# Memory Management in Operating System

Last Updated : 23 Jul, 2025

Memory is a hardware component that stores data, instructions and information temporarily or permanently for processing. It consists of an array of bytes or words, each with a unique address.

- Memory holds both input data and program instructions needed for the CPU to execute tasks.
- Memory works closely with the CPU to provide quick access to data being used.
- Memory management ensures efficient use of memory and supports multiprogramming.

## Memory Management

Memory management is a critical aspect of operating systems that ensures efficient use of the computer's memory resources. It controls how memory is allocated and deallocated to processes, which is key to both performance and stability. Below is a detailed overview of the various components and techniques involved in memory management.

Memory Management

## Why Memory Management is Required?

- Allocate and de-allocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.
- To maintain data integrity while executing of process.

*Read more about Requirements of Memory Management System here.*

## What is Main Memory?

Main memory, also known as RAM (Random Access Memory), is a large array of bytes or words that the computer's processor uses to store programs and data that are actively being processed. This memory is volatile, meaning that all data is lost when the power is turned off. Main memory is crucial for executing programs, and its size and speed directly influence the performance of the system.

## Logical and Physical Address Space

- **Logical Address Space:** An address generated by the CPU is known as a “Logical Address”. It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.
- **Physical Address Space:** It refers to the set of actual addresses used by the memory hardware. A physical address, also called a real address, is generated by the Memory Management Unit (MMU) through run-time mapping of virtual addresses. Unlike virtual addresses, physical addresses remain constant.

## Static and Dynamic Loading

Loading a process into the main memory is done by a loader. There are two different types of loading :

- **Static Loading:** Static Loading is basically loading the entire program into a fixed address. It requires more memory space.
- **Dynamic Loading:** Dynamic loading loads program routines into memory only when they are needed. This saves memory by not loading unused routines. The routines remain on disk in relocatable(can be loaded at any memory location) format until called. It allows better memory utilization, especially for large programs.

## Static and Dynamic Linking

To perform a linking task a linker is used. A linker is a program that takes one or more object files generated by a compiler and combines them into a single executable file.

- **Static Linking:** In static linking, the linker combines all necessary program modules into a single executable program. So there is no runtime dependency. Some operating systems support only static linking, in which system language libraries are treated like any other object module.

- **Dynamic Linking:** The basic concept of dynamic linking is similar to dynamic loading. In dynamic linking, "Stub" is included for each appropriate library routine reference. A stub is a small piece of code. When the stub is executed, it checks whether the needed routine is already in memory or not. If not available then the program loads the routine into memory.

## Swapping

Swapping moves processes between main memory and secondary memory to manage limited memory space. It allows multiple processes to run by temporarily swapping out lower priority processes for higher priority ones. The swapped-out process resumes once it's loaded back. Transfer time depends on the amount of data swapped.

Swapping

## Memory Management Techniques

Memory management techniques are methods used by an operating system to efficiently allocate, utilize, and manage memory resources for processes. Various techniques help the operating system manage memory effectively. They can be broadly categorized into:

Memory Management Techniques

## Memory Management with Monoprogramming (Without Swapping)

This is the simplest memory management approach the memory is divided into two sections: One part of the operating system. The second part of the user program

- In this approach, the operating system keeps track of the first and last location available for the allocation of the user program
- The operating system is loaded either at the bottom or at top
- Interrupt vectors are often loaded in low memory therefore, it makes sense to load the operating system in low memory
- Sharing of data and code does not make much sense in a single process environment
- The Operating system can be protected from user programs with the help of a fence register.

## Multiprogramming with Fixed Partitions (Without Swapping)

- A memory partition scheme with a fixed number of partitions was introduced to support multiprogramming. this scheme is based on contiguous allocation
- Each partition is a block of contiguous memory
- Memory is partitioned into a fixed number of partitions.
- Each partition is of fixed size

**Partition Table** : Once partitions are defined operating system keeps track of the status of memory partitions it is done through a data structure called a partition table.

| Starting Address of Partition | Size of Partition | Status |
|---|---|---|
| 0k | 200k | allocated |
| 200k | 100k | free |
| 300k | 150k | free |
| 450k | 250k | allocated |

## Contiguous Memory Allocation

Contiguous memory allocation is a memory management method where each process is given a single, continuous block of memory. This means all the data for a process is stored in adjacent memory locations.

Contiguous Memory Allocation

## Non-Contiguous Memory Allocation

This method allows processes to be broken into smaller parts, which are placed in different, non-adjacent memory locations. Techniques for non-contiguous memory allocation include:

- **Paging**: The process is divided into fixed-size blocks called "pages," and the memory is divided into blocks of the same size called "frames." The operating system keeps a page table to map logical pages to physical frames.
- **Segmentation**: The process is divided into segments of varying sizes, such as code, data, stack, etc. The operating system maintains a segment table to map logical segments to physical memory.

## Fragmentation

Fragmentation is defined as when the process is loaded and removed after execution from memory, it creates a small free hole. These holes can not be assigned to new processes because holes are not combined or do not fulfill the memory requirement of the process. In the operating systems two types of fragmentation are:

- **Internal fragmentation:** Happens when fixed-sized memory blocks are allocated to processes larger than needed, leaving unused space within the allocated block. For example, if a process needs 2MB of memory but is allocated a 3MB block, 1MB is wasted.
- **External fragmentation:** Occurs when free memory is scattered in small blocks across the system, making it impossible to allocate a large contiguous block to a process, even though the total free memory is enough.

## Memory Allocation Strategies

Efficient memory allocation is essential for optimal performance. There are several strategies for allocating memory blocks:

**Fixed Partition Allocation**: Memory is divided into fixed-sized partitions, and each partition can hold only one process. The OS keeps track of free and occupied partitions using a partition table.

**Dynamic Partition Allocation**: Memory is divided into variable-sized partitions based on the size of the processes. This helps avoid wastage of memory but can result in fragmentation.

**Placement Algorithms**: When allocating memory, the OS uses placement algorithms to decide which free block should be assigned to a process:

- **First Fit**: Allocates the first available partition large enough to hold the process.
- **Best Fit**: Allocates the smallest available partition that fits the process, reducing wasted space.
- **Worst Fit**: Allocates the largest available partition, leaving the largest remaining space.
- **Next Fit**: Similar to First Fit but starts searching for free memory from the point of the last allocation.