DEPARTMENT OF AIML
23CST202- OPERATING SYSTEMS
II YEAR IV SEM AIML-B
UNIT 3-MEMORY MANAGEMENT
TOPIC –DEMAND PAGING

DEMAND PAGING

Demand paging is a memory management scheme used in operating systems to improve memory usage and system performance. Let's understand demand paging with real life example Imagine you are reading a very thick book, but you don't want to carry the entire book around because it's too heavy. Instead, you decide to only bring the pages you need as you read through the book. When you finish with one page, you can put it away and grab the next page you need.

In a computer system, the book represents the entire program, and the pages are parts of the program called "pages" of memory. Demand paging works similarly: instead of loading the whole program into the computer's memory at once (which can be very large and take up a lot of space), the operating system only loads the necessary parts (pages) of the program when they are needed.

This concept says that we should not load any pages into the main memory until we need them, or keep all pages in secondary memory until we need them.

What is Demand Paging?
Demand paging is a technique used in virtual memory systems where pages enter main memory only when requested or needed by the CPU. In demand paging, the operating system loads only the necessary pages of a program into memory at runtime, instead of loading the entire program into memory at the start. A page fault occurred when the program needed to access a page that is not currently in memory.

The operating system then loads the required pages from the disk into memory and updates the page tables accordingly. This process is transparent to the running program and it continues to run as if the page had always been in memory.

What is Page Fault?
The term "page miss" or "page fault" refers to a situation where a referenced page is not found in the main memory.

When a program tries to access a page, or fixed-size block of memory, that isn't currently loaded in physical memory (RAM), an exception known as a page fault happens. Before enabling the program to access a page that is required, the operating system must bring it into memory from secondary storage (such a hard drive) in order to handle a page fault.

In modern operating systems, page faults are a common component of virtual memory management. By enabling programs to operate with more data than can fit in physical memory at once, they enable the efficient use of physical memory. The

operating system is responsible for coordinating the transfer of data between physical memory and secondary storage as needed.

What is Thrashing?

Thrashing is the term used to describe a state in which excessive paging activity takes place in computer systems, especially in operating systems that use virtual memory, severely impairing system performance. Thrashing occurs when a system's high memory demand and low physical memory capacity cause it to spend a large amount of time rotating pages between main memory (RAM) and secondary storage, which is typically a hard disc.

It is caused due to insufficient physical memory, overloading and poor memory management. The operating system may use a variety of techniques to lessen thrashing, including lowering the number of running processes, adjusting paging parameters, and improving memory allocation algorithms. Increasing the system's physical memory (RAM) capacity can also lessen thrashing by lowering the frequency of page swaps between RAM and the disc.

Pure Demand Paging

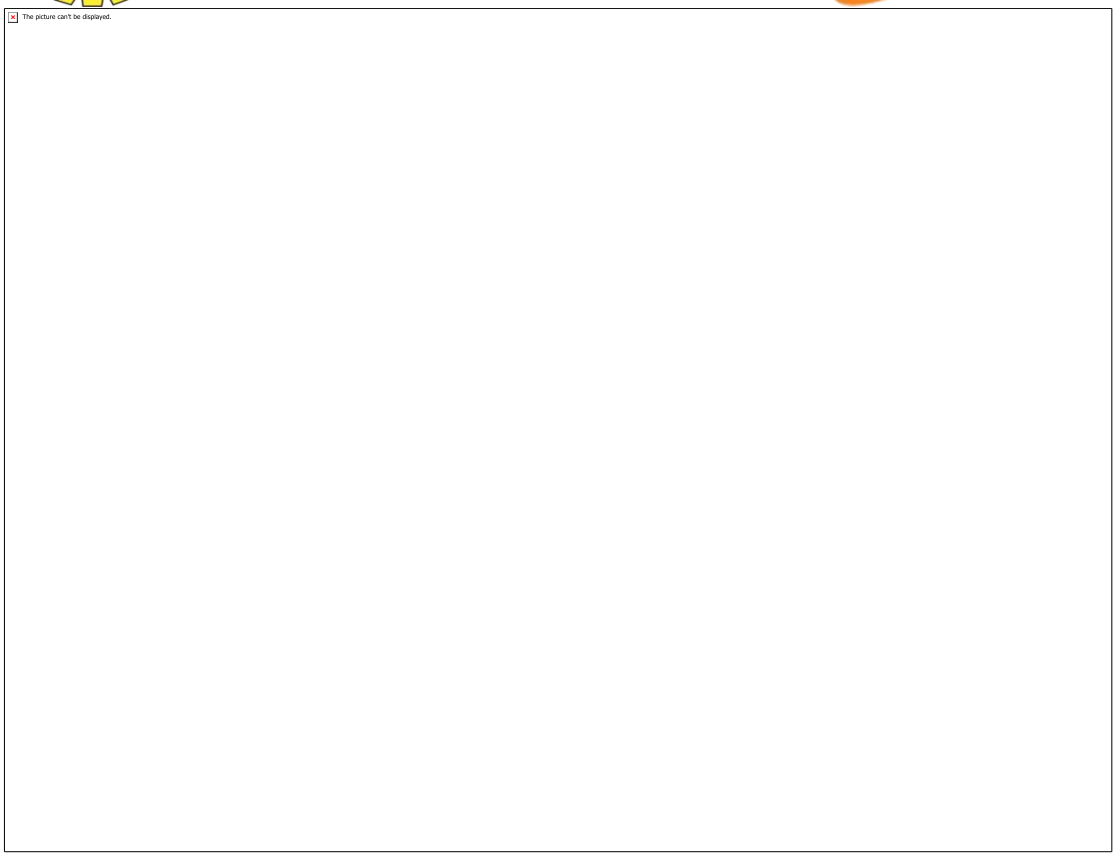
Pure demand paging is a specific implementation of demand paging. The operating system only loads pages into memory when the program needs them. In on-demand paging only, no pages are initially loaded into memory when the program starts, and all pages are initially marked as being on disk.

Operating systems that use pure demand paging as a memory management strategy do so without preloading any pages into physical memory prior to the commencement of a task. Demand paging loads a process's whole address space into memory one step at a time, bringing just the parts of the process that are actively being used into memory from disc as needed.

It is useful for executing huge programs that might not fit totally in memory or for computers with limited physical memory. If the program accesses a lot of pages that are not in memory right now, it could also result in a rise in page faults and possible performance overhead. Operating systems frequently use caching techniques and improve page replacement algorithms to lessen the negative effects of page faults on system performance as a whole.

Working Process of Demand Paging

Let us understand this with the help of an example. Suppose we want to run a process P which have four pages P0, P1, P2, and P3. Currently, in the page table, we have pages P1 and P3.



The operating system‘s demand paging mechanism follows a few steps in its operation.

- **Program Execution:** Upon launching a program, the operating system allocates a certain amount of memory to the program and establishes a process for it.
- **Creating Page Tables:** To keep track of which program pages are currently in memory and which are on disk, the operating system makes page tables for each process.
- **Handling Page Fault:** When a program tries to access a page that isn’t in memory at the moment, a page fault happens. In order to determine whether the necessary page is on disk, the operating system pauses the application and consults the page tables.
- **Page Fetch:** The operating system loads the necessary page into memory by retrieving it from the disk if it is there.
- The page’s new location in memory is then reflected in the page table.
- **Resuming The Program:** The operating system picks up where it left off when the necessary pages are loaded into memory.
- **Page Replacement:** If there is not enough free memory to hold all the pages a program needs, the operating system may need to replace one or more pages currently in memory with pages currently in memory. on the disk. The page replacement algorithm used by the operating system determines which pages are selected for replacement.
- **Page Cleanup:** When a process terminates, the operating system frees the memory allocated to the process and cleans up the corresponding entries in the page tables.

How Demand Paging in OS Affects System Performance?

Demand paging can improve system performance by reducing the memory needed for programs and allowing multiple programs to run simultaneously. However, if not implemented properly, it can cause performance issues. When a program needs a part that isn't in the main memory, the operating system must fetch it from the hard disk, which takes time and pauses the program. This can cause delays, and if the system runs out of memory, it will need to frequently swap pages in and out, increasing delays and reducing performance.

Common Algorithms Used for Demand Paging in OS

Demand paging is a memory management technique that loads parts of a program into memory only when needed. If a program needs a page that isn't currently in memory, the system fetches it from the hard disk. Several algorithms manage this process:

- **FIFO (First-In-First-Out):** Replaces the oldest page in memory with a new one. It's simple but can cause issues if pages are frequently swapped in and out, leading to thrashing.
- **LRU (Least Recently Used):** Replaces the page that hasn't been used for the longest time. It reduces thrashing more effectively than FIFO but is more complex to implement.
- **LFU (Least Frequently Used):** Replaces the page used the least number of times. It helps reduce thrashing but requires extra tracking of how often each page is used.
- **MRU (Most Recently Used):** Replaces the page that was most recently used. It's simpler than LRU but not as effective in reducing thrashing.
- **Random:** Randomly selects a page to replace. It's easy to implement but unpredictable in performance.

What is the Impact of Demand Paging in Virtual Memory Management?

With demand paging, the operating system swaps memory pages between the main memory and secondary storage based on need. When a program needs a page not currently in memory, the operating system retrieves it from secondary storage, a process called a page fault.

Demand paging significantly impacts virtual memory management by allowing the operating system to use virtual memory efficiently, improving overall system performance. Its main advantage is reducing the physical memory required, enabling more applications to run at once and allowing larger programs to run.

However, demand paging has some drawbacks. The page fault mechanism can delay program execution because the operating system must retrieve pages from secondary storage. This delay can be minimized by optimizing the page replacement algorithm.

Demand Paging in OS vs Pre-Paging

Demand paging and pre-paging are two memory management techniques used in operating systems.

Demand paging loads pages from disk into main memory only when they are needed by a program. This approach saves memory space by keeping only the required pages in memory, reducing memory allocation costs and improving

memory use. However, the initial access time for pages not in memory can delay program execution.

Pre-paging loads multiple pages into main memory before they are needed by a program. It assumes that if one page is needed, nearby pages will also be needed soon. Pre-paging can speed up program execution by reducing delays caused by demand paging but can lead to unnecessary memory allocation and waste.

Advantages of Demand Paging

So in the Demand Paging technique, there are some benefits that provide efficiency of the operating system.

- **Efficient use of physical memory**: Query paging allows for more efficient use because only the necessary pages are loaded into memory at any given time.
- **Support for larger programs:** Programs can be larger than the physical memory available on the system because only the necessary pages will be loaded into memory.
- **Faster program start:** Because only part of a program is initially loaded into memory, programs can start faster than if the entire program were loaded at once.
- **Reduce memory usage:** Query paging can help reduce the amount of memory a program needs, which can improve system performance by reducing the amount of disk I/O required.

Disadvantages of Demand Paging

- **Page Fault Overload:** The process of swapping pages between memory and disk can cause a performance overhead, especially if the program frequently accesses pages that are not currently in memory.
- **Degraded Performance:** If a program frequently accesses pages that are not currently in memory, the system spends a lot of time swapping out pages, which degrades performance.
- **Fragmentation:** Query paging can cause physical memory fragmentation, degrading system performance over time.
- **Complexity:** Implementing query paging in an operating system can be complex, requiring complex algorithms and data structures to manage page tables and swap space.