Introduction to Platform Technologies

ABOUT PLATFORM

Platform - is a group of technology that are used as a base upon other applications, processes or technologies are developed

conforms to a set of standards that enable software developers to develop software application for the platform.

MODERN COMPUTER consist of:

- One or more processors
- Main Memory
- Disk
- Printer
- Various Input/output device

GOAL OF OPERATING SYSTEM

Efficient Utilization
- Optimize use of CPU, memory, storage, and I/O devices

Multitasking
- Manage the execution of multiple processes simultaneously

Resource Sharing
- Allow multiple users/applications to share resources without conflicts

System Security
- Enforce security measures like authentication and access control

PROCESS CONTROL

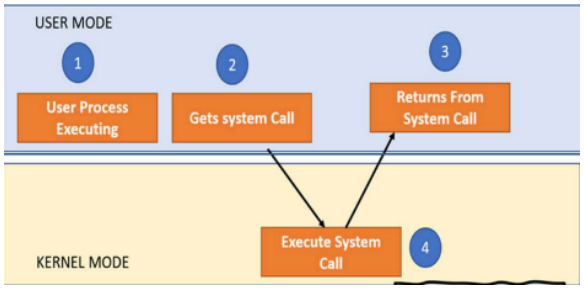
• Memory Management
• Memory Processor
• File Management
• Security
• Control over system Performance
• Job accounting
• Error detection aids
• Coordination between software and users

WHAT IS A SYSTEM CALL IN OPERATING SYSTEM?

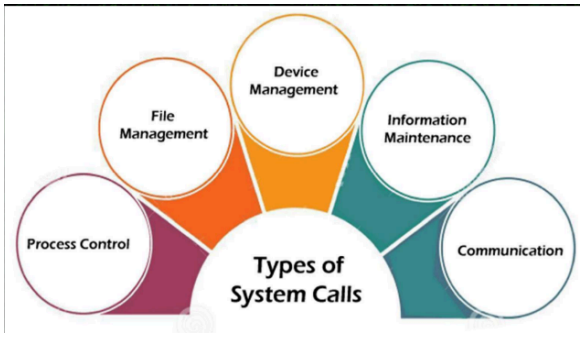
- way for a user program to interface with the operating system. The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request.

- **Application Program Interface (API)** connects the operating system's functions to user programs.

ARCHITECTURE OF SYSTEM CALLS



Type of System Calls



PROCESS CONTROL

THIS SYSTEM CALLS PERFORM THE TASK OF PROCESS CREATION, PROCESS TERMINATION, ETC.

Functions:

- End and Abort
- Load and Execute
- Create Process and Terminate Process
- Wait and Signal Event
- Allocate and free memory

FILE MANAGEMENT

FILE MANAGEMENT SYSTEM CALLS HANDLE FILE MANIPULATION JOBS LIKE CREATING A FILE, READING, AND WRITING, ETC.

Functions:

- Create a file
- Delete file
- Open and close file
- Read, write, and reposition
- Get and set file attributes

DEVICE MANAGEMENT

DEVICE MANAGEMENT DOES THE JOB OF DEVICE MANIPULATION LIKE READING FROM DEVICE BUFFERS, WRITING INTO DEVICE BUFFERS, ETC.

Functions:

- Request and release device
- Logically attach/ detach devices
- Get and Set device attributes

INFORMATION MAINTENANCE

IT HANDLES INFORMATION AND ITS TRANSFER BETWEEN THE OS AND THE USER PROGRAM.

Functions:

- Get or set time and date
- Get process and device attributes

COMMUNICATION

THESE TYPES OF SYSTEM CALLS ARE SPECIALLY USED FOR INTERPROCESS COMMUNICATIONS.

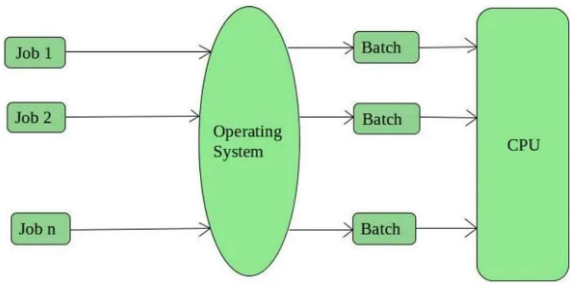
Functions:

- Create, delete communications connections
- Send, receive message
- Help OS to transfer status information
- Attach or detach remote devices

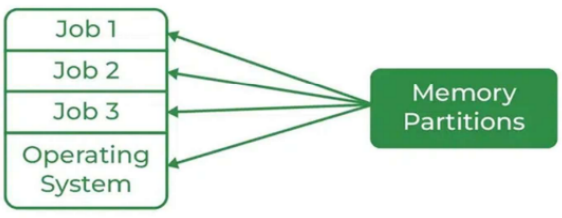
TYPES OF OS

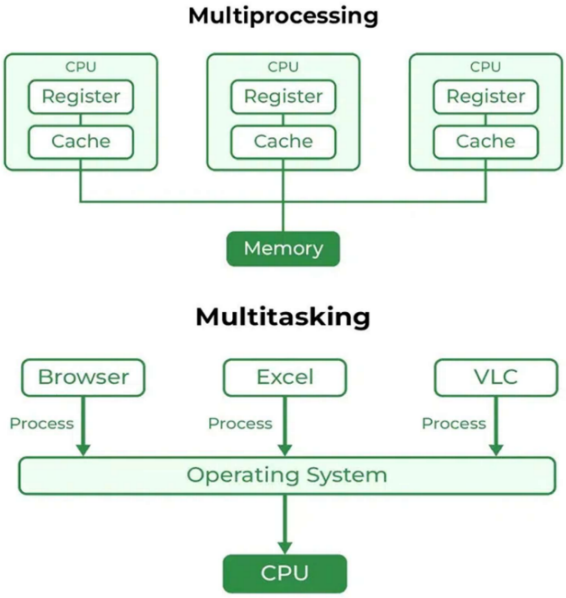
- Batch Operating System
- Multi-Programming System
- Multi-Processing System
- Multi-Tasking Operating System
- Time-Sharing Operating System
- Distributed Operating System
- Network Operating System
- Real-Time Operating System

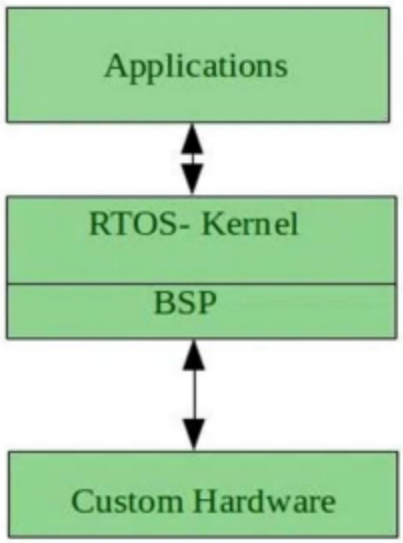
BATCH OPERATING SYSTEM



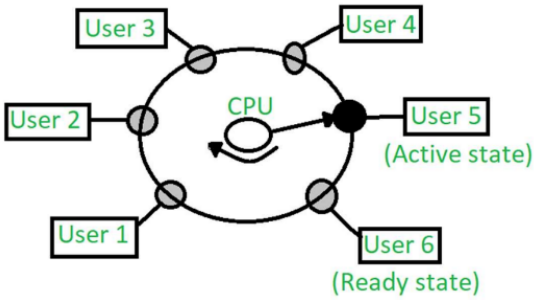
MULTI PROGRAMMING





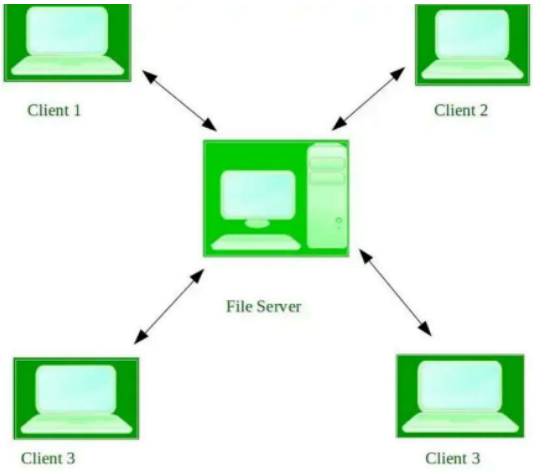


TIME-SHARING OPERATING SYSTEMS



DISTRIBUTED OPERATING SYSTEM

NETWORK OPERATING SYSTEM



REAL TIME OPERATING SYSTEM (RTOS)

Process and Threads

A **PROCESS** is basically a program in execution. The execution of a process must progress in a sequential fashion.
A **PROCESS** is defined as an entity which represents the basic unit of work to be implemented in the system.

Writing our computer **PROGRAM** in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

A PROCESS MEMORY

Stack - The process Stack contains the temporary data such as method/function parameters, return address and local variables.

Heap - This is dynamically allocated memory to a process during its run time.

Data - This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.

Text - This is dynamically allocated memory to a process during its run time.

PROGRAM

A **program** is a piece of code which may be a single line or millions of lines

A **computer program** is usually written by a computer programmer in a programming language.

A part of a computer program that performs a well-defined task is known as an **ALGORITHM.**

A collection of computer programs, libraries and related data are referred to as **SOFTWARE.**

PROCESS AND CREATION

A process may be created in the system for different operations.

Some of the events that lead to process creation are as follows −

- User request for process creation
- System Initialization
- Batch job initialization
- Execution of a process creation system call by a running process

A process may be created by another process using **fork().**

CAUSES FOR TERMINATION

Time slot expired − When the process execution does not complete within the time quantum, then the process gets terminated from the running state. The CPU picks the next job in the ready queue to execute.

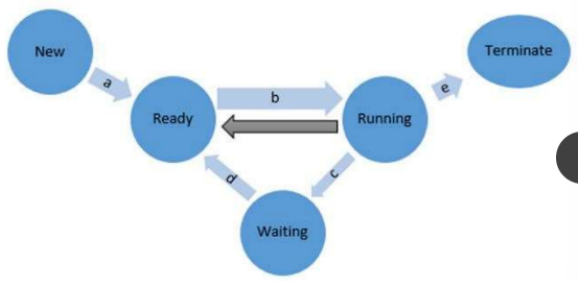
Memory bound violation − If a process needs more memory than the available memory.

I/O failure − When the operating system does not provide an I/O device, the process enters a waiting state.
Process request − If the parent process requests the child process about termination.
Invalid instruction

PROCESS TERMINATION

Process termination occurs when the process is terminated.
The exit() system call is used by most operating systems for process termination.
Whenever the process finishes executing its final statement and asks the operating system to delete it by using exit() system call.
At that point of time the process may return the status value to its parent process with the help of wait() system call.
All the resources of the process including physical and virtual memory, open files, I/O buffers are deallocated by the operating system.

Process States and Implementation of a Process



PROCESS STATES

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

1.**Running**: It means a process that is currently being executed. Assuming that there is only a single processor in the below execution process, so there will be at most one processor at a time that can be running in the state.

2.**Ready**: It means a process that is prepared to execute when given the opportunity by the OS.

3.**Blocked/Waiting**: It means that a process cannot continue executing until some event occurs like for example, the completion of an input-output operation.

4. **New**: It means a new process that has been created but has not yet been admitted by the OS for its execution. A new process is not loaded into the main memory, but its process control block (PCB) has been created.

5. **Exit/Terminate**: A process or job that has been released by the OS, either because it is completed or is aborted for some issue.

PROCESS STATE

POSSIBLE STATE TRANSITIONS

1. **Null -> New**: A new process is created for the execution of a process.

2. **New -> Ready**: The system will move the process from new to ready state and now it is ready for execution. Here a system may set a limit so that multiple processes can't occur otherwise there may be a performance issue.

3. **Ready -> Running**: The OS now selects a process for a run and the system chooses only one process in a ready state for execution.

4. **Running -> Exit:** The system terminates a process if the process indicates that it is now completed or if it has been aborted.

5. **Running -> Ready:** The reason for which this transition occurs is that when the running process has reached its maximum running time for uninterrupted execution. An example of this can be a process running in

the background that performs some maintenance or other functions periodically.

6. **Running -> Blocked**: A process is put in the blocked state if it requests something it is waiting. Like, a process may request some resources that might not be available at the time or it may be waiting for an I/O operation or waiting for some other process to finish before the process can continue.

7. **Blocked -> Ready**: A process moves from blocked state to the ready state when the event for which it has been waiting.

8. **Ready -> Exit**: This transition can exist only in some cases because, in some systems, a parent may terminate a child's process at any time.

Threads

- A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

- A thread is also called a lightweight process.

- Threads provide a way to improve application performance through parallelism.

- A thread is a basic unit of CPU utilization, consisting of a program counter, a stack, and a set of registers, (and a thread ID.)

- Traditional (heavyweight) processes have a single thread of control - There is one program counter, and one sequence of instructions that can be carried out at any given time.

- Multi-threaded applications have multiple threads within a single process, each having their own program counter, stack and set of registers, but sharing common code, data, and certain structures such as open files.

TYPES OF THREAD USER LEVEL - user managed threads.
ADVANTAGE:

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

DISADVANTAGE:

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

KERNEL- operating system manages threads acting on kernel, an operating system core.
ADVANTAGE:

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

DISADVANTAGE:

- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

Introduction to CPU Scheduling and Scheduling Algorithms

CPU Scheduling is a process that allows one process to use the CPU

while another process is delayed (in standby) due to unavailability of any resources such as I / O etc, thus making full use of the CPU.

The purpose of **CPU Scheduling** is to make the system more efficient, faster, and fairer.

Process Scheduling is the process of the process manager handling the removal of an active process from the CPU and selecting another process based on a specific strategy.

Process Scheduling is an integral part of Multi-programming applications.

Long Term or Job Scheduler

It brings the new process to the 'Ready State'.
-It is a job scheduler

Short-Term or CPU Scheduler

It is responsible for selecting one process from the ready state for scheduling it on the running state.
-It is a CPU scheduler

Medium-Term Scheduler
It is responsible for suspending and resuming the process. It mainly does swapping (moving processes from main memory to disk and vice versa)
-It is a process-swapping scheduler.

WHAT IS THE NEED FOR CPU SCHEDULING ALGORITHM?

- CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended.

- The main function of the CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line.

- In Multiprogramming, if the long-term scheduler selects multiple I / O binding processes then most of the time, the CPU remains idle. The function of an effective program is to improve resource utilization.

OBJECTIVES OF CPU SCHEDULING ALGORITHM:

- Utilization of CPU at maximum level. **Keep the CPU as busy as possible.**

- **Allocation of CPU should be fair.**
- **Throughput should be Maximum**. i.e. Number of processes that complete their execution per time unit should be maximized.

- **Minimum turnaround time**, i.e. time taken by a process to finish execution should be the least.

- There should be a **minimum waiting time** and the process should not starve in the ready queue.

- **Minimum response time**. It means that the time when a process produces the first response should be as less as possible.

WHAT ARE THE DIFFERENT TERMINOLOGIES TO TAKE CARE OF IN ANY CPU SCHEDULING ALGORITHM?

Arrival Time: Time at which the process arrives in the ready queue.

Completion Time: Time at which process completes its execution.

Burst Time: Time required by a process for CPU execution.

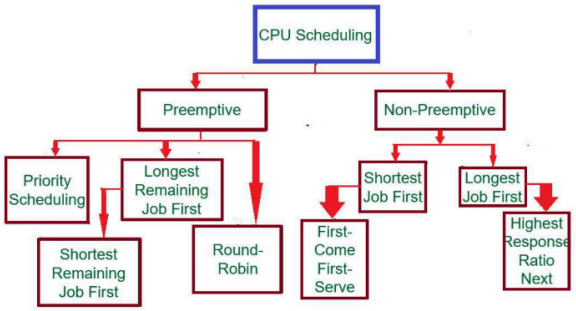
Turn Around Time: Time Difference between completion time and arrival time.

WHAT ARE THE DIFFERENT TYPES OF CPU SCHEDULING ALGORITHMS?

Preemptive Scheduling: Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.

Non-Preemptive Scheduling:
Non-Preemptive scheduling is used when a process terminates , or when a process switches from running state to waiting state.

CPU SCHEDULING SAMPLE



CPU SCHEDULING Algorithms in Operating Systems

1 .**First Come First Serve: FCFS** considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

2. **Shortest Job First(SJF):** is a scheduling process that selects the waiting process with the smallest execution time to execute next.

3. **Longest Job First(LJF)**:scheduling process is just opposite of shortest job first (SJF), as the name suggests this algorithm is based upon the fact that the process with the largest burst time is processed first. Longest Job First is non-preemptive in nature.

4. **Priority Scheduling**: is a pre-emptive method of CPU scheduling algorithm that works based on the priority of a process.

5. **Round robin:** is a CPU scheduling algorithm where each process is cyclically assigned a fixed time slot. It is the preemptive version of First come First Serve CPU Scheduling algorithm. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

6.**Shortest Remaining Time First**: is the preemptive version of the Shortest job first which we have discussed earlier where the processor is allocated to
the job closest to completion. In SRTF the process with the smallest amount of

time remaining until completion is selected to execute.

7.**Longest Remaining Time First**: s a preemptive version of the longest job first scheduling algorithm. This scheduling algorithm is used by the operating system to program incoming processes for use in a systematic way.

8. **Highest Response Ratio Next:** The name itself states that we need to find the response ratio of all available processes and select the one with the highest Response Ratio. A process once selected will run till completion.

9.**Multiple Queue Scheduling**:Processes in the ready queue can be divided into different classes.

System Processes: The CPU itself has its process to run, generally termed as System Process.

Interactive Processes: An Interactive Process is a type of process in which there should be the same type of interaction.

Batch Processes: Batch processing is generally a technique in the Operating system that collects the programs and data together in the form of a **batch** before the **processing** starts.

10. **Multilevel Feedback Queue Scheduling**: CPU Scheduling is like Multilevel Queue Scheduling but in this process can move between the queues. And thus, much more efficient than multilevel queue scheduling.

In a **multilevel queue-scheduling** algorithm, processes are permanently assigned to a queue on entry to the system, and processes are not allowed to move between queues.

CPU Scheduling Algorithms

Definition: **Scheduling** in operating systems refers to the method used to allocate CPU time to various processes

Purpose: Efficient CPU utilization and process management.

Preemptive Scheduling:

• Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state.

• The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining.

NON- Preemptive Scheduling:

• Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state.

• In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state.

Types of Scheduling Algorithm

First Come First Serve (FCFS)

• The process which arrives first in the ready queue is firstly assigned the CPU.
• In case of a tie, process with a smaller process id is executed first.
• It is always non-preemptive in nature.
• Jobs are executed on first come, first serve basis. It is a non-preemptive, pre-emptive scheduling algorithm.
• Easy to understand and implement.
• Its implementation is based on FIFO queue.
• Poor in performance as average wait time is high.

Shortest Job First (SJF)
• Process which have the shortest burst time are scheduled first.
• If two processes have the same bust time, then FCFS is used to break the tie.
• This is a non-pre-emptive, pre-emptive scheduling algorithm.
• Best approach to minimize waiting time. Easy to implement in Batch systems where required CPU time is known in advance.
• Impossible to implement in interactive systems where required CPU time is not known.
• The processer should know in advance how much time process will take.
• Pre-emptive mode of Shortest Job First is called as Shortest Remaining Time First (SRTF).

Priority Scheduling

Priority Scheduling is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc.

Types of Priority Scheduling

Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

Non-Preemptive Scheduling, the CPU has been allocated to a specific process. The process that keeps the CPU busy, will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

Characteristics of Priority Scheduling

A CPU algorithm that schedules processes based on priority.
It used in Operating systems for performing batch processes.
If two jobs having the same priority are READY, it works on a FIRST COME, FIRST SERVED basis.
In priority scheduling, a number is assigned to each process that indicates its priority level.
Lower the number, higher is the priority.
In this type of scheduling algorithm, if a newer process arrives, that is having a

higher priority than the currently running process, then the currently running process is preempted.

Round-Robin Scheduling

In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice.

This algorithm also offers starvation free execution of processes.

Advantage of Round-Robin
This scheduling method does not depend upon burst time. That's why it is easily implementable on the system.

Once a process is executed for a specific set of the period, the process is preempted, and another process executes for that given time period.

Disadvantage of Round-Robin
Round-robin scheduling doesn't give special priority to more important tasks.

Decreases comprehension

Memory Management

What is Memory Management

It ensures that blocks of memory space are properly managed and allocated so the operating system (OS), applications and other running processes have the memory they need to carry out their operations.

Memory is the important part of the computer that is used to store the data. Its management is critical to the computer system because the amount of main memory available in a computer system is very limited.

Role of Memory management

Memory manager is used to keep track of the status of memory locations, whether it is free or allocated. It addresses primary memory by providing abstractions so that software perceives a large memory is allocated to it.

Memory manager permits computers with a small amount of main memory to execute programs larger than the size or amount of available memory. It does this by moving information back and forth between primary memory and secondary memory by using the concept of swapping

Memory Management (Hardware)

In hardware, memory management involves components that physically store data, such as RAM (random access memory) chips, memory caches, and flash-based SSDs(solid-state drives).

Memory Management (OS)

In the OS, memory management involves the allocation (and constant reallocation) of specific memory blocks to individual programs as user demands change.

Memory Management (Application)

Allocation

When the program requests a block of memory, the memory manager must allocate that block out of the larger blocks it has received from the operating system. The part of the memory manager that does this is known as the allocator.

Recycling
When memory blocks have been allocated, but the data they contain is no longer required by the program, then the blocks can be recycled for reuse. There are two approaches to recycling memory: either the programmer must decide when memory can be reused (known as manual memory management); or the memory manager must be able to work it out (known as automatic memory management).

Binding of Instructions and Data to Memory

•**Compile time**: If memory location known a priori, **absolute code** can be generated; must recompile code if starting location changes
• **Load time**: Must generate **relocatable code** if memory location is not known at compile time
• **Execution time**: Binding delayed until run time if the process can be moved during its execution from one memory segment to another

Process Address Space

Memory Addresses & Description

1 **Symbolic addresses** - The addresses used in a source code. The variable names, constants, and instruction labels are the basic elements of the symbolic address space.

2 **Relative addresses** - At the time of compilation, a compiler converts symbolic addresses into relative addresses.

3 **Physical addresses** - The loader generates these addresses at the time when a program is loaded into main memory.

Logical Address

It is a virtual address generated by the CPU while a program is running. It is referred to as a virtual address because it does not exist physically. Using this address, the CPU access the actual address or physical address inside the memory, and data is fetched from there.

Physical Address

Physical Address is the actual address of the data inside the memory. The logical address is a virtual address and the program needs physical memory for its execution. The user never deals with the Physical Address. The user program generates the logical address and is mapped to the physical address by the Memory Management Unit(MMU).

Memory-Management Unit (MMU)

Hardware device that at run time maps virtual to physical address

Fragmentation

Fragmentation refers to the process or state where something is broken into

smaller, disconnected parts. Fragmentation is of two types:

External fragmentation
Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.

Internal fragmentation
Memory block assigned to process is bigger.
Some portion of memory is left unused, as it cannot be used by another process.

Multiple Partitioning
The single Contiguous memory management scheme is inefficient as it limits computers to execute only one program at a time resulting in wastage in memory space and CPU time.
The problem of inefficient CPU use can be overcome using multiprogramming that allows more than one program to run concurrently.

Fixed Partitioning
The main memory is divided into several fixed-sized partitions in a fixed partition memory management scheme or static partitioning. These partitions can be of the same size or different sizes.
Each partition can hold a single process.
The number of partitions determines the degree of multiprogramming, i.e., the maximum number of processes in memory.

Dynamic Partitioning
The dynamic partitioning was designed to overcome the problems of a fixed partitioning scheme. In a dynamic partitioning scheme, each process occupies only as much memory as they require when loaded for processing.

Non-Contiguous memory management schemes:

In a Non-Contiguous memory management scheme, the program is divided into different blocks and loaded at different portions of the memory that need not necessarily be adjacent to one another.

What is paging?

Paging is a technique that eliminates the requirements of contiguous allocation of main memory. In this, the main memory is divided into fixed-size blocks of physical memory called frames. The size of a frame should be kept the same as that of a page to maximize the main memory and avoid external fragmentation.

Segmentation

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is a different logical address space of the program.

File System

WHAT IS A File System?

A **file system** organizes and manages files stored on disk or in memory.

A collection of files is called a directory, and a file system comprises multiple directories.

WHAT IS File?

A file is a data structure storing a sequence of records.
Files can be simple (e.g., plain text) or complex (formatted data).
Directories are collections of files, and the file system is made up of directories at different levels.

File Attributes

File attributes describe properties of a file (e.g., name, type, size, location).
• Attributes can be viewed and modified (e.g., hidden, read-only, system files).
• Attributes may prevent unauthorized file deletion or modification.

Common File Attributes

1. Name: Unique identifier in the file system.
2. Type: File category (e.g., text, video, executable).
3. Location: Storage location in the file system.
4. Size: Amount of memory occupied by the file.
5. Protection: Permissions for users and groups.
6. Time/Date: Last modified timestamp.

File Operations

Overview

Common file operations: create, open, read, write, close, delete, append, truncate, rename.

File operations are performed using commands provided by the operating system.

Key File Operations

1. Create: Adds a new file to the system.
2. Open: Accesses an existing file for processing.
3. Write: Adds data to a file, increasing its size.
4. Read: Retrieves data from the file.

Additional File Operations

1. **Seek/Re-position:** Moves the file pointer for direct access.

2. **Delete:** Removes a file, freeing space.

3. **Truncate:** Deletes file content, retaining attributes.

4. **Append**: Adds data to the end of a file.

5. **Close**: Saves changes and releases resources.

File Access Methods

• Three main methods: sequential, direct, and indexed.
• Each method has specific advantages, depending on the application.

Sequential Access

• Files are accessed one record at a time, starting from the beginning.
• The pointer moves through the file as each record is read or written.

Direct Access

Direct access retrieves data from any block in a file without sequential traversal.
Efficient for databases and systems that require specific data quickly.

Indexed Access

An index is used to quickly locate specific records in a file.
Efficient for large databases but requires extra memory for the index.
Combines benefits of direct access with fast searching.

Directory Structure

• A directory lists related files, storing their attributes.
• Hard disks can be partitioned into volumes, with each partition containing directories.
• A directory entry holds information about each file.

What is a Directory?

• A directory is a container that organizes files and folders.
• Directories are often called "folders" and can contain subdirectories.
• Hierarchical directories are structured like a tree, with the root as the base (e.g., C: drive).
• Directories help logically organize files for easier access.

Common Directory Operations

File operations in a directory include:
1. File Creation
2. File Search
3. File Deletion
4. File Renaming
5. Traversing Files
6. Listing Files

File Systems in Operating Systems

• A file system organizes and manages files on storage devices.
• Common file systems:
• FAT (File Allocation Table) – used by older Windows versions.
• NTFS (New Technology File System) – modern Windows systems.
• ext – used by Linux and Unix-based systems.
• HFS/APFS – used by macOS.

Advantages of File Systems

1. Organization: Helps categorize and locate files easily.
2. Data Protection: Supports features like permissions, backup, and error correction.
3. Performance: Efficient file organization improves reading and writing speeds.

Layers of the File System

1. Logical File System: Handles file metadata and directory structure.
2. File Organization Module: Maps logical blocks to physical blocks.
3. Basic File System: Issues commands to the I/O control.
4. I/O Control: Manages device drivers and accesses the hard disk.

Allocation Methods

Disk space allocation methods impact performance:
Contiguous Allocation: Stores files in sequential blocks.
Linked Allocation: Files stored in non-contiguous blocks using pointers.
FAT: Stores pointers in a table.
Indexed Allocation: Uses an indexed block containing pointers to file blocks.

Input / Output

Categories of I/O Devices
• Human Readable:
○ Devices that interact with users (e.g., monitors, printers).
• Machine Readable:
○ Devices for electronic communication (e.g., sensors, robots).
• Communication Devices:
○ Used for networking and remote communication (e.g., modems, routers).

Programmed I/O

Programmed I/O (PIO) is a method of data transfer between the CPU and an I/O device where the CPU directly controls all aspects of the data transfer. In this method, the CPU actively monitors the status of the I/O device and

Interrupt Driven I/O

Interrupt-Driven I/O is a method of data transfer between the CPU and I/O devices that allows the CPU to perform other tasks while waiting for I/O operations to complete. Instead of the CPU actively polling the I/O device for its status, the device generates an interrupt signal when it requires attention, enabling more efficient use of CPU resources.

Interrupt Handling: Each interrupt requires context switching, which can introduce overhead, especially if interrupts occur frequently.

Direct Memory Access (DMA) is an efficient method of data transfer between an I/O device and the main memory that allows certain hardware subsystems to access the main system memory independently of the CPU.

This approach enhances performance by offloading data transfer tasks from the CPU, enabling it to execute other instructions while the data transfer is being handled

How DMA Works:

DMA Controller: A dedicated hardware component, known as the DMA controller, manages the data transfer between the I/O device and memory.

Bambi Platform Technology Reviewer

CPU Initialization: The CPU initiates a DMA transfer by sending a request to the DMA controller. This request typically includes:

The address in memory where data will be transferred.

The address of the I/O device.

The amount of data to be transferred.

Bus Control: Once the DMA controller is configured, it takes control of the system bus, allowing it to read data directly from the I/O device and write it to memory without CPU intervention.

Data Transfer: The DMA controller performs the data transfer, managing the read and write operations until the specified amount of data has been moved.

Completion Notification: After the data transfer is complete, the DMA controller sends an interrupt to the CPU to notify it that the operation has finished, allowing the CPU to resume processing other tasks.