Memory management

- Recall: Process = Set of threads + address space
- Address space
  - All the memory space the process can use as it runs
- Hardware interface: physical memory shared between processes
- Problems if processes can directly access addresses in physical memory?

Address space abstraction provided by OS

- Virtual memory: an address space can be larger than the machine’s physical memory
- Address independence: same numeric address can be used in different address spaces (i.e., different processes), yet remain logically distinct
- Protection: one process can’t access data in another process’s address space (actually controlled sharing)

Uni-programming

- 1 process runs at a time
- Always load process into same spot in memory
- Reserve space for OS
- Virtual address = physical address
- How to handle swap between processes?

Multi-programming and address translation

- Multi-programming: more than 1 process in memory at a time
- Must translate addresses accessed by a process for address independence and protection
- Two options:
  - Static address translation
  - Dynamic address translation
- How to run two processes concurrently and offer address independence with static translation?

Static Address Translation

- \texttt{fffff} 40000 frrr user process 1 20000 frrr user process 2 30000

- Does static address translation achieve the other address space abstractions?
- Achieving all address space abstractions requires doing some work on each memory reference

Dynamic address translation

- Translate every memory reference from virtual address to physical address
  - virtual address: an address issued by a process
  - physical address: an address used to access physical mem
- Translation enforces protection
  - One process can’t refer to another process’s address space
- Translation enables virtual memory
  - A virtual address only needs to be in physical memory when it’s being accessed
  - Change translations on the fly: different virtual addresses occupy the same physical memory
Address translation

- Many ways to implement translator
- Tradeoffs
  - Flexibility (sharing, growth, virtual memory)
  - Size of data needed to support translation
  - Speed of translation

Base and bounds

- Load each process into contiguous region of physical memory
  - Prevent process from accessing data outside its region
    
    ```
    if (virtual address > bound) {
        trap to kernel; kill process (core dump)
    } else {
        physical address = virtual address + base
    }
    ```

- Similar to linker-loader, but also protects processes from each other
- Only kernel can change translation data (base and bounds)

- How to handle swap between processes?
- What to do when address space grows?

Pros?
- Fast
- Simple hardware support

Cons?
- Virtual address space limited by physical memory
- No controlled sharing
- External fragmentation