**Swapping**

Swapping is the process of temporarily removing inactive programs from the main memory of a computer system. Swapping is a memory management technique and is used to temporarily remove the inactive programs from the main memory of the computer system. Any process must be in the memory for its execution but can be swapped temporarily out of memory to a backing store and then again brought back into the memory to complete its execution. Swapping is done so that other processes get memory for their execution.

Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes. It is used to improve main memory utilization. In secondary memory, the place where the swapped-out process is stored is called swap space.

The purpose of the swapping in [operating system](https://www.javatpoint.com/os-tutorial) is to access the data present in the hard disk and bring it to [RAM](https://www.javatpoint.com/ram) so that the application programs can use it. The thing to remember is that swapping is used only when data is not present in [RAM](https://www.javatpoint.com/ram-full-form).

Although the process of swapping affects the performance of the system, it helps to run larger and more than one process. This is the reason why swapping is also referred to as memory compaction.

The concept of swapping has divided into two more concepts: Swap-in and Swap-out.

* Swap-out is a method of removing a process from RAM and adding it to the hard disk.
* Swap-in is a method of removing a program from a hard disk and putting it back into the main memory or RAM.

**Example:** Suppose the user process's size is 2048KB and is a standard hard disk where swapping has a data transfer rate of 1Mbps. Now we will calculate how long it will take to transfer from main memory to secondary memory.

User process size is 2048Kb

Data transfer rate is 1Mbps = 1024 kbps

Time = process size / transfer rate

     = 2048 / 1024

     = 2 seconds

     = 2000 milliseconds

Now taking swap-in and swap-out time, the process will take 4000 milliseconds.

## **Advantages of Swapping**

1. It helps the CPU to manage multiple processes within a single main memory.
2. It helps to create and use virtual memory.
3. Swapping allows the CPU to perform multiple tasks simultaneously. Therefore, processes do not have to wait very long before they are executed.
4. It improves the main memory utilization.

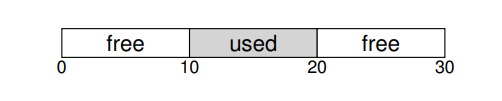
## **Disadvantages of Swapping**

1. If the computer system loses power, the user may lose all information related to the program in case of substantial swapping activity.
2. If the swapping algorithm is not good, the composite method can increase the number of Page Fault and decrease the overall processing performance.

**Free Space Management**

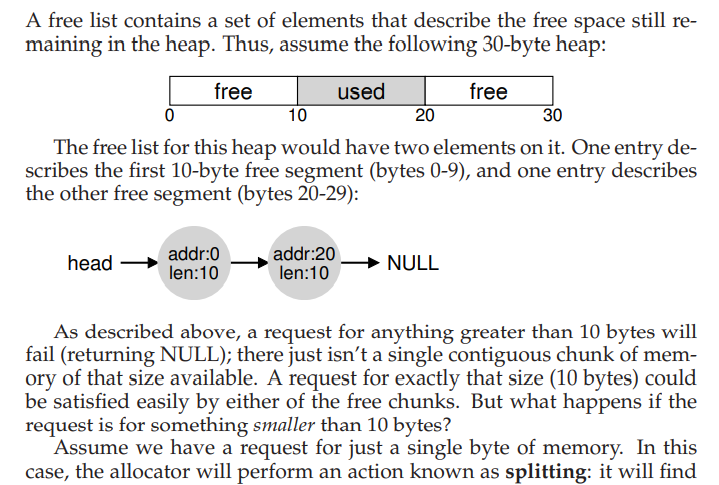
Managing free space can certainly be easy, as we will see when we discuss the concept of paging. It is easy when the space you are managing is divided into fixed-sized units; in such a case, you just keep a list of these fixed-sized units; when a client requests one of them, return the first entry.

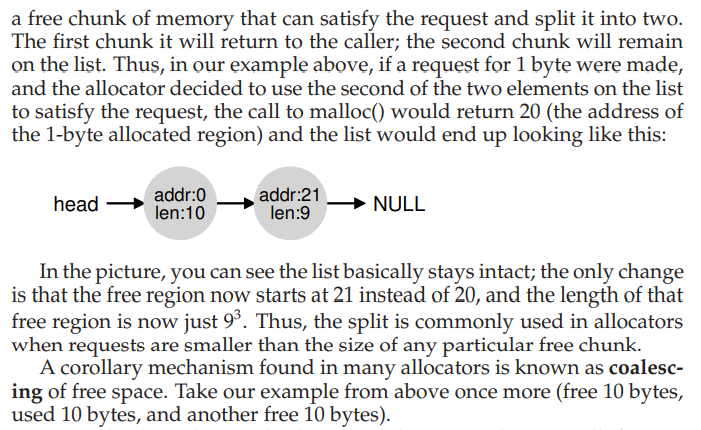
Where free-space management becomes more difficult (and interesting) is when the free space you are managing consists of variable-sized units; this arises in a user-level memory-allocation library (as in **malloc()** and **free()**) and in an OS managing physical memory when using segmentation to implement virtual memory. In either case, the problem that exists is known as external fragmentation: the free space gets chopped into little pieces of different sizes and is thus fragmented; subsequent requests may fail because there is no single contiguous space that can satisfy the request, even though the total amount of free space exceeds the size of the request.

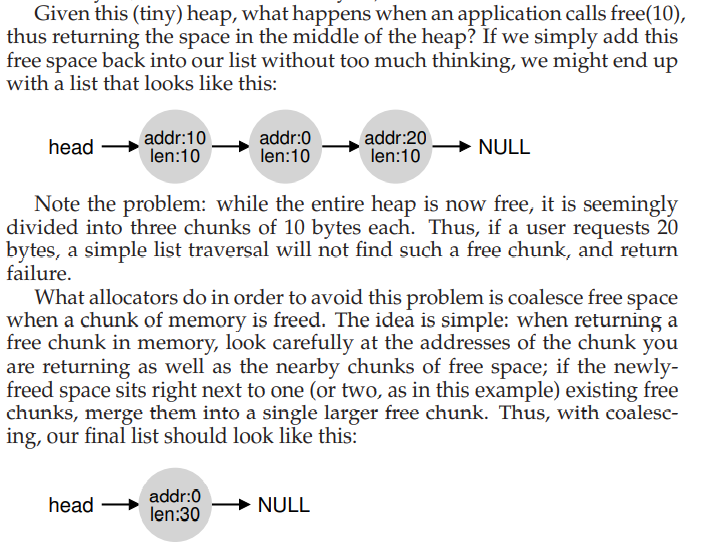
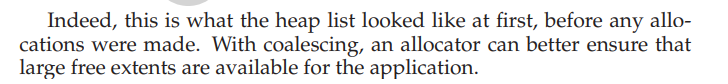


The figure shows an example of this problem. In this case, the total free space available is 20 bytes; unfortunately, it is fragmented into two chunks of size 10 each. As a result, a request for 15 bytes will fail even though there are 20 bytes free.

**Splitting and Coalescing**





# **Thrashing in Operating System**

In case, if the page fault and swapping happens very frequently at a higher rate, then the operating system has to spend more time swapping these pages. This state in the operating system is termed thrashing. Because of thrashing the CPU utilization is going to be reduced.

Let's understand by an example, if any process does not have the number of frames that it needs to support pages in active use then it will quickly page fault. And at this point, the process must replace some pages. As all the pages of the process are actively in use, it must replace a page that will be needed again right away. Consequently, the process will quickly fault again, and again, and again, replacing pages that it must bring back in immediately. This high paging activity by a process is called thrashing.

Diagram

Description automatically generated

## **Causes of Thrashing**

Thrashing affects the performance of execution in the Operating system. Also, thrashing results in severe performance problems in the Operating system. When the utilization of CPU is low, then the process scheduling mechanism tries to load many processes into the memory at the same time due to which degree of Multiprogramming can be increased. Now in this situation, there are more processes in the memory as compared to the available number of frames in the memory. Allocation of the limited amount of frames to each process.

Whenever any process with high priority arrives in the memory and if the frame is not freely available at that time then the other process that has occupied the frame is residing in the frame will move to secondary storage and after that this free frame will be allocated to higher priority process. We can also say that as soon as the memory fills up, the process starts spending a lot of time for the required pages to be swapped in. Again the utilization of the CPU becomes low because most of the processes are waiting for pages.

Thus a high degree of multiprogramming and lack of frames are two main causes of thrashing in the Operating system.

**Text & Reference books:**

1. Operating Systems: Three Easy Pieces, Remzi H. Arpaci-Dusseau and Andrea C. Arpaci- Dusseau, Arpaci-Dusseau Books, May, (2014).