Demand Paging in Operating Systems

Every process in the virtual memory contains lots of pages and in some cases, it might not be efficient to swap all the pages for the process at once. Because it might be possible that the program may need only a certain page for the application to run. Let us take an example here, suppose there is a 500 MB application and it may need as little as 100MB pages to be swapped, so in this case, there is no need to swap all pages at once.

The demand paging system is somehow similar to the paging system with swapping where processes mainly reside in the main memory(usually in the hard disk). Thus demand paging is the process that solves the above problem only by swapping the pages on Demand. This is also known as **lazy swapper**(It never swaps the page into the memory unless it is needed).

Swapper that deals with the individual pages of a process are referred to as **Pager.**

Demand Paging is a technique in which a page is usually brought into the main memory only when it is needed or demanded by the CPU. Initially, only those pages are loaded that are required by the process immediately. Those pages that are never accessed are thus never loaded into the physical memory.

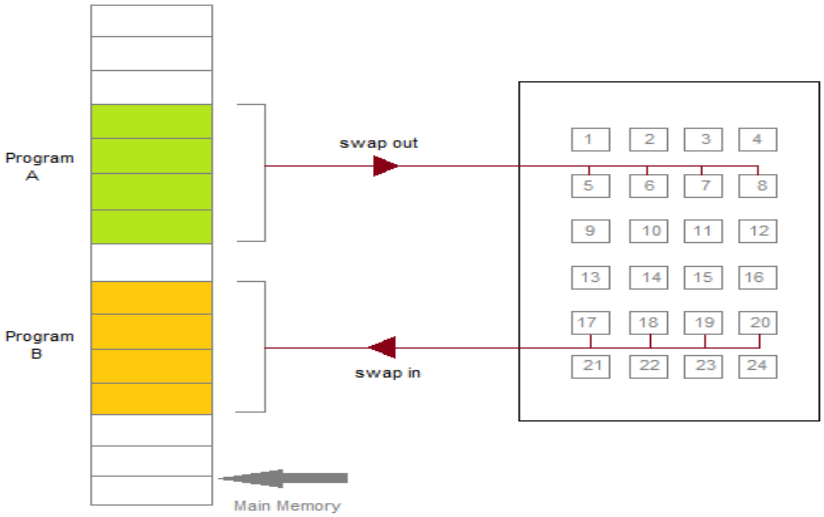


Figure: Transfer of a Paged Memory to the contiguous disk space.

Whenever a page is needed? make a reference to it;

- If the reference is **invalid** then abort it.
- If the page is Not-in-memory then bring it to memory.

Valid-Invalid Bit

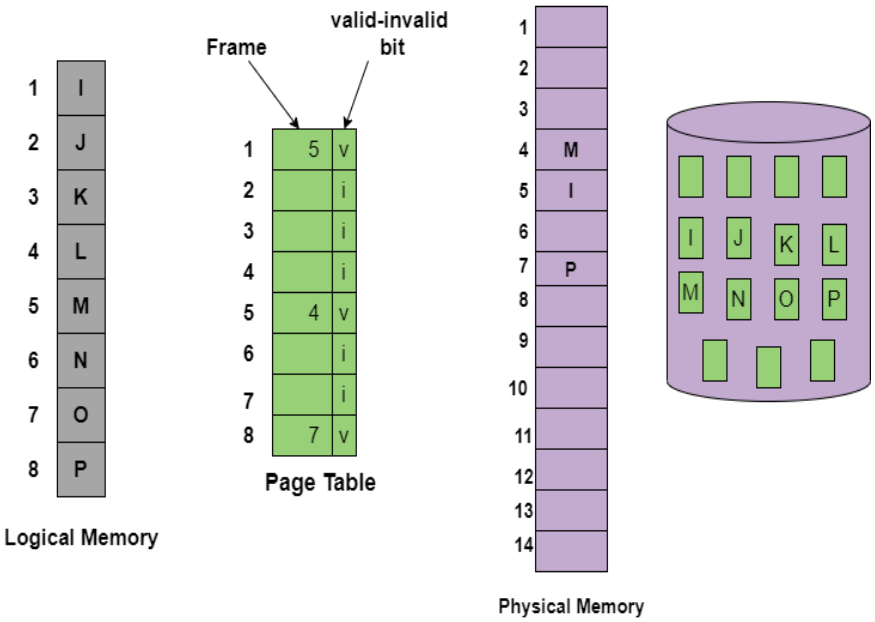
Some form of hardware support is used to distinguish between the pages that are in the memory and the pages that are on the disk. Thus for this purpose Valid-Invalid scheme is used:

- With each page table entry, a valid-invalid bit is associated(where **1** indicates **in the memory** and **0** indicates **not in the memory**)
- Initially, the valid-invalid bit is set to 0 for all table entries.

1. If the bit is set to "**valid**", then the associated page is **both legal and is in memory.**
2. If the bit is set to "**invalid**" then it indicates that the **page is either not valid** or the page is valid **but is currently not on the disk**.

- For the **pages** that are **brought into the memory**, **the page table is set as usual.**
- But for the **pages** that are **not currently in the memory**, the **page table** is either **simply marked as invalid** or it contains the **address of the page on the disk**.

During the translation of address, if the valid-invalid bit in the page table entry is 0 then it leads to **page fault.**



How Demand Paging Works?

First of all the components that are involved in the Demand paging process are as follows:

- Main Memory
- CPU
- Secondary Memory
- Interrupt
- Physical Address space
- Logical Address space
- Operating System
- Page Table

1. If a page is not available in the main memory in its active state; then a request may be made to the CPU for that page. Thus for this purpose, it has to generate an interrupt.
2. After that, the Operating system moves the process to the blocked state as an interrupt has occurred.
3. Then after this, the Operating system searches the given page in the Logical address space.
4. And Finally with the help of the page replacement algorithms, replacements are made in the physical address space. Page tables are updated simultaneously.
5. After that, the CPU is informed about that update and then asked to go ahead with the execution and the process gets back into its ready state.

When the process requires any of the pages that are not loaded into the memory, a page fault trap is triggered and the following steps are followed,

1. The memory address which is requested by the process is first checked, to verify the request made by the process.
2. If it is found to be invalid, the process is terminated.
3. In case the request by the process is valid, a free frame is located, possibly from a free-frame list, where the required page will be moved.
4. A new operation is scheduled to move the necessary page from the disk to the specified memory location. (This will usually block the process on an I/O wait, allowing some other process to use the CPU in the meantime.)

1. When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to valid.
2. The instruction that caused the page fault must now be restarted from the beginning.

Advantages of Demand Paging

The benefits of using the Demand Paging technique are as follows:

- With the help of Demand Paging, memory is utilized efficiently.
- Demand paging avoids External Fragmentation.
- Less Input/Output is needed for Demand Paging.
- This process is not constrained by the size of physical memory.
- With Demand Paging it becomes easier to share the pages.

- With this technique, portions of the process that are never called are never loaded.
- No compaction is required in demand Paging.

Disadvantages of Demand paging

Drawbacks of Demand Paging are as follows:

- There is an increase in overheads due to interrupts and page tables.
- Memory access time in demand paging is longer.

Pure Demand Paging

In some cases when initially no pages are loaded into the memory, pages in such cases are only loaded when are demanded by the process by generating page faults. It is then referred to as **Pure Demand Paging**.

- In the case of pure demand paging, there is not even a single page that is loaded into the memory initially. Thus pure demand paging causes the page fault.
- When the execution of the process starts with no pages in the memory, then the operating system sets the instruction pointer to the first instruction of the process and that is on a non-memory resident page and then in this case the process immediately faults for the page.
- After that when this page is brought into the memory then the process continues its execution, page fault is necessary until every page that it needs is in the memory.
- And at this point, it can execute with no more faults.
- This scheme is referred to as Pure Demand Paging: means never bring a page into the memory until it is required.