**A MICRO PROJECT**

## ON

**“ Multitasking in Operating Systems "**



**Submitted By**

**GANGANI VIVEK YOGESHBHAI (236010307027)**

Under the guidance of

## MRS.DIPALI.K.CHAUHAN

**Department of Computer Engineering A.Y.Dadabhai Technical Institute, Kosamba Subject Name :BASICS OF OPERATING SYSTEM(4330703)**

**Academic year: 2024-25**



# [Gujarat Technological University](https://m.facebook.com/p/GTU-Gujarat-Technological-University-100064322101451/)

**Certificate**

This is to certify that **GANGANI VIVEK YOGESHBHAI 236010307027** of **Semester – 3 in Diploma in Computer Engineering Department**, in **A.Y.Dadabhai Technical Institute, Kosamba (601)**has completed the Micro Project satisfactorily in **Subject – : BASICS OF OPERATING SYSTEM AND 4330703 for the academic year 2024- 2025.** as prescribed in the curriculum.

**Place: Kosamba**

**Subject Teacher**

## Micro-Project Proposal

**Brief Introduction**

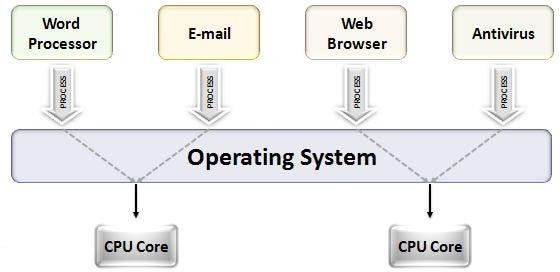
Multitasking is a fundamental capability of modern operating systems that allows them to run multiple processes simultaneously. This means that a computer can handle multiple tasks at the same time, making efficient use of its resources. The main goal of multitasking is to maximize the utilization of the CPU and improve the overall performance and responsiveness of the system.

## Aim of the Micro-Project

The aim of this microproject is to explore multitasking in operating systems, explaining its concept, history, types, and basic mechanisms. It highlights the advantages and disadvantages of multitasking and examines real-world applications to show how multitasking enhances system efficiency and user experience. The project also analyzes how multitasking impacts system performance and resource utilization, concluding with insights for further improvement in multitasking technologies.

## Definition of Multitasking:

Multitasking in operating systems refers to the capability of the system to execute multiple processes or tasks simultaneously. This is achieved by managing the execution of these processes in such a way that the CPU switches rapidly between them, giving the illusion that they are running concurrently. The primary goal of multitasking is to enhance the utilization of CPU resources, thereby improving the efficiency and responsiveness of the system.



## Types of Multitasking Operating System:

* **Cooperative Multitasking**
* **Preemptive Multitasking**

1. **Cooperative Multitasking**

In cooperative multitasking, each running process decides when to give up control of the CPU, allowing other processes to run. This type of multitasking relies on the cooperation of all processes to share CPU time effectively**.**

### Characteristics:

* + Voluntary Yielding: Each process is responsible for yielding control to other processes.
  + Less System Control: The operating system has less control over the CPU scheduling as it depends on the behavior of the processes.
  + Simpler Implementation: Easier to implement due to less complexity in context switching.

### Preemptive Multitasking

In preemptive multitasking, the operating system has complete control over the allocation of CPU time. It can preempt, or interrupt, a running process to switch to another process, ensuring fair CPU time distribution and better system responsiveness.

### Characteristics:

* + OS-Controlled Scheduling: The operating system determines which process runs and when.
  + Forced Yielding: The OS can forcibly take control from a running process to allocate CPU time to another process.
  + Complex Implementation: Requires sophisticated scheduling algorithms and efficient context switching mechanisms.

## How Multitasking Works in Operating Systems:

Multitasking in operating systems is a complex process involving several key components and mechanisms. Here's a simplified explanation of how it works:

### Process Creation:

* + When you run an application, the operating system creates a process. Each process is an instance of a program, which includes the program code and its current activity. A process is assigned resources like memory, I/O devices, and CPU time.

### Thread Management:

* + A process can contain multiple threads, which are the smallest units of execution. Threads within a process share the same memory and resources but run independently. Multithreading allows a process to perform multiple tasks concurrently.

### CPU Scheduling:

* + The operating system uses a scheduler to manage which processes get CPU time and when. Scheduling algorithms, such as Round Robin, Priority Scheduling, and Multilevel Queue, help determine the order of execution. The goal is to optimize CPU usage and ensure fair distribution of resources.

### Context Switching:

* + Multitasking requires the CPU to switch between different tasks. Context switching involves saving the state of the currently running process (like the contents of registers and the program counter) and loading the state of the next process to be executed. This happens rapidly, giving the illusion of simultaneous execution.

### Resource Allocation:

* + The operating system allocates resources such as memory, CPU, and I/O devices to processes based on their needs. Dynamic allocation ensures that resources are utilized efficiently and that processes do not interfere with each other.

### Interrupt Handling:

* + Interrupts are signals sent to the CPU to indicate that an event needs immediate attention. The operating system handles interrupts by pausing the current process, servicing the interrupt, and then resuming the process. This is crucial for managing tasks like I/O operations and real- time processing.

### Virtual Memory Management:

* + Virtual memory allows the operating system to use disk space as an extension of RAM, enabling more processes to run simultaneously than would fit in physical memory. This is managed through paging and swapping, ensuring that each process has the memory it needs.

## Challenges of Multitasking:

Multitasking in operating systems brings several benefits, but it also presents a variety of challenges that need to be addressed to ensure efficient and stable system performance. Here are some key challenges of multitasking**:**

### Resource Contention

* + **Description:** When multiple processes compete for limited resources such as CPU time, memory, and I/O devices, it can lead to contention.
  + **Impact:** This competition can result in performance bottlenecks, where processes are delayed waiting for resources, thereby reducing overall system efficiency.

### Context Switching Overhead

* + **Description:** Context switching is the process of storing the state of a currently running task and loading the state of the next task.
  + **Impact:** Frequent context switching can introduce significant overhead, consuming CPU cycles and memory. This overhead can degrade system performance, especially if the switch occurs too often.

### Synchronization Issues

* + **Description:** Processes often need to access shared resources concurrently. Proper synchronization is necessary to prevent conflicts and ensure data consistency.
  + **Impact:** Without proper synchronization mechanisms, such as semaphores and mutexes, race conditions can occur, leading to data corruption and unpredictable system behavior.

### Deadlocks

* + **Description:** A deadlock occurs when two or more processes are waiting indefinitely for resources held by each other.
  + **Impact:** Deadlocks can freeze part or all of the system, requiring manual intervention to resolve. Detecting and preventing deadlocks is a complex task.

### Priority Inversion

* + **Description:** Priority inversion happens when a lower-priority process holds a resource needed by a higher-priority process, causing the higher-priority process to wait.
  + **Impact:** This can undermine the effectiveness of priority scheduling and degrade system performance. Priority inheritance protocols are sometimes used to mitigate this issue.

### Fairness and Starvation

* + **Description:** Ensuring that all processes get a fair share of CPU time is essential for system stability. Starvation occurs when a process is perpetually denied necessary resources.
  + **Impact:** Starvation can lead to critical processes being unable to execute, affecting the overall functionality and responsiveness of the system.

## Advantages of Multitasking:

1. Increased Efficiency: By allowing multiple processes to run concurrently, multitasking makes optimal use of CPU resources, reducing idle times and improving overall system efficiency.
2. Enhanced User Experience: Users can run several applications simultaneously, such as browsing the internet, listening to music, and working on documents, without significant performance degradation.
3. Improved Productivity: Multitasking enables users and systems to perform multiple operations at once, facilitating quicker completion of tasks and more efficient workflows.
4. Effective Resource Utilization: The operating system can dynamically allocate resources such as CPU, memory, and I/O devices to processes based on their needs, ensuring that these resources are used effectively.

## Disadvantages of Multitasking:

1. Resource Contention: Multiple processes competing for limited resources can lead to contention, causing performance bottlenecks and potentially reducing overall system efficiency.
2. Context Switching Overhead: The frequent switching between tasks requires saving and loading process states, which consumes CPU cycles and memory, introducing overhead that can impact performance.
3. Complexity in Implementation: Implementing multitasking involves sophisticated scheduling algorithms and resource management techniques, increasing the complexity of the operating system's design and maintenance.
4. Potential for Bugs and Race Conditions: The complexity of managing multiple tasks and ensuring proper synchronization can lead to bugs and race conditions, particularly in systems with poorly coordinated software.

# Conclusion:

The Multitasking in operating systems allows multiple processes to run simultaneously by efficiently managing CPU resources, enhancing system performance and user experience. This capability evolved from early time-sharing systems to advanced preemptive multitasking in modern operating systems. While multitasking offers benefits like increased efficiency and productivity, it also presents challenges such as resource contention and context switching overhead. Addressing these challenges requires sophisticated scheduling algorithms and resource management techniques, ensuring effective utilization of system resources and maintaining stability and responsiveness. Overall, understanding multitasking is crucial for optimizing the performance and efficiency of operating systems.

**Signature………………………………**