II Contiguous Memory Allocation

- In this allocation type, the consecutive blocks of memory is allocated to a file/process.
- It executes quickly in comparison to non-contiguous memory.
- It is easy to be controlled by the operating system.
- Minimum overhead since there are not many address translation when a process is being executed.
- There is internal fragmentation in contiguous memory allocation.
- There are different types of partitions: Single partition allocation and multi-partition allocation.
- Memory gets wasted.
- The swapped-in process is arranged in the originally allocated space itself.

Non-contiguous Memory Allocation

- In this type of allocation, separate blocks of memory are allocated to a file/process.
- It executes slowly in comparison to contiguous memory.
- It is difficult to be controlled by the operating system.
- More overhead since it needs to translate multiple addresses.
- It includes paging.
- It includes segmentation.
- No memory is wasted.
- External fragmentation occurs in this type of allocation.
- The swapped-in processes can be arranged in any location in the memory.

Contiguous memory allocation is a memory allocation method that allocates a **single contiguous section of memory** to a process or a file. This method takes into account the size of the file or a process and also estimates the maximum size, up to what the file or process can grow?

Taking into account the future growth of the file and its request for memory, the operating system allocates sufficient contiguous memory blocks to that file. Considering this future expansion and the file's request for memory, the operating system will allocate those many contiguous blocks of memory to that file.

In this section, we will discuss contiguous memory allocation in detail along with its advantages

and disadvantages. So let us star

Contiguous Memory Allocation

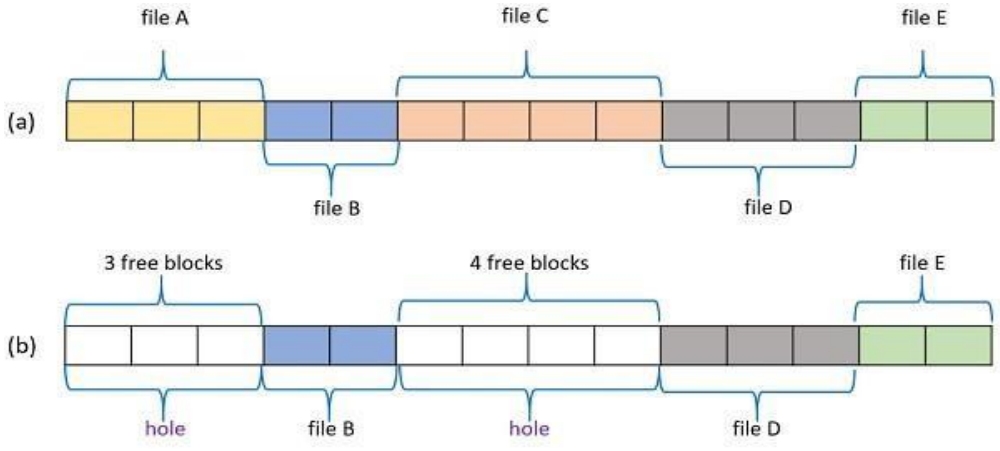
1. Memory Allocation
2. Memory Management
3. Fragmentation
4. Advantages and Disadvantages
5. Key Takeaways

1. Memory Allocation

The main memory has to accommodate both the **operating system** and **userspace**. Now, here the userspace has to accommodate various user processes. We also want these several user processes must reside in the main memory at the same time.

Now, the question arises how to allocate the available memory space to the user processes that are waiting in a ready queue?

In Contiguous memory allocation, when the process arrives from the ready queue to the main memory for execution, the contiguous memory blocks are allocated to the process according to its requirement. Now, to allocate the **contiguous** space to user processes, the memory can be divide either in the fixed-sized partition or in the variable-sized partition.



(a) Contiguous memory allocation of 5 files
(b) When the file A and C terminates and release the memory creating hole

Fixed-Sized Partition: In the fixed-sized partition, the memory is divided into fixed-sized blocks and each block contains exactly one process. But, the fixed-sized partition will limit the degree of multiprogramming as the number of the partition will decide the number of processes.

Variable-Size Partition: In the variable size partition method, the operating system maintains a table that contains the information about all memory parts that are **occupied** by the processes and all memory parts that are still **available** for the processes.

How holes are created in the memory?

Initially, the whole memory space is available for the user processes as a large block, a **hole**. Eventually, when the processes arrive in the memory, executes, terminates and leaves the memory you will see the set of holes of variable sizes.

In the figure above, you can see that when file A and file C release the memory allocated to them, creates the holes in the memory of variable size.

In the **variable size partition** method, the operating system analyses the memory requirement of the process and see whether it has a memory block of the required size. If it finds the match, then it allocates that memory block to the process. If not, then it searches the ready queue for the process that has a smaller memory requirement.

The operating system allocates the memory to the process until it cannot satisfy the memory requirement of the next process in the ready queue. It stops allocating memory to the process if it does not have a memory block (**hole**) that is large enough to hold that process.

If the memory block (**hole**) is too **large** for the process it gets **spilt** into two parts. One part of the memory block is allocated to the arrived process and the other part is returned to the set of holes. When a process terminates and releases the memory allocated to it, the released memory is then placed back to the set of holes. The **two holes that are adjacent** to each other, in the set of holes, are merged to form one **large hole**.

Now, at this point, the operating system checks whether this newly formed free large hole is able to satisfy the memory requirement of any process waiting in the ready queue.

Memory Management

In the above section, we have seen, how the operating system allocates the contiguous memory to the processes. Here, we will see how the operating system selects a free hole from the set of holes?

The operating system uses either the block allocation list or the bit map to select the hole from the set of

holes.

BLOCK ALLOCATION LIST

Block allocation list maintains two tables. One table contains the entries of the blocks that are allocated to the various files. The other table contains the entries of the holes that are free and can be allocated to the process in the waiting queue.

Now, as we have entries of free blocks which one must be chosen can be decided using either of these strategies: first-fit, best-fit, worst-fit strategies.

2. *First-fit*

Here, the searching starts either at the beginning of the table or from where the previous first-fit search has ended. While searching, the first hole that is found to be large enough for a process to accommodate is selected.

3. *Best-fit*

This method needs the list of free holes to be sorted according to their size. Then the smallest hole that is large enough for the process to accommodate is selected from the list of free holes. This strategy reduces the wastage of memory as it does not allocate a hole of larger size which leaves some amount of memory even after the process accommodates the space.

4. *Worst-fit*

This method requires the entire list of free holes to be sorted. Here, again the largest hole among the free holes is selected. This strategy leaves the largest leftover hole which may be useful for the other process.

Bit Map

The bit map method only keeps track of the free or allocated block. One block is represented by one bit, bit 0 resembles the free block and bit 1 resembles that the block is allocated to a file or a process.

It does not have entries of the files or processes to which the specific blocks are allocated.

FRAGMENTATION:

Fragmentation can either be external fragmentation or internal fragmentation. **External fragmentation** is when the free memory blocks available in memory are too small and even non-contiguous. Whereas, the **internal fragmentation** occurs when the process does not fully utilize the memory allocated to it.

The solution to the problem of external fragmentation is **memory compaction**. Here, all the memory contents are shuffled to the one area of memory thereby creating a large block of memory which can be allocated to one or more new processes.

Key Takeaways

- Memory allocation can be done either by a fixed-sized partition scheme or by variable-sized partition scheme.
- Block allocation list has three methods to select a hole from the list of free holes first-fit, best-fit and worse-fit.
- Bit map keeps track of free blocks in memory, it has one bit for one memory block, bit 0 shows that the block is free and bit 1 shows the block is allocated to some file or a process.
- Contiguous memory allocation leads to fragmentation. Further fragmentation can either be external or internal.
- Contiguous memory allocation leads to memory wastage and inflexibility.
- If the operating system uses buffered I/O during processing, then contiguous memory allocation can enhance processing speed.

his is all about contiguous memory allocation. We have discussed how the memory is allocated, managed. How does mentation occur in contiguous