What is Memory Management?

In a multiprogramming computer, the Operating System resides in a part of memory, and the rest is used by multiple processes. The task of subdividing the memory among different processes is called Memory Management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

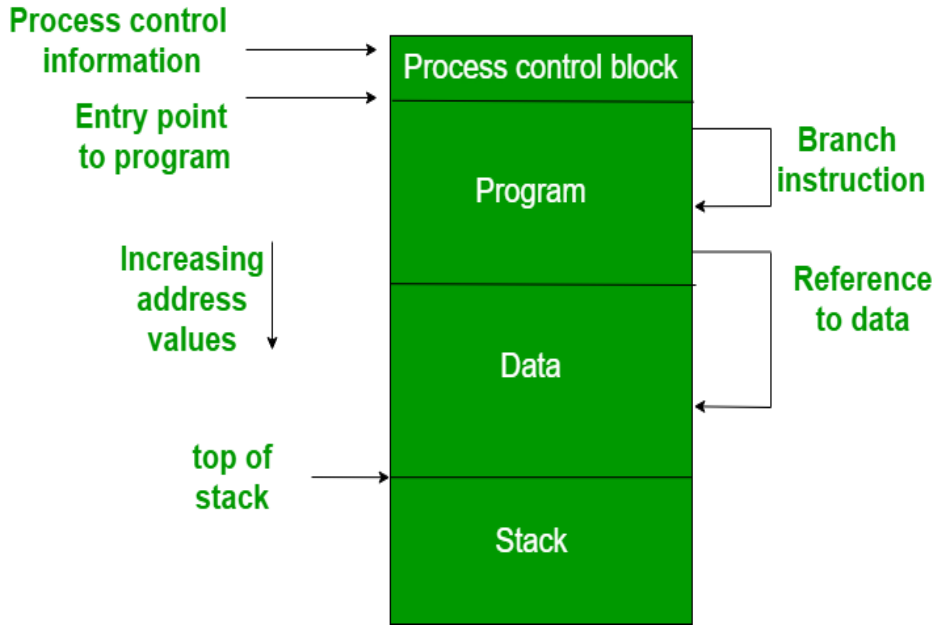
Memory Management Requirements

Memory management keeps track of the status of each memory location, whether it is allocated or free. It allocates the memory dynamically to the programs at their request and frees it for reuse when it is no longer needed. Memory management meant to satisfy some requirements that we should keep in mind.

These Requirements of memory management are:

1. **Relocation –** The available memory is generally shared among a number of processes in a multiprogramming system, so it is not possible to know in advance which other programs will be resident in main memory at the time of execution of this program. Swapping the active processes in and out of the main memory enables the operating system to have a larger pool of ready-to-execute process.

   When a program gets swapped out to a disk memory, then it is not always possible that when it is swapped back into main memory then it occupies the previous memory location, since the location may still be occupied by another process. We may need to **relocate** the process to a different area of memory. Thus there is a possibility that program may be moved in main memory due to swapping.



The figure depicts a process image. The process image is occupying a continuous region of main memory. The operating system will need to know many things including the location of process control information, the execution stack, and the code entry. Within a program, there are memory references in various instructions and these are called logical addresses. After loading of the program into main memory, the processor and the operating system must be able to translate logical addresses into physical addresses. Branch instructions contain the address of the next instruction to be executed. Data reference instructions contain the address of byte or word of data referenced.

- **Protection –** There is always a danger when we have multiple programs at the same time as one program may write to the address space of another program. So every process must be protected against unwanted interference when other process tries to write in a process whether accidental or incidental. Between relocation and protection requirement a trade-off occurs as the satisfaction of relocation requirement increases the difficulty of satisfying the protection requirement.

  Prediction of the location of a program in main memory is not possible, that's why it is impossible to check the absolute address at compile time to assure protection. Most of the programming language allows the dynamic calculation of address at run time. The memory protection requirement must be satisfied by the processor rather than the operating system because the operating system can hardly control a process when it occupies the processor. Thus it is possible to check the validity of memory references.
- **Sharing –** A protection mechanism must have to allow several processes to access the same portion of main memory. Allowing each processes access to the same copy of the program rather than have their own separate copy has an advantage.

  For example, multiple processes may use the same system file and it is natural to load one copy of the file in main memory and let it shared by those processes. It is the task of Memory management to allow controlled access to the shared areas of memory without compromising the protection. Mechanisms are used to support relocation supported sharing capabilities.
- **Logical organization –** Main memory is organized as linear or it can be a one-dimensional address space which consists of a sequence of bytes or words. Most of the programs can be organized into modules, some of those are unmodifiable (read-only, execute only) and some of those contain data that can be modified. To effectively deal with a user program, the operating system and computer hardware must support a basic module to provide the required protection and sharing. It has the following advantages:

- Modules are written and compiled independently and all the references from one module to another module are resolved by `the system at run time.
- Different modules are provided with different degrees of protection.
- There are mechanisms by which modules can be shared among processes. Sharing can be provided on a module level that lets the user specify the sharing that is desired.

- **Physical organization –** The structure of computer memory has two levels referred to as main memory and secondary memory. Main memory is relatively very fast and costly as compared to the secondary memory. Main memory is volatile. Thus secondary memory is provided for storage of data on a long-term basis while the main memory holds currently used programs. The major system concern between main memory and secondary memory is the flow of information and it is impractical for programmers to understand this for two reasons:
- The programmer may engage in a practice known as overlaying when the main memory available for a program and its data may be insufficient. It allows different modules to be assigned to the same region of memory. One disadvantage is that it is time-consuming for the programmer.
- In a multiprogramming environment, the programmer does not know how much space will be available at the time of coding and where that space will be located inside the memory.

Memory Management Partioning

In operating systems, Memory Management is the function responsible for allocating and managing a computer's main memory. Memory Management function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory.

- There are two Memory Management Techniques:
- **Contiguous**
- **Non-Contiguous**

In Contiguous Technique, executing process must be loaded entirely in the main memory.

- Contiguous Technique can be divided into:
- Fixed (or static) partitioning
  Variable (or dynamic) partitioning

Fixed (or static) Partitioning

Fixed partitioning, also known as static partitioning, is a memory allocation technique used in operating systems to divide the physical memory into fixed-size partitions or regions, each assigned to a specific process or user. Each partition is typically allocated at system boot time and remains dedicated to a specific process until it terminates or releases the partition.

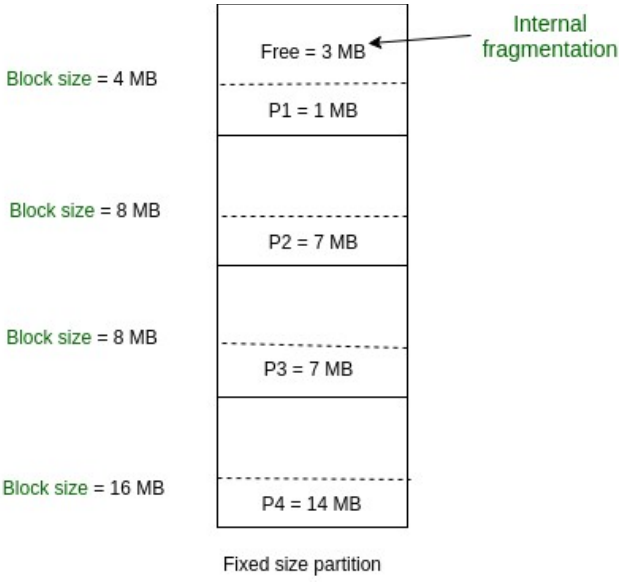
1. In fixed partitioning, the memory is divided into fixed-size chunks, with each chunk being reserved for a specific process. When a process requests memory, the operating system assigns it to the appropriate partition. Each partition is of the same size, and the memory allocation is done at system boot time.

2. Fixed partitioning has several advantages over other memory allocation techniques. First, it is simple and easy to implement. Second, it is predictable, meaning the operating system can ensure a minimum amount of memory for each process. Third, it can prevent processes from interfering with each other's memory space, improving the security and stability of the system.
3. However, fixed partitioning also has some disadvantages. It can lead to internal fragmentation, where memory in a partition remains unused. This can happen when the process's memory requirements are smaller than the partition size, leaving some memory unused. Additionally, fixed partitioning limits the number of processes that can run concurrently, as each process requires a dedicated partition.

Overall, fixed partitioning is a useful memory allocation technique in situations where the number of processes is fixed, and the memory requirements for each process are known in advance. It is commonly used in embedded systems, real-time systems, and systems with limited memory resources.

Fixed Partitioning:

This is the oldest and simplest technique used to put more than one process in the main memory. In this partitioning, the number of partitions (non-overlapping) in RAM is **fixed but the size** of each partition may or **may not be the same**. As it is a **contiguous** allocation, hence no spanning is allowed. Here partitions are made before execution or during system configure.



Fixed size partition

As illustrated in above figure, first process is only consuming 1MB out of 4MB in the main memory.
Hence, Internal Fragmentation in first block is (4-1) = 3MB.
Sum of Internal Fragmentation in every block = (4-1)+(8-7)+(8-7)+(16-14)= 3+1+1+2 = 7MB.

Suppose process P5 of size 7MB comes. But this process cannot be accommodated in spite of available free space because of contiguous allocation (as spanning is not allowed). Hence, 7MB becomes part of External Fragmentation.

There are some advantages and disadvantages of fixed partitioning.

Advantages of Fixed Partitioning

- **Easy to implement:** The algorithms needed to implement Fixed Partitioning are straightforward and easy to implement.
- **Low overhead:** Fixed Partitioning requires minimal overhead, which makes it ideal for systems with limited resources.
- **Predictable:** Fixed Partitioning ensures a predictable amount of memory for each process.
- **No external fragmentation:** Fixed Partitioning eliminates the problem of external fragmentation.
- **Suitable for systems with a fixed number of processes:** Fixed Partitioning is well-suited for systems with a fixed number of processes and known memory requirements.
- **Prevents processes from interfering with each other:** Fixed Partitioning ensures that processes do not interfere with each other's memory space.
- **Efficient use of memory:** Fixed Partitioning ensures that memory is used efficiently by allocating it to fixed-sized partitions.
- **Good for batch processing:** Fixed Partitioning is ideal for batch processing environments where the number of processes is fixed.
- **Better control over memory allocation:** Fixed Partitioning gives the operating system better control over the allocation of memory.
- **Easy to debug:** Fixed Partitioning is easy to debug since the size and location of each process are predetermined.

Disadvantages of Fixed Partitioning –

1.Internal Fragmentation:
Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This can cause internal fragmentation.

2. **External Fragmentation:**
   The total unused space (as stated above) of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form (as spanning is not allowed).

3. **Limit process size:**
   Process of size greater than the size of the partition in Main Memory cannot be accommodated. The partition size cannot be varied according to the size of the incoming process size. Hence, the process size of 32MB in the above-stated example is invalid.

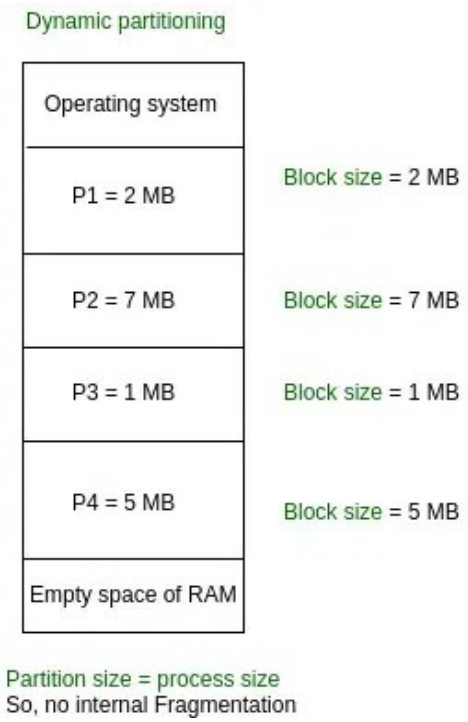
4. **Limitation on Degree of Multiprogramming:**
   Partitions in Main Memory are made before execution or during system configure. Main Memory is divided into a fixed number of partitions. Suppose if there are n1 partitions in

RAM and n2 are the number of processes, then n2<=n1 condition must be fulfilled. Number of processes greater than the number of partitions in RAM is invalid in Fixed Partitioning.

Variable (or Dynamic) Partitioning

It is a part of the Contiguous allocation technique. It is used to alleviate the problem faced by Fixed Partitioning. In contrast with fixed partitioning, partitions are not made before the execution or during system configuration. Various **features** associated with variable Partitioning-

- Initially, RAM is empty and partitions are made during the run-time according to the process's need instead of partitioning during system configuration.
- The size of the partition will be equal to the incoming process.
- The partition size varies according to the need of the process so that internal fragmentation can be avoided to ensure efficient utilization of RAM.
- The number of partitions in RAM is not fixed and depends on the number of incoming processes and the Main Memory's size.



Advantages of Variable Partitioning

- **No Internal Fragmentation:** In variable Partitioning, space in the main memory is allocated strictly according to the need of the process, hence there is no case of internal fragmentation. There will be no unused space left in the partition.
- **No restriction on the Degree of Multiprogramming:** More processes can be accommodated due to the absence of internal fragmentation. A process can be loaded until the memory is empty.
- **No Limitation on the Size of the Process:** In Fixed partitioning, the process with a size greater than the size of the largest partition could not be loaded and the process can not be

divided as it is invalid in the contiguous allocation technique. Here, In variable partitioning, the process size can’t be restricted since the partition size is decided according to the process size.

Disadvantages of Variable Partitioning

- **Difficult Implementation:** Implementing variable Partitioning is difficult as compared to Fixed Partitioning as it involves the allocation of memory during run-time rather than during system configuration.
- **External Fragmentation:** There will be external fragmentation despite the absence of internal fragmentation. For example, suppose in the above example- process P1(2MB) and process P3(1MB) completed their execution. Hence two spaces are left i.e. 2MB and 1MB. Let’s suppose process P5 of size 3MB comes. The space in memory cannot be allocated as no spanning is allowed in contiguous allocation. The rule says that the process must be continuously present in the main memory to get executed. Hence it results in External Fragmentation.