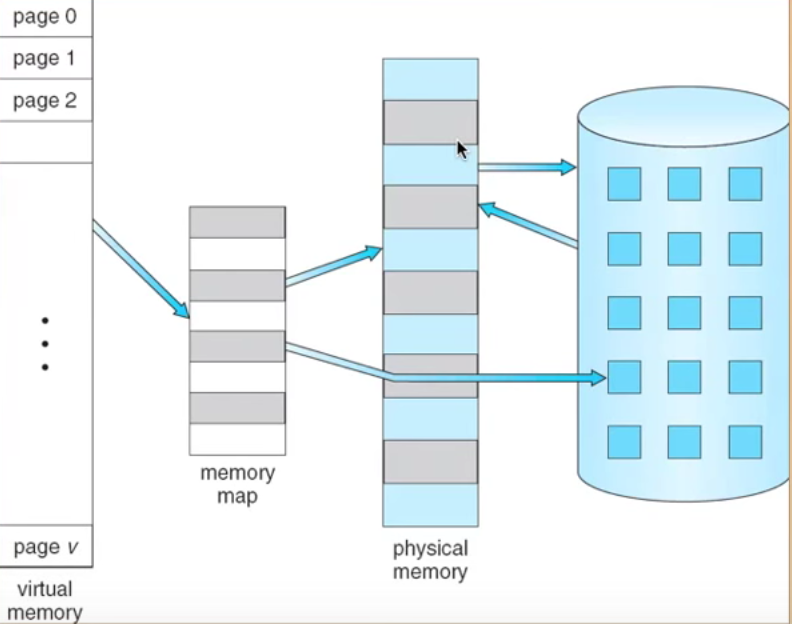
**Chapter 9 Virtual Memory**

Background

* **Virtual memory** involves the separation of logical memory as perceived by user from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when a smaller physical memory is available.
* Only part of program needs to be in memory for execution
* Logical address space can therefore be much **larger than** physical address space



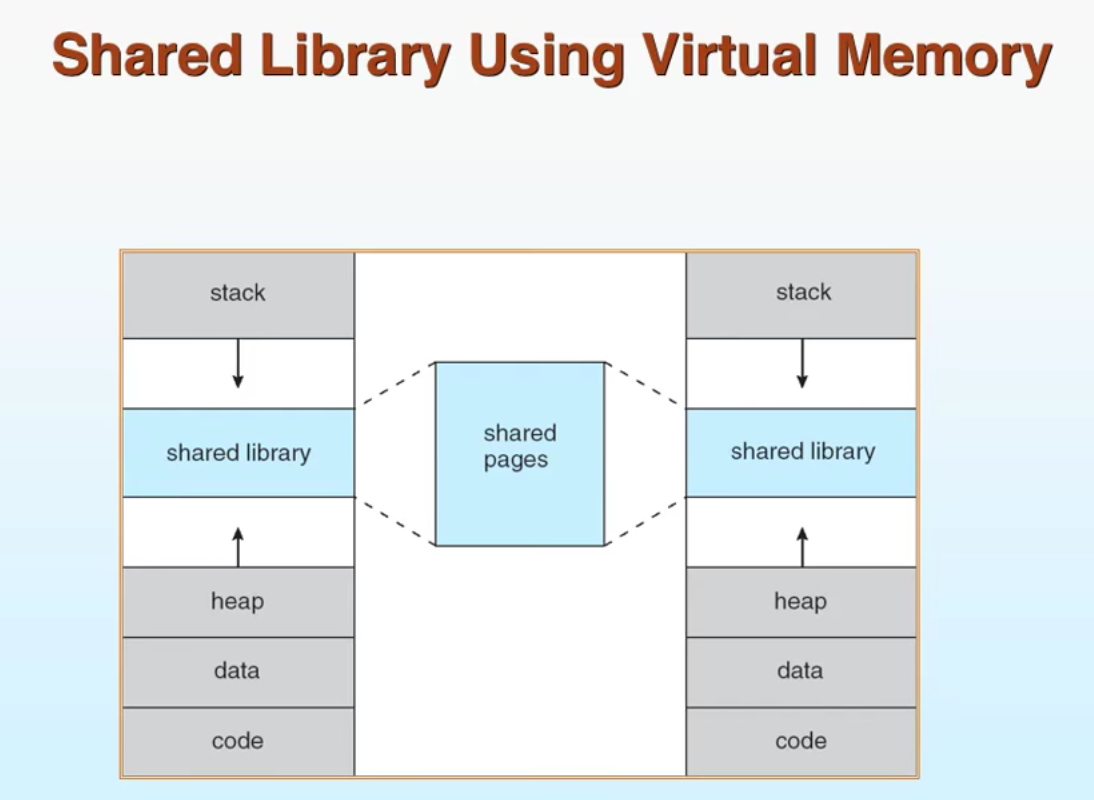
The ability to execute a program that is only partially in memory would be confer many benefits:

1.A program would no longer be constrained by the amount of physical memory that is available.

2.Because each user program could take less physical memory, more programs could be run at the same time, with a corresponding increase in CPU utilization and throughout but with no increase in response time or turnaround time

3. Less I/O would be needed to load swap each user program into memory, so user program would run faster

* Separating logical memory from physical memory, VM allows files and memory to be shared by two or more process through page sharing. Benefits



1. System libraries can be shared by several processes through mapping of the shared object into a virtual address space. Although each process considers the shared libraries to be part of its virtual address space, the actual pages where the libraries reside in physical memory are shared by all process. Typically, a library is mapped read-only into the space of each process that is linked with it.

2.VM enables processes to share memory which process can communicate with other processes. VM allows one processes to create a region of memory that it can share with other process. Processes sharing this region consider it part of VM space, yet the actual physical pages of memory are shared

* Allows for more efficient process creation, because the programmer no longer needs to worry about the amount of physical memory available.
* Virtual memory can be implemented via:
* Demand paging --- bring a page into memory only when it is needed

1. Less I/O needed
2. Less memory needed
3. Faster response
4. More users

* Page is needed => reference to it

1. Invalid reference => abort
2. Not in memory => bring it to memory

* **Lazy swapper** --- news swaps a page into memory unless page will be needed

1. Swapper that deals with pages is a **pager**

* Demand segmentation