Operating Systems Design (CS 423)

Elsa L Gunter
2112 SC, UIUC
http://www.cs.illinois.edu/class/cs423/

Based on slides by Roy Campbell, Sam King, and Andrew S Tanenbaum

Address Spaces and Memory

- Process = one or more threads in an address space
- Thread: stream of execution
  - Unit of concurrency
- Address space: memory space that threads use
  - Unit of data

Address Space Abstrac2on

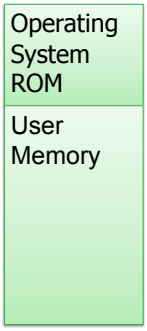
- Address space: all the memory data
  - Program code, stack, data segment
- Hardware interface (physical reality)
  - One memory, small, shared
- Application interface (illusion)
  - Each process has own memory, large

Illusions Provided by Address Apace

- Address independence
  - Same address in different processes not conflicting with each other
  - Eg Same address for stack
- Protection
  - One process cannot access the data of another
  - Secret data, protected code
- Virtual memory
  - 64 bit address space, memory many 4 G

Uni-programming

- One process runs at a time
- Always load process into the same spot
- How do you switch processes?
- What abstractions
- does this provide?
- Problems?



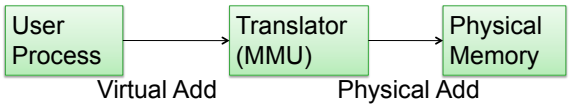
Multi-programming

- Multi-programming: more than one process in memory at a time
- Need address translation
  - Need protection
- Address translation
  - Avoid conflicting addresses
  - Static: before you execute
  - Dynamic: during execution, could change

Dynamic translation

- Translate every memory reference from virtual address to physical address
- Virtual address: an address used by the user process to reference a location in memory
- Physical address: an address used by the physical memory

Dynamic Address Translation



- Translation enforces protection
  - One process can't even refer to another process's address space
- Translation enables virtual memory
  - A virtual address only needs to be in physical memory when it is being accessed
- Change translations on the fly as different virtual addresses occupy physical memory
- Do you need hardware support?

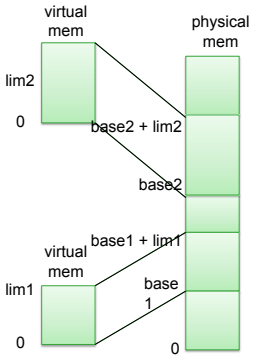
Address translation

- Lots of ways to implement, remember the big picture
- Tradeoffs:
  - Flexibility (e.g., sharing, growth, virtual memory)
  - Size of translation data
  - Speed of translation

Base and Limit

- Load each process into contiguous regions of physical memory

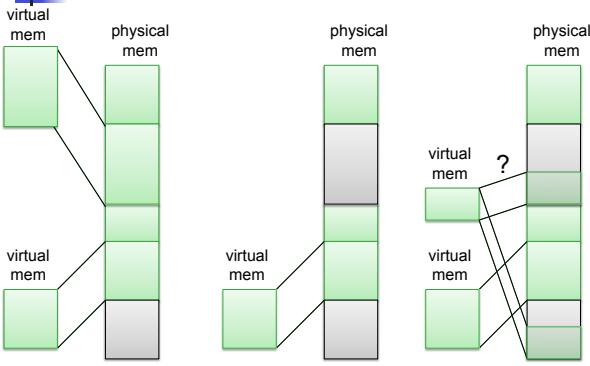
```
If(virt addr > bound){
   trap to kern
} else {
   phys addr =
    virt addr + base
}
```



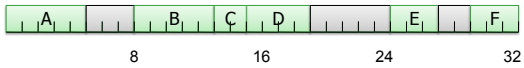
Base and limit

- What must change during a context switch?
- Can a process change its own base and limit?
- Can you share memory with another process?
- How does the kernel handle the address space growing
  - You are the OS designer, come up with an algorithm for allowing processes to grow

Memory Allocation in the small

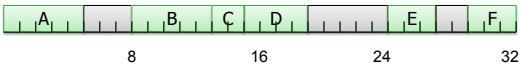


Memory Management w/ Linked Lists



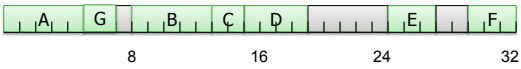
- Info for each hole
  - Both ends:
    - Size of hole
    - Bit saying free or allocated
    - Together
  - Next and Previous pointers to other holes
- Info for each allocated "chunk:
  - Size and allocation bit, together, on each end

Memory Allocation Algorithms

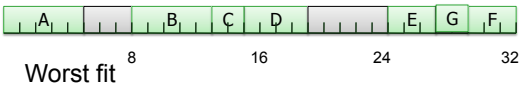


Allocated 2 units:

- First fit



- Best fit



- Worst fit