Chapter 5 Memory Management

- Background – Basic hardware, Address binding, Logical versus physical address space, Dynamic loading, Dynamic linking and shared libraries.

Address binding is the process of mapping from one address space to another address space. Logical addresses are generated by CPU during execution whereas physical address refers to location in a physical memory unit (the one that is loaded into memory). Note that users deal only with logical address (virtual address)

Static Linking: When we click the .exe (executable) file of the program and it starts running, all the necessary contents of the binary file have been loaded into the process's virtual address space. However, most programs also need to run functions from the system libraries, and these library functions also need to be loaded. In the simplest case, the necessary library functions are embedded directly in the program's executable binary file. Such a program is statically linked to its libraries, and statically linked executable codes can commence running as soon as they are loaded.

Dynamic Linking: Every dynamically linked program contains a small, statically linked function that is called when the program starts. This static function only maps the link library into memory and runs the code that the function contains. The link library determines what are all the dynamic libraries which the program requires along with the names of the variables and functions needed from those libraries by reading the information contained in sections of the library. After which it maps the libraries into the middle of virtual memory and resolves the references to the symbols contained in those libraries. We don't know where in the memory these shared libraries are actually mapped:

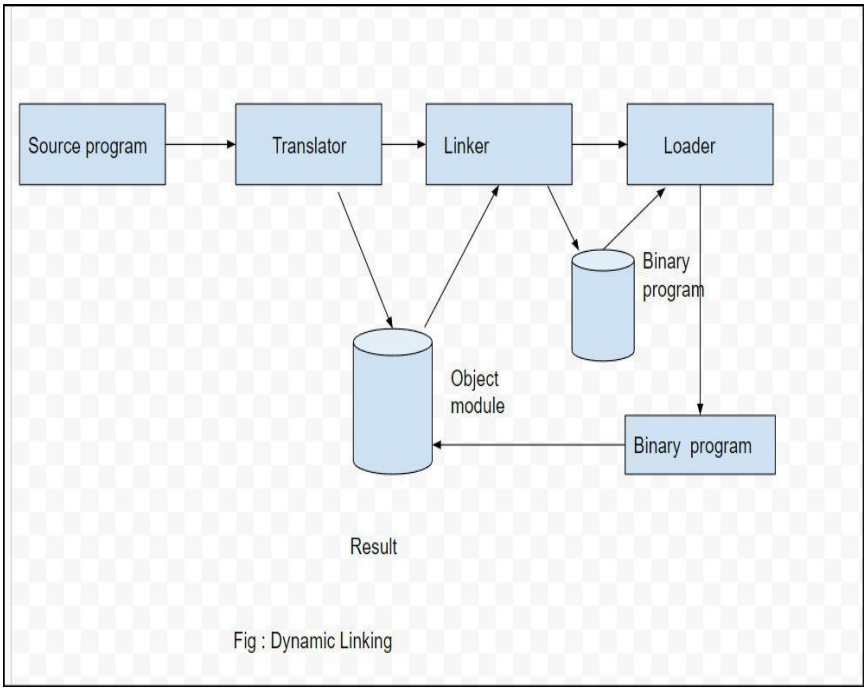


Fig : Dynamic Linking

SWAPPING:- To increase CPU utilization in multiprogramming, a memory management scheme known as swapping can be used. Swapping is the process of bringing a process into memory and then temporarily copying it to the disc after it has run for a while. The purpose of swapping in an operating system is to access data on a hard disc and move it to RAM so that application programs can use it Swapping has been subdivided into two concepts: swap-in and swap-out.

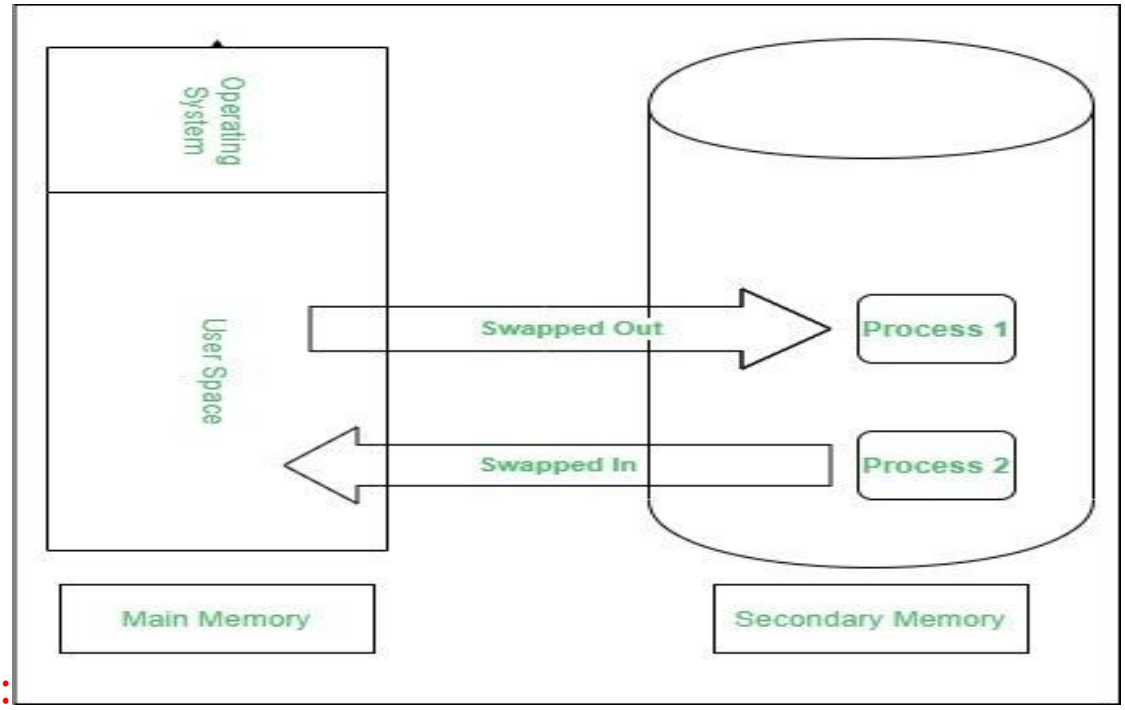
- Swap-out is a technique for moving a process from RAM to the hard disc.
- Swap-in is a method of transferring a program from a hard disc to main memory, or RAM.

Advantages:-

- If there is low main memory so some processes may has to wait for much long but by using swapping process do not have to wait long for execution on CPU.
- It utilize the main memory.
- Using only single main memory, multiple process can be run by CPU using swap partition.
- The concept of virtual memory start from here and it utilize it in better way.
- This concept can be useful in priority based scheduling to optimize the swapping process.

Disadvantages:-

- If there is low main memory resource and user is executing too many processes and suddenly the power of system goes off there might be a scenario where data get erase of the processes which are took parts in swapping.
- Chances of number of page faults occur
- Low processing performance



:

• Contiguous Memory Allocation – Memory mapping and protection, Memory allocation, Fragmentation:-

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.

Memory Mapping

Modern computers have a virtual memory that is not physically present. We can achieve it by setting up a Hard disk, this way extended memory is called virtual memory. So any program that is inside the computer will have a virtual memory address to store data. This data cannot be used unless it is converted to a physical address. So, In short, Memory Mapping is the process done by the Operating System to translate Virtual memory addresses to physical addresses. So, the program can run anytime when the OS loads it as it is required.

Fragmentation:-

Fragmentation is the phenomenon of having unused or wasted memory space between the allocated blocks. There are two types of fragmentation: internal and external. Internal fragmentation occurs when the allocated block is larger than the requested size, leaving some space within the block unused.

• Paging – Basic Method, Hardware support, Protection, Shared Pages:-

In paging, the physical memory is divided into fixed-size blocks called page frames, which are the same size as the pages used by the process. The process's logical address space is also divided into fixed-size blocks called pages, which are the same size as the page frames. When a process requests memory, the operating system allocates one or more page frames to the process and maps the process's logical pages to the physical page frames.
The mapping between logical pages and physical page frames is maintained by the page table, which is used by the memory management unit to translate logical addresses into physical addresses. The page table maps each logical page number to a physical page frame number.

Protection and sharing of memory: Paging allows for the protection and sharing of memory between processes, as each process has its page table that maps its logical address.
Swapping is easy between equal-sized pages and page frames: As pages and frames are of equal size, swapping between them becomes very easy.

- Segmentation – Basic concept, Hardware

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the exact sizes are called segments. Segmentation gives the user's view of the process which paging does not provide. Here the user's view is mapped to physical memory.

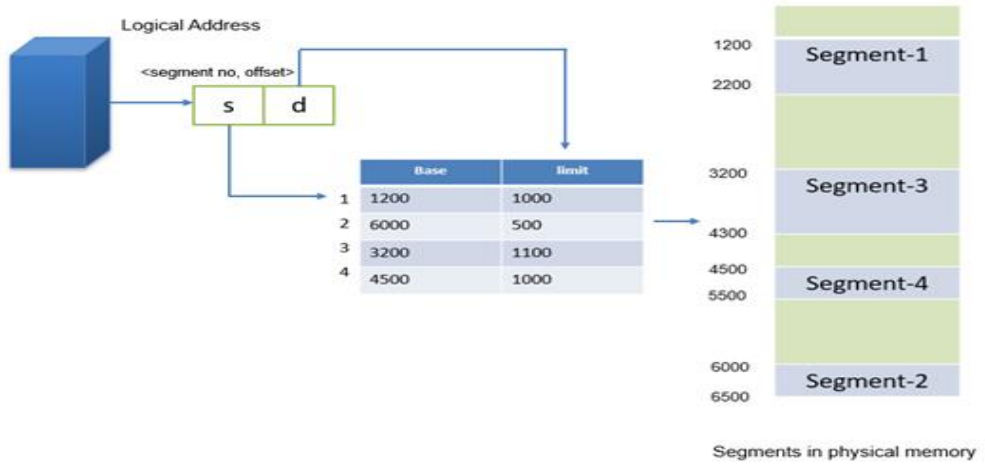
Segment Table

It maps a two-dimensional Logical address into a one-dimensional Physical address. It's each table entry has:

- **Base Address:** It contains the starting physical address where the segments reside in memory.
- **Segment Limit:** Also known as segment offset. It specifies the length of the segment.

The address generated by the CPU is divided into:

- **Segment number (s):** Number of bits required to represent the segment.
- **Segment offset (d):** Number of bits required to represent the size of the segment


Logical Address
<segment no, offset>
s
d
Base
limit
1 1200 1000
2 6000 500
3 3200 1100
4 4500 1000
1200
2200
3200
4300
4500
5500
6000
6500
Segment-1
Segment-3
Segment-4
Segment-2
Segments in physical memory

• Virtual Memory Management – Background, Demand paging, Performance of demand paging,

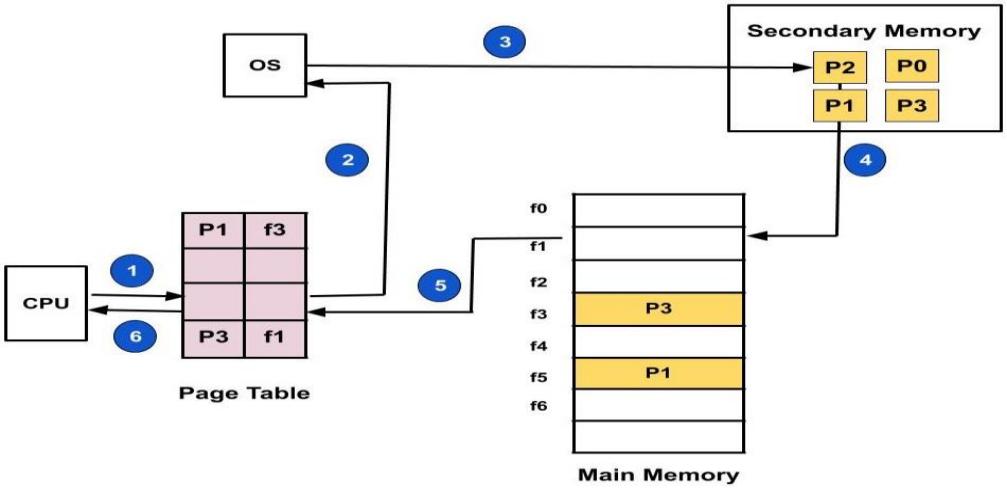
Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites and program-generated addresses are translated automatically to the corresponding machine addresse.

Demand Paging:-

The process of loading the page into memory on demand (whenever a page fault occurs) is known as demand paging. The process includes the following steps are as follows:

1. If the CPU tries to refer to a page that is currently not available in the main memory, it generates an interrupt indicating a memory access fault.
2. The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.
3. The OS will search for the required page in the logical address space.

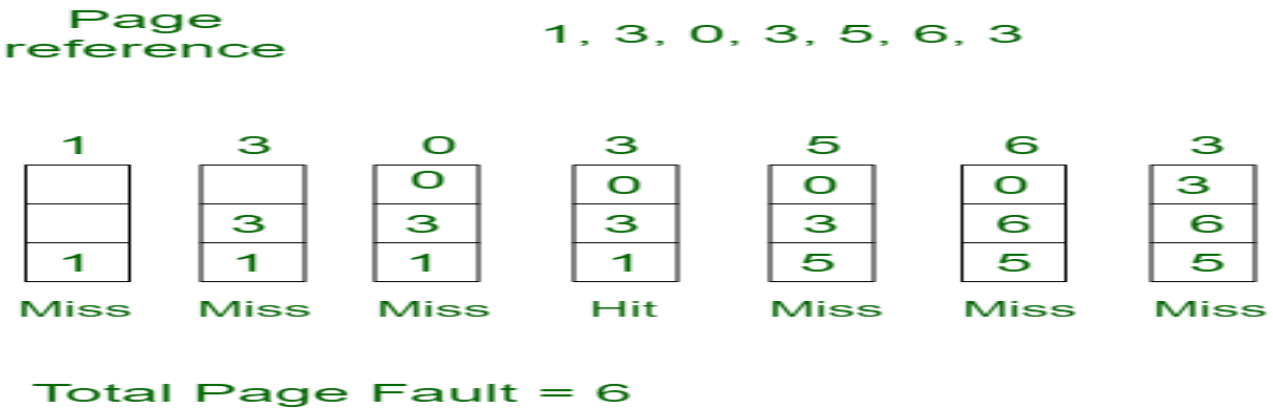
4. The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision-making of replacing the page in physical address space.
5. The page table will be updated accordingly.
6. The signal will be sent to the CPU to continue the program execution and it will place the process back into the ready state.



Page replacement – FIFO, Optimal, LRU, MFU.

. First In First Out (FIFO): This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

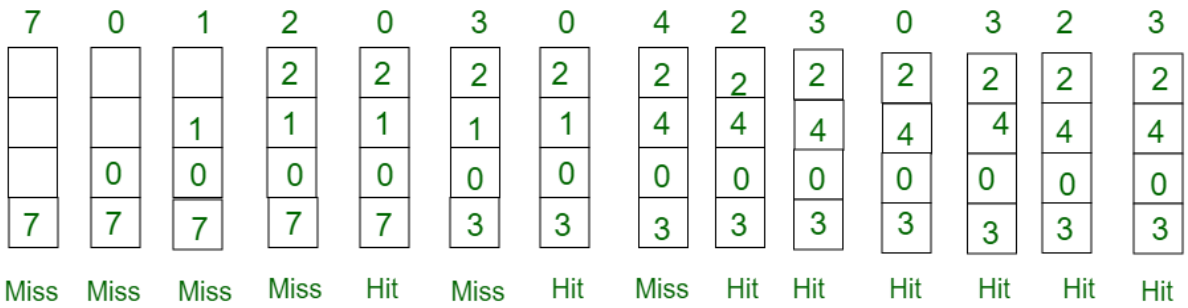
Example 1: Consider page reference string 1, 3, 0, 3, 5, 6, 3 with 3 page frames.Find the number of page faults.



Initially, all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**
when 3 comes, it is already in memory so —> **0 Page Faults.** Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.** 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.** Finally, when 3 come it is not available so it replaces 0 **1 page fault.**
Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference strings 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4, and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10-page faults.

2. Optimal Page replacement: In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.
Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.

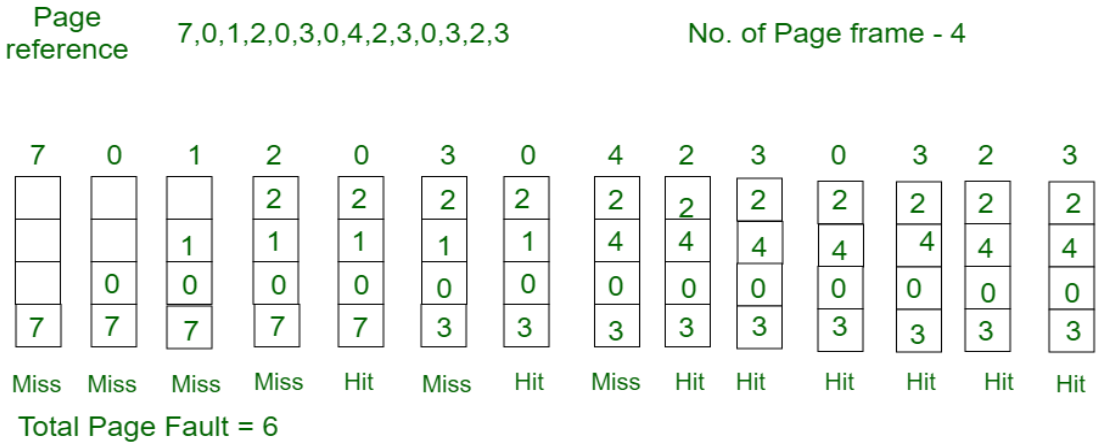
Page reference 7,0,1,2,0,3,0,4,2,3,0,3,2,3 No. of Page frame - 4



Total Page Fault = 6

Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
0 is already there so —> **0 Page fault.** when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.** 0 is already there so —> **0 Page fault.** 4 will takes place of 1 —> **1 Page Fault.**
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.
Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

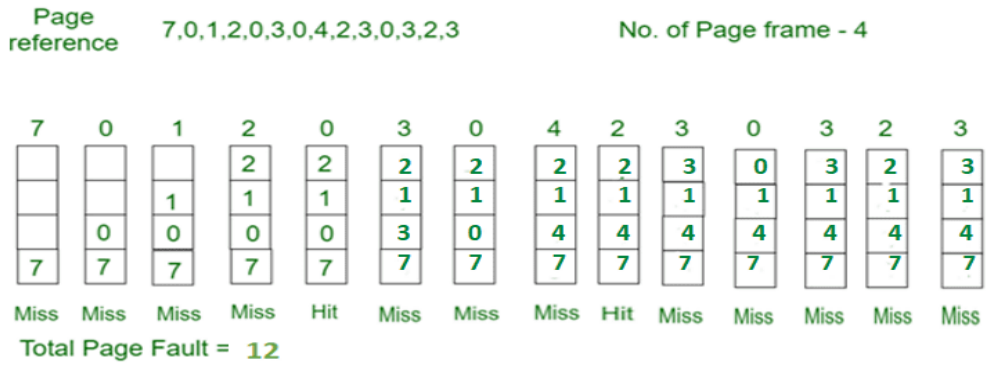
3. Least Recently Used: In this algorithm, page will be replaced which is least recently used.
Example-3: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frames. Find number of page faults.



Here LRU has same number of page fault as optimal but it may differ according to question.

Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
0 is already their so —> **0 Page fault.** when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**
0 is already in memory so —> **0 Page fault**.
4 will takes place of 1 —> **1 Page Fault**
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

4. Most Recently Used (MRU): In this algorithm, page will be replaced which has been used recently. Belady's anomaly can occur in this algorithm.



Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**
0 is already their so–> **0 page fault**
when 3 comes it will take place of 0 because it is most recently used —>**1 Page fault**
when 0 comes it will take place of 3 —>**1 Page fault**
when 4 comes it will take place of 0 —>**1 Page fault**
2 is already in memory so —> **0 Page fault**
when 3 comes it will take place of 2 —>**1 Page fault**
when 0 comes it will take place of 3 —>**1 Page fault**
when 3 comes it will take place of 0 —>**1 Page fault**
when 2 comes it will take place of 3 —>**1 Page fault**
when 3 comes it will take place of 2 —>**1 Page fault**

******************Thank You*************************

Prepared by Prof.Y.J.Patangade

Department of Computer Science

Hutatma Rajgurumhavidyalay

Ref link.:- ref. https://www.geeksforgeeks.org and TYBCS OS-I Vison Text book