Operating System (OS)
CS232

Beyond Physical Memory: Mechanisms

Dr. Muhammad Mobeen Movania

Outlines

- Illusion of large virtual address space
- What is swap space?
- The Present Bit
- What happens at page fault?
- What if memory is full?
- Page fault control flow
- When replacements really occur?
- Summary

Illusion of large virtual address space

- Usually, we assume that process address space can fit within the available physical memory
- To give illusion of large virtual address space, OS uses the next level of memory hierarchy, the hard disk
  - Pages are moved to hard disk if they cant be accommodated in system memory
  - Slow system response if there is a lot of page swapping

Swap Space

- Some space on the hard disk for moving pages back and forth from main memory

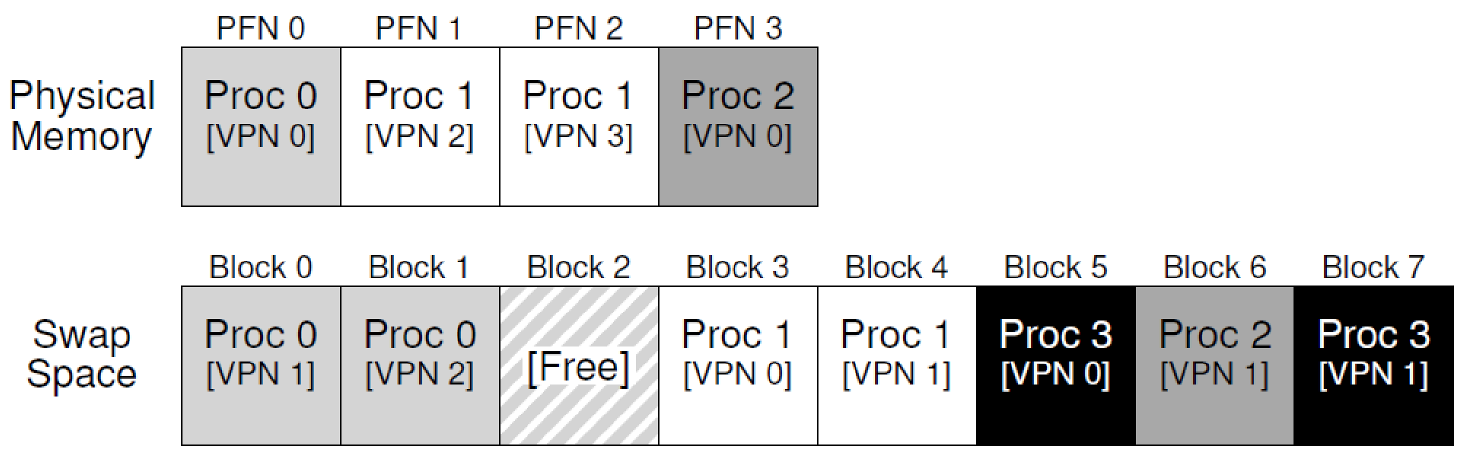

Figure 21.1: **Physical Memory and Swap Space**

The Present Bit

- A bit in the page table entry
- Tells if this page is in physical memory or on disk

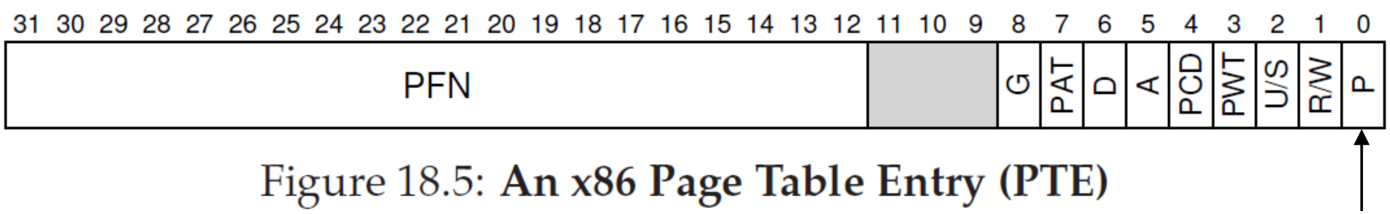

Figure 18.5: **An x86 Page Table Entry (PTE)**

- Accessing a page which is not in physical memory is called ***page fault***

What happens at page fault?

- Irrespective of hardware or software managed TLB, OS handles page fault
  - OS page fault handler runs
  - If a page is not present and has been swapped to disk, the OS will need to swap the page into memory in order to service the page fault
  - When the disk I/O completes, the OS will then
    - update the page table to mark the page as present
    - update the PFN field of the page-table entry (PTE) to record the in-memory location of the newly-fetched page
    - Finally, retry the instruction

What if memory is full?

- OS might like to first **page out** one or more pages to make room for the new page(s) the OS is about to bring in.
- The process of picking a page to kick out, or **replace** is known as the **page-replacement policy**.

Page Fault Control Flow

```
VPN = (VirtualAddress & VPN_MASK) >> SHIFT
(Success, TlbEntry) = TLB_Lookup(VPN)
if (Success == True)   // TLB Hit
    if (CanAccess(TlbEntry.ProtectBits) == True)
        Offset   = VirtualAddress & OFFSET_MASK
        PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
        Register = AccessMemory(PhysAddr)
    else
        RaiseException(PROTECTION_FAULT)
else                   // TLB Miss
    PTEAddr = PTBR + (VPN * sizeof(PTE))
    PTE = AccessMemory(PTEAddr)
    if (PTE.Valid == False)
        RaiseException(SEGMENTATION_FAULT)
    else
        if (CanAccess(PTE.ProtectBits) == False)
            RaiseException(PROTECTION_FAULT)
        else if (PTE.Present == True)
            // assuming hardware-managed TLB
            TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
            RetryInstruction()
        else if (PTE.Present == False)
            RaiseException(PAGE_FAULT)
```

Figure 21.2: **Page-Fault Control Flow Algorithm (Hardware)**

When replacements really occur?

- Most operating systems thus have some kind of **high watermark** (HW) and **low watermark** (LW) to help decide when to start evicting pages from memory
  - when the OS notices that there are fewer than LW pages available, a background thread (swap daemon) is responsible for freeing memory runs
  - It evicts pages until there are HW pages available

Summary

- We have introduced the notion of accessing more memory than is physically present within a system
- Page table must have a bit (**present bit**) to tell if a requested page in memory or not
- If the page is not present
  - OS **page-fault handler** runs to service the **page fault**, and thus arranges for the transfer of the desired page from disk to memory
- All of this happens transparent to the process