Memory Management

Questions answered in these notes

- How do processes share main memory?
- What are the advantages and disadvantages of static relocation?
- What are the advantages and disadvantages of dynamic relocation?
- How does the memory management interact with hardware?

Readings for this lecture

Silberschatz/Galvin: Chapter 8

Goals for Multiprogramming

Sharing

- Several processes to coexist in main memory

Transparency

- Processes not aware that memory is shared
- Run regardless of number and/or locations of processes

Protection

- Cannot corrupt OS or other processes
- Privacy: Cannot read data of other processes

Efficiency should not be severely degraded

- Purpose of sharing is to increase efficiency
- Do not waste CPU or memory resources

Static Relocation

Transparency --> Relocation

- Processes can run anywhere in memory (can’t predict in advance)
- Modify addresses statically (similar to linker) when load process

Advantages

- Allows multiple processes to run
- Requires no hardware support
Disadvantages of Static Relocation

Process allocation must be contiguous
- Fragmentation: May not be able to allocate new process
- What kind?
- Processes may not be able to increase address space
- Can’t move process after it has been placed?

No protection: Destroy other processes and/or the OS

Dynamic Relocation

Change address dynamically at every reference

Program-generated address translated to hardware address
- Program addresses are called logical or virtual addresses
- Hardware addresses are called physical or real addresses.

Address space: View of memory for each process

Address Spaces

Translation from logical to physical addresses

Hardware Support

Two operating modes
- **Privileged (protected, kernel) mode**: When OS runs
  - When trap into OS (system calls, interrupts, errors)
  - Allows certain instructions to be executed
  - Allows OS to access all of physical memory
- **User mode**: When user processes run
  - Performs translation of logical address to physical address

Base and Bounds registers for translation
- **Base register**: Start location for address space
- **Bounds register**: Last valid address the process may generate
  - Appears to have private memory of size equal to bounds register
### Implementation

**Translation on every memory access in user process**
- Compare logical address to bounds register
  - If logical address is greater, then generate error
- Add base register to logical address to generate physical address

**CPU registers**
- logical address
- 32 bits
- bounds
- relative
- base

**Memory relocation**
- physical address
- error
- <= bounds
- no
- yes

### Managing Processes w/ Base and Bounds

**Context-Switch**
- Add base and bounds registers to PCB
- Steps during context-switch
  - Change to privileged mode
  - Save base and bounds registers of old process
  - Load base and bounds registers of new process
  - Change to user mode and jump to new process

**Protection Requirement**
- User process can not change base and bounds register
- User process can not change to privileged mode

**What if don't change base and bounds register when switch?**

### Pros and Cons of Base and Bounds

**Advantages**
- Supports dynamic relocation of address spaces
- Supports protection across multiple address spaces
- Cheap: Few registers and little logic
- Fast: Add and compare can be done in parallel

**Disadvantages**
- Each process must be allocated contiguously in real memory
  - Fragmentation: Can not allocate a new process
  - Solution: Swapping
- Must allocate memory that may not be used
- No Sharing: Cannot share limited parts of address space (e.g., cannot share code with private data)