**Memory partitioning** means dividing the main memory into chunks of the same or different sizes so that they can be assigned to processes in the main memory.

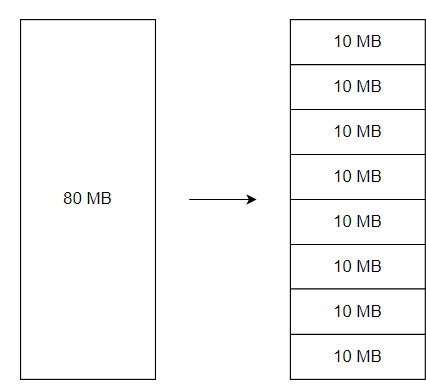
There are two types of memory partitioning techniques:

* Fixed-sized memory partitioning
* Variable-sized memory partitioning

**Fixed-sized memory partitioning**

In **fixed-sized** memory partitioning, the main memory is divided into blocks of the same or different sizes. Fixed-size memory partitioning can take place before executing any processes or during the configuration of the system.

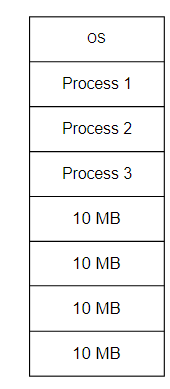
Consider an example of a main memory of 80MB. Let’s suppose that the main memory is divided into blocks of 10MB as shown below:



The operating system is assigned the first block in the memory.

Let’s say we bring 3 processes, *process 1*, *process 2*, and *process 3*, from hard disk to main memory. *Process 1*, *process 2*, and *process 3* are of sizes 6MB, 8MB, and 5MB respectively.

To assign the processes in the main memory, the operating system will try to find a single block that is large enough to store the process. In other words, the operating system will find a block whose size is greater than or equal to the size of the process. After loading the three processes into the memory, the memory will look like this:



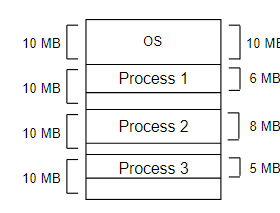
**Advantages**

* As the partitions of the main memory are pre-defined before execution, there is very little OS overhead. This is because the OS does not have to partition the memory before allocating it to processes.
* As the fixed-sized memory partitioning technique is very simple, it is easy to implement.

**Disadvantages**

**Internal fragmentation**

Consider the example of three processes, *process 1*, *process 2*, and *process 3*, of sizes 6MB, 8MB, and 5MB respectively. The processes will be assigned to a memory of 40MB divided into blocks of 10MB, as shown below:

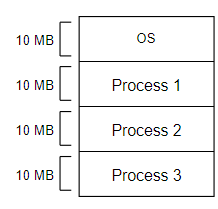


Notice that the memory blocks are not utilized completely. *Process 1* consumes 6MB and is assigned to a block of size 10MB; this means that 4MB memory remains unutilized. Similarly, 2MB and 5MB memory is unutilized by 10MB memory blocks assigned to *process 2* and *process 3*. Thus, out of 40MB of main memory, 4+2+5=11MB of main memory remains unutilized.

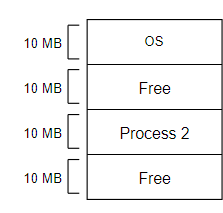
Let’s suppose we try to load another process, *process 4*, of size 6MB in the memory. Although we have 11MB of free memory space, the process cannot be loaded into the memory. This is because we need a contiguous block of 6MB to load *process 4*.

**External fragmentation**

Consider the example of three processes, *process 1*, *process 2*, and *process 3*, all of size 10MB. The processes will be assigned to a memory of 40MB divided into blocks of 10MB as shown below:



Now consider that *process 1* and *process 3* have freed the memory blocks because they have completed their execution. After removing *process 1* and *process 3*, the main memory explained above will look like this:



Let’s suppose we try to load another process, *process 4*, of size 11MB in the memory. Although we have 20MB of free memory space, the process cannot be loaded into the memory. This is because we need a contiguous block of 11MB to load *process 4*.

**Degree of concurrency**

The **degree of concurrency** means the maximum number of processes that can be loaded into the main memory at the same time.

The fixed-sized partitioning limits the degree of concurrency. The maximum number of processes that can be loaded into the memory at the same time is equal to the number of fixed-sized partitions of the main memory.

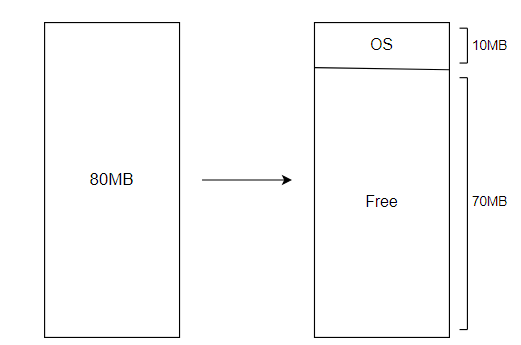
**Limitation on maximum process size**

The fixed-sized partitioning limits the maximum size of processes that are loaded into memory. Even if we have a main memory of 4GB, we cannot load a process of even 10MB if the maximum block size we have is 5MB.

**Variable-sized memory partitioning**

In **variable-sized memory partitioning**, the main memory is divided into blocks of the same or different sizes. Variable-sized memory partitioning takes place at run-time when a process asks for a block of the main memory. If enough main memory is available, the process is assigned a block of the main memory of exactly the same size that is required.

Consider an example of a main memory of 80MB. In the beginning, the memory will contain only the operating system. Let’s say that the operating system consumes 10B of the main memory; the main memory will look like this:

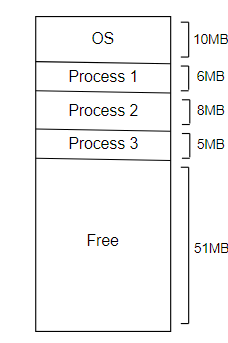


Let’s say we bring three processes, *process 1*, *process 2*, and *process 3*, from hard disk to main memory. *Process 1*, *process 2*, and *process 3* are of sizes 6MB, 8MB, and 5MB respectively.

To assign the processes in main memory, the operating system will try to find a single block that is large enough to store the process. In other words, the operating system will find a block whose size is greater than or equal to the size of the process.

If the block is exactly the same size as the process, it will be assigned to the process. If the size of the block is greater than the size of the process, the block will be broken down into two blocks, one equal in size to the block, and the other being *size = size of the block before breaking - the size of the process*.

After breaking down, the process will be assigned the first block that was of equal size to the process.﻿ After loading the three processes into the memory, the memory will look like this:



**Advantages**

**No internal fragmentation**

**Internal fragmentation** occurs when a process is assigned a block of greater size than that of the process.﻿ In variable-sized memory partitioning, there is no internal fragmentation.

**Degree of concurrency**

The **degree of concurrency** means the maximum number of processes that can be loaded into the main memory at the same time.

The variable-sized partitioning does not limit the degree of concurrency. This is because any number of processes can be loaded into the memory as long as there is a contiguous memory block available that is large enough to load the process.

**No limitation on maximum process size**

In *fixed-sized memory partitioning*, a process of greater size than that of the maximum memory block cannot be loaded into the memory. This limits the maximum process size that can be loaded into the memory.

In *variable-sized partitioning*, there is no limit of the maximum size of processes that can be loaded into memory. We can load a process of any size into the memory as long as there is a free block large enough to load the process.

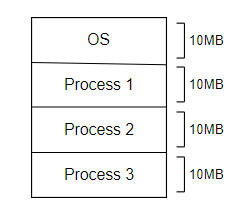
**Disadvantages**

**Difficult to implement**

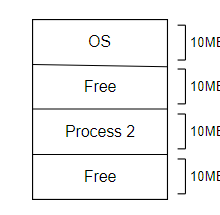
As main memory is partitioned at run time to assign the blocks of the exact size as required by the processes, variable-sized memory partitioning is difficult to implement.

**External fragmentation**

Consider an example of three processes, *process 1*, *process 2*, and *process 3*, all of size 10MB. The processes will be assigned to a memory of 40MB divided into blocks of 10MB, as shown below:



Now consider that *process 1* and *process 3* have freed the memory blocks because they have completed their execution. After removing *process 1* and *process 3*, the main memory explained above will look like this:



Let’s suppose we try to load another process, *process 4*, of size 11MB in the memory. Although we have 20MB of free memory space, the process cannot be loaded into the memory. This is because we need a contiguous block of 11MB to load *process 4*.