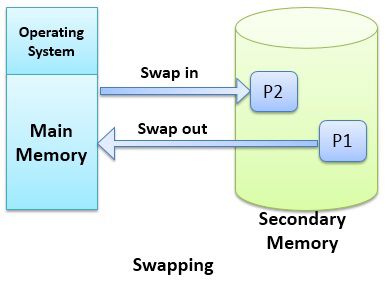
***Swapping*** *is a mechanism in which a process can be swapped/moved temporarily out of main memory to a backing store , and then brought back into memory for continued execution.*  
• For example, assume a multiprogramming environment with a round-robin CPU-scheduling algorithm. When a quantum expires, the memory manager will start to swap out (move) the process that just finished, and to swap in(bring back) another process to the memory space that has been freed.  
• A process must be loaded into memory in order to execute.  
• If there is not enough memory available to keep all running processes in memory at the same time, then some processes who are not currently using the CPU may have their memory swapped out to a fast local disk called the backing store.



STANDARD SWAPPING

 Swap only totally idle processes, or do I/O operations only into and out of OS buffers, which are then transferred to or from process's main memory as a second step.

• Most modern OSes no longer use swapping, because it is too slow and there are faster alternatives available. ( e.g. Paging. )

• It is important to swap processes out of memory only when they are idle, or more to the point, only when there are no pending I/O operations.

**Contiguous memory allocation**

**Memory management techniques**

The memory management techniques is divided into two parts...

1. **Uniprogramming:**  
   In the uniprogramming technique, the RAM is divided into two parts one part is for the resigning the operating system and other portion is for the user process. Here the fence register is used which contain the last address of the operating system parts. The operating system will compare the user data addresses with the fence register and if it is different that means the user is not entering in the OS area. Fence register is also called boundary register and is used to prevent a user from entering in the operating system area. Here the CPU utilization is very poor and hence multiprogramming is used.
2. **Multiprogramming:**  
   In the multiprogramming, the multiple users can share the memory simultaneously. By multiprogramming we mean there will be more than one process in the main memory and if the running process wants to wait for an event like I/O then instead of sitting ideal CPU will make a context switch and will pick another process.
   1. Contiguous memory allocation
   2. Non-contiguous memory allocation

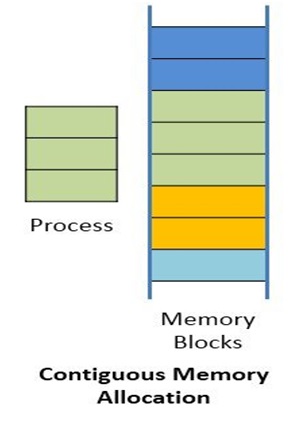
a) Contiguous memory allocation

In **contiguous memory allocation**, all the available memory space remain together in one place. It means freely available memory partitions are not scattered here and there across the whole memory space.

In the **contiguous memory allocation**, both the operating system and the user must reside in the main memory. The main memory is divided into two portions one portion is for the operating and other is for the user program.

In the **contiguous memory allocation** when any user process request for the memory a single section of the contiguous memory block is given to that process according to its need. We can achieve contiguous memory allocation by dividing memory into the fixed-sized partition.

A single process is allocated in that fixed sized single partition. But this will increase the degree of multiprogramming means more than one process in the main memory that bounds the number of fixed partition done in memory. Internal fragmentation increases because of the contiguous memory allocation.



**Fixed sized partition**

In the fixed sized partition the system divides memory into fixed size partition (may or may not be of the same size) here entire partition is allowed to a process and if there is some wastage inside the partition is allocated to a process and if there is some wastage inside the partition then it is called internal fragmentation.

**Advantage:** Management or book keeping is easy.

**Disadvantage:** Internal fragmentation

**→ Variable size partition**

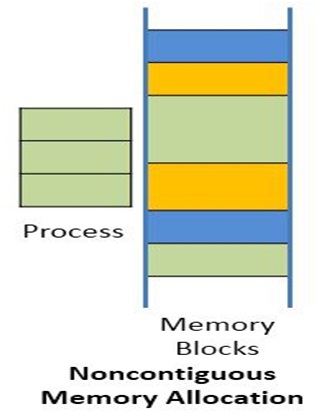
In the variable size partition, the memory is treated as one unit and space allocated to a process is exactly the same as required and the leftover space can be reused again.

**Advantage:** There is no internal fragmentation.

**Disadvantage:** Management is very difficult as memory is becoming purely fragmented after some time.

## b) Non-contiguous memory allocation

In the **non-contiguous memory allocation** the available free memory space are scattered here and there and all the free memory space is not at one place. So this is time-consuming. In the **non-contiguous memory allocation**, a process will acquire the memory space but it is not at one place it is at the different locations according to the process requirement. This technique of **non-contiguous memory allocation** reduces the wastage of memory which leads to internal and external fragmentation. This utilizes all the free memory space which is created by a different process.



**Non-contiguous memory allocation** is of different types,

1. Paging
2. Segmentation
3. Segmentation with paging

**i) Paging**

A non-contiguous policy with a fixed size partition is called paging. A computer can address more memory than the amount of physically installed on the system. This extra memory is actually called virtual memory. Paging technique is very important in implementing virtual memory. Secondary memory is divided into equal size partition (fixed) called pages. Every process will have a separate page table. The entries in the page table are the number of pages a process. At each entry either we have an invalid pointer which means the page is not in main memory or we will get the corresponding frame number. When the frame number is combined with instruction of set D than we will get the corresponding physical address. Size of a page table is generally very large so cannot be accommodated inside the PCB, therefore, PCB contains a register value PTBR( page table base register) which leads to the page table.

**Advantages:** It is independent of external fragmentation.

**Disadvantages:**

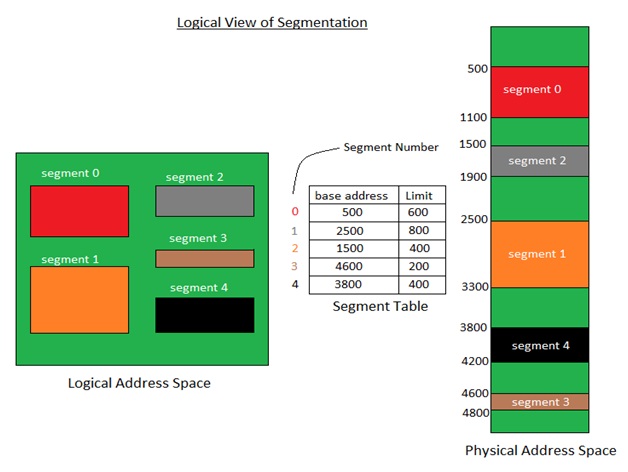
1. It makes the translation very slow as main memory access two times.
2. A page table is a burden over the system which occupies considerable space.

**ii) Segmentation**

Segmentation is a programmer view of the memory where instead of dividing a process into equal size partition we divided according to program into partition called segments. The translation is the same as paging but paging segmentation is independent of internal fragmentation but suffers from external fragmentation. Reason of external fragmentation is program can be divided into segments but segment must be contiguous in nature.

**iii) Segmentation with paging**

In segmentation with paging, we take advantages of both segmentation as well as paging. It is a kind of multilevel paging but in multilevel paging, we divide a page table into equal size partition but here in segmentation with paging, we divide it according to segments. All the properties are the same as that of paging because segments are divided into pages.



# Page Table

Page Table is a data structure used by the virtual memory system to store the mapping between logical addresses and physical addresses.

Logical addresses are generated by the CPU for the pages of the processes therefore they are generally used by the processes.

Physical addresses are the actual frame address of the memory. They are generally used by the hardware or more specifically by RAM subsystems.

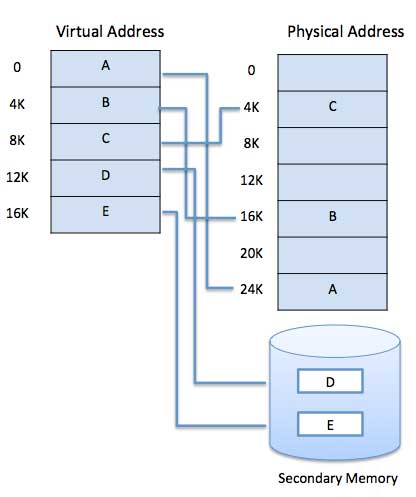
**Virtual memory**

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.

The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

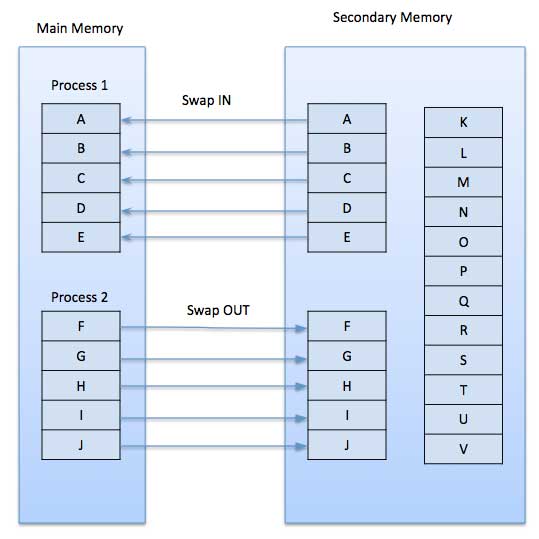
* User written error handling routines are used only when an error occurred in the data or computation.
* Certain options and features of a program may be used rarely.
* Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
* The ability to execute a program that is only partially in memory would counter many benefits.
* Less number of I/O would be needed to load or swap each user program into memory.
* A program would no longer be constrained by the amount of physical memory that is available.
* Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.



Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

## Demand Paging

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program’s pages out to the disk or any of the new program’s pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program’s pages as they are referenced.



### Advantages

Following are the advantages of Demand Paging −

* Large virtual memory.
* More efficient use of memory.
* There is no limit on degree of multiprogramming.

### Disadvantages

* Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

## Page Replacement Algorithm

Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

**Copy on write**

**Copy-on-write** (**CoW** or **COW**), sometimes referred to as **implicit sharing**[[1]](https://en.wikipedia.org/wiki/Copy-on-write#cite_note-1) or **shadowing**,[[2]](https://en.wikipedia.org/wiki/Copy-on-write#cite_note-2) is a resource-management technique used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming) to efficiently implement a "duplicate" or "copy" operation on modifiable resources.[[3]](https://en.wikipedia.org/wiki/Copy-on-write#cite_note-Linux-3) If a resource is duplicated but not modified, it is not necessary to create a new resource; the resource can be shared between the copy and the original. Modifications must still create a copy, hence the technique: the copy operation is deferred to the first write. By sharing resources in this way, it is possible to significantly reduce the resource consumption of unmodified copies, while adding a small overhead to resource-modifying operations.

Copy-on-write finds its main use in sharing the [virtual memory](https://en.wikipedia.org/wiki/Virtual_memory) of [operating system](https://en.wikipedia.org/wiki/Operating_system) [processes](https://en.wikipedia.org/wiki/Computer_process), in the implementation of the [fork system call](https://en.wikipedia.org/wiki/Fork_(system_call)).

Copy-on-write can be implemented efficiently using the [page table](https://en.wikipedia.org/wiki/Page_table) by marking certain pages of [memory](https://en.wikipedia.org/wiki/Computer_storage) as read-only and keeping a count of the number of references to the page.

# Allocation Of Frames

When a page fault occurs, there is a free frame available to store a new page into a frame. While the page swap is taking place, a replacement can be selected, which is written to the disk as the user process continues to execute. The operating system allocates all its buffer and tablespace from the free-frame list for the new page.

Two major allocation Algorithm/schemes.

#### ****Equal allocation****

#### ****Proportional allocation****

**Equal allocation:** The easiest way to split m frames among n processes is to give everyone an equal share, m/n frames. This is known as **equal allocation.**

**proportional allocation:** Here, it allocates available memory to each process according to its size. Let the size of the virtual memory for process **pi** be **si,**and define S= ∑ Si

Then, if the total number of available frames is m, we allocate **ai** frames to process pi, where **ai**is approximately **ai =**Si/ S x m.

#### ****Global Versus Local Allocation****

We can classify page-replacement algorithms into two broad categories: **global replacement**and **local replacement.**

Global replacement allows a process to select a replacement frame from the set of all frames, even if that frame is currently allocated to some other process; one process can take a frame from another.

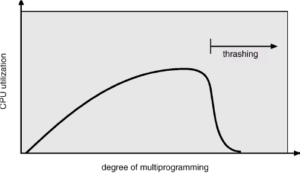
Local replacement requires that each process selects from only its own set of allocated frames.

### ****Thrashing:-****

The system spends most of its time shuttling pages between main memory and secondary memory due to frequent page faults. This behavior is known as thrashing.

A process is thrashing if it is spending more time paging than executing. This leads to:

low CPU utilization and the operating system thinks that it needs to increase the degree of multiprogramming.



**Memory mapped files**

Consider a sequential read of a file on disk using the standard system calls openQ, readO, and writeQ. Each file access requires a system call and disk access. Alternatively, we can use the virtual memory techniques discussed so far to treat file I/O as routine memory accesses. This approach, known as memory mapping a file, allows a part of the virtual address space to be logically associated with the file.

Basic Mechanism

 Memory mapping a file is accomplished by mapping a disk block to a page (or pages) in memory. Initial access to the file proceeds through ordinary demand paging, resulting in a page fault. However, a page-sized portion of the file is read from the file system into a physical page (some systems may opt  to read in more than a page-sized chunk of memory at a time).

Subsequent reads and writes to the file are handled as routine memory accesses, thereby simplifying file access and usage by allowing the system to manipulate files through memory rather than incurring the overhead of using the readQ and writeO system calls. Note that writes to the file mapped in memory are not necessarily immediate (synchronous) writes to the file on disk. Some systems may choose to update the physical file when the operating system periodically checks whether the page in memory has been modified. When the file is closed, all the memory-mapped data are written back to disk and removed from the virtual memory of the process.