Topics in Memory Management (Review)

- Uniprogrammed operating systems
  - Assembling, linking, loading
  - Static memory allocation
  - Dynamic memory allocation
    - Stacks, heaps
    - Managing the free list, memory reclamation
- Multiprogrammed operating systems
  - Includes most of the above topics
  - Static relocation
  - Dynamic relocation
    - Virtual vs. physical address
    - Partitioning (and compaction)
    - Segmentation
    - Paging
  - Swapping
  - Demand paging

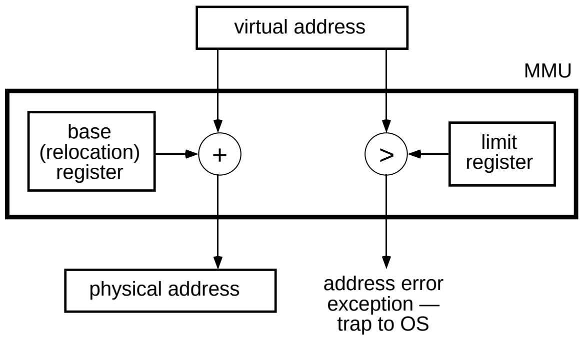
Dynamic Relocation (Review)

- There are now two different views of the address space:
  - The *physical address space* — seen only by the OS — is as large as there is physical memory on the machine
  - The *virtual (logical) address space* —seen by the process — can be as large as the instruction set architecture allows
    - For now, we'll assume it's much smaller than the physical address space
  - Multiple processes share the physical memory, but each can see only its own virtual address space
- The OS and hardware must now manage two different addresses:
  - *Virtual address* — seen by the process
  - *Physical address* — address in physical memory (seen by OS)

Static vs. Dynamic Relocation (Review)

- Problems with static relocation:
  - Safety — not satisfied — one process can access / corrupt another's memory, can even corrupt OS's memory
  - Processes can not change size (why…?)
  - Processes can not move after beginning to run (why would they want to?)
  - Used by MS-DOS, and early versions of Windows and Mac OS
- An alternative: dynamic relocation
  - The basic idea is to change each memory address dynamically <u>as the process runs</u>
  - Translation done by hardware — between the CPU and the memory is a *memory management unit* (MMU) that converts virtual addresses to physical addresses
    - This translation happens for every memory reference the process makes

Implementing Dynamic Relocation



- MMU protects address space, and translates virtual addresses
  - *Base register* holds base physical address of process, *limit register* holds highest virtual address of process
  - Translation:
    physical address = virtual address + base
  - Protection:
    if virtual address > limit, then trap to the OS with an address exception

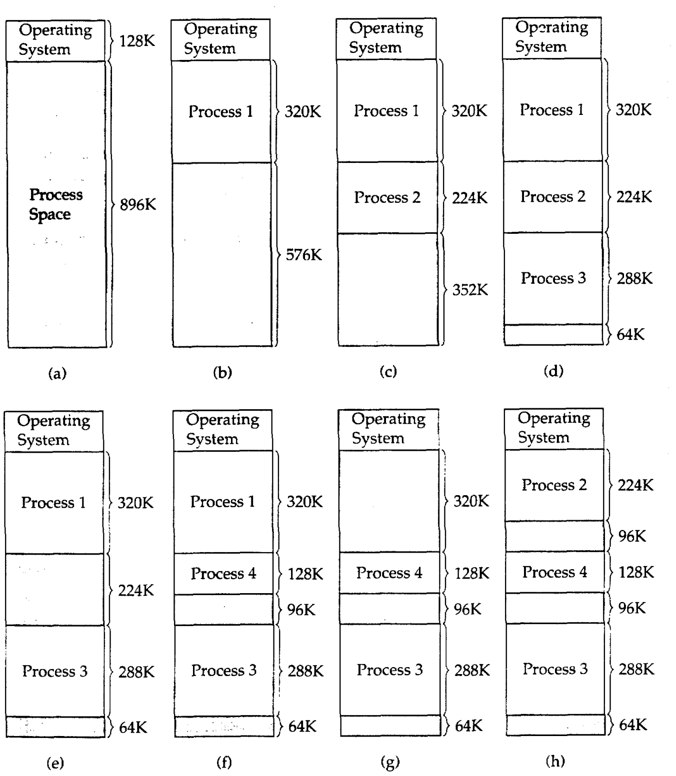
Dynamic Relocation — OS vs. User Programs

- User programs (processes) address their own virtual memory
  - Run in relocation mode — indicated by a bit in the PSW — and in user mode
    - User programs can not change the relocation mode
- OS directly addresses physical memory
  - OS runs with relocation turned off, and in kernel mode
- When user program makes a system call:
  - CPU atomically goes into kernel mode, turns off relocation, traps to trap handler
  - OS trap handler accesses physical memory and does whatever is necessary to service the system call
  - CPU atomically turns on relocation, goes into user mode, returns to user program

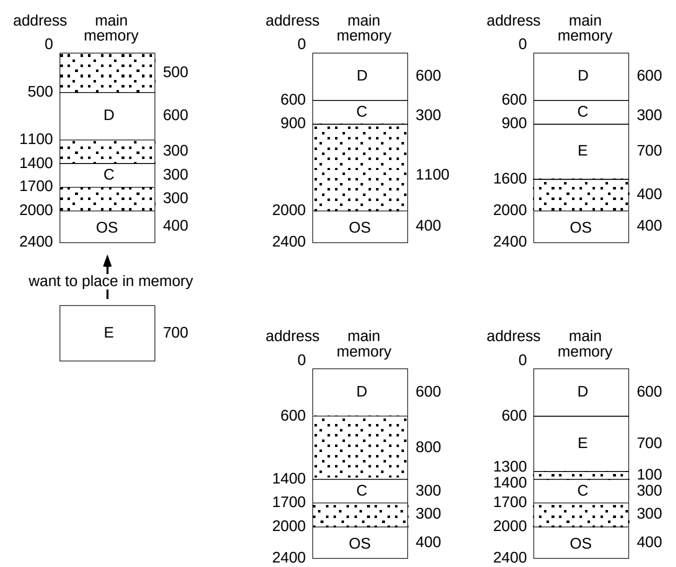
Dynamic Relocation and Partitioning

- Physical memory is divided into *partitions*
  - A process is loaded into a free partition (a "hole" in the memory space)
- Fixed-size partitions:
  - Memory is divided into a predetermined number of fixed-size partitions
    - Partitions may be either of equal size, or of different (although fixed) sizes
  - Use first-fit, best-fit, etc. as discussed for dynamic allocation of heaps
  - Number of partitions limits the *degree of multiprogramming* — number of active processes
- Dynamic (variable-size) partitions:
  - When a process gets brought into memory, it is allocated a partition of exactly the right size

Effect of Dynamic Relocation with Dynamic Partitioning



Compaction



- Evaluation:
  - Memory moved =
  - Space created =

Swapping (Medium-Term Scheduling)

- If there isn't room enough in memory for all processes, some processes can be swapped out to make room
  - OS *swaps a process out* by storing its complete state to disk
  - OS can reclaim space used (not really…) by ready or blocked processes
- When process becomes active again, OS must *swap* it back *in* (into memory)
  - With static relocation, the process must be replaced in the same location
  - With dynamic relocation, OS can place the process in any free partition (must update the relocation and limit registers)
- Swapping and dynamic relocation make it easy to increase the size of a process and to compact memory (although slow!)

UNIX Process Model (From Lecture 06)

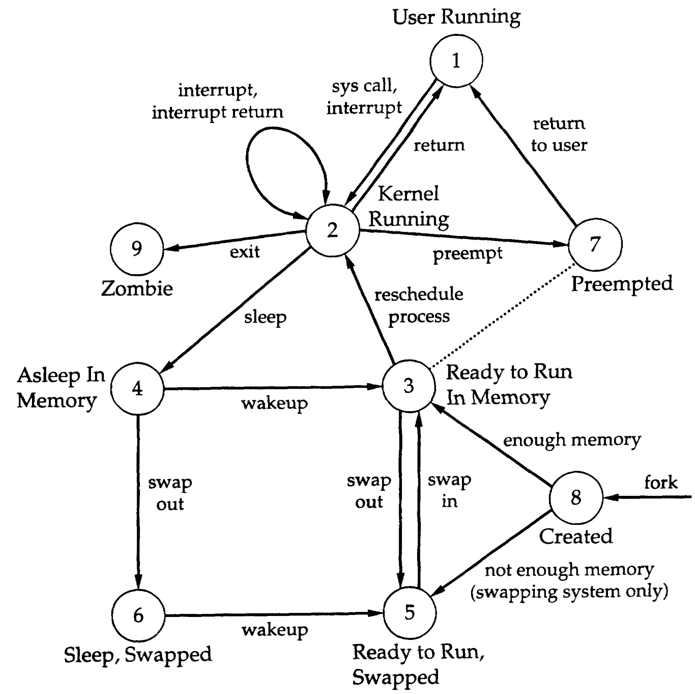


FIGURE 3.16 UNIX process state transition diagram [BACH86]

Figure from *Operating Systems*, 2nd edition, Stallings, Prentice Hall, 1995

Original diagram from *The Design of the UNIX Operating System*, M. Bach, Prentice Hall, 1986

Evaluation of Dynamic Relocation

- Advantages:
  - OS can easily move a process
  - OS can allow processes to grow
  - Hardware changes are minimal, but fairly fast and efficient
  - ➥Transparency, safety, and efficiency are all satisfied, although there is some small overhead to dynamic relocation
- Disadvantages:
  - Compared to static relocation, memory addressing is slower due to translation
  - Memory allocation is complex (partitions, holes, fragmentation, etc.)
  - If process grows, OS may have to move it
  - Process limited to physical memory size
  - Not possible to share code or data between processes