Unix process creation

System calls to start a process:

1. Fork() creates a copy of current process
2. Exec(program, args) replaces current address space with specified program
Multi-process issues

- How to partition physical memory allocation among processes?
  - Fairness versus efficiency

- Global replacement
  - Can evict pages from this process or other processes

- Local replacement
  - Can evict pages only from this process
  - Must still determine how many pages to allocate to this process

- Pros and Cons?
Thrashing

- What would happen if many large processes all actively used their entire address space?

- Performance degrades rapidly as miss rate goes up
  - Avg access time = hit rate * hit time + miss rate * miss time
  - E.g., hit time = .0001 ms; miss time = 10 ms
    » Average access time (100% hit rate) = .0001 ms
    » Average access time (99% hit rate) = ?
    » Average access time (90% hit rate) = ?
Solutions to Thrashing

- Buy more DRAM
  - Price per GB fallen by 4x since 2009

- Run fewer processes for longer
  - Example: Longer time slice
  - Reduces page faults
Working set

- Thrashing depends on portion of address space actively used by each process
  - What do we mean by “actively using”?  

- Working set = all pages used in last $T$ seconds
  - Larger working set $\Rightarrow$ process needs more physical memory to run well (i.e., avoid thrashing)

- Sum of all working sets should fit in memory
  - Only run subset of processes that fit in memory

- How to measure size of working set?
  - Periodic sweep of clock hand in LRU clock
Project 3

- Hope you have a state machine for swap-backed pages by now

- Things to consider
  - Transitions?
  - Properties that capture the state of a page?
  - Protection bits?

- Don’t translate state machine into if-else cases
- Think ahead in designing data structures
Project 3

- Adopt and incremental workflow
- Add support for file-mapped pages
  - Without fork = 54/75
- Add support for fork()
OS view vs. Application view

- Protection
  - All pages can be read from and written to
  - Using R/W bits to track reference, dirty, etc.

- Sharing
  - File-mapped pages
  - Copy-on-write