Fragmentation in Operating System

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused.

Contiguous memory allocation allocates space to processes whenever the processes enter **RAM**. These **RAM** spaces are divided either by fixed partitioning or by dynamic partitioning. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to coming processes.

In this article, you will learn about fragmentation and its types.

What is Fragmentation?

Fragmentation is an unwanted problem in the operating system in which the processes are loaded and unloaded from memory, and free memory space is fragmented. Processes can't be assigned to memory blocks due to their small size, and the memory blocks stay unused. It is also necessary to understand that as programs are loaded and deleted from memory, they generate free space or a hole in the memory. These small blocks cannot be allotted to new arriving processes, resulting in inefficient memory use.

The conditions of fragmentation depend on the memory allocation system. As the process is loaded and unloaded from memory, these areas are fragmented into small pieces of memory that cannot be allocated to incoming processes. It is called **fragmentation**.

Causes of Fragmentation

User processes are loaded and unloaded from the main memory, and processes are kept in memory blocks in the main memory. Many spaces remain after process loading and swapping that another process cannot load due to their size. Main memory is available, but its space is insufficient to load another process because of the dynamical allocation of main memory processes.

Types of Fragmentation

There are mainly two types of fragmentation in the operating system. These are as follows:

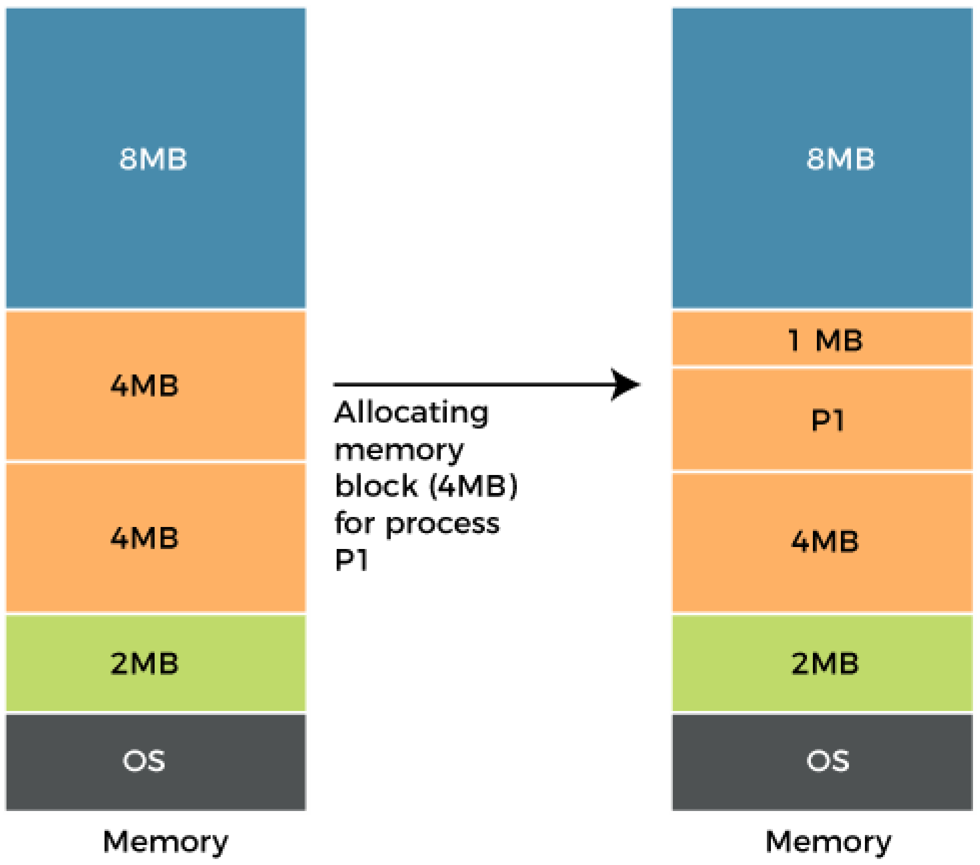
1. **Internal Fragmentation**
2. **External Fragmentation**

Internal Fragmentation

When a process is allocated to a memory block, and if the process is smaller than the amount of memory requested, a free space is created in the given memory block. Due to this, the free space of the memory block is unused, which causes **internal** fragmentation.

For Example:

Assume that memory allocation in RAM is done using fixed partitioning (i.e., memory blocks of fixed sizes). **2MB, 4MB, 4MB**, and **8MB** are the available sizes. The Operating System uses a part of this RAM.



Let's suppose a process **P1** with a size of **3MB** arrives and is given a memory block of **4MB**. As a result, the **1MB** of free space in this block is unused and cannot be used to allocate memory to another process. It is known as **internal fragmentation**.

How to avoid internal fragmentation?

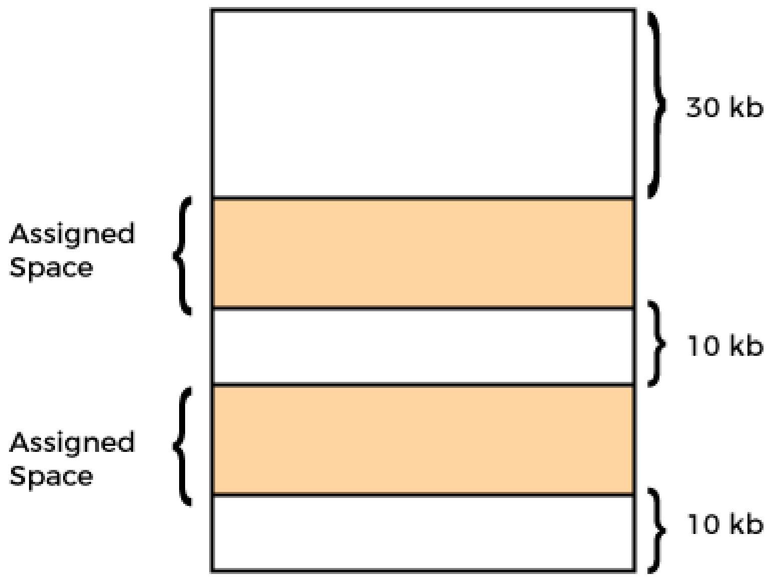
The problem of internal fragmentation may arise due to the fixed sizes of the memory blocks. It may be solved by assigning space to the process via dynamic partitioning.

Dynamic partitioning allocates only the amount of space requested by the process. As a result, there is no internal fragmentation.

External Fragmentation

External fragmentation happens when a dynamic memory allocation method allocates some memory but leaves a small amount of memory unusable. The quantity of available memory is substantially reduced if there is too much external fragmentation. There is enough memory space to complete a request, but it is not contiguous. It's known as **external** fragmentation.

For Example:



Process 05 needs 45kb memory space

Let's take the example of external fragmentation. In the above diagram, you can see that there is sufficient space **(50 KB)** to run a process **(05) (need 45KB)**, but the memory is not contiguous. You can use compaction, paging, and segmentation to use the free space to execute a process.

How to remove external fragmentation?

This problem occurs when you allocate RAM to processes continuously. It is done in paging and segmentation, where memory is allocated to processes non-contiguously. As a result, if you remove this condition, external fragmentation may be decreased.

Compaction is another method for removing external fragmentation. External fragmentation may be decreased when dynamic partitioning is used for memory allocation by combining all free memory into a single large block. The larger memory block is used to allocate space based on the requirements of the new processes. This method is also known as defragmentation.

Advantages and disadvantages of fragmentation

There are various advantages and disadvantages of fragmentation. Some of them are as follows:

Advantages

There are various advantages of fragmentation. Some of them are as follows:

Fast Data Writes

Data write in a system that supports data fragmentation may be faster than reorganizing data storage to enable contiguous data writes.

Fewer Failures

If there is insufficient sequential space in a system that does not support fragmentation, the write will fail.

Storage Optimization

A fragmented system might potentially make better use of a storage device by utilizing every available storage block.

Disadvantages

There are various disadvantages of fragmentation. Some of them are as follows:

Need for regular defragmentation

A more fragmented storage device's performance will degrade with time, necessitating the requirement for time-consuming defragmentation operations.

Slower Read Times

The time it takes to read a non-sequential file might increase as a storage device becomes more fragmented.

Conclusion

In short, both internal and external fragmentation are natural processes that result in either memory wasting or empty memory space. However, the problems in both cases cannot be completely overcome, although they can be reduced to some extent using the solutions provided above.

Internal vs. External Fragmentation

What is Fragmentation?

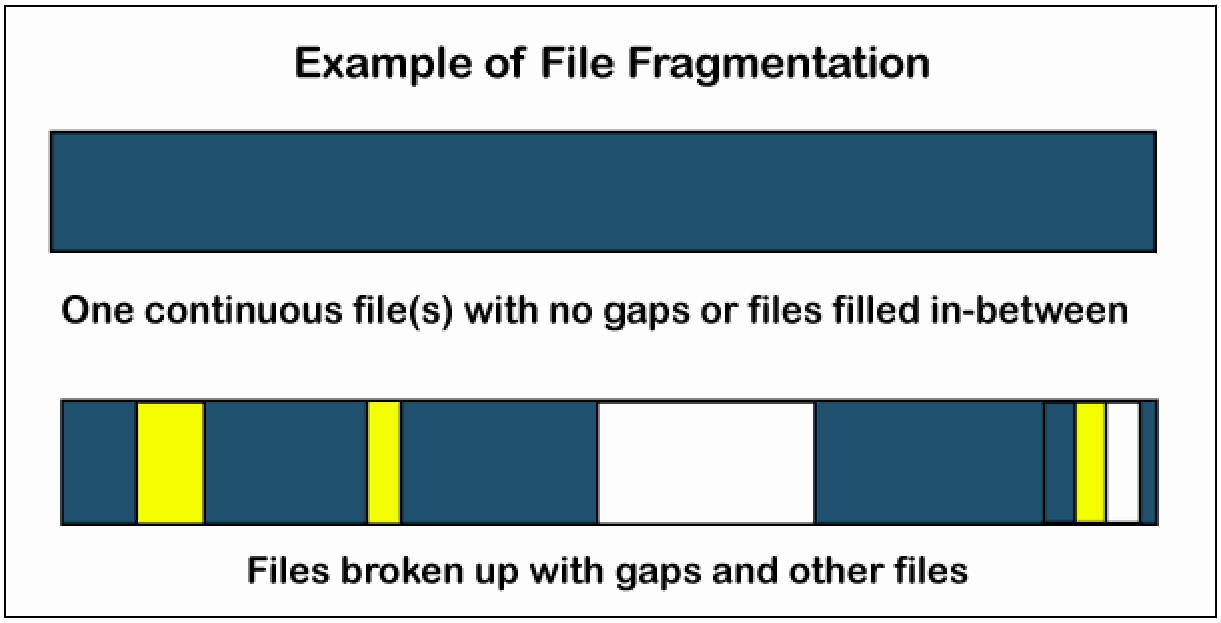
"Fragmentation is a process of data storage in which memory space is used inadequately, decreasing ability or efficiency and sometimes both." The precise implications of fragmentation depend on the specific storage space allocation scheme in operation and the particular fragmentation type. In certain instances, fragmentation contributes to "unused" storage capacity, and the concept also applies to the unusable space generated in that situation. The memory used to preserve the data set (- for example file format) is similar for other systems (- for example, the FAT file system), regardless of the amount of fragmentation (from null to the extreme).

There are three distinct fragmentation kinds: internal fragmentation, external fragmentation, and data fragmentation that can exist beside or a combination. In preference for enhancements, inefficiency, or usability, fragmentation is often acknowledged. For other tools, such as processors, similar things happen.

The Fundamental concept of Fragmentation

When a computer program demands fragments of storage from the operating system (OS), the elements are assigned in chunks. When a chunk of the software program is completed, it can be released back to the system, making it ready to be transferred to the next or the similar program again afterward. Software differs in the size and duration of time a chunk is kept by it. A computer program can demand and release several blocks of storage throughout its lifetime.

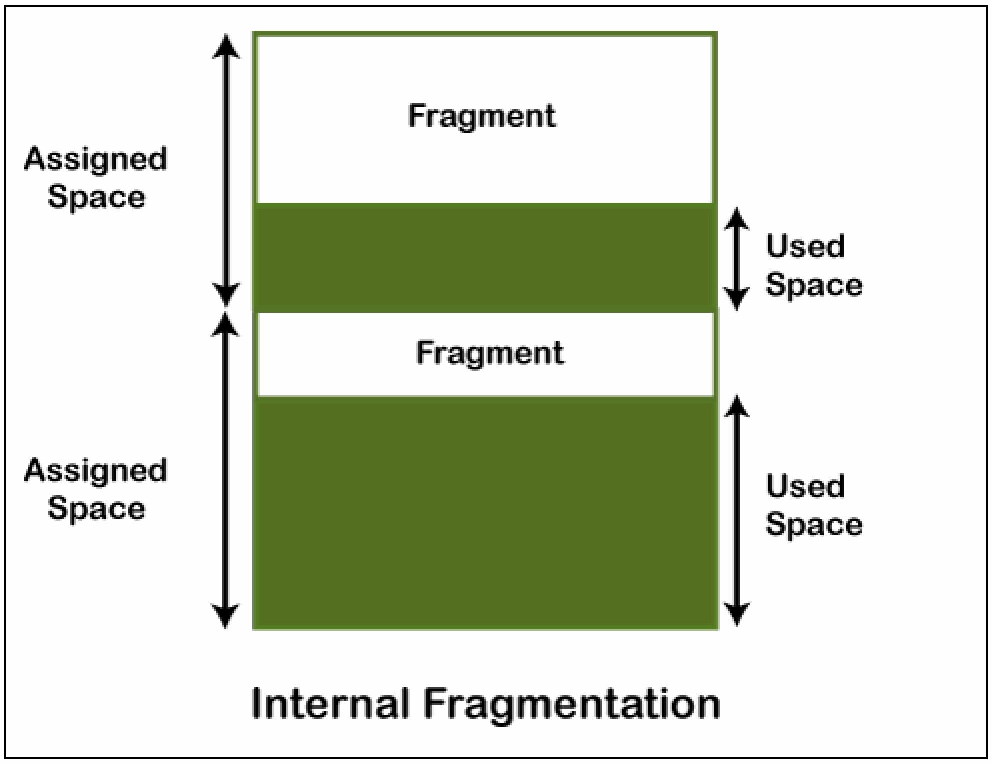
The unused memory sections are large and continuous when a system is initiated. The large continuous sectors become fragmented into a smaller part of the regions through time and utilization. Ultimately, accessing large contiguous blocks of storage can become difficult for the system.


Example of File Fragmentation
One continuous file(s) with no gaps or files filled in-between
Files broken up with gaps and other files

Internal Fragmentation

Most memory space is often reserved than is required to adhere to the restrictions regulating storage space. For instance, memory can only be supplied in blocks (multiple of 4) to systems, and as an outcome, if a program demands maybe 29 bytes, it will get a coalition of 32 bytes. The surplus storage goes to waste when this occurs. The useless space is found inside an assigned area in this case. This structure, called fixed segments, struggles from excessive memory-any process consumes an enormous chunk, no matter how insignificant. Internal fragmentation is what this garbage is termed. Unlike many other forms of fragmentation, it is impossible to restore inner fragmentation, typically, the only way to eliminate it is with a new design.

For instance, in dynamic storage allocation, storage reservoirs reduce internal fragmentation significantly by extending the space overhead over a more significant number of elements.



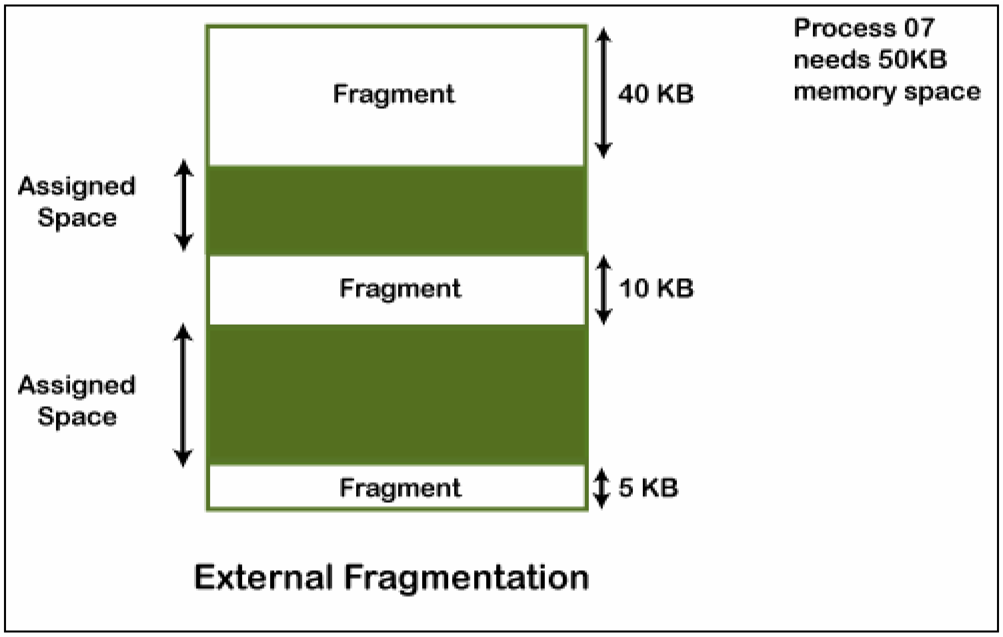
The figure mentioned above demonstrates internal fragmentation because internal fragmentation is considered the distinction between the assigned storage space and the needed space or memory.

External Fragmentation

When used storage is differentiated into smaller lots and is punctuated by assigned memory space, external fragmentation occurs. It is a weak point of many storage allocation methodologies when they cannot effectively schedule memory used by systems. The consequence is that, while unused storage is available, it is essentially inaccessible since it is separately split into fragments that are too limited to meet the software's requirements. The word "external" derives from the fact that the inaccessible space is stored outside the assigned regions.

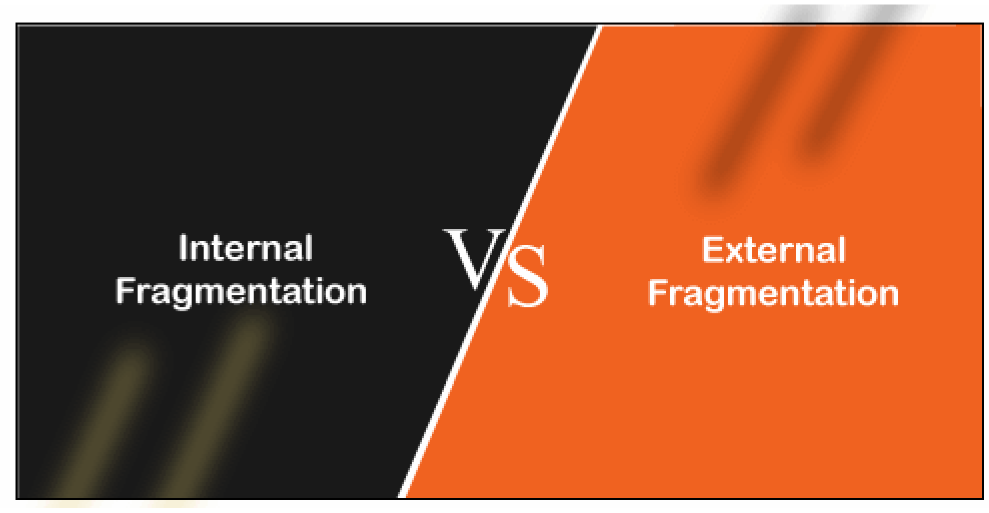
Consider, for instance, a scenario in which a system assigns three consecutive memory blocks and then relieves the middle block. The memory allocator can use this unused allocation of the storage for future assignments. Fortunately, if the storage to be reserved is more generous in size than this available region, it will not use this component.

In data files, external fragmentation often exists when several files of various sizes are formed, resized, and discarded. If a broken document into several small chunks is removed, the impact is much worse since this retains equally small free space sections.



You can see in the figure mentioned above that there is sufficient memory space (55 KB) to execute a process-07 (50 KB mandated), but the storage (fragment) is not adjacent. Here, to use the empty room to run a procedure, you can use compression, paging, or segmentation strategies.

Internal fragmentation vs. External fragmentation



Here, the differences between Internal and External Fragmentation are discussed below in the tabular format.

Sr. No.	Internal Fragmentation	External Fragmentation
1.	Frames square measure designated for processing in internal fragmentation of fixed-sized storage.	Variable-sized memory frames square measure designated to the process during external fragmentation.

2.	When the system or procedure is greater than the storage, internal fragmentation occurs.	Whenever the system or procedure is withdrawn, external fragmentation occurs.
3.	The internal fragmentation approach is the frame with the perfect match.	Compression, paging, and differentiation are alternatives to external fragmentation.
4.	Internal fragmentation happens whenever the storage is split into fragments of a fixed length.	External fragmentation happens whenever the storage is split into segments of variable size depending on the process length.
5.	The distinction between the assigned memory and the storage or memory needed is considered as internal fragmentation.	The empty spaces created among non-contiguous pieces of storage are too tiny for a new system to operate, considered as external fragmentation.