Good afternoon maam..my name is garima pandey and I am here to present the topic..virtual memory in mac os
so before moving further..i would like to brief what is virtual memory
Virtual memory allows an operating system to escape the limitations of physical RAM. A virtual memory manager creates a logical address space (or "virtual" address space) that is larger than the installed physical memory (RAM) and divides it up into uniformly-sized chunks of memory called pages. Each page in the logical address space has a corresponding page on the disk, in a special file known as the backing store.

There are two key features of the processor and its memory management unit (MMU) that you must grasp in order to understand how virtual memory works. The first is the page table, which is a table that maps all logical pages into their corresponding physical pages. When the processor accesses a logical address, the MMU uses the page table to translate the access into a physical address, which is the address that's actually passed to the computer's memory subsystem.
If the translation from a logical page address to a physical address fails, a page fault occurs. The virtual memory system invokes a special page-fault handler to stop executing the current code and respond to the fault. The page-fault handler finds a free page of physical memory, transfers the data from the backing store to the physical page, and then updates the page table so that the page now appears to be at the correct logical address.
If no free pages are available in physical memory, the handler must first release an existing page. If that page contains modified data, the handler writes its contents to the backing store before releasing it. This process is known as paging.

NEXT SLIDE

In Mac OS X, each process has its own sparse 32-bit virtual address space. Thus, each process has an address space that can grow dynamically up to a limit of four gigabytes. As an application uses up space, the virtual memory system allocates additional swap file space on the root file system.

The virtual address space of a process consists of mapped regions of memory. Each region of memory in the process represents a specific set of virtual memory pages. A region has specific attributes controlling such things as inheritance (portions of the region may be mapped from "parent" regions), write-protection, and whether it is "wired" (that is, it cannot be paged out). Because regions contain a given number of pages, they are page-aligned, meaning the starting address of the region is also the starting address of a page and the ending address also defines the end of a page.

NEXT SLIDE

Now..the role of kernel

The kernel associates a VM object with each region of the virtual address space. The kernel uses the VM object to track and manage the resident and nonresident pages of that region. A region can map either to an area of memory in the backing store or to a specific filemapped file in the file system. The VM object maps regions in the backing store through the default pager and maps file-mapped files through the vnode pager. The default pager is a system manager that maps the nonresident virtual memory pages to backing store and fetches those pages when requested. The vnode pager implements file mapping. The vnode pager uses the paging mechanism to provide a window directly into a file. This mechanism lets you read and write portions of the file as if they were located in memory.

A VM object may point to a pager or to another VM object. The kernel uses this self referencing to implement a form of page-level sharing known as copy-on-write. Copy-on-write allows multiple blocks of code (including different processes) to share a page as long as none write to that page. If one process writes to the page, a new, writable copy of the page is created in the address space of the process doing the writing. This mechanism allows the system to copy large quantities of data efficiently.

NEXT SLIDE

Now some key points or terminologies related..firstly

The kernel maintains and queries three system-wide lists of physical memory pages:
- The active list contains pages that are currently mapped into memory and have been recently accessed.
- The inactive list contains pages that are currently resident in physical memory but have not been accessed recently.
- The free list contains pages of physical memory that are not associated with any address space of VM object.

now talking about the jobs of vm manager

Allocating and Accessing Virtual Memory

Applications usually allocate memory using the malloc routine. This routine finds free space on an existing page or allocates new pages using vm_allocate to create space for the new memory block.

At this point there are no pages resident in physical memory and no pages in the backing store. Everything is mapped virtually within the system. When a program accesses the region, by reading or writing to a specific address in it, a fault occurs because that address has not been mapped to physical memory. The kernel also recognizes that the VM object has no backing store for the page on which this address occurs.

It maps the virtual page to the physical page by filling in a data structure called the pmap. The pmap contains the page table used by the processor (or by a separate memory management unit) to map a given virtual address to the actual hardware address

The next one is paging the virtual memory in and out

Lastly I would brief the concept of shared and wired memory

Shared memory is memory that can be written to or read from by two or more processes.

Wired memory (also called resident memory) stores kernel code and data structures that should never be paged out to disk.

Thank you maam..