**CONTIGUOUS MEMORY ALLOCATION**

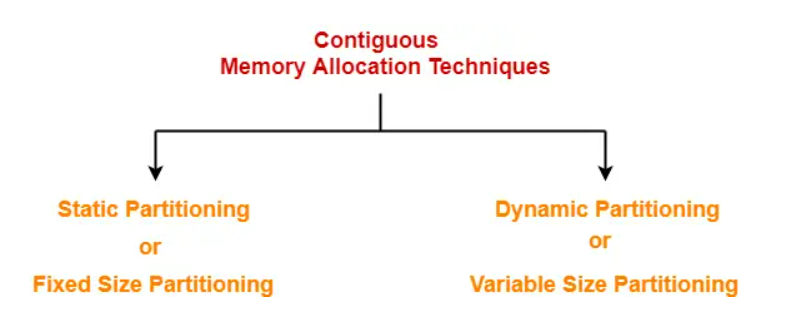
The memory is usually divided into two partitions: one for the resident operating system and one for the user processes. We can place the operating system in either low memory or high memory. Allocating space to software applications is referred to as memory allocation. Contiguous and non-contiguous memory allocation are the two basic types of memory allocation.

In **Contiguous memory allocation** which is a memory management technique, whenever there is a request by the user process for the memory then a single section of the contiguous memory block is given to that process according to its requirement. For example, if a user process needs some x bytes of contiguous memory, then all the x bytes will reside in one place in the memory that is defined by a range of memory addresses: 0x0000 to 0x00FF.

**Types of the Contiguous Memory Allocation Techniques:**

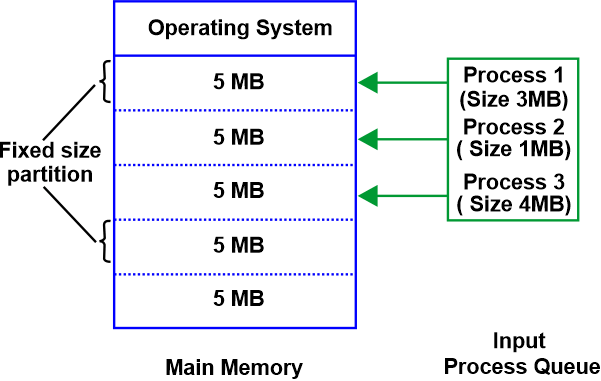
There are mainly two contiguous memory allocation techniques, which can be used to allocate the memory to processes.

* Fixed-size partition scheme
* Variable-size partition scheme

****

## 1)Fixed-size Partition Scheme:

This technique is also known as Static partitioning. In this scheme, the system divides the memory into fixed-size partitions. The size of each partition is fixed as indicated by the name of the technique and it cannot be changed. In this partition scheme, each partition may contain exactly one process. There is a problem that this technique will limit the degree of multiprogramming because the number of partitions will basically decide the number of processes. Whenever any process terminates then the partition becomes available for another process. This method was originally used by the IBM OS/360 operating system (called MFT)but is no longer in use.



## In the above example, we can see that the operating system has the same size partitions of 5 MB each and we have three processes having three different sizes 3 Mb, 1 Mb, and 4 Mb. Process 1 will be assigned to the first partition of 5 MB and process 1 will acquire whole partition of 5 MB. Similarly, Process 2 and Process 3 will acquire partitions two and three respectively. Both processes, Process 2 and Process 3 will acquire an entire partition of size 5 MB.

## Advantages:

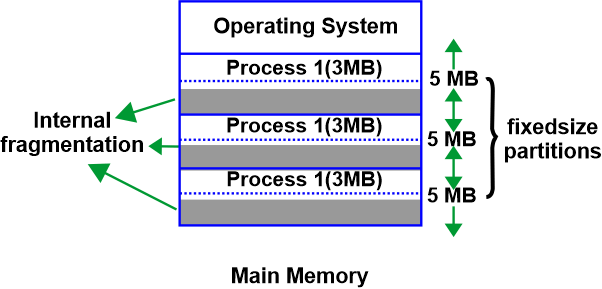
* This strategy is easy to employ because each block is the same size. Now all that is left to do is allocate processes to the fixed memory blocks that have been divided up.
* It is simple to keep track of how many memory blocks are still available, which determines how many further processes can be allocated memory.

## Disadvantages:

## Some disadvantages of using this scheme are as follows:

**1. Internal Fragmentation**

Suppose the size of the process is lesser than the size of the partition in that case some size of the partition gets wasted and remains unused. This wastage inside the memory is generally termed as Internal fragmentation

**2**. **Limitation on the size of the process**

If in a case size of a process is more than that of a maximum-sized partition then that process cannot be loaded into the memory. Due to this, a condition is imposed on the size of the process and it is: the size of the process cannot be larger than the size of the largest partition.

**3. External Fragmentation**

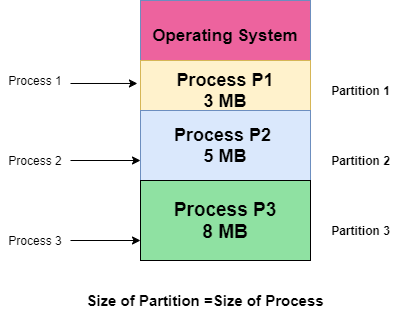
It is another drawback of the fixed-size partition scheme as total unused space by various partitions cannot be used in order to load the processes even though there is the availability of space but it is not in the contiguous fashion.

**4. Degree of multiprogramming is less**

In this partition scheme, as the size of the partition cannot change according to the size of the process. Thus the degree of multiprogramming is very less and is fixed.

## 2)Variable-size Partition Scheme:

This scheme is also known as **Dynamic partitioning** and is came into existence to overcome the drawback i.e internal fragmentation that is caused by **Static partitioning**. In this partitioning, scheme allocation is done dynamically. The size of the partition is not declared initially. Whenever any process arrives, a partition of size equal to the size of the process is created and then allocated to the process. Thus the size of each partition is equal to the size of the process.As partition size varies according to the need of the process so in this partition scheme there is no **internal fragmentation.**



**Advantages:**

Some Advantages of using this partition scheme are as follows:

1. **No Internal Fragmentation**  
   As in this partition scheme space in the main memory is allocated strictly according to the requirement of the process thus there is no chance of internal fragmentation. Also, there will be no unused space left in the partition.
2. **Degree of Multiprogramming is Dynamic**  
   As there is no internal fragmentation in this partition scheme due to which there is no unused space in the memory. Thus more processes can be loaded into the memory at the same time.
3. **No Limitation on the Size of Process**  
   In this partition scheme as the partition is allocated to the process dynamically thus the size of the process cannot be restricted because the partition size is decided according to the process size.

**Disadvantages:**

Some Disadvantages of using this partition scheme are as follows:

1. **External Fragmentation:**  
   As there is no internal fragmentation which is an advantage of using this partition scheme does not mean there will no external fragmentation. Let us understand this with the help of an example: In the above diagram- process P1(3MB) and process P3(8MB) completed their execution. Hence there are two spaces left i.e. 3MB and 8MB. Let’s there is a Process P4 of size 15 MB comes. But the empty space in memory cannot be allocated as no spanning is allowed in contiguous allocation. Because the rule says that process must be continuously present in the main memory in order to get executed. Thus it results in External Fragmentation.
2. **Difficult Implementation:**  
   The implementation of this partition scheme is difficult as compared to the Fixed Partitioning scheme as it involves the allocation of memory at run-time rather than during the system configuration. As we know that OS keeps the track of all the partitions but here allocation and deallocation are done very frequently and partition size will be changed at each time so it will be difficult for the operating system to manage everything.

## Strategies Used for Contiguous Memory Allocation Input Queues:

So far, we've seen the two types of schemes for contiguous memory allocation. But what happens when a new process comes in and has to be allotted a space in the main memory? How is it decided which block or segment it will get?

**Processes that have been assigned continuous blocks of memory will fill the main memory at any given time.** However, when a process completes, it leaves behind an empty block known as a hole. This space could also be used for a new process. Hence, the main memory consists of processes and holes, and any one of these holes can be allotted to a new incoming process. We have three strategies to allot a hole to an incoming process:

# **First Fit:**

The first-fit algorithm searches for the first free partition that is large enough to accommodate the process. The operating system starts searching from the beginning of the memory and allocates the first free partition that is large enough to fit the process.

For example, suppose we have the following memory partitions:

| 10 KB | 20 KB | 15 KB | 25 KB | 30 KB |

Now, a process requests 18 KB of memory. The operating system starts searching from the beginning and finds the first free partition of 20 KB. It allocates the process to that partition and keeps the remaining 2 KB as free memory.

# **Best Fit:**

The best-fit algorithm searches for the smallest free partition that is large enough to accommodate the process. The operating system searches the entire memory and selects the free partition that is closest in size to the process.

For example, suppose we have the following memory partitions:

| 10 KB | 20 KB | 15 KB | 25 KB | 30 KB |

Now, a process requests 18 KB of memory. The operating system searches for the smallest free partition that is larger than 18 KB, and it finds the partition of 20 KB. It allocates the process to that partition and keeps the remaining 2 KB as free memory.

# **Worst Fit:**

The worst-fit algorithm searches for the largest free partition and allocates the process to it. This algorithm is designed to leave the largest possible free partition for future use.

For example, suppose we have the following memory partitions:

| 10 KB | 20 KB | 15 KB | 25 KB | 30 KB |

Now, a process requests 18 KB of memory. The operating system searches for the largest free partition, which is 30 KB. It allocates the process to that partition and keeps the remaining 12 KB as free memory.

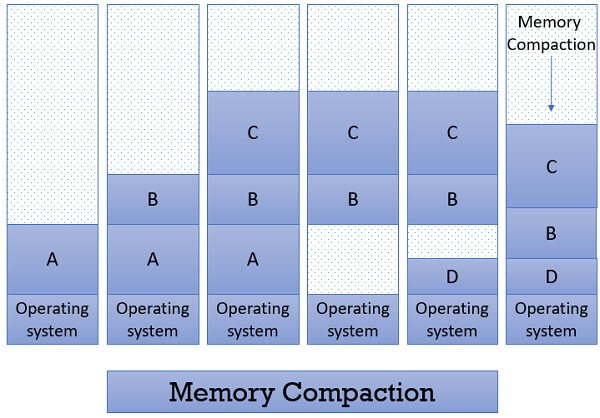
**Fragmentation:**

Fragmentation is an unwanted problem where the memory blocks cannot be allocated to the processes due to their small size and the blocks remain unused. That means, when the processes are loaded and removed from the memory, they create free space or hole in the memory and these small blocks cannot be allocated to new upcoming processes and results in inefficient use of memory. Basically, there are two types of fragmentation:

* Internal Fragmentation
* External Fragmentation

Both the first-fit and best-fit strategies for memory allocation suffer from **external fragmentation**. As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when there is enough total memory space to satisfy a request but the available spaces are not contiguous: storage is fragmented into a large number of small holes. This fragmentation problem can be severe. In the worst case, we could have a block of free (or wasted) memory between every two processes. If all these small pieces of memory were in one big free block instead, we might be able to run several more processes.

One solution to the problem of external fragmentation is compaction. The goal is to shuffle the memory contents so as to place all free memory together in one large block.



Another possible solution to the external-fragmentation problem is to permit the logical address space of the processes to be non-contiguous, thus allowing a process to be allocated physical memory wherever such memory is available.