UNIT 4:
Memory Management

What is a Logical Address?

The **logical address** is a virtual address created by the CPU of the computer system. The logical address of a program is generated when the program is running. A group of several logical address is referred to a **logical address space**. The logical address is basically used as a reference to access the physical memory locations.

In computer systems, a hardware device named **memory management unit** (MMU) is used to map the logical address to its corresponding physical address. However, the logical address of a program is visible to the computer user.

What is a Physical Address?

The **physical address** of a computer program is one that represents a location in the memory unit of the computer. The physical address is not visible to the computer user. The MMU of the system generates the physical address for the corresponding logical address.

The physical address is accessed through the corresponding logical address because a user cannot directly access the physical address. For running a computer program, it requires a physical memory space. Therefore, the logical address has to be mapped with the physical address before the execution of the program.

Difference between Logical and Physical Address in Operating System

The following table highlights all the major differences between logical and physical address in operating system −

S. No.	Logical Address	Physical Address
1.	This address is generated by the CPU.	This address is a location in the memory unit.
2.	The address space consists of the set of all logical addresses.	This address is a set of all physical addresses that are mapped to the corresponding logical addresses.

3.	These addresses are generated by CPU with reference to a specific program.	It is computed using Memory Management Unit (MMU).
4.	The user has the ability to view the logical address of a program.	The user can't view the physical address of program directly.
5.	The user can use the logical address in order to access the physical address.	The user can indirectly access the physical address.

Swapping in Operating System

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 is to access the data present in the hard disk and bring it to 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.

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.

1. User process size is 2048Kb
2. Data transfer rate is 1Mbps = 1024 kbps
3. Time = process size / transfer rate
4. = 2048 / 1024

5. = 2 seconds
6. = 2000 milliseconds
7. 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

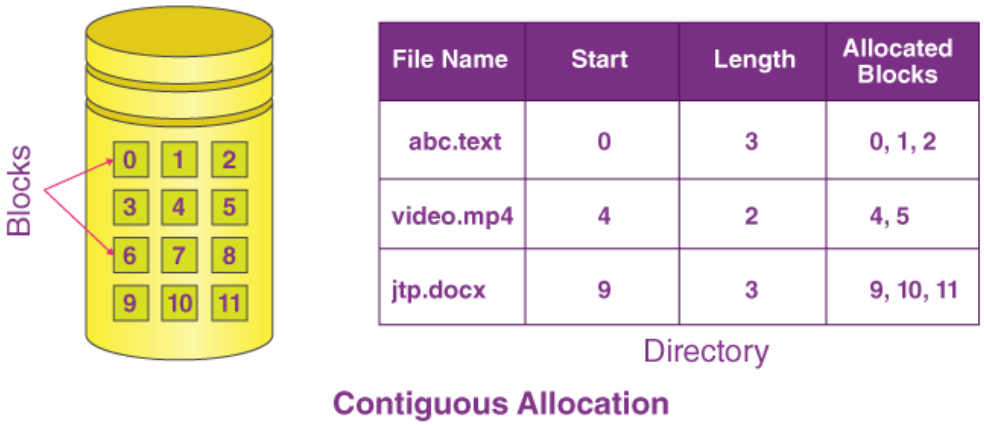
1. If the computer system loses power, the user may lose all information related to the program in case of substantial swapping activity.
2. If the swapping algorithm is not good, the composite method can increase the number of Page Fault and decrease the overall processing performance.

Note:

- In a single tasking operating system, only one process occupies the user program area of memory and stays in memory until the process is complete.
- In a multitasking operating system, a situation arises when all the active processes cannot coordinate in the main memory, then a process is swap out from the main memory so that other processes can enter it.

Contiguous Memory Allocation

Contiguous memory allocation refers to a memory management technique in which whenever there occurs a request by a user process for the memory, one of the sections of the contiguous memory block would be given to that process, in accordance with its requirement.



As you can see in the illustration shown above, three files are there in the directory. The starting block, along with the length of each file, is mentioned in the table. Thus, we can see in this table that all the contiguous blocks get assigned to every file as per their need.

Types of Partitions

Contiguous memory allocation can be achieved when we divide the memory into the following types of partitions:

1. Fixed-Sized Partitions

Another name for this is static partitioning. In this case, the system gets divided into multiple fixed-sized partitions. In this type of scheme, every partition may consist of exactly one process. This very process limits the extent at which multiprogramming would occur, since the total number of partitions decides the total number of processes. Read more on fixed-sized partitions here.

2. Variable-Sized Partitions

Dynamic partitioning is another name for this. The scheme allocation in this type of partition is done dynamically. Here, the size of every partition isn't declared initially. Only once we know the process size, will we know the size of the partitions. But in this case, the size of the process and the partition is equal; thus, it helps in preventing internal fragmentation.

On the other hand, when a process is smaller than its partition, some size of the partition gets wasted (internal fragmentation). It occurs in static partitioning, and dynamic partitioning solves this issue. Read more on dynamic partitions here.

Pros of Contiguous Memory Allocation

1. It supports a user's random access to files.

2. The user gets excellent read performance.

3. It is fairly simple to implement.

Cons of Contiguous Memory Allocation

1. Having a file grow might be somewhat difficult.

2. The disk may become fragmented.

Partition Allocation Methods

In Partition Allocation Methods, when there is more than one partition is free available to accommodate a process's request. A partition must be selected.`To choose a particular partition, a partition allocation method is most needed. A partition allocation method considers better if it avoids internal fragmentation.

Single Partition Allocation:

In this memory allocation method, the operating system resides in the low memory. The remaining memory treated as a single partition. This single partition is available for userspace. Only one job can be loaded in this user space.

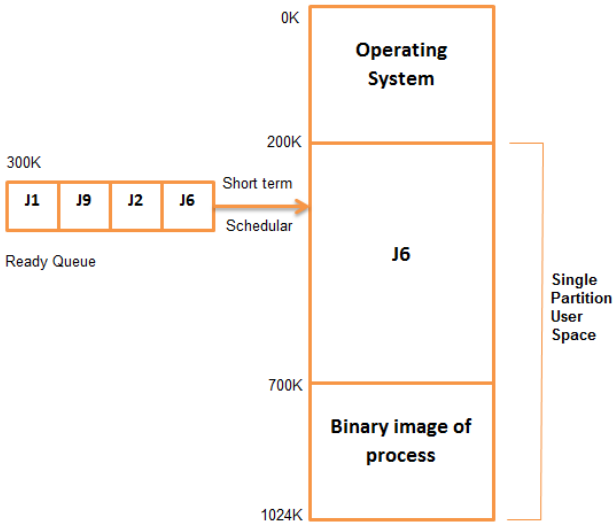


Fig: Memory Allocation for Single Partitioned

Multiple Partitions Memory Management:

This method can be implemented in 3 ways. These are:
i. Fixed-equal Multiple Partitions Memory Management
ii. Fixed-variable Partitions Memory Management
iii. Dynamic Partitions Memory Management

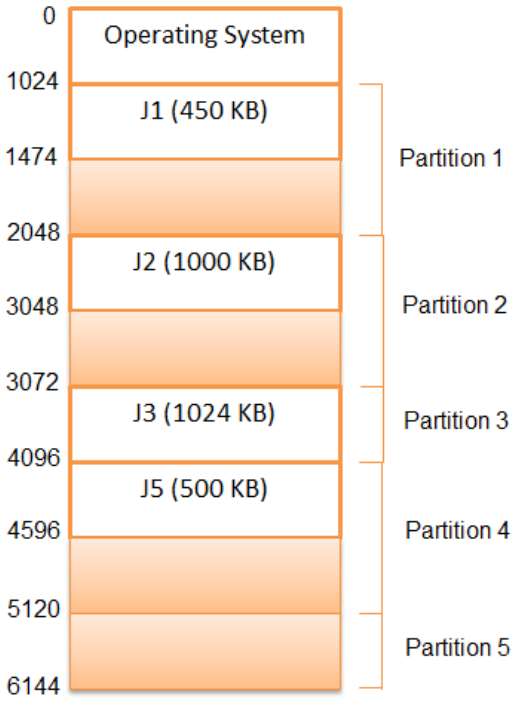


Fig: Memory Allocation for Multiple Partitioned

Fixed-equal Multiple Partitions Memory Management:

In this memory management, the operating system occupies the low memory and the rest of the main memory is available for user space. The user space divides into fixed partitions. The partition sizes are depending on the operating system.

Fixed-variable Partitions Memory Management:

In this memory management, the user space of the main memory is divided into several partitions. But the partition sizes are different lengths. The operating system keeps a table indicating which partitions of memory are available and which are occupied.

Dynamic Partitions Memory Management:

To eliminate some of the problems with fixed partitions, an approach known as **Dynamic Partitions**. In this method, partitions are created dynamically. So that each process loads into a partition of the same size as that process.

First fit:

Allocate the partition that is big enough, searching can start either from low memory or high memory. We can stop searching as soon as we find a free partition that is large enough.

Advantages of the First fit:

1. It is generally faster than the Worst fit.
2. It is better than the Worst fit in terms of decreasing time and storage utilization.

Best fit:

Allocate the smallest partition that is big enough (or) select a partition which had the least internal fragmentation.

Worst fit:

Search the entire partitions and select a partition which is the largest of all. Select a partition that had the maximum internal fragmentation.

Next fit:

Next fit is similar to the first fit. But it'll search for the first sufficient partition from the last allocation point.

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.

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:

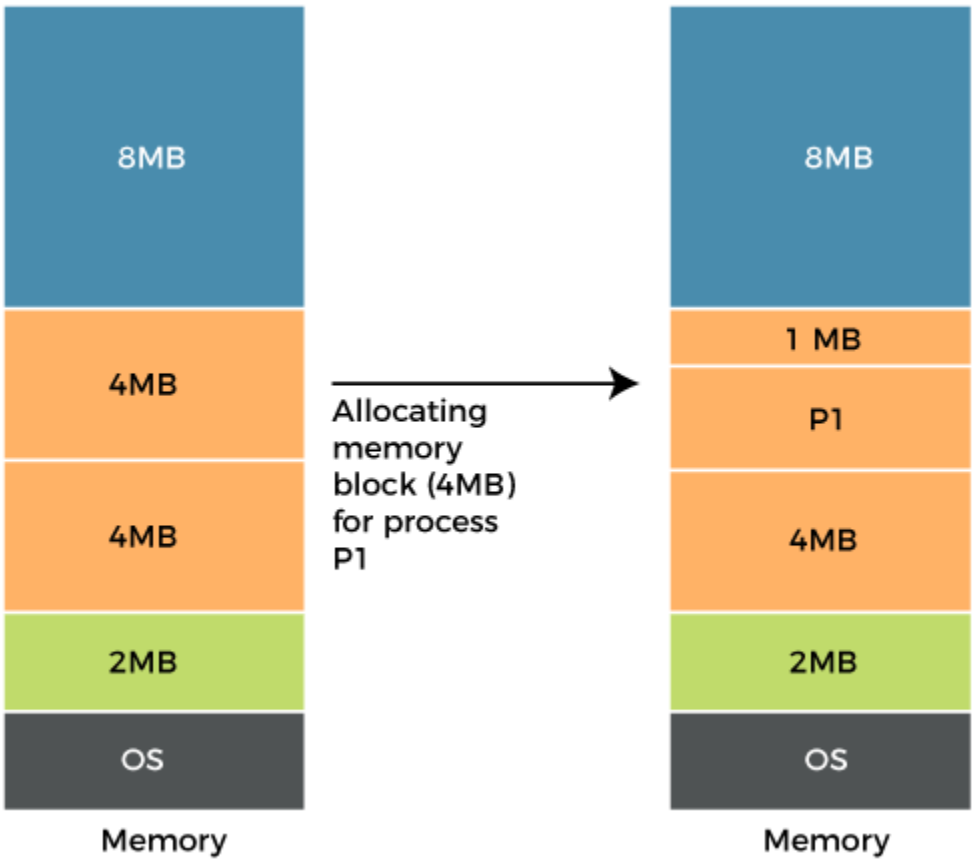
1. **Internal Fragmentation**
2. **External Fragmentation**

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**.

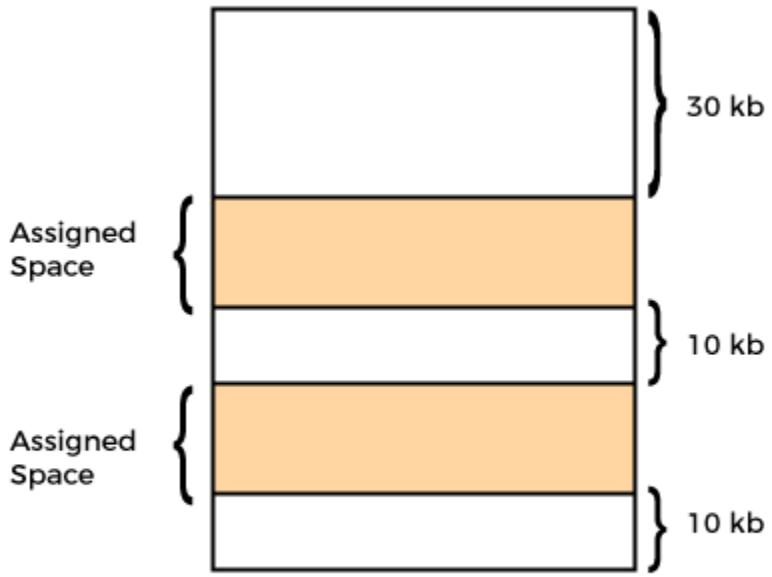
How to avoid internal fragmentation?

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.

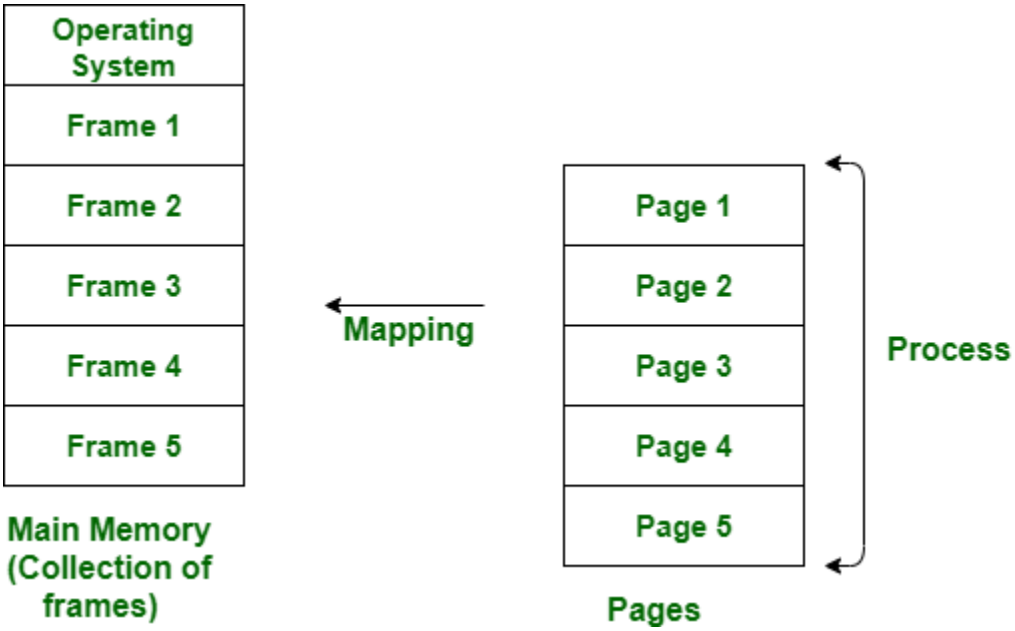
Conclusion

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.

Paging:

Paging is a method or technique which is used for non-contiguous memory allocation. It is a fixed-size partitioning theme (scheme). In paging, both main memory and secondary memory are divided into equal fixed-size partitions. The partitions of the secondary memory area unit and main memory area unit are known as pages and frames respectively.

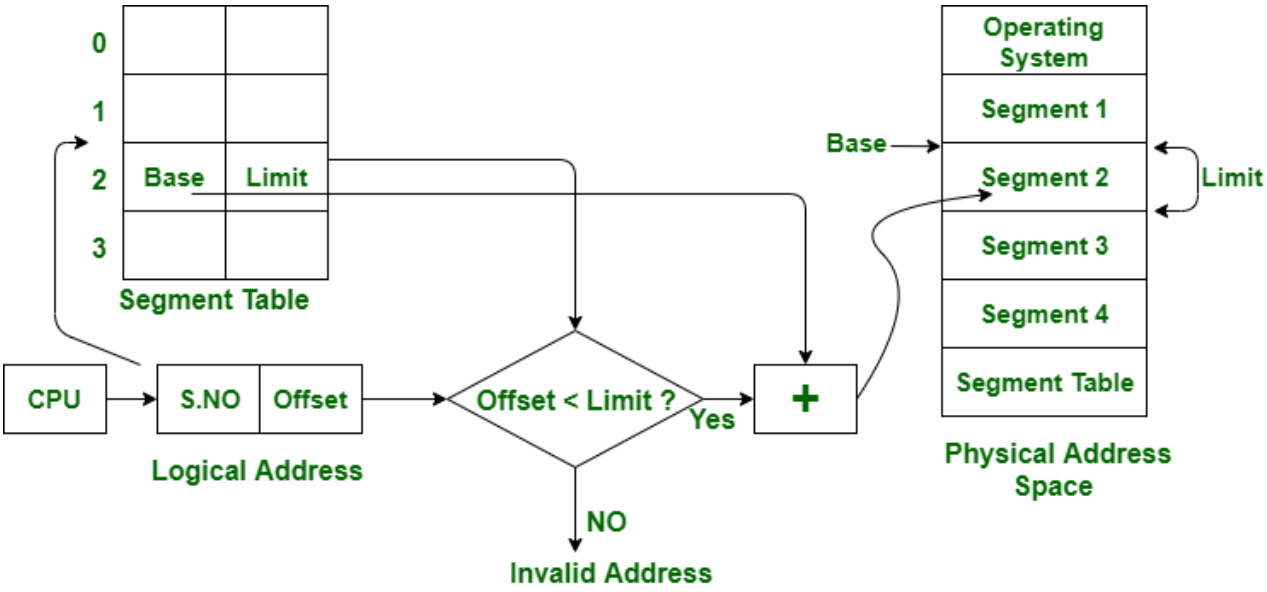
Paging is a memory management method accustomed fetch processes from the secondary memory into the main memory in the form of pages. in paging, each process is split into parts wherever the size of every part is the same as the page size. The size of the last half could also be but the page size. The pages of the process area unit hold on within the frames of main memory relying upon their accessibility.



Segmentation:

Segmentation is another non-contiguous memory allocation scheme like paging. like paging, in segmentation, the process isn't divided indiscriminately into mounted(fixed) size pages. It is a variable-size partitioning theme. like paging, in segmentation,

secondary and main memory are not divided into partitions of equal size. The partitions of secondary memory area units are known as segments. The details concerning every segment are hold in a table known as segmentation table. Segment table contains two main data concerning segment, one is Base, which is the bottom address of the segment and another is Limit, which is the length of the segment.
In segmentation, the CPU generates a logical address that contains the Segment number and segment offset. If the segment offset is a smaller amount than the limit then the address called valid address otherwise it throws miscalculation because the address is invalid.



The above figure shows the translation of a logical address to a physical address.

S.NO	Paging	Segmentation
1.	In paging, the program is divided into fixed or mounted size pages.	In segmentation, the program is divided into variable size sections.
2.	For the paging operating system is accountable.	For segmentation compiler is accountable.
3.	Page size is determined by hardware.	Here, the section size is given by the user.

S.NO	Paging	Segmentation
4.	It is faster in comparison to segmentation.	Segmentation is slow.
5.	Paging could result in internal fragmentation.	Segmentation could result in external fragmentation.
6.	In paging, the logical address is split into a page number and page offset.	Here, the logical address is split into section number and section offset.
7.	Paging comprises a page table that encloses the base address of every page.	While segmentation also comprises the segment table which encloses segment number and segment offset.
8.	The page table is employed to keep up the page data.	Section Table maintains the section data.
9.	In paging, the operating system must maintain a free frame list.	In segmentation, the operating system maintains a list of holes in the main memory.
10.	Paging is invisible to the user.	Segmentation is visible to the user.
11.	In paging, the processor needs the page number, and offset to calculate the absolute address.	In segmentation, the processor uses segment number, and offset to calculate the full address.
12.	It is hard to allow sharing of procedures between processes.	Facilitates sharing of procedures between the processes.
13	In paging, a programmer cannot efficiently handle data structure.	It can efficiently handle data structures.
14.	This protection is hard to apply.	Easy to apply for protection in segmentation.

S.NO	Paging	Segmentation
15.	The size of the page needs always be equal to the size of frames.	There is no constraint on the size of segments.
16.	A page is referred to as a physical unit of information.	A segment is referred to as a logical unit of information.
17.	Paging results in a less efficient system.	Segmentation results in a more efficient system.

What is Virtual Memory in OS (Operating System)?

Virtual Memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.

By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

How Virtual Memory Works?

In modern word, virtual memory has become quite common these days. In this scheme, whenever some pages needs to be loaded in the main memory for the execution and the memory is not available for those many pages, then in that case, instead of stopping the pages from entering in the main memory, the OS search for the RAM area that are least used in the recent times or that are not referenced and copy that into the secondary memory to make the space for the new pages in the main memory.

Since all this procedure happens automatically, therefore it makes the computer feel like it is having the unlimited RAM.

Demand Paging

Demand Paging is a popular method of virtual memory management. In demand paging, the pages of a process which are least used, get stored in the secondary memory.

A page is copied to the main memory when its demand is made or page fault occurs. There are various page replacement algorithms which are used to determine the pages which will be replaced. We will discuss each one of them later in detail.

Snapshot of a virtual memory management system

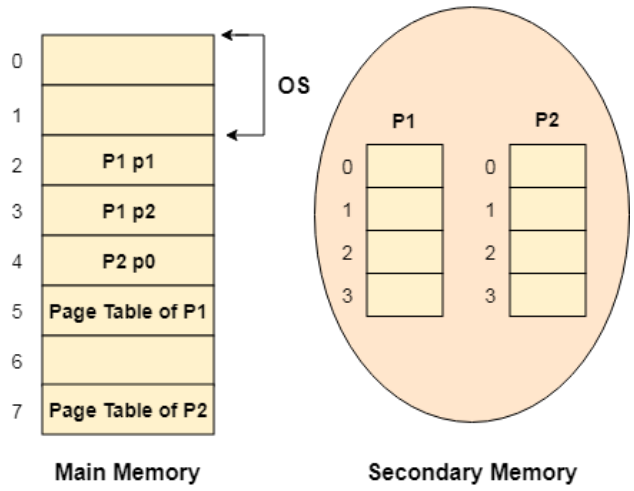
Let us assume 2 processes, P1 and P2, contains 4 pages each. Each page size is 1 KB. The main memory contains 8 frame of 1 KB each. The OS resides in the first two partitions. In the third partition, 1st page of P1 is stored and the other frames are also shown as filled with the different pages of processes in the main memory.

The page tables of both the pages are 1 KB size each and therefore they can be fit in one frame each. The page tables of both the processes contain various information that is also shown in the image.

The CPU contains a register which contains the base address of page table that is 5 in the case of P1 and 7 in the case of P2. This page table base address will be added to the page number of the Logical address when it comes to accessing the actual corresponding entry.

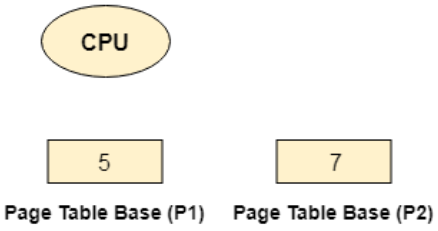
	Frame	Present / Absent	D Bit	Reference bit	Protection
0		0	0	0	
1	2	1	0	1	
2	3	1	1	0	
3		0	0	0	

Page Table of P1



	Frame	Present / Absent	D Bit	Reference bit	Protection
0	4	1	0	1	
1		0	0	0	
2		0	0	0	
3		0	0	0	

Page Table of P2



Advantages of Virtual Memory

1. The degree of Multiprogramming will be increased.
2. User can run large application with less real RAM.
3. There is no need to buy more memory RAMs.

Disadvantages of Virtual Memory

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have the lesser hard disk space for its use.

Page Replacement Algorithms in Operating Systems (OS)

Today we are going to learn about Page Replacement Algorithms in Operating Systems (OS). Before knowing about Page Replacement Algorithms in Operating Systems let us learn about Paging in Operating Systems and also a little about Virtual Memory.

Only after understanding the concept of Paging we will understand about Page Replacement Algorithms.

Paging in Operating Systems (OS)

Paging is a storage mechanism. Paging is used to retrieve processes from secondary memory to primary memory.

The main memory is divided into small blocks called pages. Now, each of the pages contains the process which is retrieved into main memory and it is stored in one frame of memory.

It is very important to have pages and frames which are of equal sizes which are very useful for mapping and complete utilization of memory.

Virtual Memory in Operating Systems (OS)

A storage method known as virtual memory gives the user the impression that their main memory is quite large. By considering a portion of secondary memory as the main memory, this is accomplished.

By giving the user the impression that there is memory available to load the process, this approach allows them to load larger size programs than the primary memory that is accessible.

The Operating System loads the many components of several processes in the main memory as opposed to loading a single large process there.

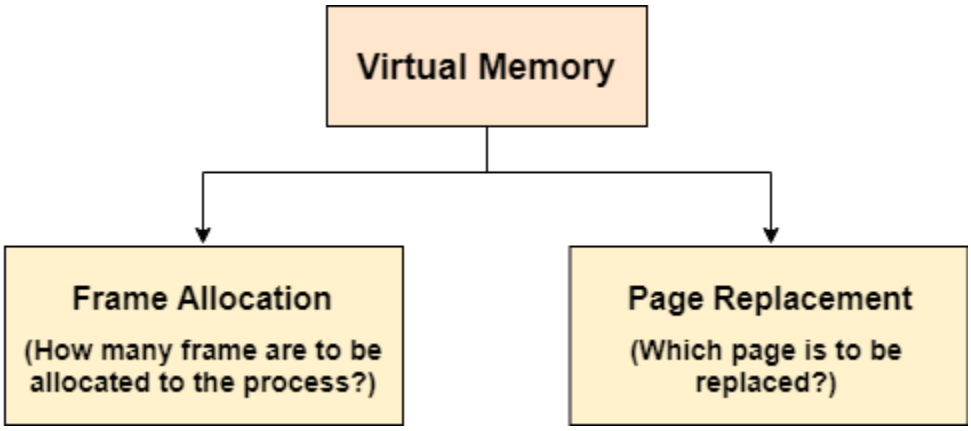
By doing this, the level of multiprogramming will be enhanced, which will increase CPU consumption.

Demand Paging

The Demand Paging is a condition which is occurred in the Virtual Memory. We know that the pages of the process are stored in secondary memory. The page is brought to the main memory

when required. We do not know when this requirement is going to occur. So, the pages are brought to the main memory when required by the Page Replacement Algorithms.

So, the process of calling the pages to main memory to secondary memory upon demand is known as Demand Paging.



The important jobs of virtual memory in Operating Systems are two. They are:

- Frame Allocation
- Page Replacement.

Frame Allocation in Virtual Memory

Demand paging is used to implement virtual memory, an essential component of operating systems. A page-replacement mechanism and a frame allocation algorithm must be created for demand paging. If you have numerous processes, frame allocation techniques are utilized to determine how many frames to provide to each process.

A Physical Address is required by the Central Processing Unit (CPU) for the frame creation and the physical Addressing provides the actual address to the frame created. For each page a frame must be created.

Frame Allocation Constraints

- The Frames that can be allocated cannot be greater than total number of frames.
- Each process should be given a set minimum amount of frames.
- When fewer frames are allocated then the page fault ration increases and the process execution becomes less efficient

- There ought to be sufficient frames to accommodate all the many pages that a single instruction may refer to

Frame Allocation Algorithms

There are three types of Frame Allocation Algorithms in Operating Systems. They are:

1) Equal Frame Allocation Algorithms

Here, in this Frame Allocation Algorithm we take number of frames and number of processes at once. We divide the number of frames by number of processes. We get the number of frames we must provide for each process.

This means if we have 36 frames and 6 processes. For each process 6 frames are allocated.

It is not very logical to assign equal frames to all processes in systems with processes of different sizes. A lot of allocated but unused frames will eventually be wasted if a lot of frames are given to a little operation.

2) Proportionate Frame Allocation Algorithms

Here, in this Frame Allocation Algorithms we take number of frames based on the process size. For big process more number of frames is allocated. For small processes less number of frames is allocated by the operating system.

The problem in the Proportionate Frame Allocation Algorithm is number of frames are wasted in some rare cases.

The advantage in Proportionate Frame Allocation Algorithm is that instead of equally, each operation divides the available frames according to its demands.

3) Priority Frame Allocation Algorithms

According to the quantity of frame allocations and the processes, priority frame allocation distributes frames. Let's say a process has a high priority and needs more frames; in such case, additional frames will be given to the process. Processes with lower priorities are then later executed in future and first only high priority processes are executed first.

Page Replacement Algorithms

There are three types of Page Replacement Algorithms. They are:

- Optimal Page Replacement Algorithm

- First In First Out Page Replacement Algorithm
- Least Recently Used (LRU) Page Replacement Algorithm

First in First out Page Replacement Algorithm

This is the first basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as Page Hit.

If the page to be searched is not found among the frames then, this process is known as Page Fault.

When Page Fault occurs this problem arises, then the First In First Out Page Replacement Algorithm comes into picture.

The First In First Out (FIFO) Page Replacement Algorithm removes the Page in the frame which is allotted long back. This means the useless page which is in the frame for a longer time is removed and the new page which is in the ready queue and is ready to occupy the frame is allowed by the First In First Out Page Replacement.

Let us understand this First In First Out Page Replacement Algorithm working with the help of an example.

Example:

Consider the reference string 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0 for a memory with three frames and calculate number of page faults by using FIFO (First In First Out) Page replacement algorithms.

Points to Remember

Page Not Found - - - > Page Fault

Page Found - - - > Page Hit

Reference String:

6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

S. no	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F3				2	2	2	4	4	4	2	2	2	2	2	2	2	2	2	2	0
F2		1	1	1	1	3	3	3	0	0	0	0	0	0	0	3	3	3	3	3
F1	6	6	6	6	0	0	0	6	6	6	1	1	1	1	1	1	1	1	1	1
Hit (H)/ Fault (F)	F	F	H	F	F	F	F	F	F	F	F	H	H	H	H	F	H	H	H	F

Number of Page Hits = 8

Number of Page Faults = 12

The Ratio of Page Hit to the Page Fault = 8 : 12 - - - > 2 : 3 - - - > 0.66

The Page Hit Percentage = 8 *100 / 20 = 40%

The Page Fault Percentage = 100 - Page Hit Percentage = 100 - 40 = 60%

Explanation

First, fill the frames with the initial pages. Then, after the frames are filled we need to create a space in the frames for the new page to occupy. So, with the help of First in First Out Page Replacement Algorithm we remove the frame which contains the page is older among the pages. By removing the older page we give access for the new frame to occupy the empty space created by the First in First out Page Replacement Algorithm.

OPTIMAL Page Replacement Algorithm

This is the second basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as Page Hit.

If the page to be searched is not found among the frames then, this process is known as Page Fault.

When Page Fault occurs this problem arises, then the OPTIMAL Page Replacement Algorithm comes into picture.

The OPTIMAL Page Replacement Algorithms works on a certain principle. The principle is:

Replace the Page which is not used in the Longest Dimension of time in future

This principle means that after all the frames are filled then, see the future pages which are to occupy the frames. Go on checking for the pages which are already available in the frames. Choose the page which is at last.

Example:

Suppose the Reference String is:

0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

6, 1, 2 are in the frames occupying the frames.

Now we need to enter 0 into the frame by removing one page from the page

So, let us check which page number occurs last

From the sub sequence 0, 3, 4, 6, 0, 2, 1 we can say that 1 is the last occurring page number. So we can say that 0 can be placed in the frame body by removing 1 from the frame.

Let us understand this OPTIMAL Page Replacement Algorithm working with the help of an example.

Example:

Consider the reference string 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 4, 0 for a memory with three frames and calculate number of page faults by using OPTIMAL Page replacement algorithms.

Points to Remember

Page Not Found - - - > Page Fault

Page Found - - - > Page Hit

Reference String:

6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 4, 0

S. no	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F3				2	2	2	4	4	4	4	4	4	4	4	4	3	3	3	4	4
F2		1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
F1	6	6	6	6	0	0	0	6	6	2	2	2	2	2	2	2	2	2	2	0
Hit (H)/ Faul t (F)	F	F	H	F	F	F	F	H	H	F	F	H	H	H	H	F	H	F	F	F

Number of Page Hits = 8

Number of Page Faults = 12

The Ratio of Page Hit to the Page Fault = 8 : 12 - - - > 2 : 3 - - - > 0.66

The Page Hit Percentage = 8 *100 / 20 = 40%

The Page Fault Percentage = 100 - Page Hit Percentage = 100 - 40 = 60%

Explanation

First, fill the frames with the initial pages. Then, after the frames are filled we need to create a space in the frames for the new page to occupy.

Here, we would fill the empty spaces with the pages we and the empty frames we have. The problem occurs when there is no space for occupying of pages. We have already known that we would replace the Page which is not used in the Longest Dimension of time in future.

There comes a question what if there is absence of page which is in the frame.

Suppose the Reference String is:

0, 2, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

6, 1, 5 are in the frames occupying the frames.

Here, we can see that page number 5 is not present in the Reference String. But the number 5 is present in the Frame. So, as the page number 5 is absent we remove it when required and other page can occupy that position.

Least Recently Used (LRU) Replacement Algorithm

This is the last basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as Page Hit.

If the page to be searched is not found among the frames then, this process is known as Page Fault.

When Page Fault occurs this problem arises, then the Least Recently Used (LRU) Page Replacement Algorithm comes into picture.

The Least Recently Used (LRU) Page Replacement Algorithms works on a certain principle. The principle is:

Replace the page with the page which is less dimension of time recently used page in the past.

Example:

Suppose the Reference String is:

6, 1, 1, 2, 0, 3, 4, 6, 0

The pages with page numbers 6, 1, 2 are in the frames occupying the frames.

Now, we need to allot a space for the page numbered 0.

Now, we need to travel back into the past to check which page can be replaced.

6 is the oldest page which is available in the Frame.

So, replace 6 with the page numbered 0.

Let us understand this Least Recently Used (LRU) Page Replacement Algorithm working with the help of an example.

Example:

Consider the reference string 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0 for a memory with three frames and calculate number of page faults by using Least Recently Used (LRU) Page replacement algorithms.

Points to Remember

Page Not Found - - - > Page Fault

Page Found - - - > Page Hit

Reference String:

6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

S. no	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
F3				2	2	2	4	4	4	2	2	2	2	2	2	2	2	2	2	2
F2		1	1	1	1	3	3	3	0	0	0	0	0	0	0	0	0	1	1	1
F1	6	6	6	6	0	0	0	6	6	6	1	1	1	1	1	3	3	3	3	0
Hit (H)/ Fault (F)	F	F	H	F	F	F	F	F	F	F	F	H	H	H	H	F	H	F	H	F

Number of Page Hits = 7

Number of Page Faults = 13

The Ratio of Page Hit to the Page Fault = 7 : 12 - - - > 0.5833 : 1

The Page Hit Percentage = 7 * 100 / 20 = 35%

The Page Fault Percentage = 100 - Page Hit Percentage = 100 - 35 = 65%

Explanation

First, fill the frames with the initial pages. Then, after the frames are filled we need to create a space in the frames for the new page to occupy.

Here, we would fill the empty spaces with the pages we and the empty frames we have. The problem occurs when there is no space for occupying of pages. We have already known that we

would replace the Page which is not used in the Longest Dimension of time in past or can be said as the Page which is very far away in the past.

What is Thrash?

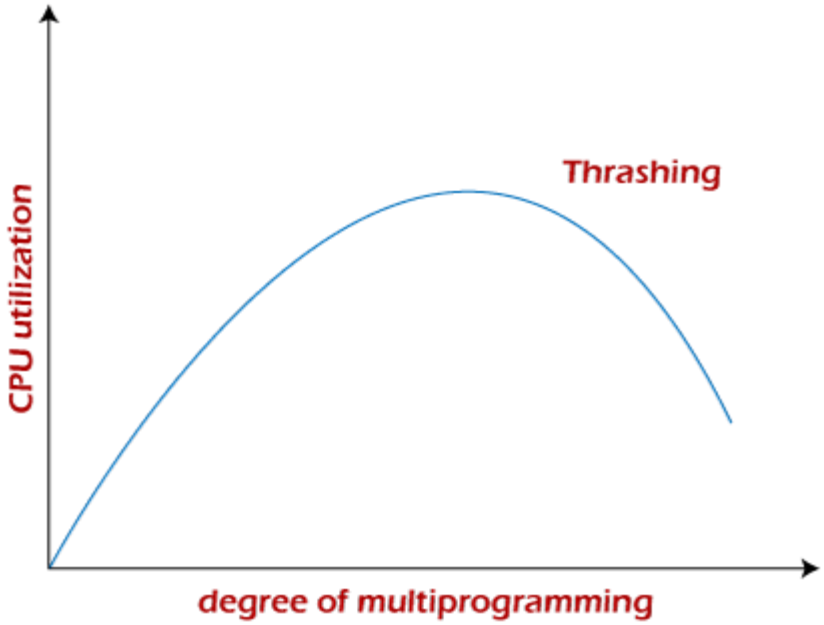
In computer science, ***thrash*** is the poor performance of a virtual memory (or paging) system when the same pages are being loaded repeatedly due to a lack of main memory to keep them in memory. Depending on the configuration and algorithm, the actual throughput of a system can degrade by multiple orders of magnitude.

In computer science, ***thrashing*** occurs when a computer's virtual memory resources are overused, leading to a constant state of paging and page faults, inhibiting most application-level processing. It causes the performance of the computer to degrade or collapse. The situation can continue indefinitely until the user closes some running applications or the active processes free up additional virtual memory resources.

To know more clearly about thrashing, first, we need to know about page fault and swapping.

- **Page fault:** We know every program is divided into some pages. A page fault occurs when a program attempts to access data or code in its address space but is not currently located in the system RAM.
- **Swapping:** Whenever a page fault happens, the operating system will try to fetch that page from secondary memory and try to swap it with one of the pages in RAM. This process is called swapping.

Thrashing is when the page fault and swapping happens very frequently at a higher rate, and then the operating system has to spend more time swapping these pages. This state in the operating system is known as thrashing. Because of thrashing, the CPU utilization is going to be reduced or negligible.



The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults. As a result, no valuable work would be done by the CPU, and the CPU utilization would fall drastically.

The long-term scheduler would then try to improve the CPU utilization by loading some more processes into the memory, thereby increasing the degree of multiprogramming. Unfortunately, this would result in a further decrease in the CPU utilization, triggering a chained reaction of higher page faults followed by an increase in the degree of multiprogramming, called thrashing.

Algorithms during Thrashing

Whenever thrashing starts, the operating system tries to apply either the Global page replacement Algorithm or the Local page replacement algorithm.

1. Global Page Replacement

Since global page replacement can bring any page, it tries to bring more pages whenever thrashing is found. But what actually will happen is that no process gets enough frames, and as a result, the thrashing will increase more and more. Therefore, the global page replacement algorithm is not suitable when thrashing happens.

2. Local Page Replacement

Unlike the global page replacement algorithm, local page replacement will select pages which only belong to that process. So there is a chance to reduce the thrashing. But it is proven that

there are many disadvantages if we use local page replacement. Therefore, local page replacement is just an alternative to global page replacement in a thrashing scenario.

Causes of Thrashing

Programs or workloads may cause thrashing, and it results in severe performance problems, such as:

- If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new system. A global page replacement algorithm is used. The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming.
- CPU utilization is plotted against the degree of multiprogramming.
- As the degree of multiprogramming increases, CPU utilization also increases.
- If the degree of multiprogramming is increased further, thrashing sets in, and CPU utilization drops sharply.
- So, at this point, to increase CPU utilization and to stop thrashing, we must decrease the degree of multiprogramming.

How to Eliminate Thrashing

Thrashing has some negative impacts on hard drive health and system performance. Therefore, it is necessary to take some actions to avoid it. To resolve the problem of thrashing, here are the following methods, such as:

- **Adjust the swap file size:**If the system swap file is not configured correctly, disk thrashing can also happen to you.
- **Increase the amount of RAM:** As insufficient memory can cause disk thrashing, one solution is to add more RAM to the laptop. With more memory, your computer can handle tasks easily and don't have to work excessively. Generally, it is the best long-term solution.
- **Decrease the number of applications running on the computer:** If there are too many applications running in the background, your system resource will consume a lot. And the remaining system resource is slow that can result in thrashing. So while closing, some applications will release some resources so that you can avoid thrashing to some extent.
- **Replace programs:** Replace those programs that are heavy memory occupied with equivalents that use less memory.

Techniques to Prevent Thrashing

The Local Page replacement is better than the Global Page replacement, but local page replacement has many disadvantages, so it is sometimes not helpful. Therefore below are some other techniques that are used to handle thrashing:

1. Locality Model

A locality is a set of pages that are actively used together. The locality model states that as a process executes, it moves from one locality to another. Thus, a program is generally composed of several different localities which may overlap.

For example, when a function is called, it defines a new locality where memory references are made to the function call instructions, local and global variables, etc. Similarly, when the function is exited, the process leaves this locality.

2. Working-Set Model

This model is based on the above-stated concept of the Locality Model.

The basic principle states that if we allocate enough frames to a process to accommodate its current locality, it will only fault whenever it moves to some new locality. But if the allocated frames are lesser than the size of the current locality, the process is bound to thrash.

According to this model, based on parameter A, the working set is defined as the set of pages in the most recent 'A' page references. Hence, all the actively used pages would always end up being a part of the working set.

The accuracy of the working set is dependent on the value of parameter A. If A is too large, then working sets may overlap. On the other hand, for smaller values of A, the locality might not be covered entirely.

If D is the total demand for frames and WSS_i is the working set size for process i,

$D = \sum WSS_i$

Now, if 'm' is the number of frames available in the memory, there are two possibilities:

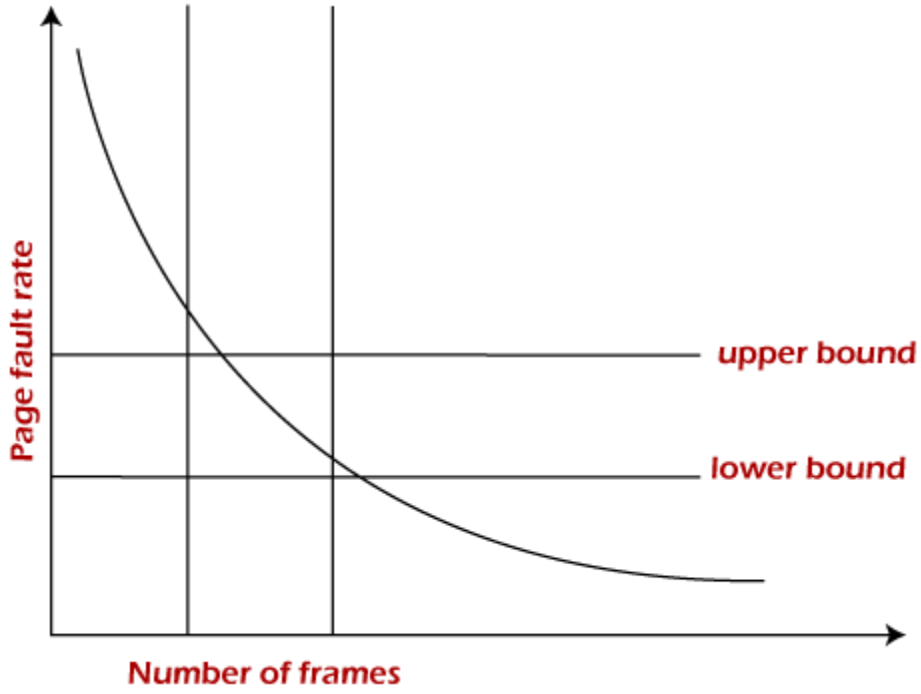
- D>m, i.e., total demand exceeds the number of frames, then thrashing will occur as some processes would not get enough frames.
- D<=m, then there would be no thrashing.

If there are enough extra frames, then some more processes can be loaded into the memory. On the other hand, if the summation of working set sizes exceeds the frames' availability, some of the processes have to be suspended (swapped out of memory).

This technique prevents thrashing along with ensuring the highest degree of multiprogramming possible. Thus, it optimizes CPU utilization.

3. Page Fault Frequency

A more direct approach to handle thrashing is the one that uses the Page-Fault Frequency concept.



The problem associated with thrashing is the high page fault rate, and thus, the concept here is to control the page fault rate.

If the page fault rate is too high, it indicates that the process has too few frames allocated to it. On the contrary, a low page fault rate indicates that the process has too many frames.

Upper and lower limits can be established on the desired page fault rate, as shown in the diagram.

If the page fault rate falls below the lower limit, frames can be removed from the process. Similarly, if the page faults rate exceeds the upper limit, more frames can be allocated to the process.

In other words, the graphical state of the system should be kept limited to the rectangular region formed in the given diagram.

If the page fault rate is high with no free frames, some of the processes can be suspended and allocated to them can be reallocated to other processes. The suspended processes can restart later.