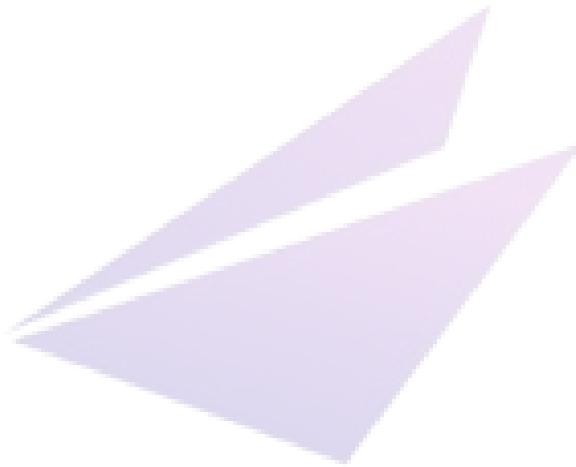


# Operating System

PANA ACADEMY

# Evolution of Operating system

- The First Generation
- The Second Generation
- The Third Generation
- The Fourth Generation



# Types of Operating System

## **Batch operating system**

In the 1970s, Batch processing was very popular. In this technique, similar types of jobs were batched together and executed in time. People were used to having a single computer which was called a mainframe.

**General Motors Research Laboratories (GMRL)** first batch operating system.

## **Multiprogramming operating system**

Multiprogramming is an extension to batch processing where the CPU is always kept busy. Each process needs two types of system time: CPU time and IO time.

In a multiprogramming environment, when a process does its I/O, The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.

# Types of Operating system

## Multiprocessing Operating System

In Multiprocessing, Parallel computing is achieved. There are more than one processors present in the system which can execute more than one process at the same time. This will increase the throughput of the system.

## Multitasking Operating System

The multitasking operating system is a logical extension of a multiprogramming system that enables **multiple** programs simultaneously. It allows a user to perform more than one computer task at the same time.

## Real Time Operating System

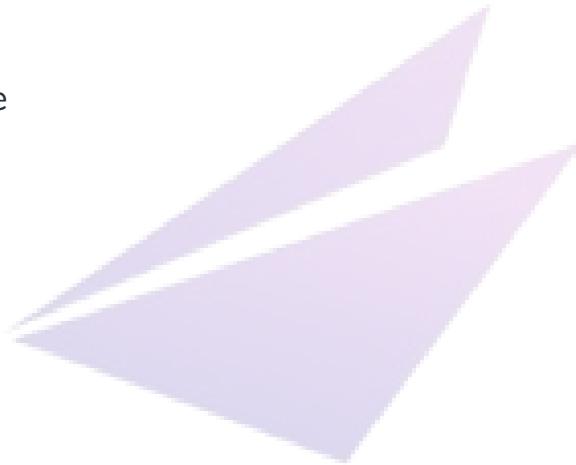
In Real-Time Systems, each job carries a certain deadline within which the job is supposed to be completed, otherwise, the huge loss will be there, or even if the result is produced, it will be completely useless.

# Components of Operating system

- 1. Process Management
- 2. I/O Device Management
- 3. File Management
- 4. Network Management
- 5. Main Memory Management
- 6. Secondary Storage Management
- 7. Security Management
- 8. Command Interpreter System

# Structures of operating system

1. Simple/Monolithic Structure
2. MicroKernel Structure
3. Hybrid-Kernel Structure





# Monolithic kernel vs Microkernel

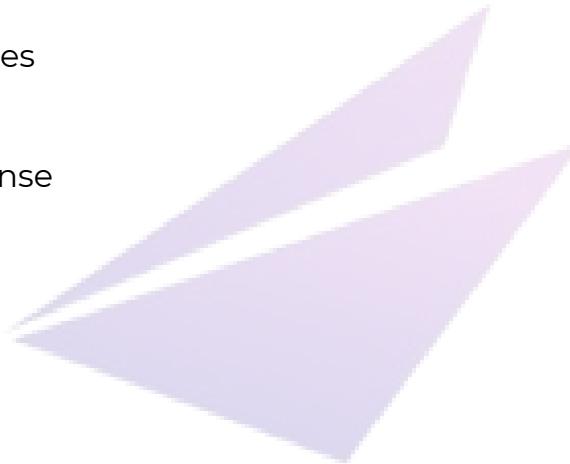
- **Monolithic**
  - a single large processes running entirely in a single address space
  - All kernel services exist and execute in kernel address space.
  - The kernel can invoke functions directly
  - It is a single static binary file
- **Micro**
  - broken down into separate processes (known as servers )
  - All servers are kept separate and run in different address spaces.
  - servers communicate through IPC.

# Summary: Kernels

- **Monolithic kernels**
  - Advantages: performance
  - Disadvantages: difficult to debug and maintain
- **Microkernels**
  - Advantages: more reliable and secure
  - Disadvantages: more overhead
- **Hybrid Kernels**
  - Advantages: benefits of monolithic and microkernels
  - Disadvantages: same as monolithic kernels
- **Nano kernel & Exo kernels**
  - Advantages: minimal and simple
  - Disadvantages: more work for application developers

# Services of Operating system

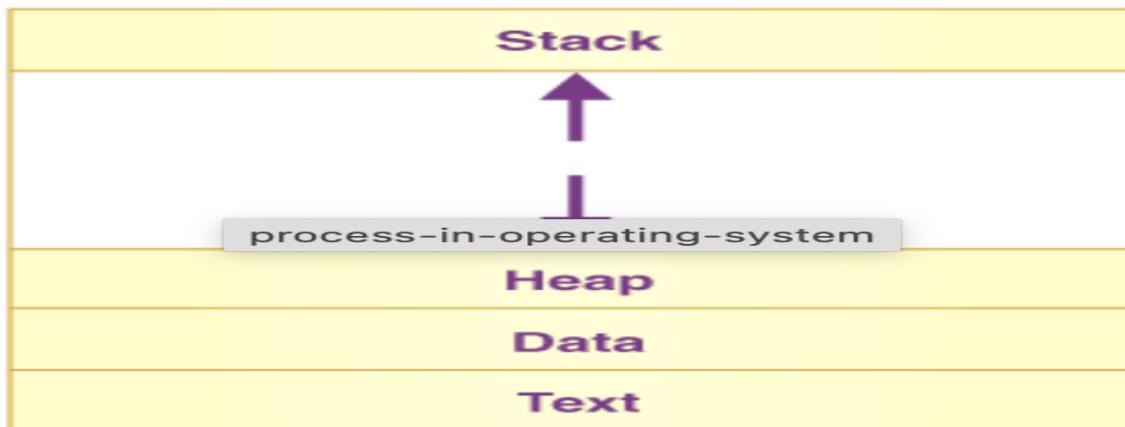
- o Program execution
- o Control Input/output devices
- o Program creation
- o Error Detection and Response
- o Accounting
- o Security and Protection
- o File Management
- o Communication



# Process

A process is basically a program in execution.

Components of process



# Components of process

## Stack

Temporary data like method or function parameters, return address, and local variables are stored in the process stack.

## Heap

This is the memory that is dynamically allocated to a process during its execution.

## Text

This comprises the contents present in the processor's registers as well as the current activity reflected by the value of the program counter.

## Data

The global as well as static variables are included in this section.

# Process Life Cycle

When a process runs, it goes through many states. Distinct operating systems have different stages, and the names of these states are not standardised. In general, a process can be in one of the five states listed below at any given time.

## Start

When a process is started/created first, it is in this state.

## Ready

Here, the process is waiting for a processor to be assigned to it. Ready processes are waiting for the operating system to assign them a processor so that they can run. The process may enter this state after starting or while running, but the scheduler may interrupt it to assign the CPU to another process.

## **Running**

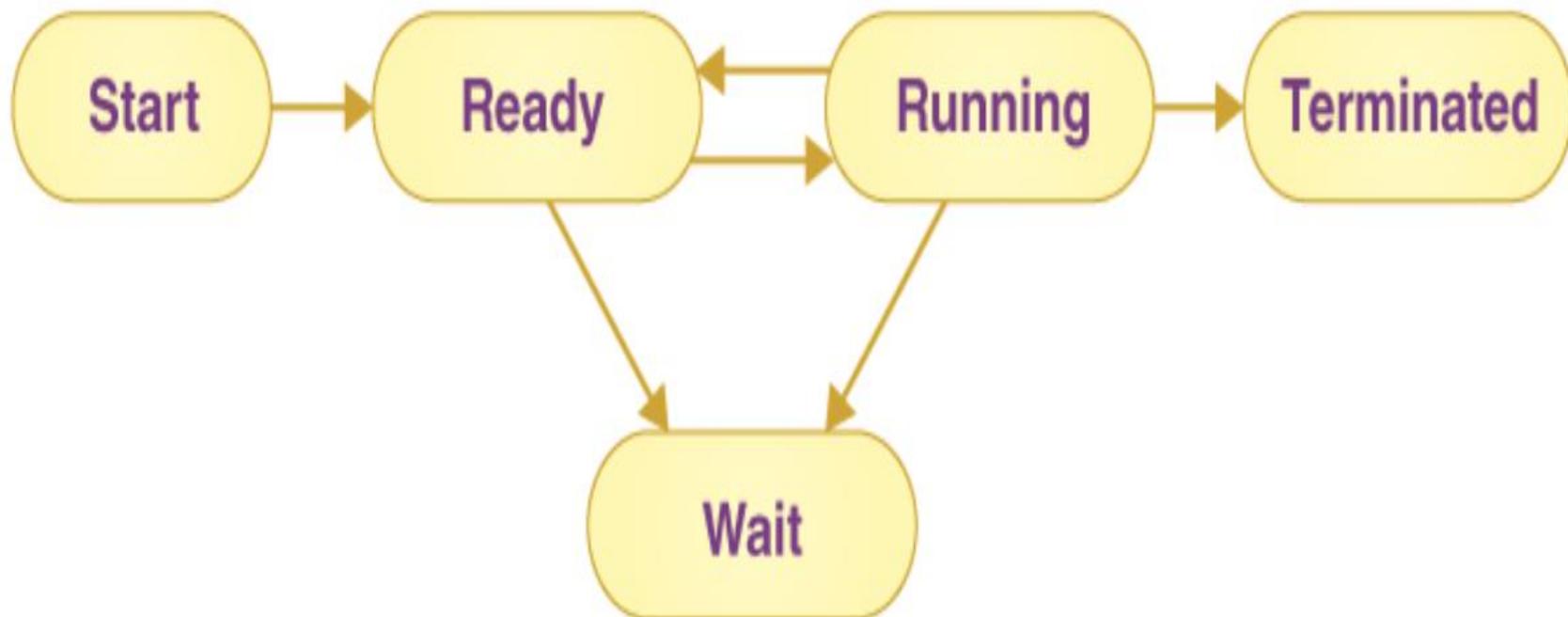
When the OS scheduler assigns a processor to a process, the process state gets set to running, and the processor executes the process instructions.

## **Waiting**

If a process needs to wait for any resource, such as for user input or for a file to become available, it enters the waiting state.

## **Terminated or Exit**

The process is relocated to the terminated state, where it waits for removal from the main memory once it has completed its execution or been terminated by the operating system.



# Description of states

- NEW – The creation of the process.
- READY – The waiting for the process that is to be assigned to any processor.
- RUNNING – Execution of the instructions.
- WAITING – The waiting of the process for some event that is about to occur (like an I/O completion, a signal reception, etc.).
- TERMINATED – A process has completed execution.

# Process Control

To identify the processes, it assigns a process identification number (PID) to each process.

Process control block (PCB) is used to track the process execution status.

Each block of memory contains information about the process state, program counter, stack pointer, status of opened files, scheduling algorithms, etc.

A Process Control Block (PCB) is a data structure used by the operating system to manage information about a process.

S. No	<b>Process</b>	<b>Thread</b>
1.	When a program is under execution, then it is known as a process.	A segment of a process is known as thread.
2.	It consumes maximum time to stop.	It consumes minimum time to stop.
3.	It needs more time for work and conception.	It needs less time for work and conception.
4.	Context switching takes maximum time here.	Here, context switching takes minimum time.
5.	It is not that effective in terms of communication.	It is effective in terms of communication.
6.	It takes more resources.	It takes less resources.
7.	It is a heavy weight process.	It is a light weight process.
8.	If one process is obstructed then it will not affect the operation of another process.	If one thread is obstructed then it will affect the execution of another process.

## **Concept of CPU scheduling**

CPU scheduling in an operating system is the process of determining which processes in the ready queue should be assigned to the CPU for execution. It involves deciding when to switch between processes, aiming to maximize efficiency and throughput.

## **Terminologies**

The ready queue is a data structure that holds all the processes that are ready to be executed by the CPU.

When a process is added to the ready queue for execution, it marks the arrival time.

The burst time signifies how long a process needs to finish its execution cycle. It helps determine the duration of each task completion.

## First-Come, First-Served (FCFS)

When a new process enters the ready queue, it takes its position at the end. The CPU serves processes based on their arrival time.

For example, if Process A arrives before Process B, it will be executed first. However, a drawback is that shorter processes may get stuck behind longer ones.

## Shortest Job Next (SJN)

In this method, when the CPU becomes available, it selects the process with the smallest execution time remaining. If two processes have equal burst times, FCFS scheduling is used to break ties.

SJN can lead to situations where short-term jobs suffer from starvation due to continuously arriving long-term jobs.

## Round Robin (RR)

With [Round Robin scheduling](#), each process gets an equal share of the processor's time. If a process doesn't complete within its quantum, it goes back to the end of the queue and waits for its turn again.

While RR prevents starvation and provides reasonable response times for all tasks, too small a quantum can lead to context-switching overheads.

# Important Points

Non-preemptive scheduling allows a process to finish its execution before another one starts. This method is suitable for tasks that should run without interruption.

In preemptive scheduling, the operating system can stop a running process to give the CPU a more important task.

**Criteria for effective CPU scheduling:**

**Maximizing CPU Utilization**

**Maximizing Throughput**

**Minimizing Waiting Time**

# MCQ

<https://www.sanfoundry.com/operating-system-questions-answers-basics/>

<https://www.includehelp.com/mcq/operating-system-mcqs.aspx>

<https://testbook.com/objective-questions/mcq-on-process-and-thread--5eea6a1539140f30f369f32a>