# Partitioning in Operating System

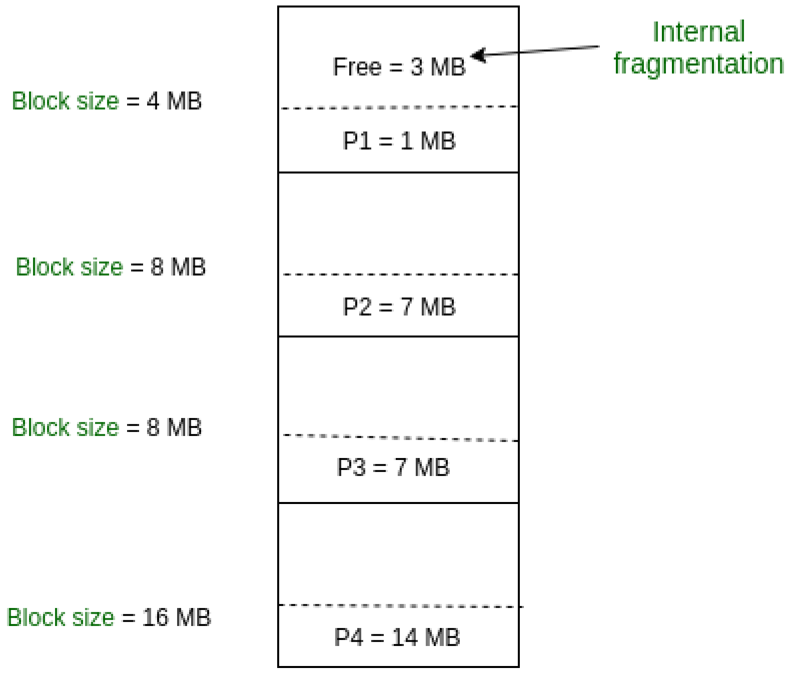
## Fixed (or static) Partitioning

In operating systems, Memory Management is the function responsible for allocating and managing 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**, and **Non-Contiguous**. In Contiguous Technique, executing process must be loaded entirely in main-memory. Contiguous Technique can be divided into:

1. Fixed (or static) partitioning
2. Variable (or dynamic) partitioning

**Fixed Partitioning:**
This is the oldest and simplest technique used to put more than one processes in the main memory. In this partitioning, number of partitions (non-overlapping) in RAM are **fixed but size** of each partition may or **may not be same**. As it is **contiguous** allocation, hence no spanning is allowed. Here partition 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 inspite 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 –**

1. **Easy to implement:**
   Algorithms needed to implement Fixed Partitioning are easy to implement. It simply requires putting a process into certain partition without focussing on the emergence of Internal and External Fragmentation.
2. **Little OS overhead:**
   Processing of Fixed Partitioning require lesser excess and indirect computational power.

**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 size of partition in Main Memory cannot be accommodated. Partition size cannot be varied according to the size of incoming process's size. Hence, process size of 32MB in above stated example is invalid.

**Limitation on Degree of Multiprogramming:**
Partition in Main Memory are made before execution or during system configure.
Main Memory is divided into fixed number of partition. Suppose if there are $n1$ parti-
tions in RAM and $n2$ are the number of processes, then $n2 <= n1$ condition
must be fulfilled. Number of processes greater than number of partitions in RAM is
invalid in Fixed Partitioning.

-

## Variable (or dynamic) Partitioning

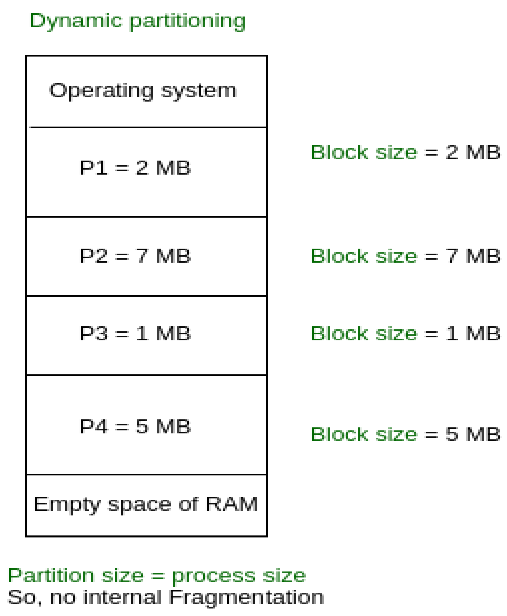
In operating systems, Memory Management is the function responsible for allocating and managing computer's main memory. Memory Management function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory.

There are two Memory Management Techniques: **Contiguous**, and **Non-Contiguous**. In Contiguous Technique, executing process must be loaded entirely in main-memory. Contiguous Technique can be divided into:

1. Fixed (or static) partitioning
2. Variable (or dynamic) partitioning

**Variable Partitioning –**
It is a part of 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 configure. Various **features** associated with variable Partitioning-

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



There are some advantages and disadvantages of variable partitioning over fixed partitioning as given below.

**Advantages of Variable Partitioning –**

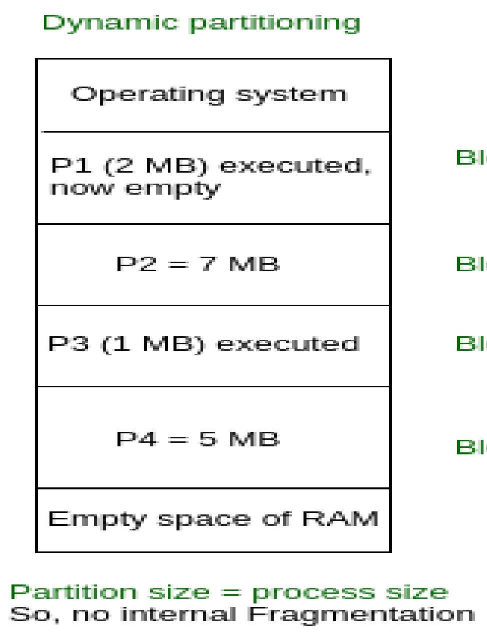
1. **No Internal Fragmentation:**
   In variable Partitioning, space in main memory is allocated strictly according to the need of process, hence there is no case of internal fragmentation. There will be no unused space left in the partition.
2. **No restriction on Degree of Multiprogramming:**
   More number of processes can be accommodated due to absence of internal fragmentation. A process can be loaded until the memory is empty.

3. **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 loaded and process can not be divided as it is invalid in 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 –**

1. **Difficult Implementation:**
Implementing variable Partitioning is difficult as compared to Fixed Partitioning as it involves allocation of memory during run-time rather than during system configure.
2. **External Fragmentation:**
There will be external fragmentation inspite of absence of internal fragmentation.

   For example, suppose in 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 empty space in memory cannot be allocated as no spanning is allowed in contiguous allocation. The rule says that process must be contiguously present in main memory to get executed. Hence it results in External Fragmentation.



   Now P5 of size 3 MB cannot be accommodated in spite of required available space because in contiguous no spanning is allowed.

## Memory allocation strategies

1. First Fit
2. Best fit
3. Worst fit
4. Buddy's system
5. Next fit

**Answer:**

### First Fit

In the first fit approach is to allocate the first free partition or hole large enough which can accommodate the process. It finishes after finding the first suitable free partition.

#### Advantage

Fastest algorithm because it searches as little as possible.

#### Disadvantage

The remaining unused memory areas left after allocation become waste if it is too smaller. Thus request for larger memory requirement cannot be accomplished.

### Best Fit

The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole which is close to actual process size needed.

#### Advantage

Memory utilization is much better than first fit as it searches the smallest free partition first available.

#### Disadvantage

It is slower and may even tend to fill up memory with tiny useless holes.

### Worst fit

In worst fit approach is to locate largest available free portion so that the portion left will be big enough to be useful. It is the reverse of best fit.

#### Advantage

Reduces the rate of production of small gaps.

#### Disadvantage

If a process requiring larger memory arrives at a later stage then it cannot be accommodated as the largest hole is already split and occupied.

## Buddy's System

In buddy system, sizes of free blocks are in form of integral power of 2. E.g. 2, 4, 8, 16 etc. Up to the size of memory. When a free block of size 2k is requested, a free block from the list of free blocks of size 2k is allocated. If no free block of size 2k is available, the block of next larger size, 2k+1 is split in two halves called buddies to satisfy the request.

### Example

Let total memory size be 512KB and let a process P1, requires 70KB to be swapped in. As the hole lists are only for powers of 2, 128KB will be big enough. Initially no 128KB is there, nor are blocks 256KB. Thus 512KB block is split into two buddies of 256KB each, one is further split into two 128KB blocks and one of them is allocated to the process. Next P2 requires 35KB. Rounding 35KB up to a power of 2, a 64KB block is required.

So when 128KB block is split into two 64KB buddies. Again a process P3(130KB) will be adjusted in the whole 256KB. After satisfying the request in this way when such block is free, the two blocks/buddies can be recombined to form the twice larger original block when it is second half buddy is also free.

### Advantage

Buddy system is faster. When a block of size 2k is freed, a hole of 2k memory size is searched to check if a merge is possible, whereas in other algorithms all the hole list must be searched.

### Disadvantage

It is often become inefficient in terms of memory utilization. As all requests must be rounded up to a power of 2, a 35KB process is allocated to 64KB, thus wasting extra 29KB causing internal fragmentation. There may be holes between the buddies causing external fragmentation.

## Next fit

Next fit is a modified version of first fit. It begins as first fit to find a free partition. When called next time it starts searching from where it left off, not from the beginning.