

NESO ACADEMY

Implementation of Round Robin scheduling:

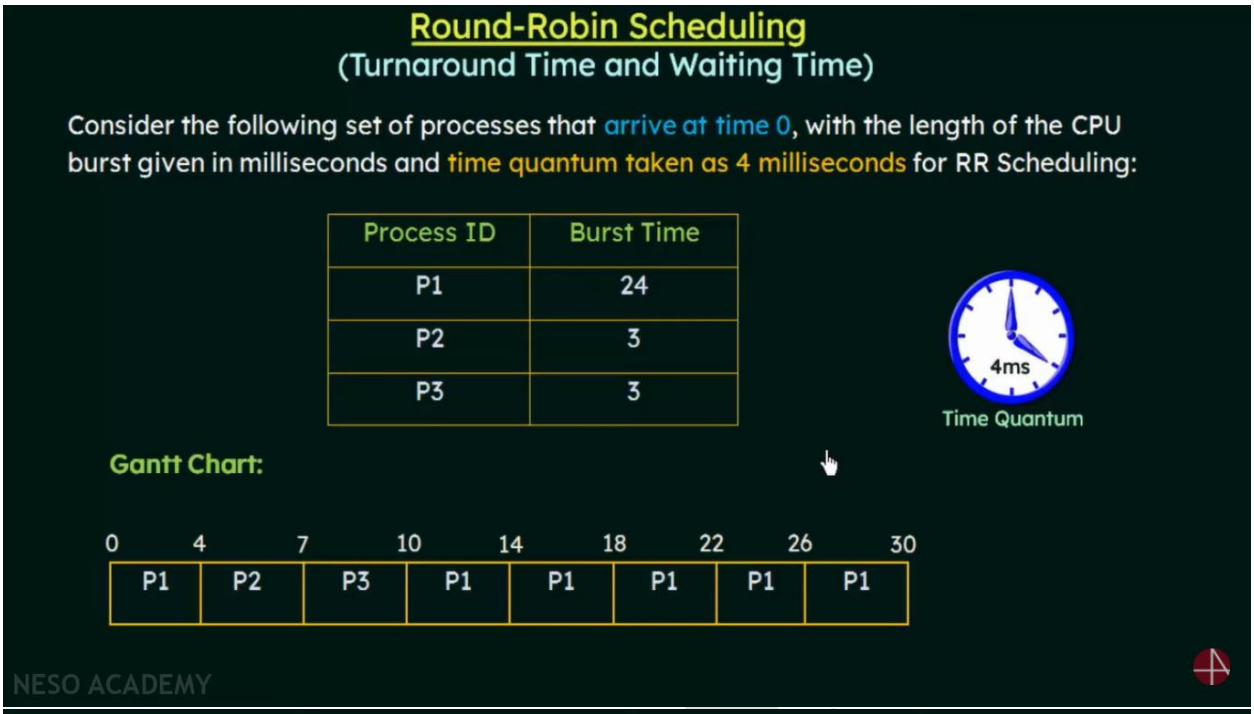
- We keep the ready queue as a FIFO queue of processes.
- New processes are added to the tail of the ready queue.
- The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

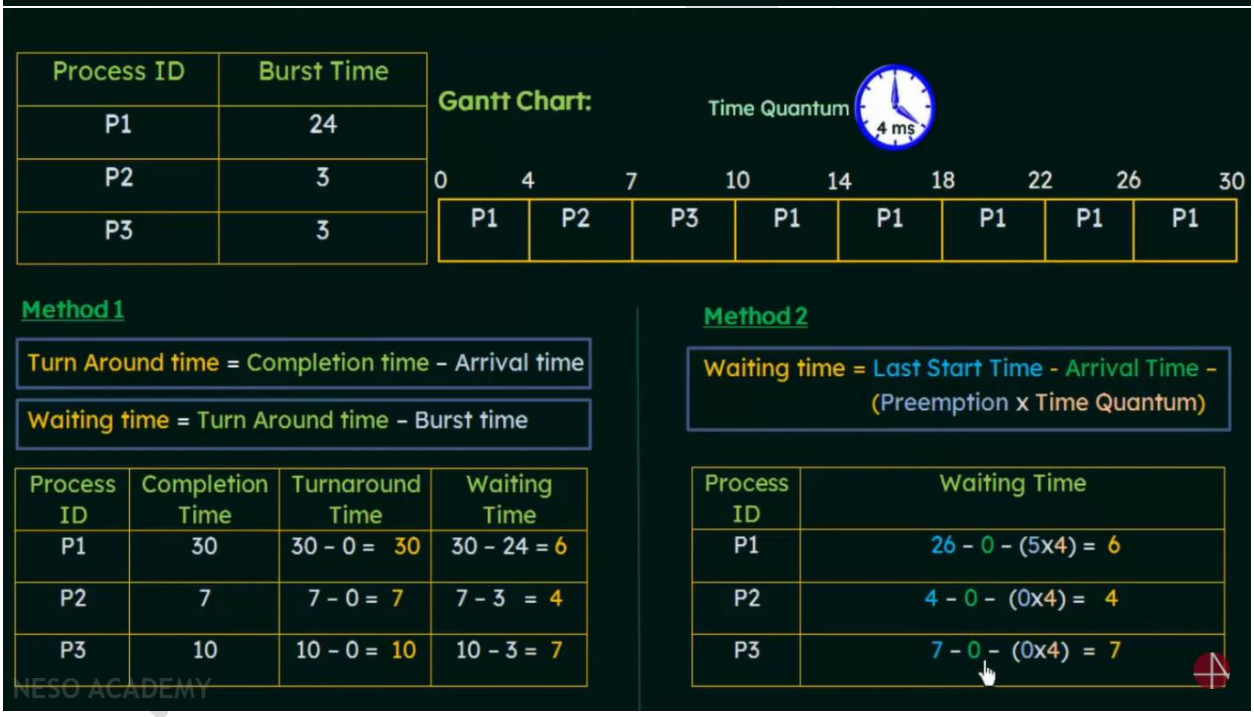
One of two things will then happen

- The process may have a CPU burst of less than 1 time quantum
  - The process itself will release the CPU voluntarily
  - The CPU scheduler will then proceed to the next process in the ready queue
- The CPU burst of the currently running process is longer than 1 time quantum, the timer will go off and will cause an interrupt to the O.S.
  - A context switch will be executed, and the process will be put at the tail of the ready queue
  - The CPU scheduler will then select the next process in the ready queue

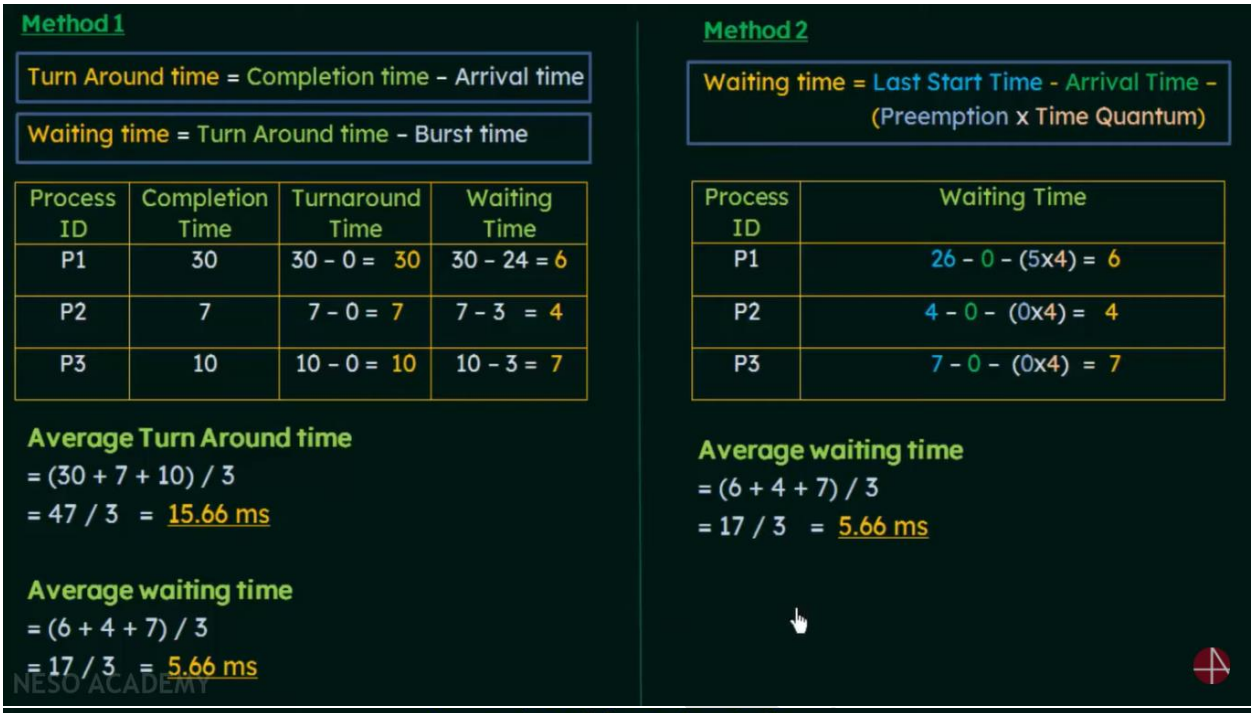
NESO ACADEMY

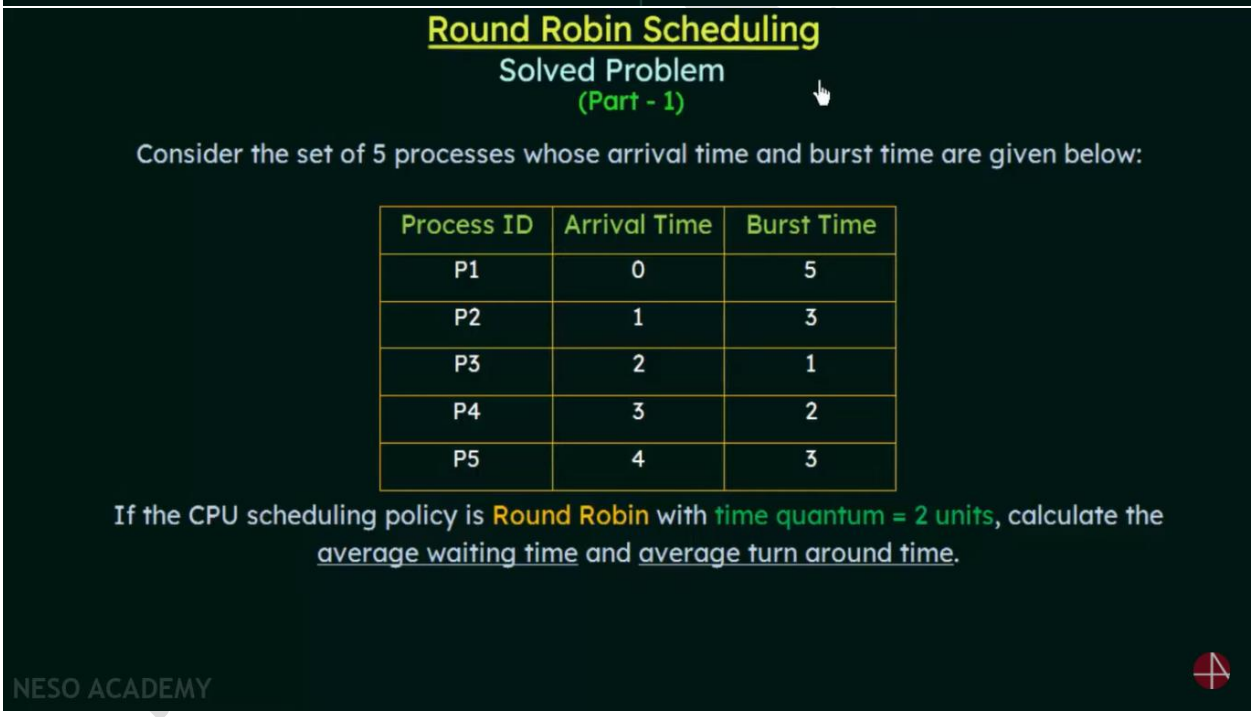
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



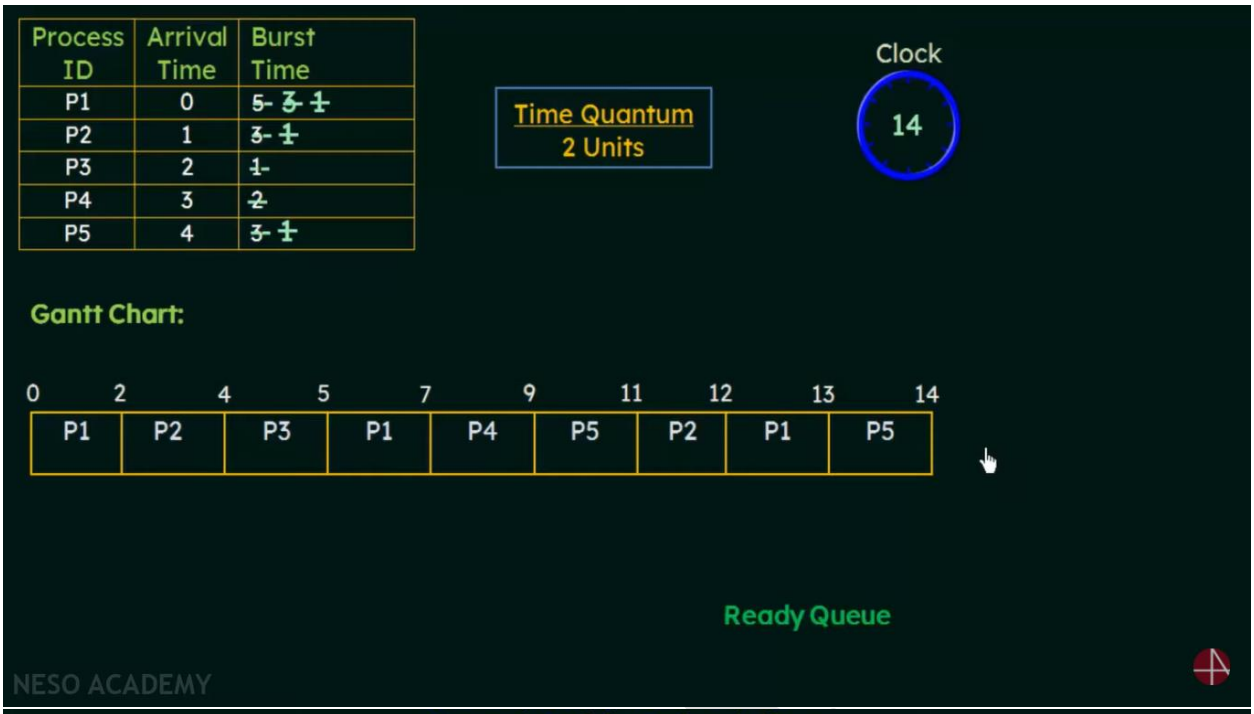


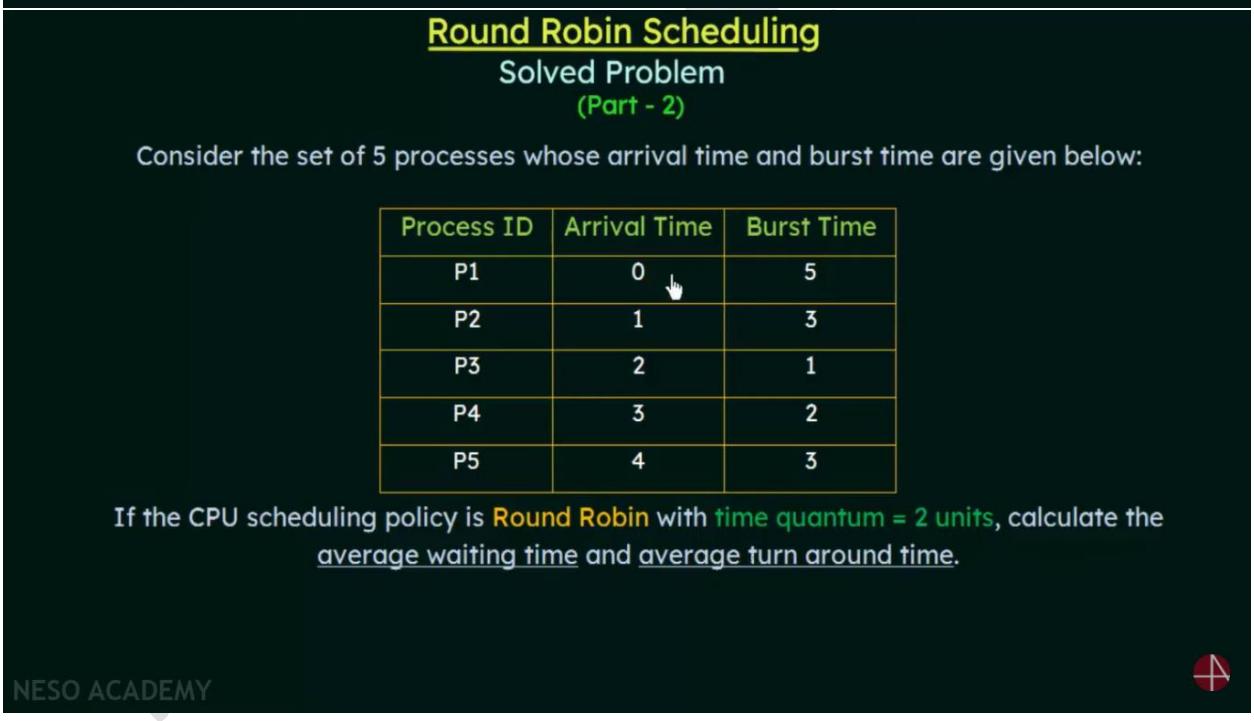
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





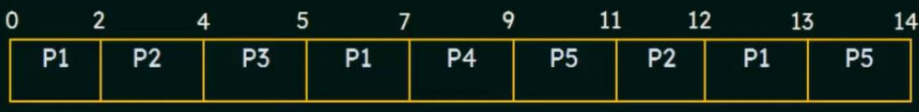
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Gantt Chart:



Turn Around time = Completion time – Arrival time

Waiting time = Turn Around time – Burst time

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

Turn Around time = Completion time – Arrival time

Waiting time = Turn Around time – Burst time

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

Average Turn Around time

= (13 + 11 + 3 + 6 + 10) / 5

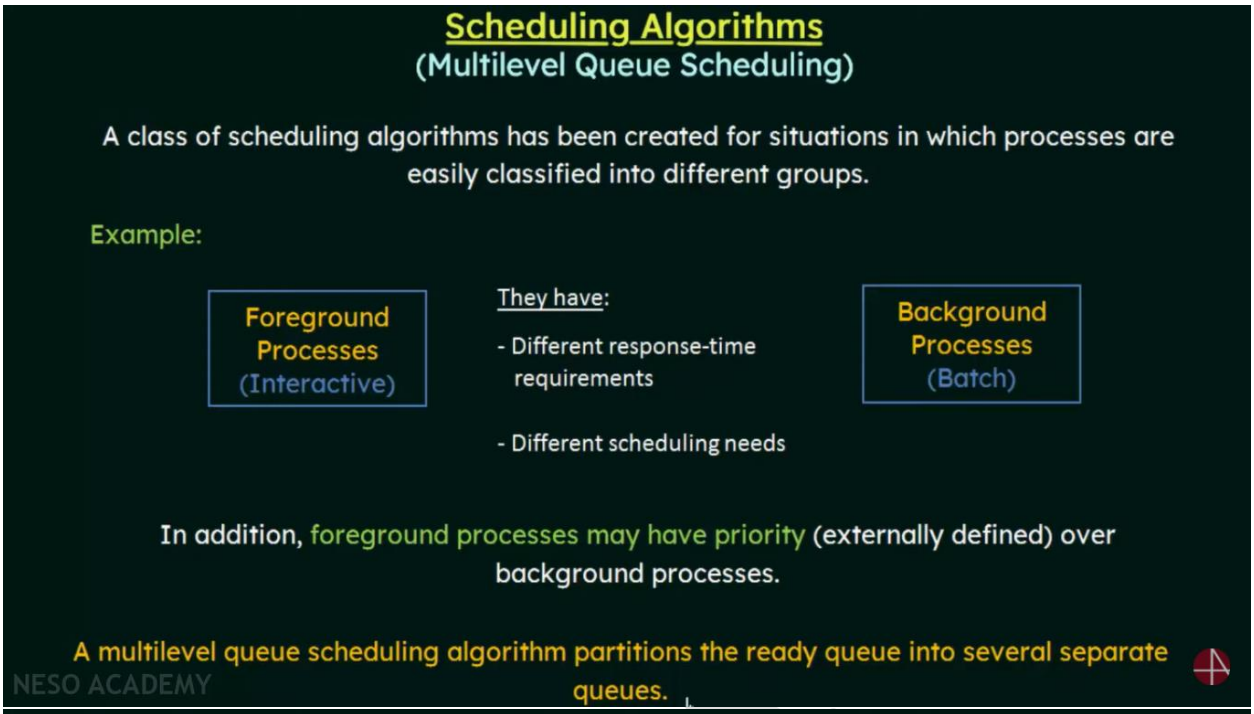
= 43 / 5 = **8.6 units**

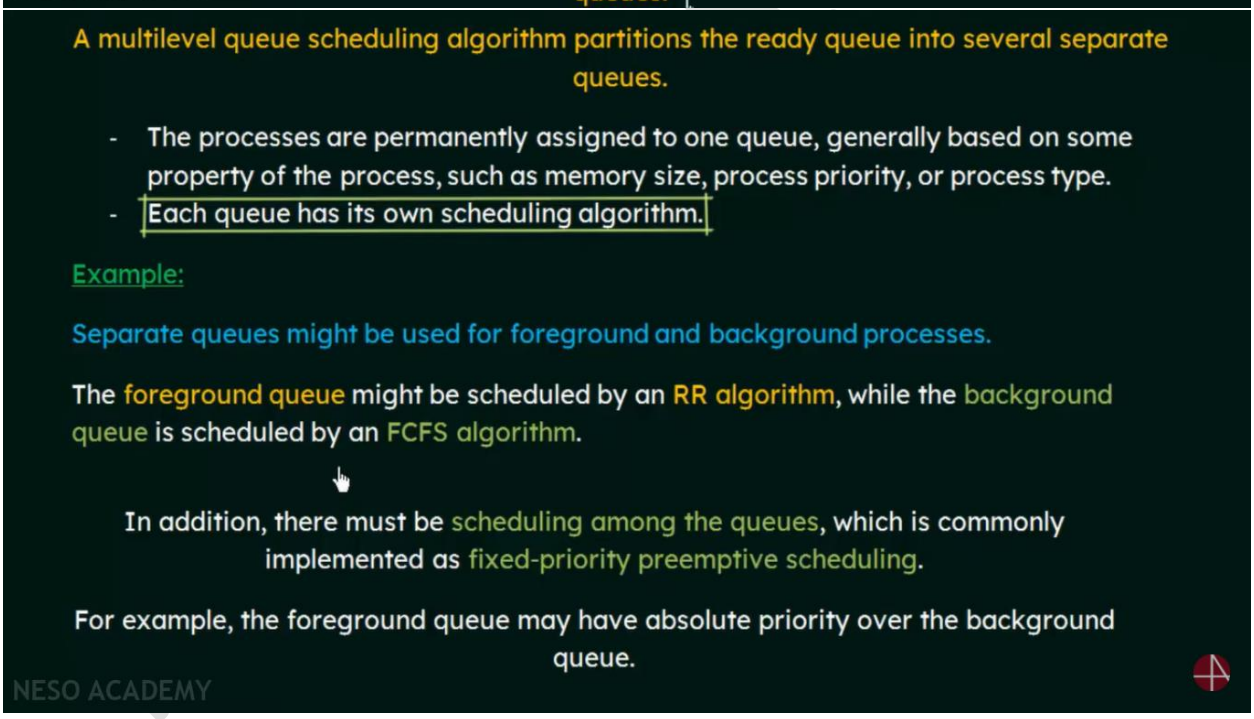
Average waiting time

= (8 + 8 + 2 + 4 + 7) / 5

= 29 / 5 = **5.8 units**

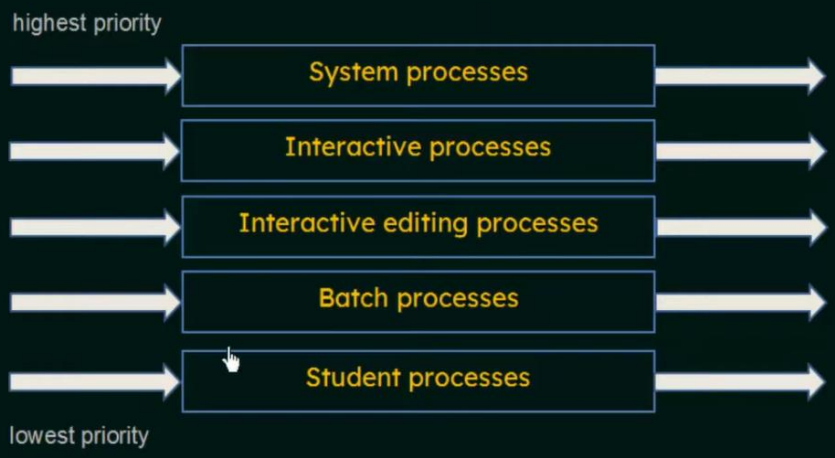
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

An example of a multilevel queue scheduling algorithm with five queues, listed below in order of priority:



NESO ACADEMY

Scheduling Algorithms
(Multilevel Feedback-Queue Scheduling)

The multilevel feedback-queue scheduling algorithm allows a process to move between queues.

- The idea is to separate processes according to the characteristics of their CPU bursts.
- If a process uses too much CPU time, it will be moved to a lower-priority queue.
- This scheme leaves I/O-bound and interactive processes in the higher-priority queues.
- In addition, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue.

This form of aging prevents starvation.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

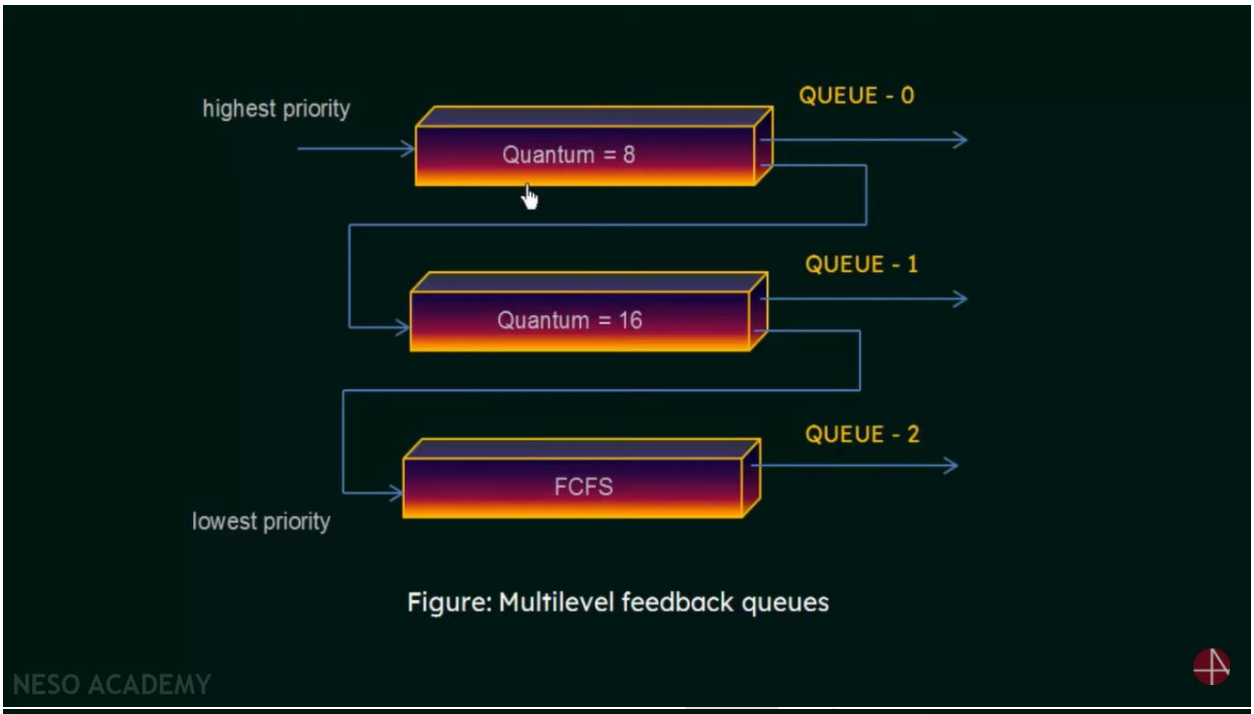


Figure: Multilevel feedback queues

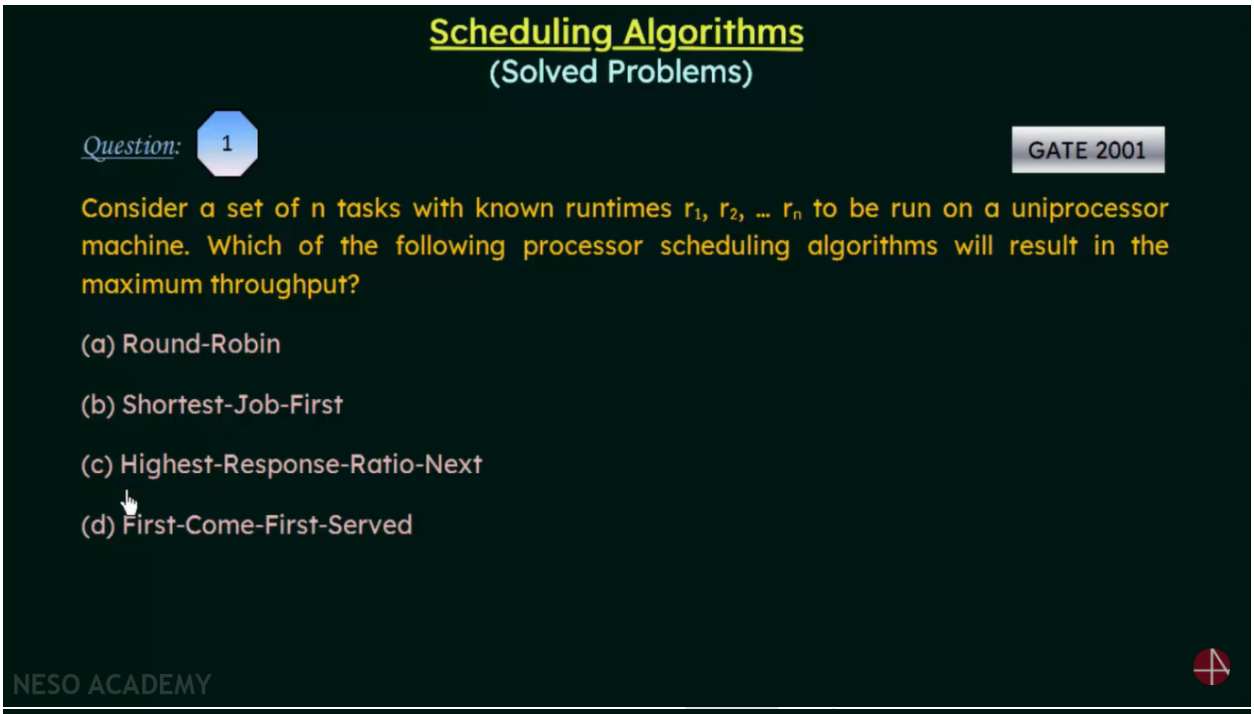
NESO ACADEMY

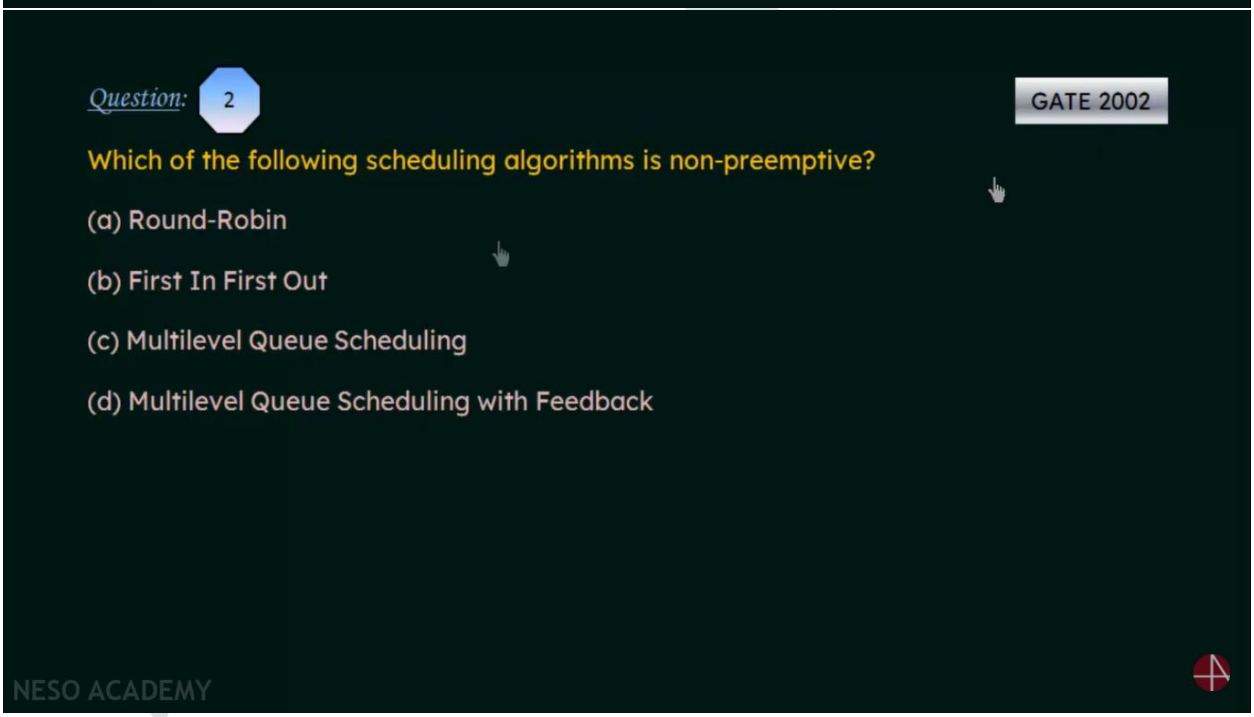
In general, a multilevel feedback-queue scheduler is defined by the following parameters:

- The number of queues
- The scheduling algorithm for each queue
- The method used to determine when to upgrade a process to a higher priority queue
- The method used to determine when to demote a process to a lower priority queue
- The method used to determine which queue a process will enter when that process needs service

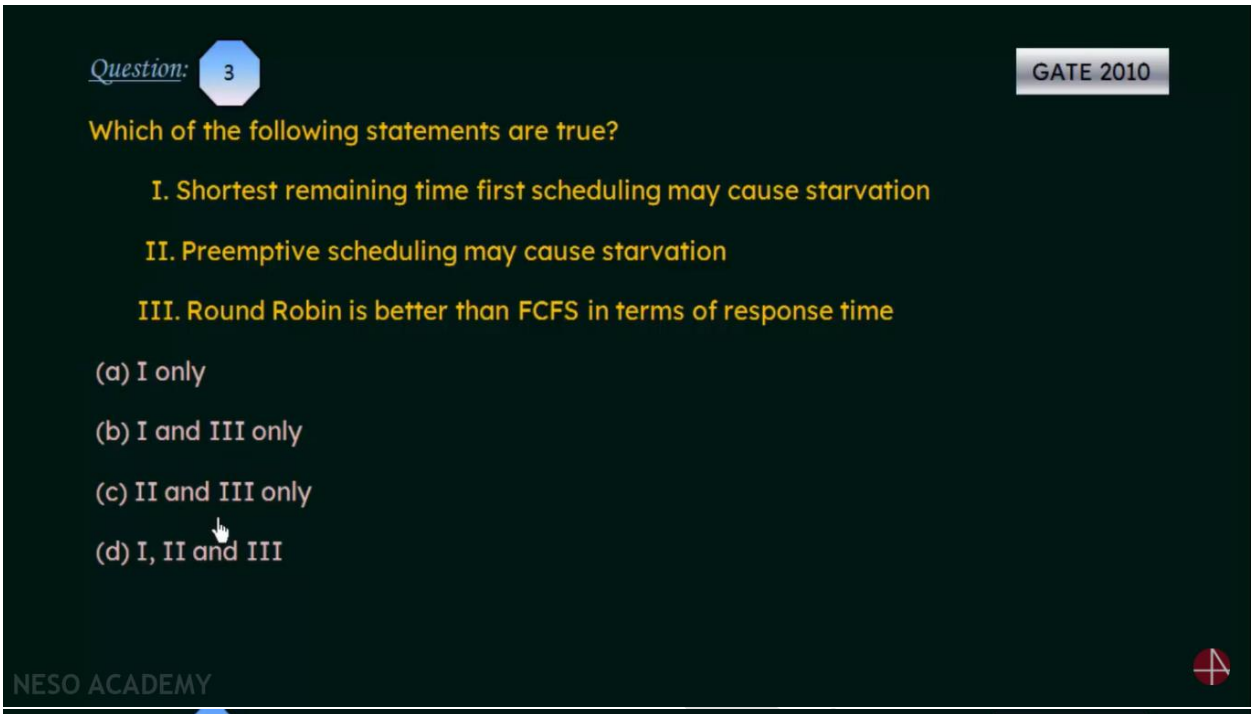
NESO ACADEMY

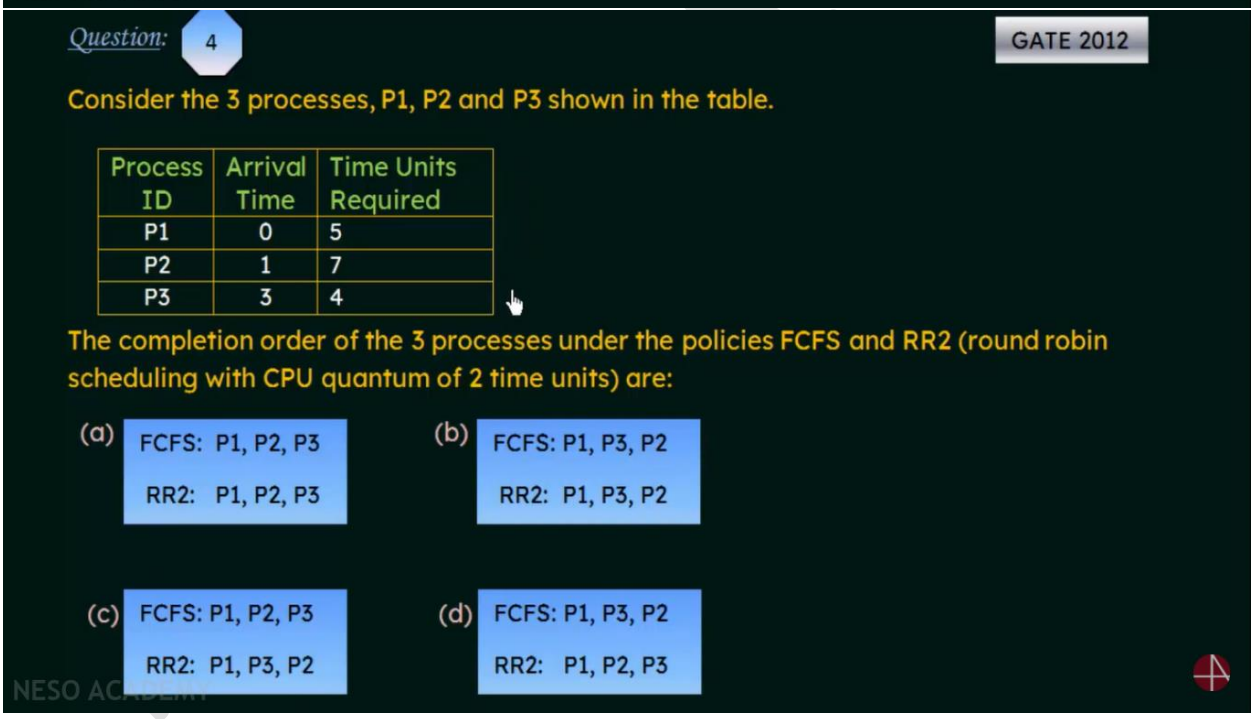
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Question: 4

GATE 2012

Consider the 3 processes, P1, P2 and P3 shown in the table.

Process ID	Arrival Time	Time Units Required
P1	0	~~5~~-~~3~~- 1
P2	1	~~7~~-5-3
P3	3	~~4~~ 2

P3 P1

The completion order of the 3 processes under the policies FCFS and RR2 (round robin scheduling with CPU quantum of 2 time units) are:

(a) FCFS: P1, P2, P3
RR2: P1, P2, P3

(b) FCFS: P1, P3, P2
RR2: P1, P3, P2

(c) FCFS: P1, P2, P3
RR2: P1, P3, P2

(d) FCFS: P1, P3, P2
RR2: P1, P2, P3

Question: 5

GATE 2013

A scheduling algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (the lowest priority). The scheduler re-evaluates the process priorities every T time units and decides the next process to schedule. Which one of the following is TRUE if the processes have no I/O operations and all arrive at time zero?

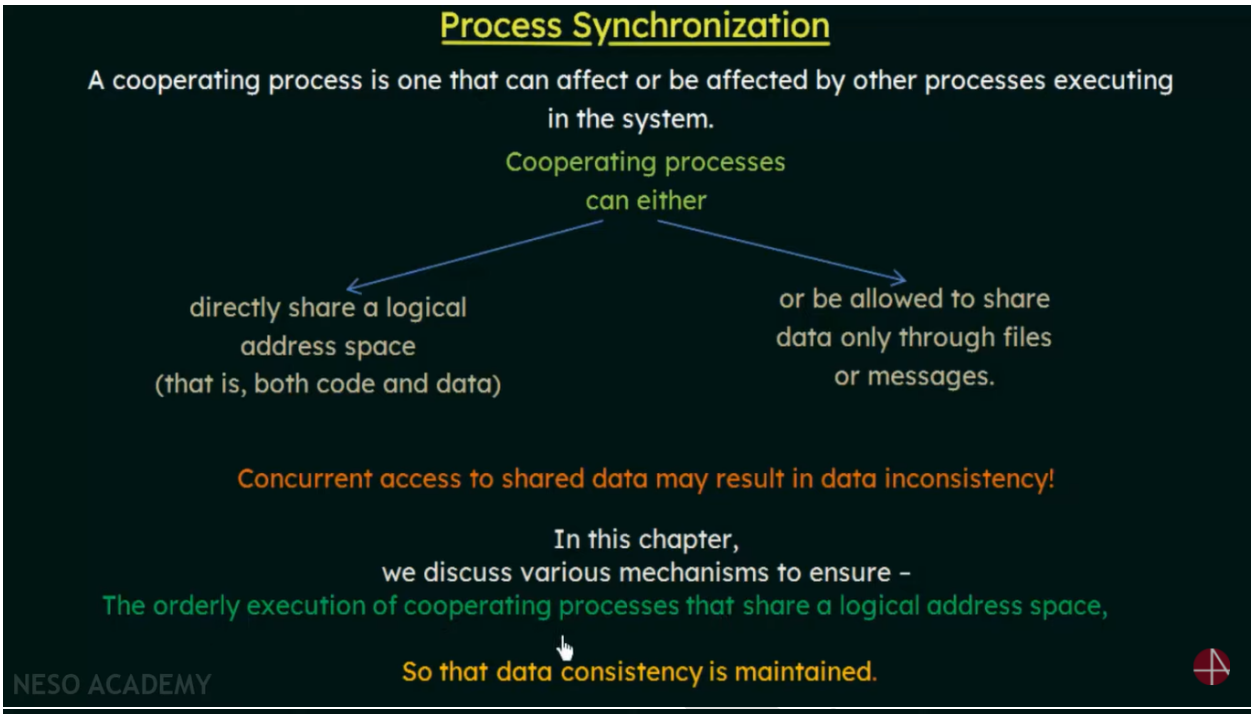
(a) This algorithm is equivalent to the first-come-first-serve algorithm.

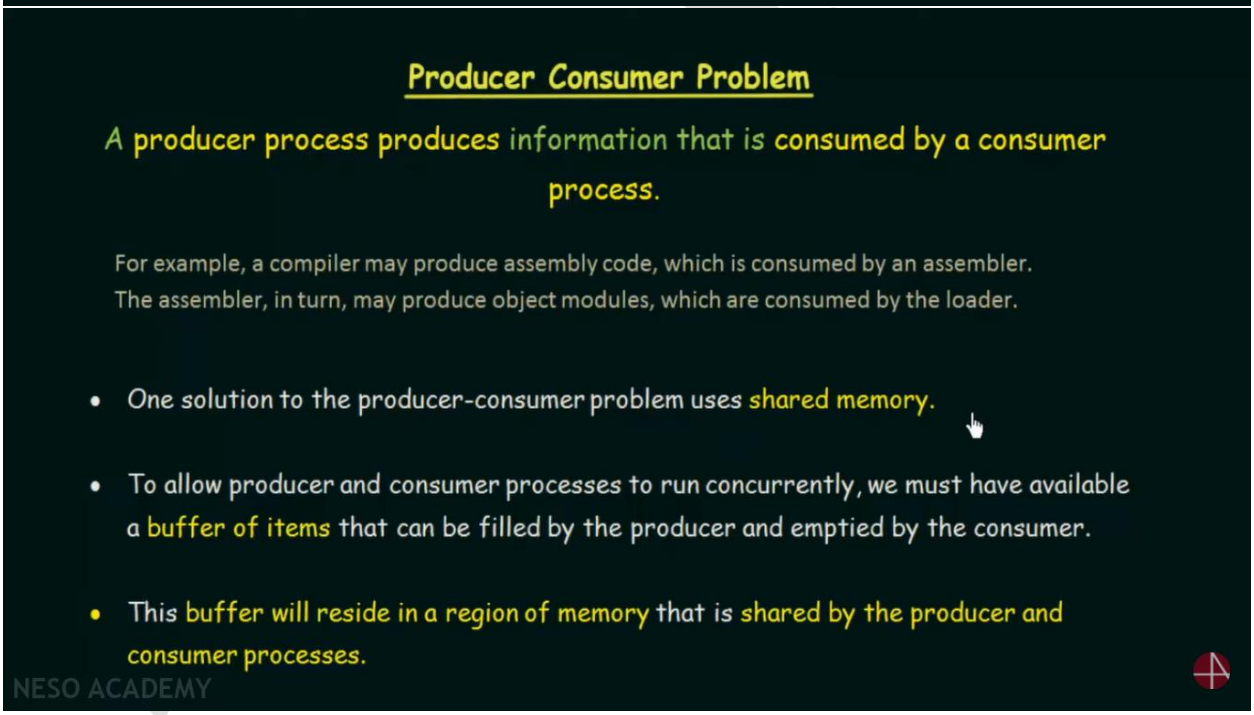
(b) This algorithm is equivalent to the round-robin algorithm.

(c) This algorithm is equivalent to the shortest-job-first algorithm.

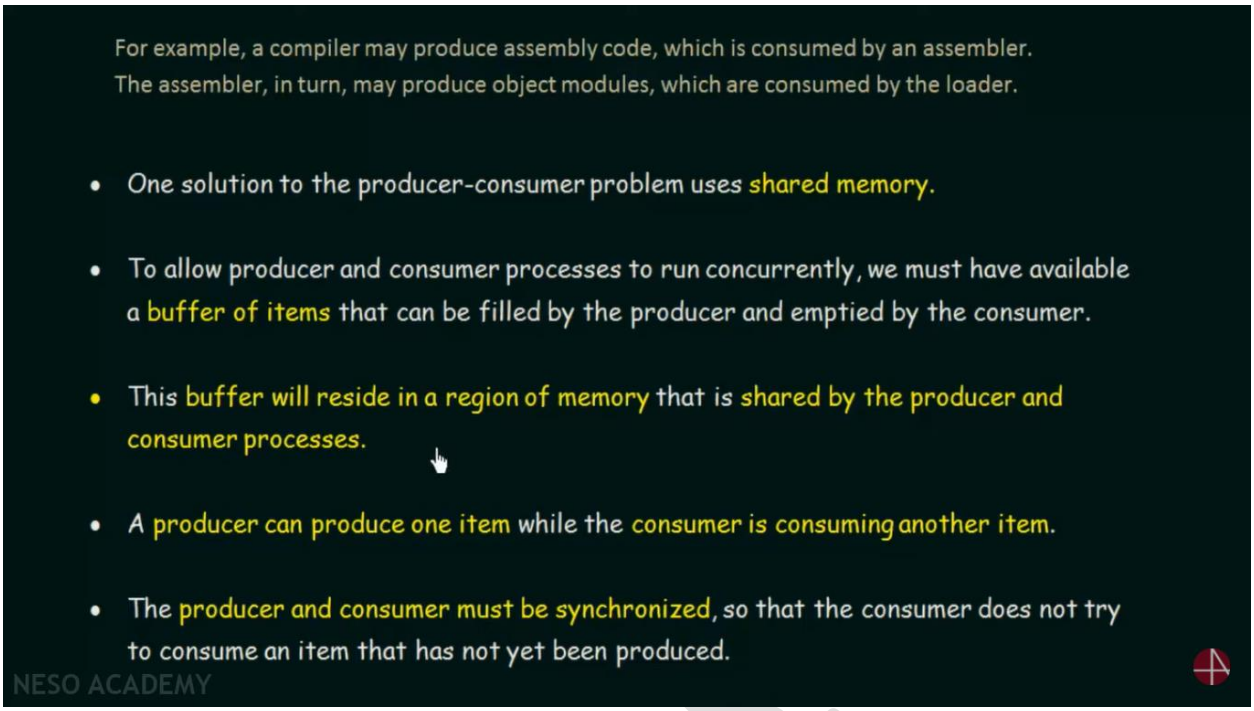
(d) This algorithm is equivalent to the shortest-remaining-time-first algorithm.

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

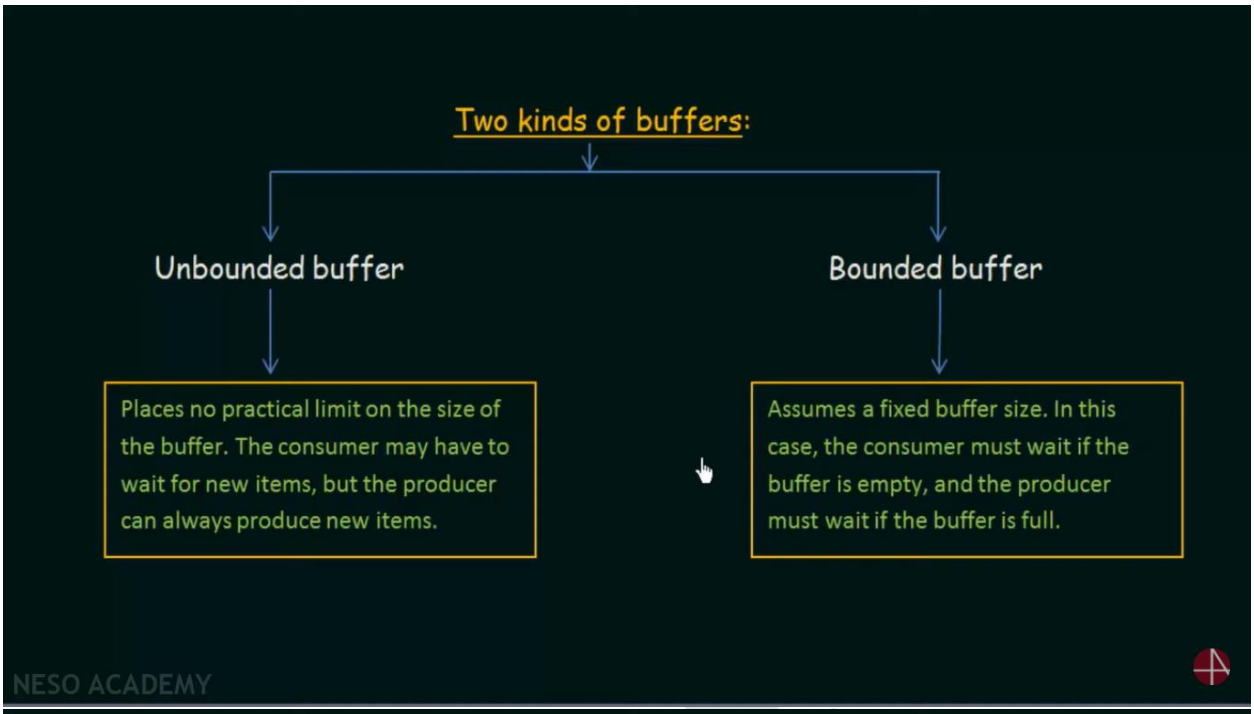


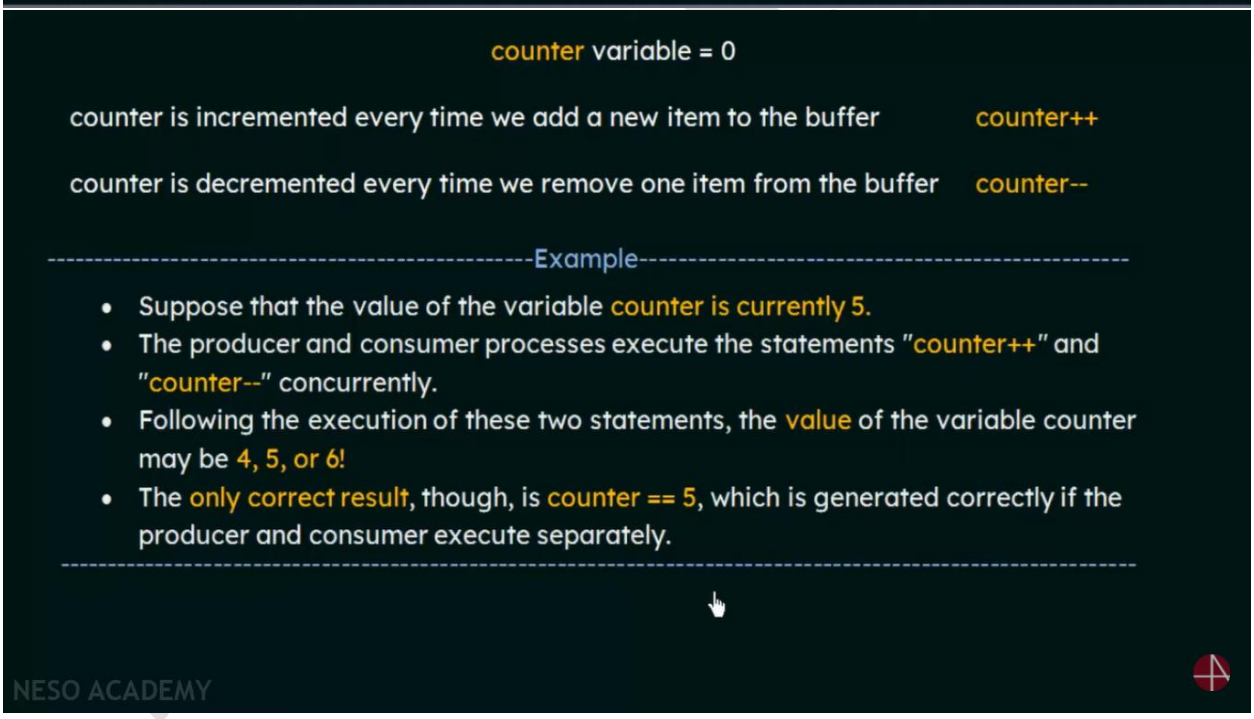


NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

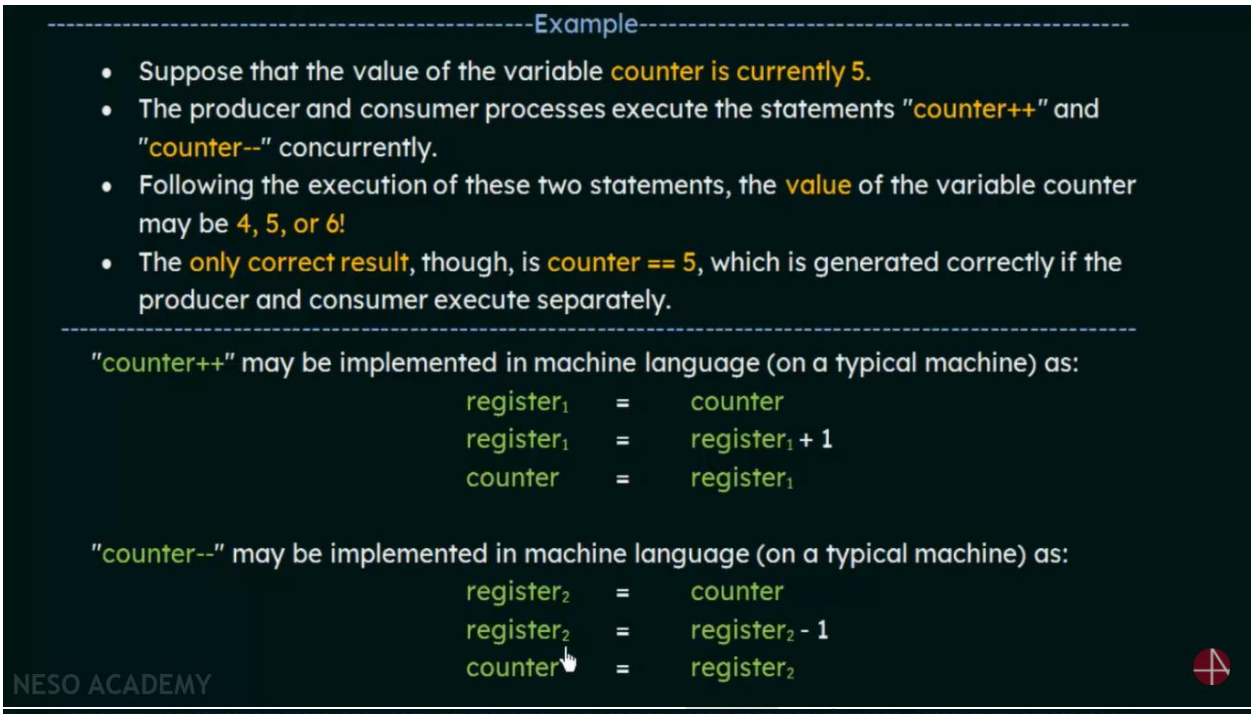


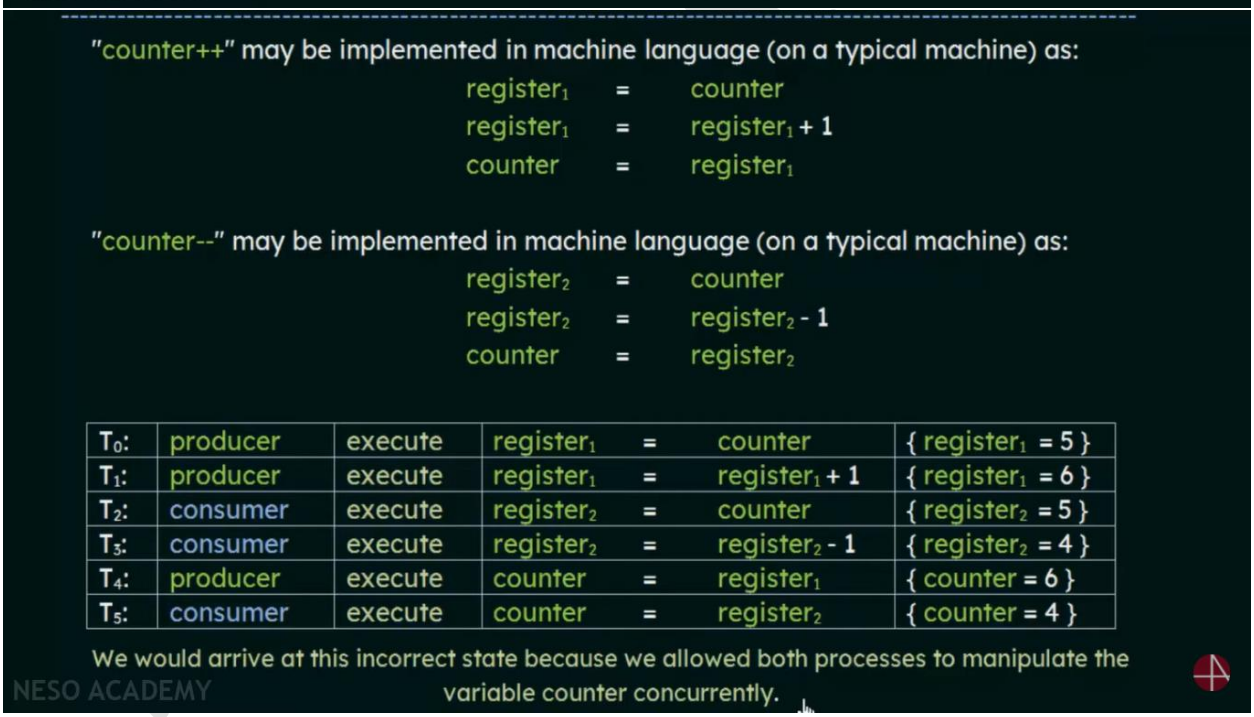
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

T_0:	producer	execute	$register_1$	=	counter	{ $register_1$ = 5 }
T_1:	producer	execute	$register_1$	=	$register_1$ + 1	{ $register_1$ = 6 }
T_2:	consumer	execute	$register_2$	=	counter	{ $register_2$ = 5 }
T_3:	consumer	execute	$register_2$	=	$register_2$ - 1	{ $register_2$ = 4 }
T_4:	producer	execute	counter	=	$register_1$	{ counter = 6 }
T_5:	consumer	execute	counter	=	$register_2$	{ counter = 4 }

We would arrive at this incorrect state because we allowed both processes to manipulate the variable counter concurrently.

A situation like this, where several processes access and manipulate the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called a race condition.

Clearly, we want the resulting changes not to interfere with one another. Hence we need process synchronization.

NESO ACADEMY

The Critical-Section Problem

Consider a system consisting of n processes { P_0, P_1, ..., P_n }.

Each process has a segment of code, called a

critical section

in which the process may be changing common variables, updating a table, writing a file, and so on.

When one process is executing in its critical section, no other process is to be allowed to execute in its critical section.

That is, no two processes are executing in their critical sections at the same time.

The critical-section problem is to design a protocol that the processes can use to cooperate.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

- Each process must request permission to enter its critical section.
- The section of code implementing this request is the entry section.
- The critical section may be followed by an exit section.
- The remaining code is the remainder section.

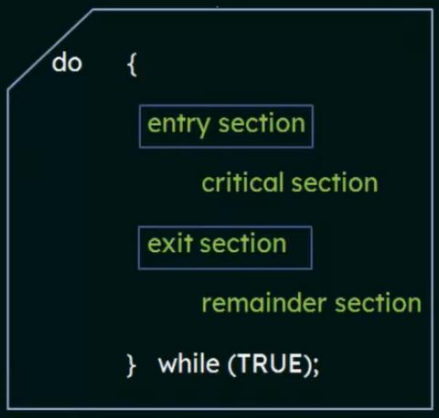


Figure: General structure of a typical process.

A solution to the critical-section problem must satisfy the following three requirements:

1. Mutual exclusion:

If process P_i is executing in its critical section, then no other processes can be executing in their critical sections.

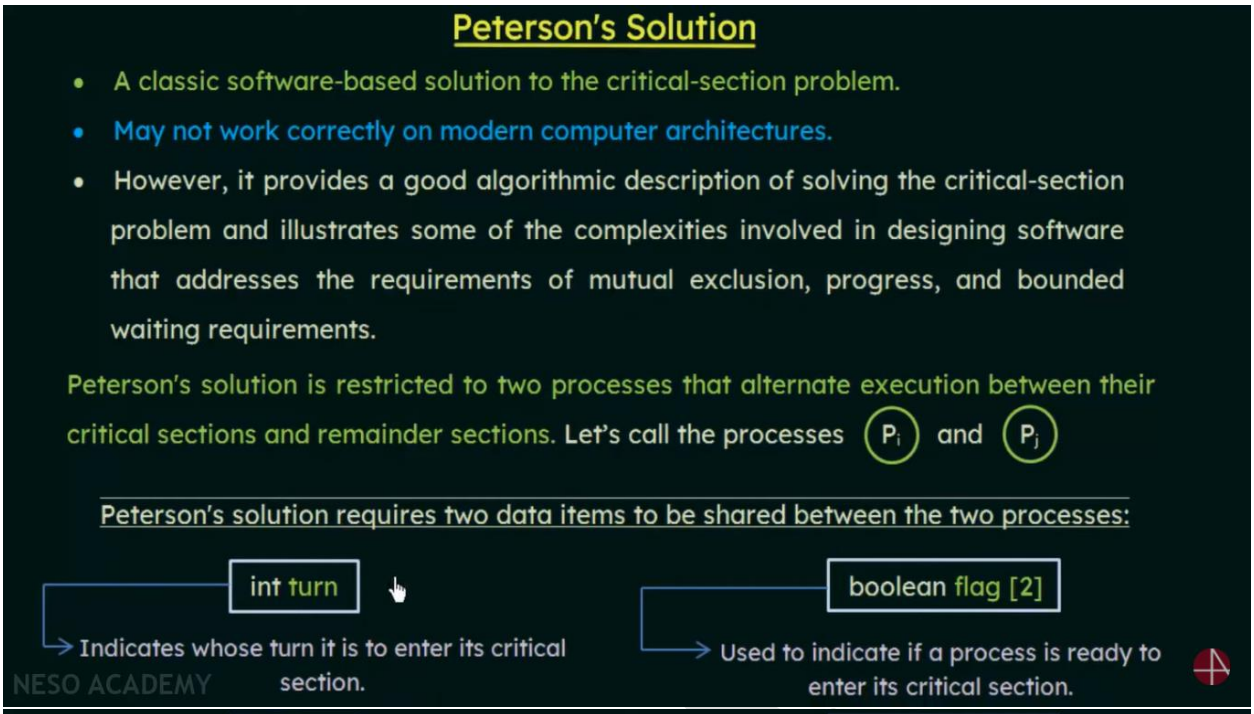
2. Progress:

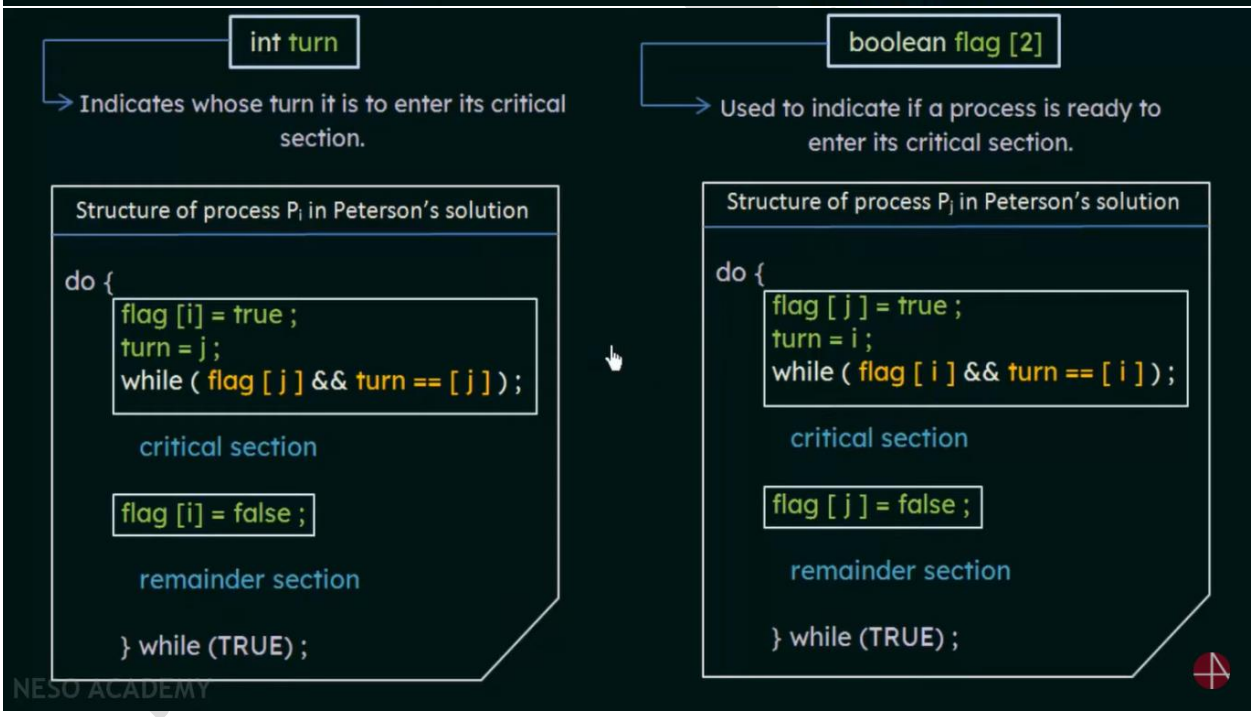
If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in the decision on which will enter its critical section next, and this selection cannot be postponed indefinitely.

3. Bounded waiting:

There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

A solution to the critical-section problem must satisfy the following three requirements:

1. Mutual exclusion:

If process P_i is executing in its critical section, then no other processes can be executing in their critical sections.

2. Progress:

If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in the decision on which will enter its critical section next, and this selection cannot be postponed indefinitely.

3. Bounded waiting:

There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

NESO ACADEMY

Test and Set Lock

- A hardware solution to the synchronization problem.
- There is a shared lock variable which can take either of the two values, 0 or 1.
- Before entering into the critical section, a process inquires about the lock.
- If it is locked, it keeps on waiting till it becomes free.
- If it is not locked, it takes the lock and executes the critical section.

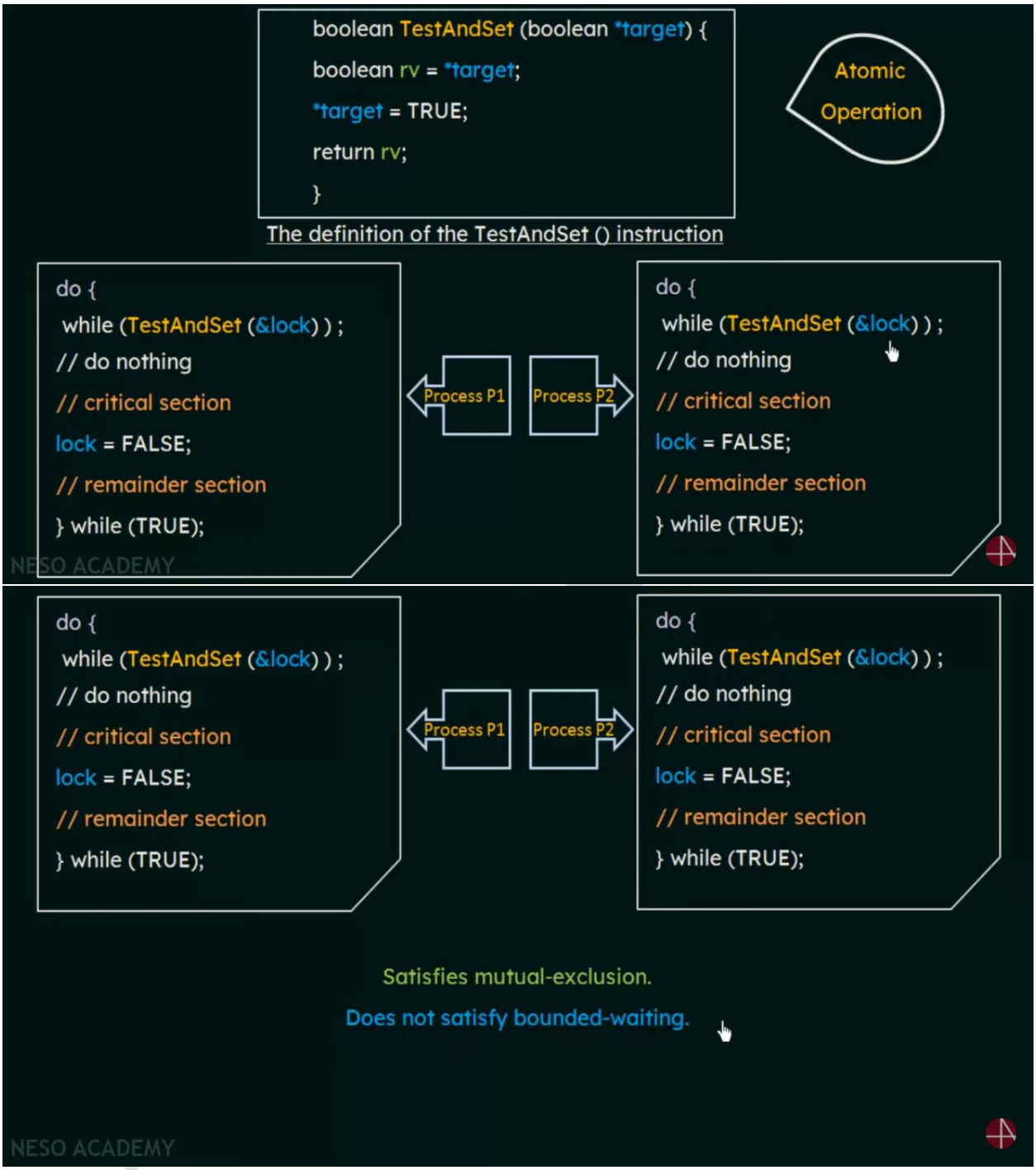
```
boolean TestAndSet (boolean *target) {
boolean rv = *target;
*target = TRUE;
return rv;
}
```

Atomic Operation

The definition of the TestAndSet () instruction

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Semaphores

- Semaphore proposed by Edsger Dijkstra, is a technique to manage concurrent processes by using a simple integer value, which is known as a semaphore.

NESO ACADEMY

Semaphores

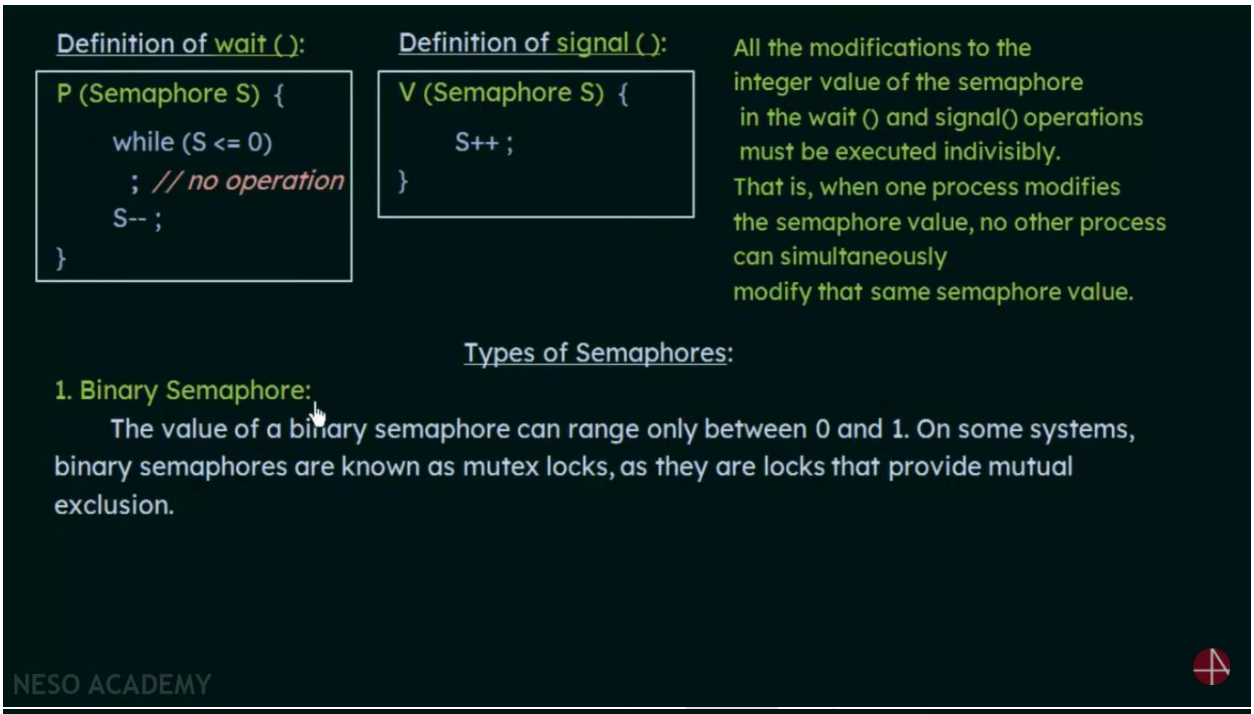
- Semaphore proposed by Edsger Dijkstra, is a technique to manage concurrent processes by using a simple integer value, which is known as a semaphore.
- Semaphore is simply a variable which is non-negative and shared between threads. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment.
- A semaphore S is an integer variable that, apart from initialization, is accessed only through two standard atomic operations: wait ( ) and signal ( ).

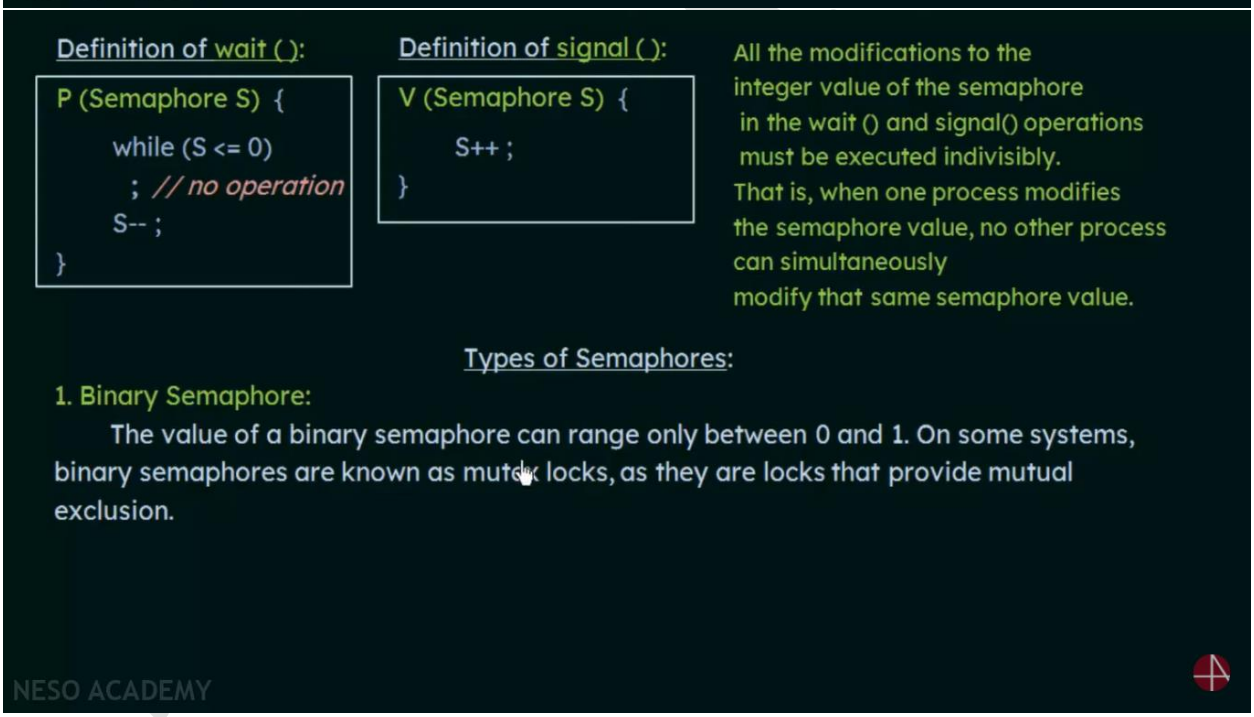
wait ( ) → P [from the Dutch word proberen, which means "to test"]

signal ( ) → V [from the Dutch word verhogen, which means " to increment"]

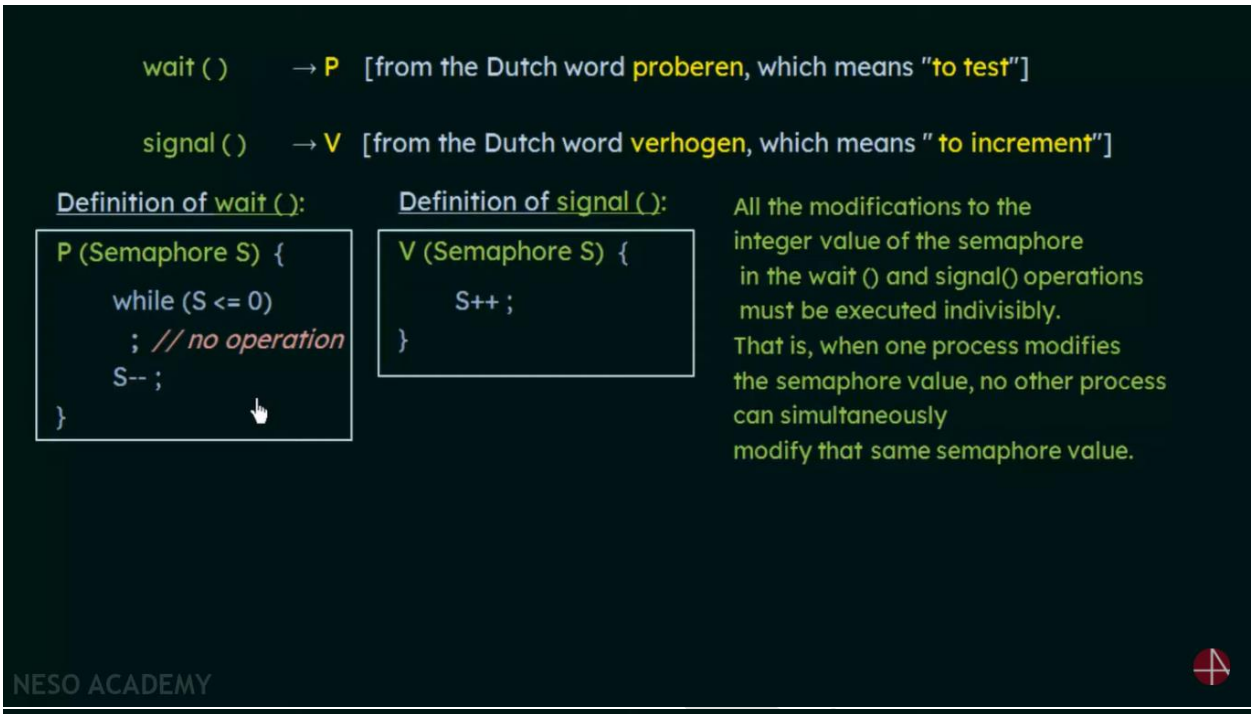
NESO ACADEMY

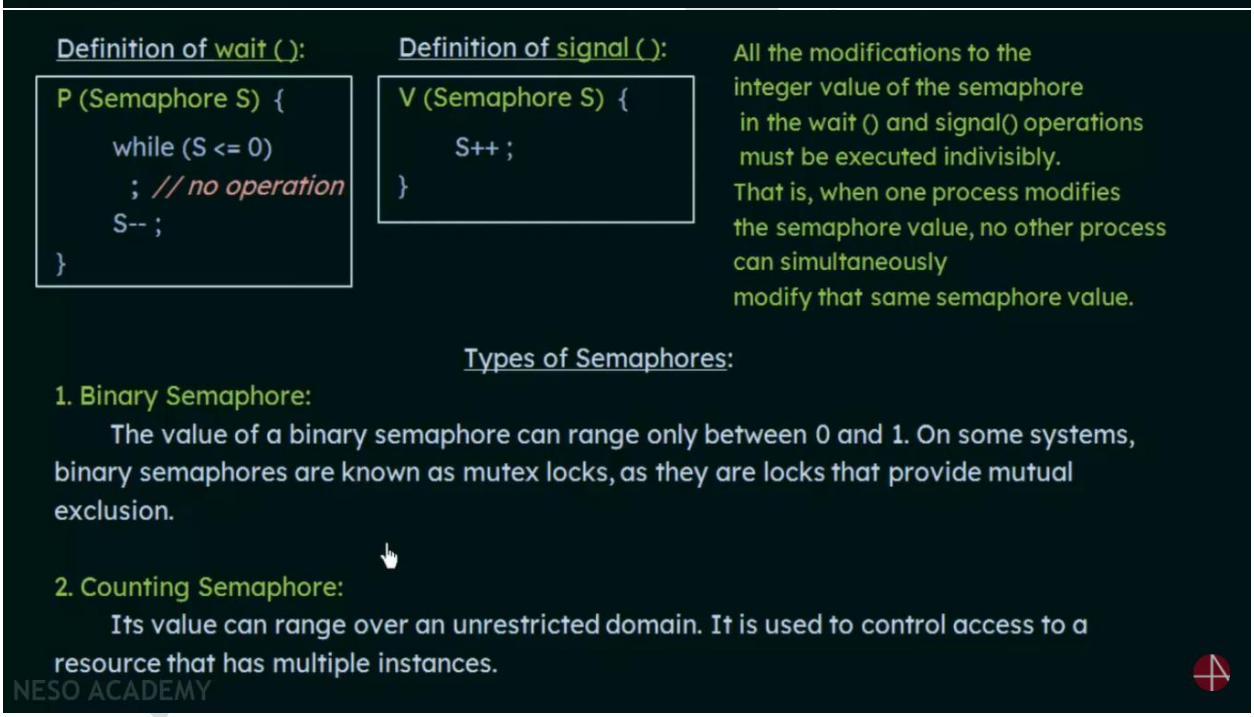
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



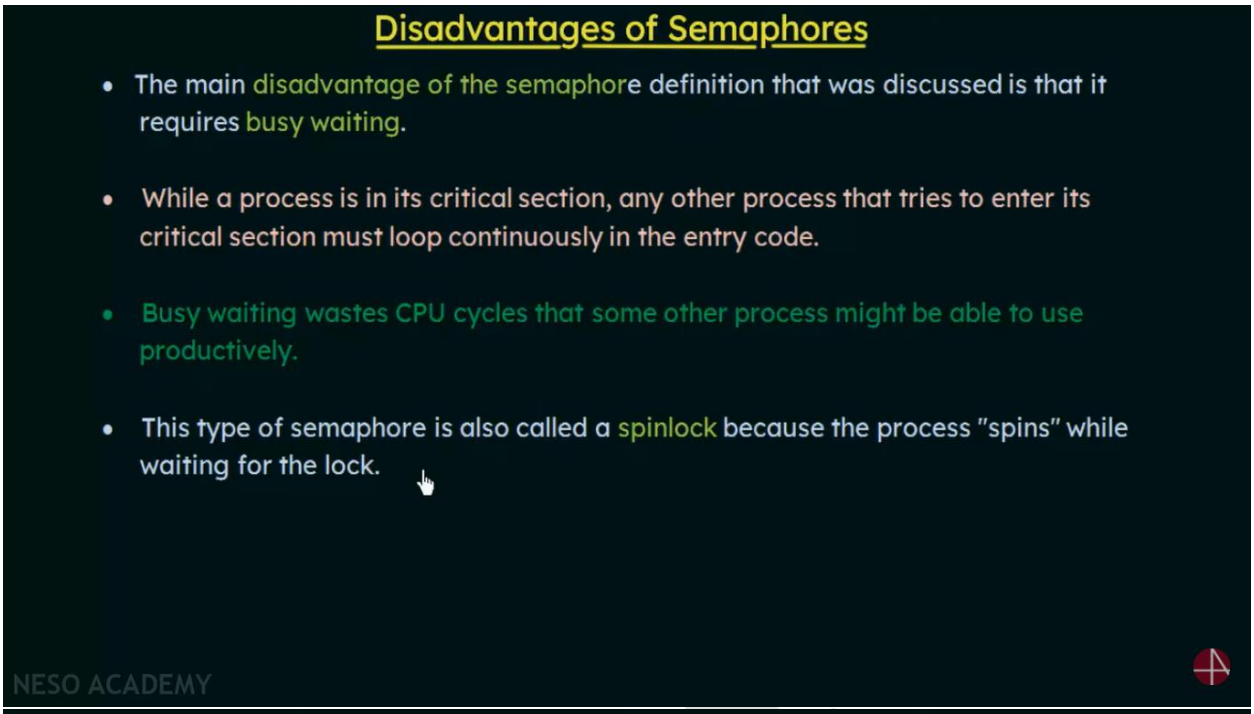


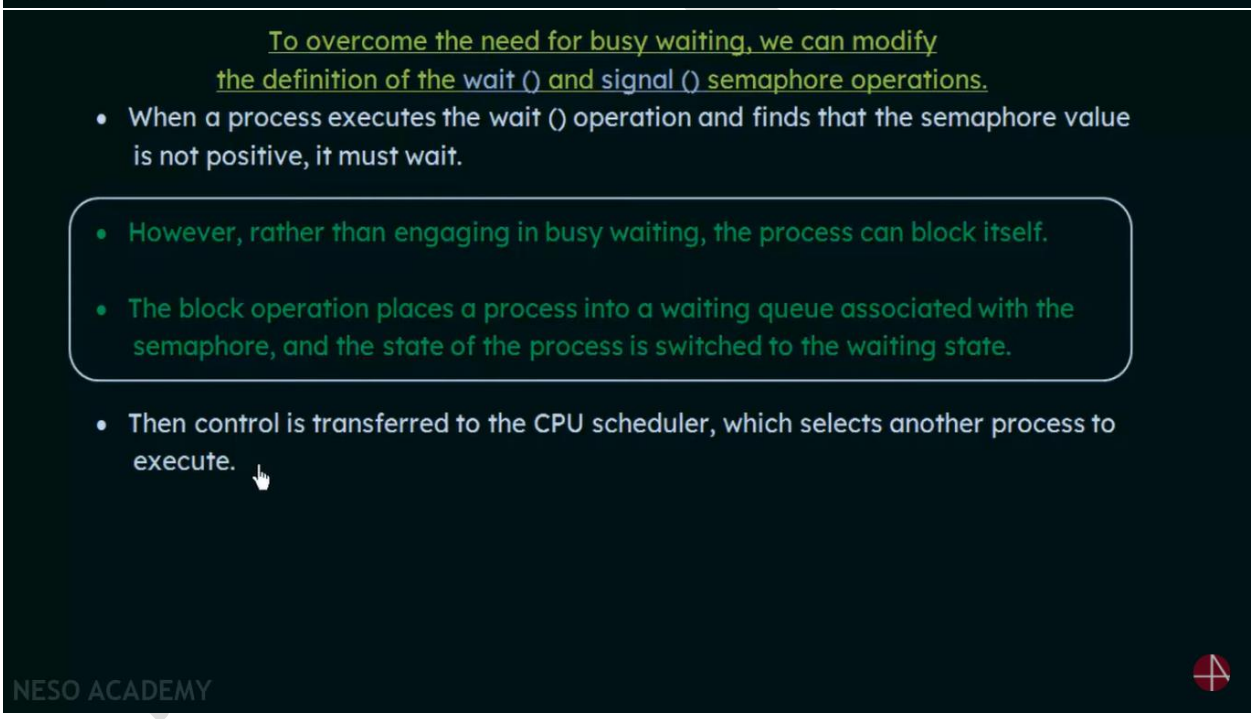
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Deadlocks and Starvation

- The implementation of a semaphore with a waiting queue may result in a situation where two or more processes are waiting indefinitely for an event that can be caused only by one of the waiting processes.
- The event in question is the execution of a signal() operation. When such a state is reached, these processes are said to be deadlocked.

P0
wait (S);
wait (Q);
.
.
.
signal (S);
signal (Q);

P1
wait (Q);
wait (S);
.
.
.
signal (Q);
signal (S);

NESO ACADEMY

Classic Problems of Synchronization

(The Bounded-Buffer Problem)

The Bounded Buffer Problem **(Producer Consumer Problem)**, is one of the classic problems of synchronization.

There is a buffer of n slots and each slot is capable of storing one unit of data.

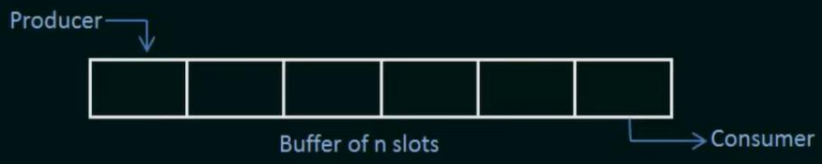
There are two processes running, namely, **Producer** and **Consumer**, which are operating on the buffer.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

There is a buffer of n slots and each slot is capable of storing one unit of data.

There are two processes running, namely, Producer and Consumer, which are operating on the buffer.



- The producer tries to insert data into an empty slot of the buffer.
- The consumer tries to remove data from a filled slot in the buffer.
- The Producer must not insert data when the buffer is full.
- The Consumer must not remove data when the buffer is empty.
- The Producer and Consumer should not insert and remove data simultaneously.

NESO ACADEMY

Solution to the Bounded Buffer Problem using Semaphores:

We will make use of three semaphores:

1. m (mutex), a binary semaphore which is used to acquire and release the lock.
2. empty, a counting semaphore whose initial value is the number of slots in the buffer, since, initially all slots are empty.
3. full, a counting semaphore whose initial value is 0.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

1. m (mutex), a binary semaphore which is used to acquire and release the lock.
2. empty, a counting semaphore whose initial value is the number of slots in the buffer, since, initially all slots are empty.
3. full, a counting semaphore whose initial value is 0.

Producer

```
do {
  wait (empty); // wait until empty>0
                and then decrement 'empty'
  wait (mutex); // acquire lock
  /* add data to buffer */
  signal (mutex); // release lock
  signal (full);  // increment 'full'
} while(TRUE)
```

Consumer

```
do {
  wait (full); // wait until full>0 and
               then decrement 'full'
  wait (mutex); // acquire lock
  /* remove data from buffer */
  signal (mutex); // release lock
  signal (empty);  // increment 'empty'
} while(TRUE)
```

NESO ACADEMY

Classic Problems of Synchronization
(The Readers-Writers Problem)

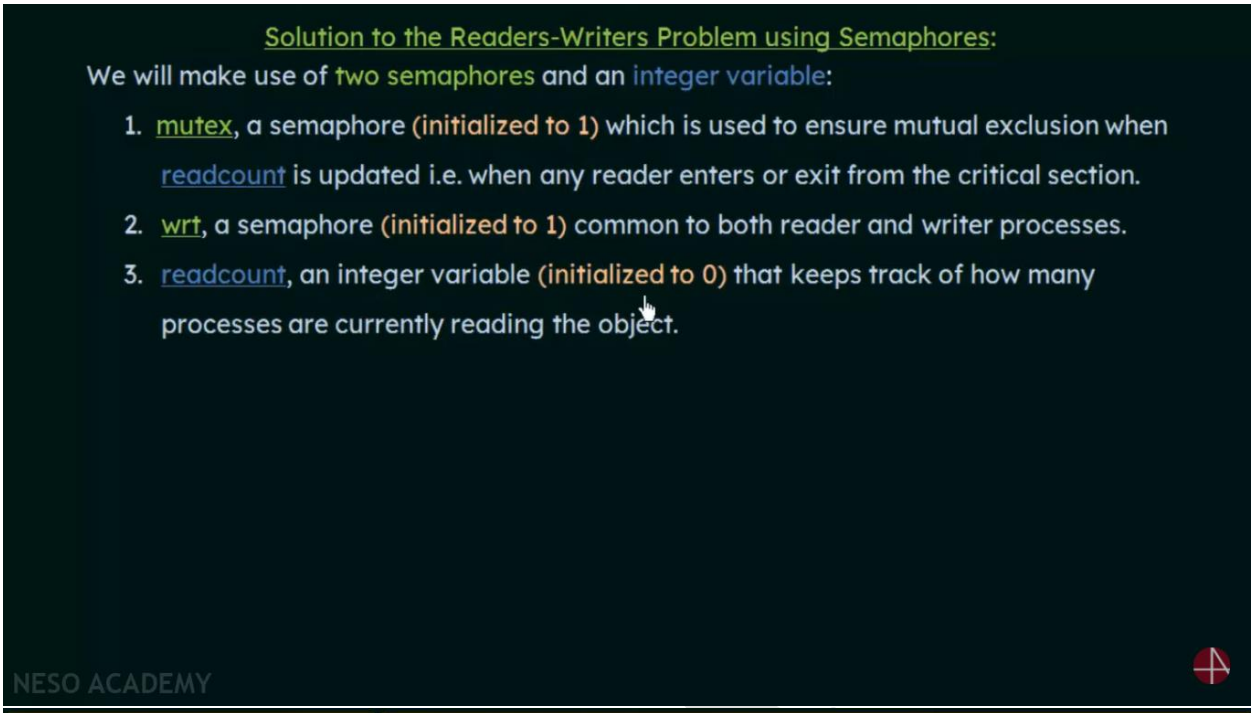
- A database is to be shared among several concurrent processes.
- Some of these processes may want only to read the database, whereas others may want to update (that is, to read and write) the database.
- We distinguish between these two types of processes by referring to the former as **Readers** and to the latter as **Writers.**
- Obviously, if two readers access the shared data simultaneously, no adverse affects will result.
- However, if a writer and some other thread (either a reader or a writer) access the database simultaneously, chaos may ensue.

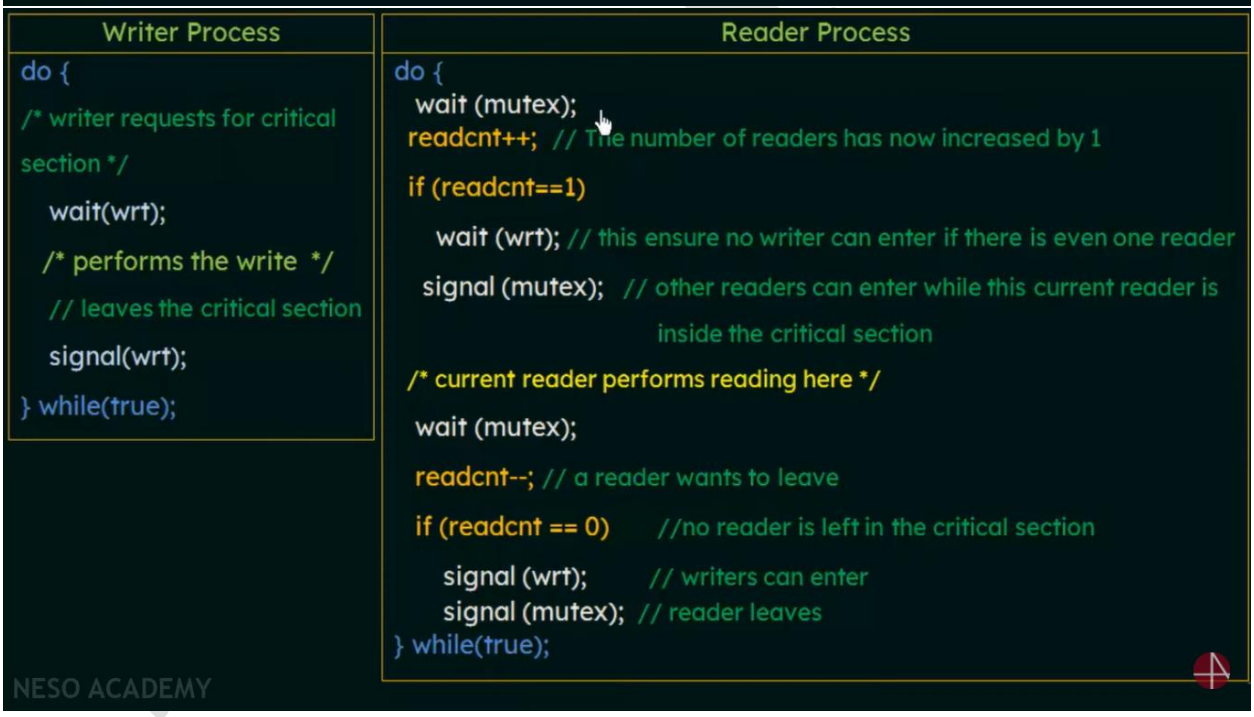
To ensure that these difficulties do not arise, we require that the writers have exclusive access to the shared database.

This synchronization problem is referred to as the readers-writers problem.

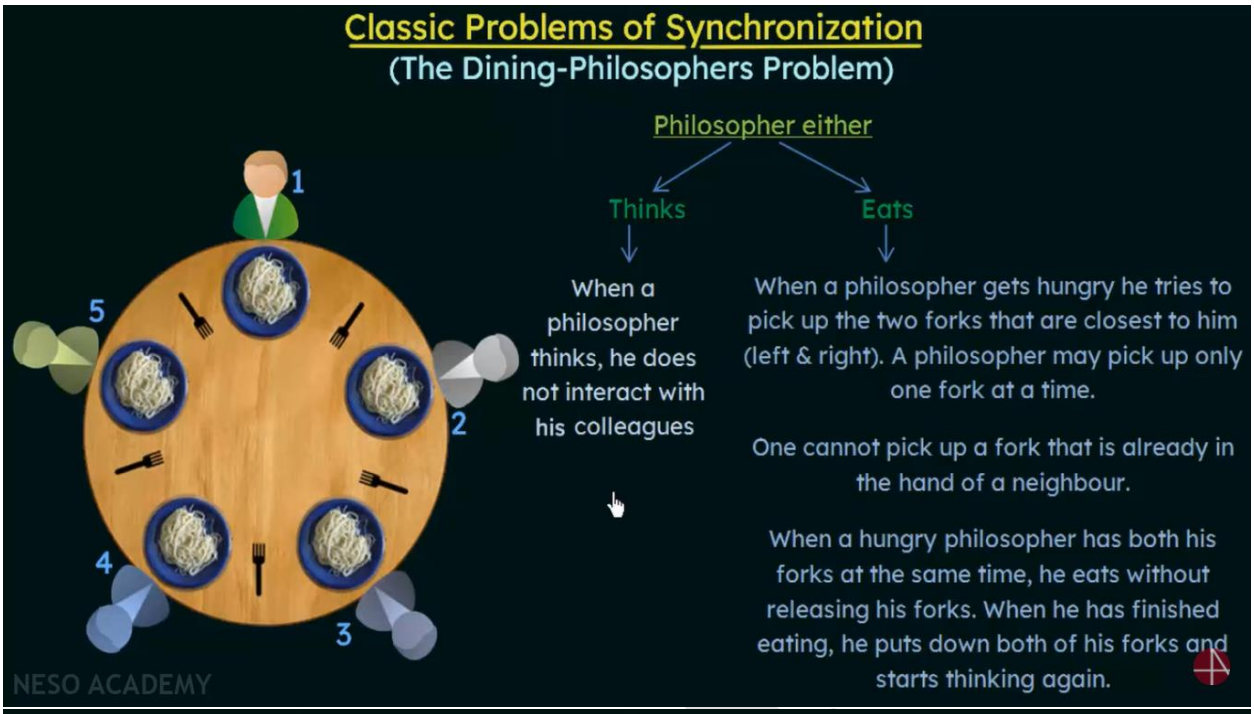
NESO ACADEMY

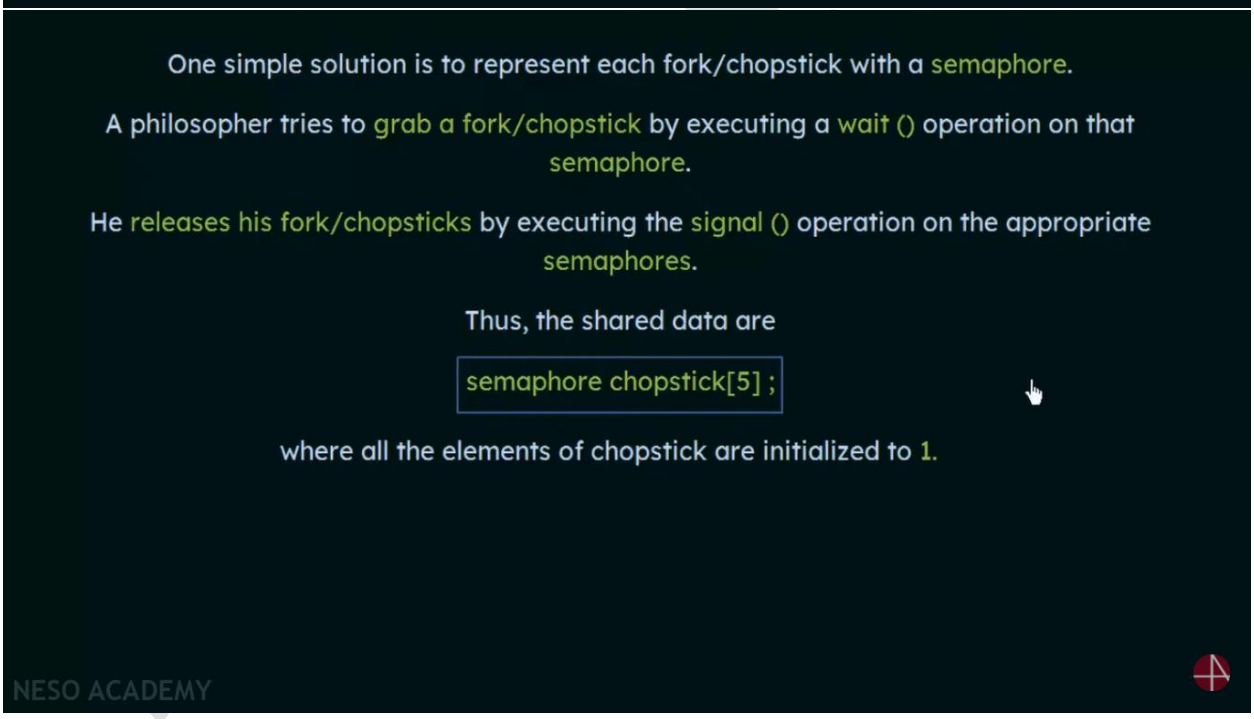
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.



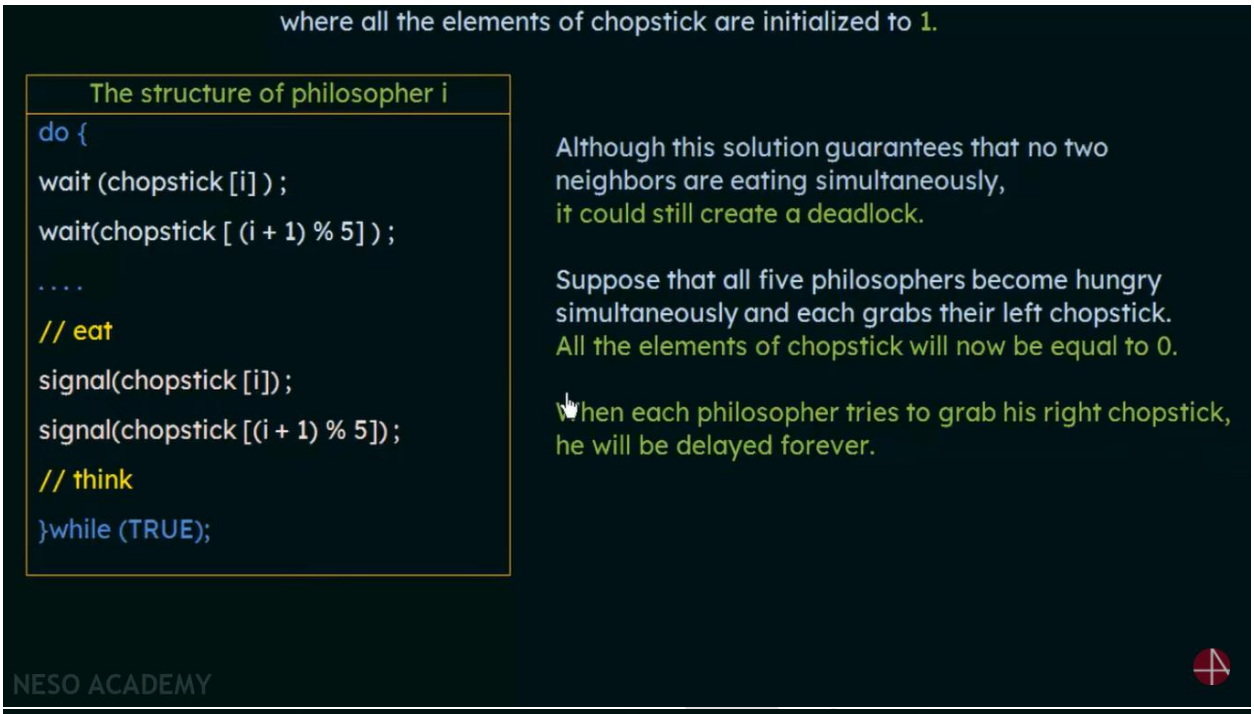


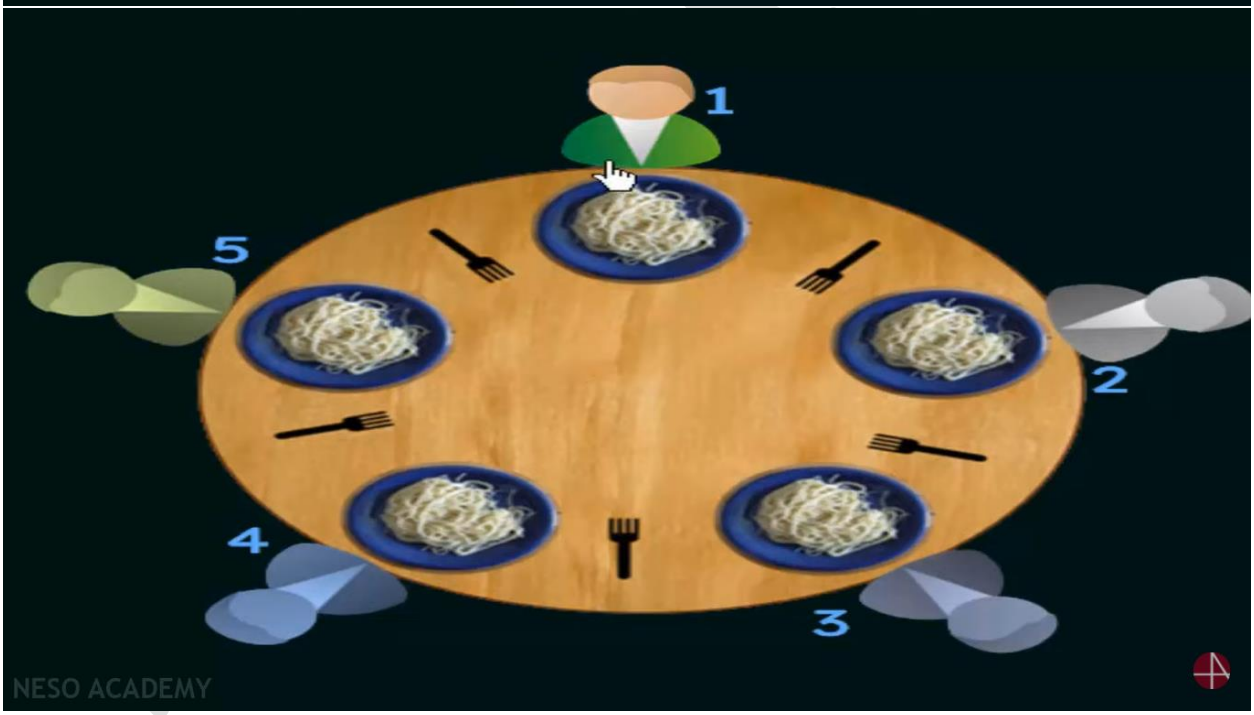
NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.





NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

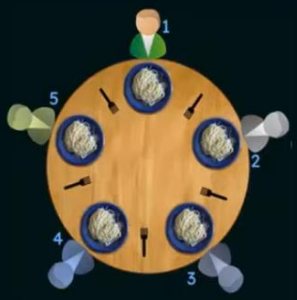




NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Some possible remedies to avoid deadlocks:

- Allow at most four philosophers to be sitting simultaneously at the table.
- Allow a philosopher to pick up his chopsticks only if both chopsticks are available (to do this he must pick them up in a critical section).
- Use an asymmetric solution; that is, an odd philosopher picks up first his left chopstick and then his right chopstick, whereas an even philosopher picks up his right chopstick and then his left chopstick.



NESO ACADEMY

Monitors

- A high level abstraction that provides a convenient and effective mechanism for process synchronization.
- A monitor type presents a set of programmer-defined operations that provide mutual exclusion within the monitor.
- The monitor type also contains the declaration of variables whose values define the state of an instance of that type, along with the bodies of procedures or functions that operate on those variables.

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Syntax of a Monitor

```
monitor monitor_name
{
// shared variable declarations
procedure P1 ( . . . ) {
. . .
}
procedure P2 ( . . . ) {
. . .
}
.
.
procedure Pn ( . . . ) {
. . .
}
initialization code ( . . . ) {
. . .
}
}
```

NESO ACADEMY

- A procedure defined within a monitor can access only those variables declared locally within the monitor and its formal parameters.
- Similarly, the local variables of a monitor can be accessed by only the local procedures.
- The monitor construct ensures that only one process at a time can be active within the monitor.

Condition Construct- **condition x, y;**
The only operations that can be invoked on a condition variable are wait ( ) and signal ( ).

The operation x.wait( ) ; means that the process invoking this operation is suspended until another process invokes x.signal ( );
The x. signal ( ) operation resumes exactly one suspended process.

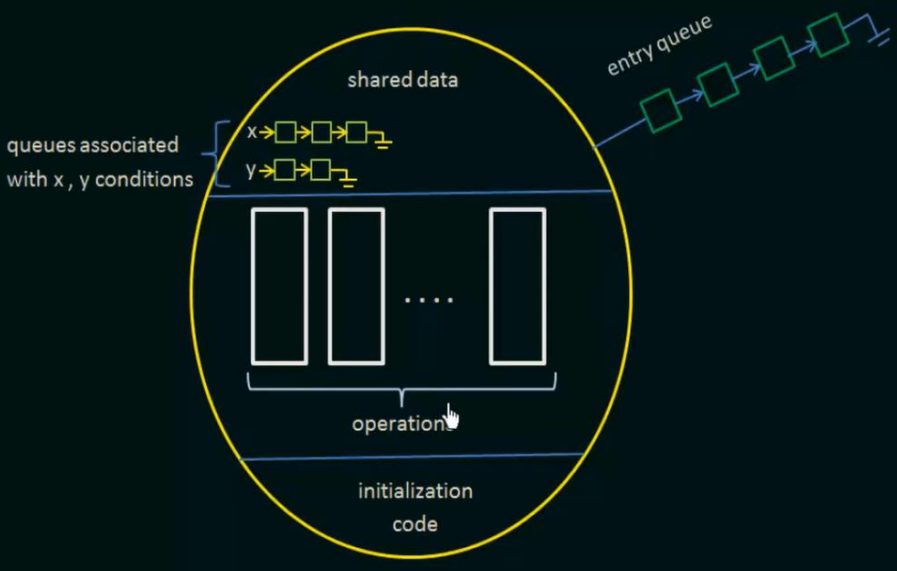


Fig: Schematic view of a monitor

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Dining-Philosophers Solution Using Monitors

- We now illustrate monitor concepts by presenting a deadlock-free solution to the dining-philosophers problem.
- This solution imposes the restriction that a philosopher may pick up his chopsticks only if both of them are available.
- To code this solution, we need to distinguish among three states in which we may find a philosopher. For this purpose, we introduce the following data structure:

```
enum { thinking, hungry, eating } state [5] ;
```

- Philosopher i can set the variable state [i] = eating only if his two neighbors are not eating: (state [(i+4) % 5] != eating) and (state [(i+1)% 5] != eating).
- We also need to declare `condition self [5] ;` where philosopher i can delay himself when he is hungry but is unable to obtain the chopsticks he needs.

NESO ACADEMY

A monitor solution to the dining-philosopher problem

```
monitor dp  {
    enum { THINKING, HUNGRY, EATING } state [5];
    condition self [5] ;

    void pickup (int i)  {
            state [i] = HUNGRY;
            test (i) ;
            if (state [i] != EATING)
            self [i] .wait() ;
            }
    void putdown(int i)  {
            state [i] = THINKING;
            test ((i + 4) % 5) ;
            test ((i + 1) % 5) ;
            }

    void test (int i)  {
            if ((state [(i + 4) % 5] != EATING) &&
                    (state [i] == HUNGRY) &&
                    (state [(i + 1) % 5] != EATING)) {
                            state [i] = EATING;
                            self [i].signal() ;
                            }
                    }
    initialization-code () {
            for (int i = 0; i < 5; i++)
            state [i] = THINKING ;
            }
}
```

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Process Synchronization
Solved Problems (Part-1)

Question:

GATE 2010

Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared boolean variables S1 and S2 are randomly assigned.

Method Used by P1	Method Used by P2
while (S1 == S2) ; Critical Section S1 = S2;	while (S1 != S2) ; Critical Section S2 = not (S1);

NESO ACADEMY

Question:

GATE 2010

Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared boolean variables S1 and S2 are randomly assigned.

Method Used by P1	Method Used by P2
while (S1 == S2) ; Critical Section S1 = S2;	while (S1 != S2) ; Critical Section S2 = not (S1);

Which one of the following statements describes the properties achieved?

(a) Mutual exclusion but not progress

(b) Progress but not mutual exclusion

(c) Neither mutual exclusion nor progress

(d) Both mutual exclusion and progress

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

A solution to the critical-section problem must satisfy the following three requirements:

1. Mutual exclusion:

If process P_i is executing in its critical section, then no other processes can be executing in their critical sections.

2. Progress:

If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in the decision on which will enter its critical section next, and this selection cannot be postponed indefinitely.

3. Bounded waiting:

There exists a bound, or limit, on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section and before that request is granted.

NESO ACADEMY

Process Synchronization
Solved Problems (Part-2)

Question:

GATE 2010

The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as S0=1, S1=0, S2=0.

Process P0	Process P1	Process P2

while (true) {
wait (S0);
print (0);
release (S1);
release (S2);
}
	wait (S1);
	release (S0);
		wait (S2);
		release (S0);

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Question:

GATE 2010

The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as S0=1, S1=0, S2=0.

Process P0	Process P1	Process P2

while (true) {
wait (S0);
print (0);
release (S1);
release (S2);
}
	wait (S1);
	release (S0);
		wait (S2);
		release (S0);

How many times will process P0 print '0' ?

(a) At least twice

(b) Exactly twice

(c) Exactly thrice

(d) Exactly once

NESO ACADEMY

Process Synchronization
Solved Problems (Part-3)

Question:

GATE 2007

Two processes, P1 and P2, need to access a critical section of code. Consider the following synchronization construct used by the processes:

Process P1	Process P2

while (true) {
wants1 = true;
while (wants2 == true);
/* Critical Section */
wants1=false;
}
/* Remainder section */
	while (true) {
	wants2 = true;
	while (wants1==true);
	/* Critical Section */
	wants2 = false;
	}
	/* Remainder section */

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Process P1	Process P2

while (true) {
wants1 = true;
while (wants2 == true);
/* Critical Section */
wants1=false;
}
/* Remainder section */
	while (true) {
	wants2 = true;
	while (wants1==true);
	/* Critical Section */
	wants2 = false;
	}
	/* Remainder section */

Here, wants1 and wants2 are shared variables, which are initialized to false. Which one of the following statements is TRUE about the above construct?

(a) It does not ensure mutual exclusion.

(b) It does not ensure bounded waiting.

(c) It requires that processes enter the critical section in strict alternation.

(d) It does not prevent deadlocks, but ensures mutual exclusion.

NESO ACADEMY

Process Synchronization
Solved Problems (Part- 4)

Question:

GATE 2019

Consider three concurrent processes P1, P2 and P3 as shown below, which access a shared variable D that has been initialized to 100.

P1	P2	P3
. . D = D + 20 . .	. . D = D - 50 . .	. . D = D + 10 . .

NESO ACADEMY

NAME – RUSHIKESH SHARAD MANE
SUBJECT- OPERATING SYSTEM(OS)
TOPIC-1. INTRODUCTION TO OPERATING SYSTEM.

Consider three concurrent processes P1, P2 and P3 as shown below, which access a shared variable D that has been initialized to 100.

P1	P2	P3
. . D = D + 20 . .	. . D = D - 50 . .	. . D = D + 10 . .

The process are executed on a uniprocessor system running a time-shared operating system. If the minimum and maximum possible values of D after the three processes have completed execution are X and Y respectively, then the value of Y-X is _____.

(a) 80

(b) 130

(c) 50

(d) None of these

Process Synchronization
Solved Problems (Part-5)

Question:

GATE 1998

A counting semaphore was initialized to 10. Then 6 P (wait) operations and 4V (signal) operations were completed on this semaphore. The resulting value of the semaphore is

(a) 0

(b) 8

(c) 10

(d) 12