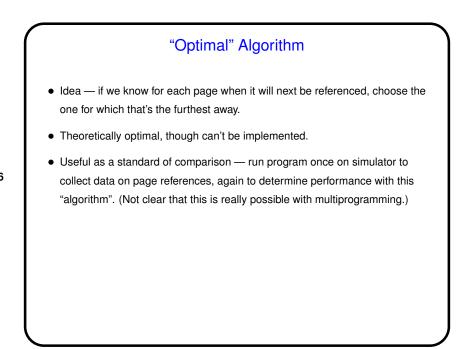
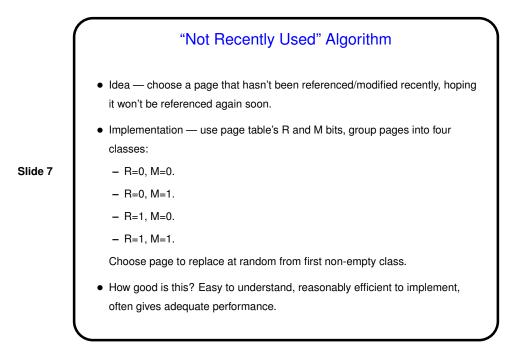
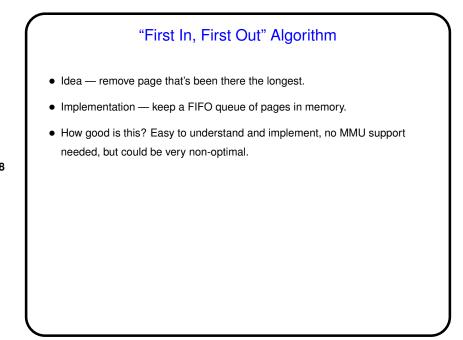
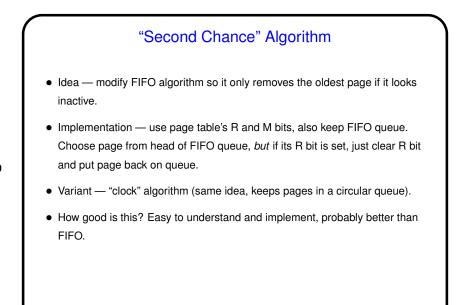


• Choice usually constrained by what MMU provides (though that is influenced by what would help o/s designers).



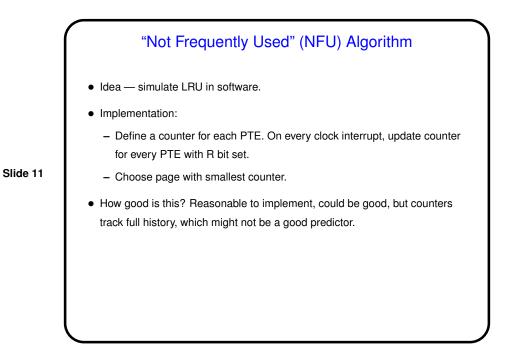


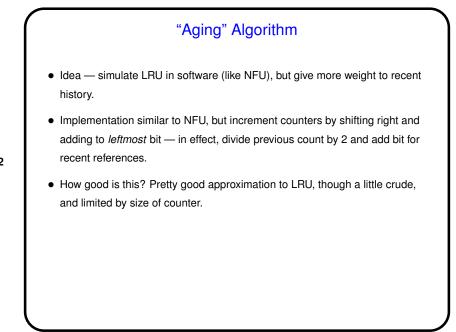




Slide 10

"Least Recently Used" (LRU) Algorithm Idea — replace least-recently-used page, on the theory that pages heavily used in the recent past will be heavily used in the near future. (Usually true). Implementation: Full implementation requires keeping list of pages ordered by time of reference. Must update this list on every memory reference. Only practical with special hardware — e.g.: Build 64-bit counter C, incremented after each instruction. On every memory reference, store C's value in PTE. To find LRU page, scan page table for smallest stored value of C. How good is this? Could be pretty good, but requires hardware we probably won't have.





Intermezzo — Demand Paging, Prepaging, and Working Sets

- The purest form of paging is "demand paging" processes are started with no pages in memory, and pages are loaded into memory on demand only.
- An alternative is "prepaging" try to load pages in advance of demand. How?
- Most programs exhibit "locality of reference", so a process usually isn't using all its pages.
- A process's "working set" is the pages it's using. Changes over time, with size a function of time and also of how far back we look.

"Working Set" Algorithm
 Idea — steal / replace page not in recent working set. Define working set by looking back τ time units (w.r.t. process's virtual time). Value of τ is a tuning parameter, to be set by o/s designer or sysadmin.
Implementation:
 For each entry in page table, keep track of time of last reference.
 When we need to choose a page to replace, scan through page table and for each entry:
If R bit is set, update time of last reference.
Compute time elapsed since last use. If more than $ au,$ page can be replaced.
 If we don't find a page to replace that way, pick the one with oldest time or last use. If a tie, pick at random.
 How good is this? Good, but could be slow.

