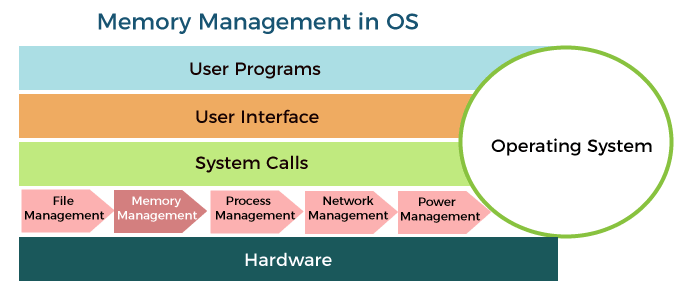
**What do you mean by memory management?**

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. At any time, many processes are competing for it. Moreover, to increase performance, several processes are executed simultaneously. For this, we must keep several processes in the main memory, so it is even more important to manage them effectively.



Memory management plays several roles in a computer system.

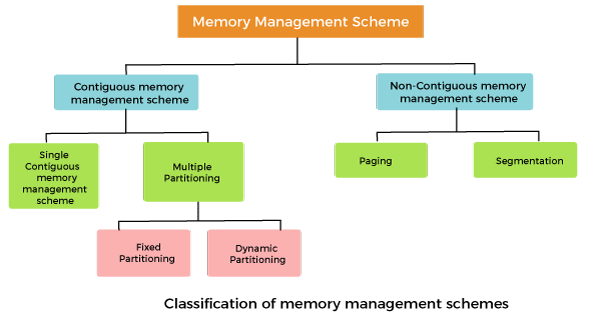
Following are the important roles in a computer system:

* 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.
* The memory manager is responsible for protecting the memory allocated to each process from being corrupted by another process. If this is not ensured, then the system may exhibit unpredictable behavior.
* Memory managers should enable sharing of memory space between processes. Thus, two programs can reside at the same memory location although at different times.

**Memory management Techniques:**

**The Memory management Techniques can be classified into following main categories:**

* Contiguous memory management schemes
* Non-Contiguous memory management schemes



Contiguous memory management schemes:

In a Contiguous memory management scheme, each program occupies a single contiguous block of storage locations, i.e., a set of memory locations with consecutive addresses.

Single contiguous memory management schemes:

The Single contiguous memory management scheme is the simplest memory management scheme used in the earliest generation of computer systems. In this scheme, the main memory is divided into two contiguous areas or partitions. The operating systems reside permanently in one partition, generally at the lower memory, and the user process is loaded into the other partition.

**Advantages of Single contiguous memory management schemes:**

* Simple to implement.
* Easy to manage and design.
* In a Single contiguous memory management scheme, once a process is loaded, it is given full processor's time, and no other processor will interrupt it.

**Disadvantages of Single contiguous memory management schemes:**

* Wastage of memory space due to unused memory as the process is unlikely to use all the available memory space.
* The CPU remains idle, waiting for the disk to load the binary image into the main memory.
* It can not be executed if the program is too large to fit the entire available main memory space.
* It does not support multiprogramming, i.e., it cannot handle multiple programs simultaneously.

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. To switch between two processes, the operating systems need to load both processes into the main memory. The operating system needs to divide the available main memory into multiple parts to load multiple processes into the main memory. Thus multiple processes can reside in the main memory simultaneously.

**The multiple partitioning schemes can be of two types:**

* Fixed Partitioning
* Dynamic Partitioning

Fixed Partitioning

The earliest and one of the simplest technique which can be used to load more than one processes into the main memory is Fixed partitioning or Contiguous memory allocation.

In this technique, the main memory is divided into partitions of equal or different sizes. The operating system always resides in the first partition while the other partitions can be used to store user processes. The memory is assigned to the processes in contiguous way.

In fixed partitioning,

1. The partitions cannot overlap.
2. A process must be contiguously present in a partition for the execution.

There are various cons of using this technique.

**1. Internal Fragmentation**

If the size of the process is lesser then the total size of the partition then some size of the partition get wasted and remain unused. This is wastage of the memory and called internal fragmentation.

As shown in the image below, the 4 MB partition is used to load only 3 MB process and the remaining 1 MB got wasted.

**2. External Fragmentation**

The total unused space of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form.

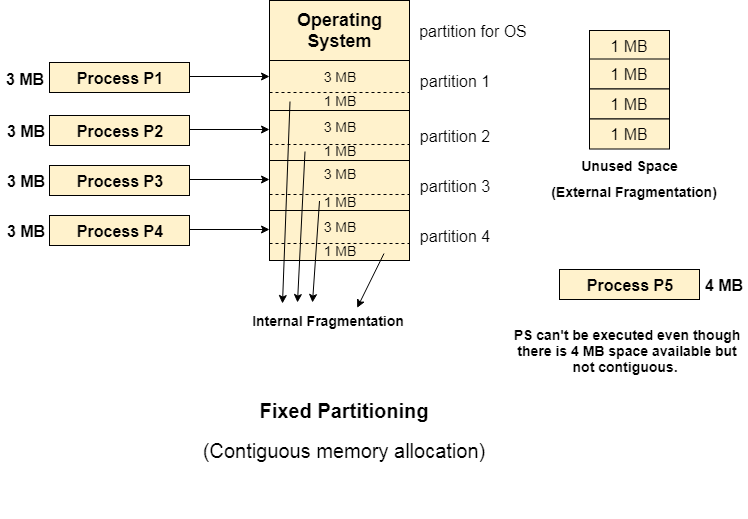
As shown in the image below, the remaining 1 MB space of each partition cannot be used as a unit to store a 4 MB process. Despite of the fact that the sufficient space is available to load the process, process will not be loaded.

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

If the process size is larger than the size of maximum sized partition then that process cannot be loaded into the memory. Therefore, a limitation can be imposed on the process size that is it cannot be larger than the size of the largest partition.

**4. Degree of multiprogramming is less**

By Degree of multi programming, we simply mean the maximum number of processes that can be loaded into the memory at the same time. In fixed partitioning, the degree of multiprogramming is fixed and very less due to the fact that the size of the partition cannot be varied according to the size of processes.



**Advantages of Fixed Partitioning memory management schemes:**

* Simple to implement.
* Easy to manage and design.

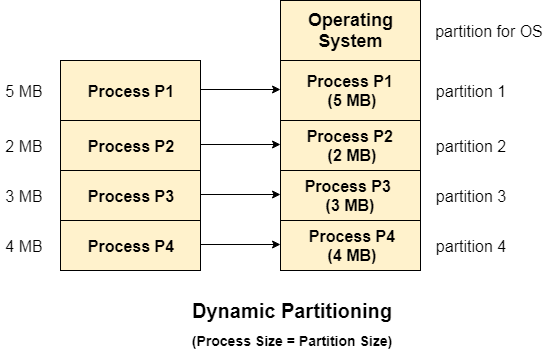
**Disadvantages of Fixed Partitioning memory management schemes:**

* This scheme suffers from internal fragmentation.
* The number of partitions is specified at the time of system generation.

# **Dynamic Partitioning**

Dynamic partitioning tries to overcome the problems caused by fixed partitioning. In this technique, the partition size is not declared initially. It is declared at the time of process loading.

The first partition is reserved for the operating system. The remaining space is divided into parts. The size of each partition will be equal to the size of the process. The partition size varies according to the need of the process so that the internal fragmentation can be avoided.



## **Advantages of Dynamic Partitioning over fixed partitioning**

### 1. No Internal Fragmentation

Given the fact that the partitions in dynamic partitioning are created according to the need of the process, It is clear that there will not be any internal fragmentation because there will not be any unused remaining space in the partition.

### 2. No Limitation on the size of the process

In Fixed partitioning, the process with the size greater than the size of the largest partition could not be executed due to the lack of sufficient contiguous memory. Here, In Dynamic partitioning, the process size can't be restricted since the partition size is decided according to the process size.

### 3. Degree of multiprogramming is dynamic

Due to the absence of internal fragmentation, there will not be any unused space in the partition hence more processes can be loaded in the memory at the same time.

## **Disadvantages of dynamic partitioning**

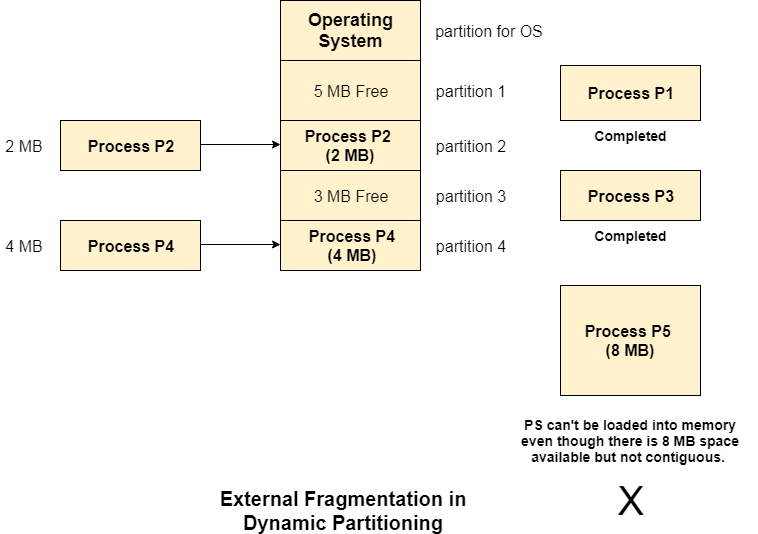
### External Fragmentation

Absence of internal fragmentation doesn't mean that there will not be external fragmentation.

Let's consider three processes P1 (1 MB) and P2 (3 MB) and P3 (1 MB) are being loaded in the respective partitions of the main memory.

After some time P1 and P3 got completed and their assigned space is freed. Now there are two unused partitions (1 MB and 1 MB) available in the main memory but they cannot be used to load a 2 MB process in the memory since they are not contiguously located.

The rule says that the process must be contiguously present in the main memory to get executed. We need to change this rule to avoid external fragmentation.



### Complex Memory Allocation

In Fixed partitioning, the list of partitions is made once and will never change but in dynamic partitioning, the allocation and deallocation is very complex since the partition size will be varied every time when it is assigned to a new process. OS has to keep track of all the partitions.

Due to the fact that the allocation and deallocation are done very frequently in dynamic memory allocation and the partition size will be changed at each time, it is going to be very difficult for OS to manage everything.

**Advantages of Dynamic Partitioning memory management schemes:**

* Simple to implement.
* Easy to manage and design.

**Disadvantages of Dynamic Partitioning memory management schemes:**

* This scheme also suffers from internal fragmentation.
* The number of partitions is specified at the time of system segmentation.

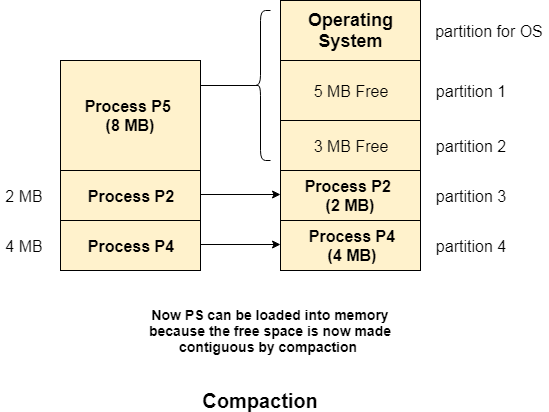
# **Compaction**

We got to know that the dynamic partitioning suffers from external fragmentation. However, this can cause some serious problems.

To avoid compaction, we need to change the rule which says that the process can't be stored in the different places in the memory.

We can also use compaction to minimize the probability of external fragmentation. In compaction, all the free partitions are made contiguous and all the loaded partitions are brought together.

By applying this technique, we can store the bigger processes in the memory. The free partitions are merged which can now be allocated according to the needs of new processes. This technique is also called defragmentation.



As shown in the image above, the process P5, which could not be loaded into the memory due to the lack of contiguous space, can be loaded now in the memory since the free partitions are made contiguous.

### Problem with Compaction

The efficiency of the system is decreased in the case of compaction due to the fact that all the free spaces will be transferred from several places to a single place.

Huge amount of time is invested for this procedure and the CPU will remain idle for all this time. Despite of the fact that the compaction avoids external fragmentation, it makes system inefficient.

Let us consider that OS needs 6 NS to copy 1 byte from one place to another.

1. 1 B transfer needs 6 NS
2. 256 MB transfer needs 256 X 2^20 X 6 X 10 ^ -9 secs

hence, it is proved to some extent that the larger size memory transfer needs some huge amount of time that is in seconds.

# **Bit Map for Dynamic Partitioning**

The Main concern for dynamic partitioning is keeping track of all the free and allocated partitions. However, the Operating system uses following data structures for this task.

1. Bit Map
2. Linked List

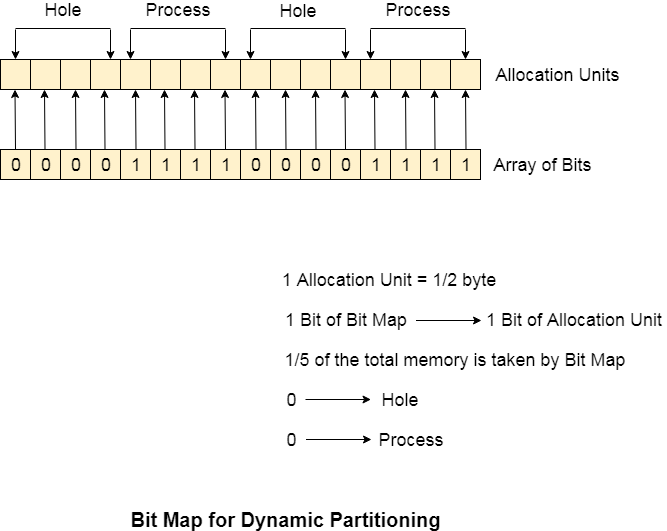
Bit Map is the least famous data structure to store the details. In this scheme, the main memory is divided into the collection of allocation units. One or more allocation units may be allocated to a process according to the need of that process. However, the size of the allocation unit is fixed that is defined by the Operating System and never changed. Although the partition size may vary but the allocation size is fixed.

The main task of the operating system is to keep track of whether the partition is free or filled. For this purpose, the operating system also manages another data structure that is called bitmap.

The process or the hole in Allocation units is represented by a flag bit of bitmap. In the image shown below, a flag bit is defined for every bit of allocation units. However, it is not the general case, it depends on the OS that, for how many bits of the allocation units, it wants to store the flag bit.

The flag bit is set to 1 if there is a contiguously present process at the adjacent bit in allocation unit otherwise it is set to 0.

A string of 0s in the bitmap shows that there is a hole in the relative Allocation unit while the string of 1s represents the process in the relative allocation unit.



## **Disadvantages of using Bitmap**

1. The OS has to assign some memory for bitmap as well since it stores the details about allocation units. That much amount of memory cannot be used to load any process therefore that decreases the degree of multiprogramming as well as throughput.

In the above image,

The allocation unit is of 4 bits that is 0.5 bits. Here, 1 bit of the bitmap is representing 1 bit of allocation unit.

1. Size of 1 allocation unit = 4 bits
2. Size of bitmap = 1/(4+1) = 1/5 of total main memory.

Therefore, in this bitmap configuration, 1/5 of total main memory is wasted.

2. To identify any hole in the memory, the OS need to search the string of 0s in the bitmap. This searching takes a huge amount of time which makes the system inefficient to some extent

# **Linked List for Dynamic Partitioning**

The better and the most popular approach to keep track the free or filled partitions is using Linked List.

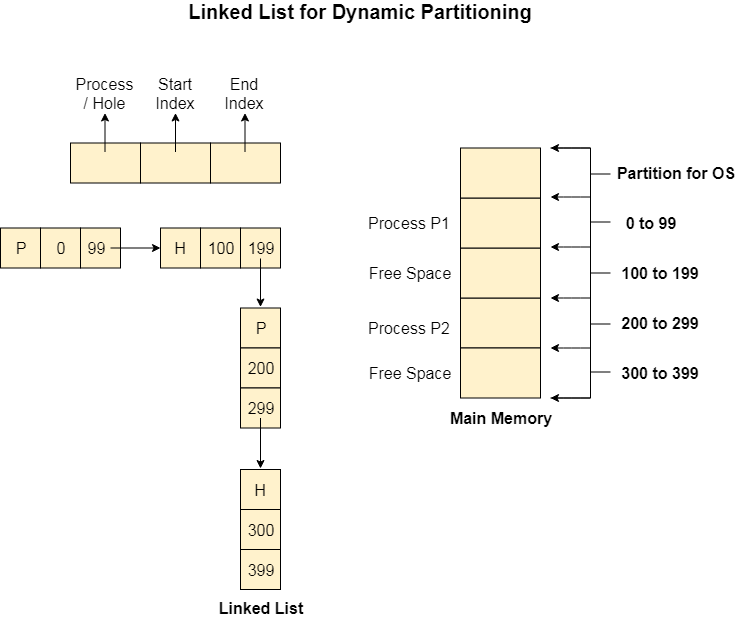
In this approach, the Operating system maintains a linked list where each node represents each partition. Every node has three fields.

1. First field of the node stores a flag bit which shows whether the partition is a hole or some process is inside.
2. Second field stores the starting index of the partition.
3. Third filed stores the end index of the partition.

If a partition is freed at some point of time then that partition will be merged with its adjacent free partition without doing any extra effort.

There are some points which need to be focused while using this approach.

1. The OS must be very clear about the location of the new node which is to be added in the linked list. However, adding the node according to the increasing order of starting index is suggestible.
2. Using a doubly linked list will make some positive effects on the performance due to the fact that a node in the doubly link list can also keep track of its previous node.



# **Partitioning Algorithms**

There are various algorithms which are implemented by the Operating System in order to find out the holes in the linked list and allocate them to the processes.

The explanation about each of the algorithm is given below.

**1. First Fit Algorithm**

First Fit algorithm scans the linked list and whenever it finds the first big enough hole to store a process, it stops scanning and load the process into that hole. This procedure produces two partitions. Out of them, one partition will be a hole while the other partition will store the process.

First Fit algorithm maintains the linked list according to the increasing order of starting index. This is the simplest to implement among all the algorithms and produces bigger holes as compare to the other algorithms.

**2. Next Fit Algorithm**

Next Fit algorithm is similar to First Fit algorithm except the fact that, Next fit scans the linked list from the node where it previously allocated a hole.

Next fit doesn't scan the whole list, it starts scanning the list from the next node. The idea behind the next fit is the fact that the list has been scanned once therefore the probability of finding the hole is larger in the remaining part of the list.

Experiments over the algorithm have shown that the next fit is not better then the first fit. So it is not being used these days in most of the cases.

**3. Best Fit Algorithm**

The Best Fit algorithm tries to find out the smallest hole possible in the list that can accommodate the size requirement of the process.

Using Best Fit has some disadvantages.

1. 1. It is slower because it scans the entire list every time and tries to find out the smallest hole which can satisfy the requirement the process.
2. Due to the fact that the difference between the whole size and the process size is very small, the holes produced will be as small as it cannot be used to load any process and therefore it remains useless.  
   Despite of the fact that the name of the algorithm is best fit, It is not the best algorithm among all.

**4. Worst Fit Algorithm**

The worst fit algorithm scans the entire list every time and tries to find out the biggest hole in the list which can fulfill the requirement of the process.

Despite of the fact that this algorithm produces the larger holes to load the other processes, this is not the better approach due to the fact that it is slower because it searches the entire list every time again and again.

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. This scheme can be classified depending upon the size of blocks and whether the blocks reside in the main memory or not.