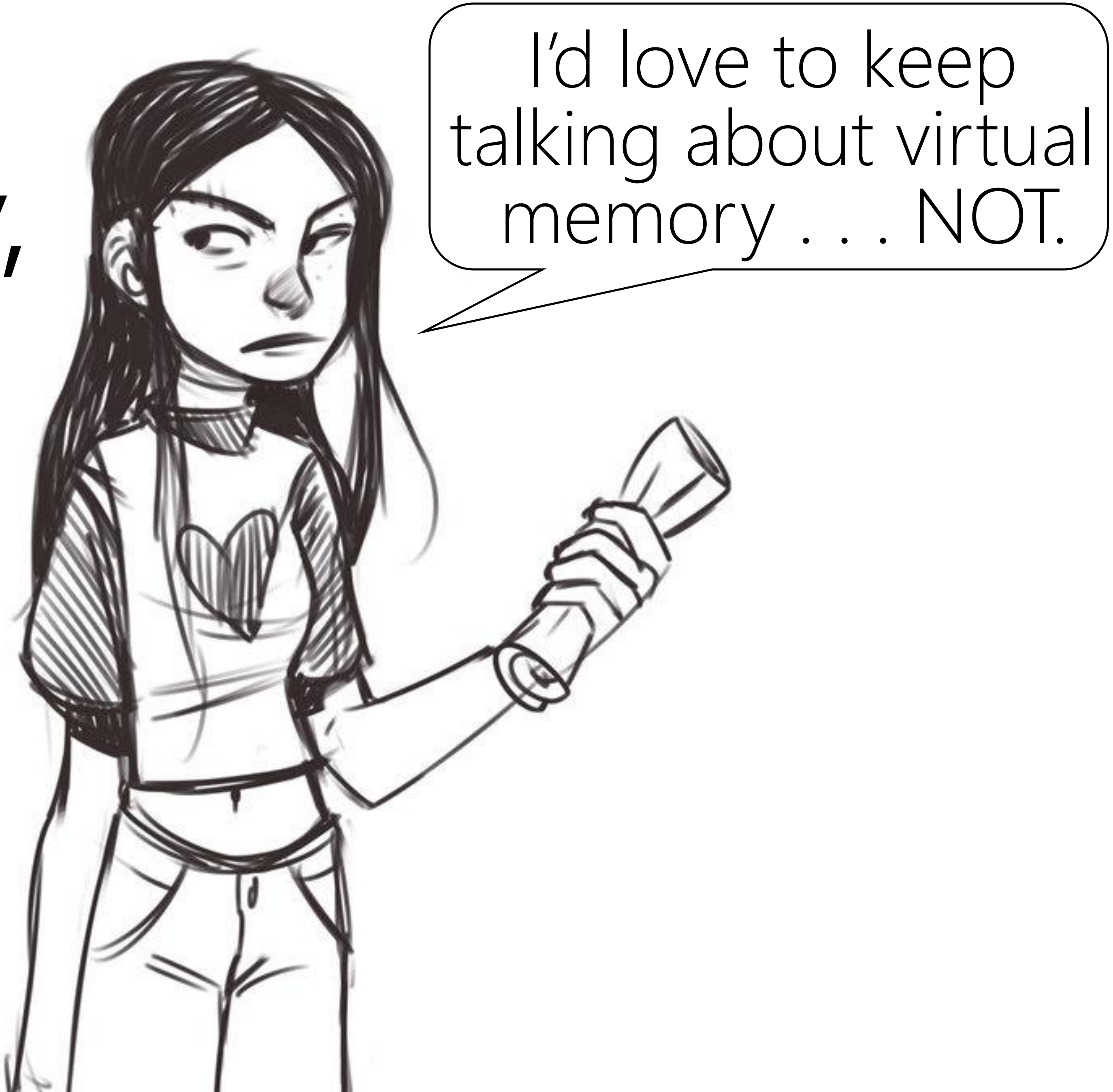
Virtual Memory, Part IV

CS 161: Lecture 9

2/28/17



Working Sets

- At any given time, a process’s working set is the set of actively-referenced virtual pages
  - If a process’s working set is not completely in physical RAM, then the process will thrash
  - When someone asks “How much memory does Program X require?”, they are asking “How big is X’s typical working set?”
- Some schedulers are aware of working sets
  - Scheduling a process with an evicted working set will lead to a bunch of swapping
  - So, scheduler can preferentially run processes that have their working sets in RAM

Balance Sets: Working-set-aware Schedulers

- Partition runnable processes into two groups
  - Active processes have their working sets in physical RAM
  - Inactive processes have swapped-out working sets
- Balance set: the pages in the working sets of the active processes
  - If the balance set grows larger than physical RAM, the scheduler forces some active processes to become inactive (ideally, the newly inactive processes were already in a waiting state, e.g., waiting on user input)
  - If the balance set shrinks to be less than physical RAM, the scheduler makes some inactive processes active
  - Scheduler must avoid starvation—eventually, all inactive processes must become active
- RAM is wasted if it's not being used, so OS should try to use it all!
  - If there's no thrashing, then having extremely high RAM utilization is desirable

Swapping in Practice

- Swapping is typically rare on desktops/laptops, since RAM is plentiful
  - However, on low-cost desktops and laptops with little RAM, swapping can be frequent and painful
  - Even on machines with a lot of RAM, OS must be prepared for RAM oversubscription
- By default, Android and iOS do not swap!
  - Mobile devices use flash for storage
  - Flash devices only support a limited number of writes
  - So, Android and iOS use virtual memory and paging (i.e., a layer of indirection between virtual and physical addresses) . . .
  - . . . but Android and iOS do not swap to avoid wearing out the flash

Swapping in Practice

- When memory pressure is high, Android+iOS forcibly evict entire apps from memory
  - iOS fires the `applicationWillTerminate()` callback of the about-to-be-evicted app, allowing the app to serialize app-specific state before eviction (this state is typically much smaller than the entire working set of the app!); later, when the app is resurrected, it uses the serialized state to reinitialize itself
    - `applicationWillTerminate()` only invoked on background apps
    - Suspended apps, i.e., apps that aren't running code but have pages in memory, are terminated without notification
  - Android invokes `onTrimMemory(int urgency)` method to inform apps that memory pressure is high and apps should deallocate unneeded memory (iOS defines similar notifications)
    - Android can kill paused apps at any time . . .
    - . . . so apps should serialize critical state when they transition to a paused state!