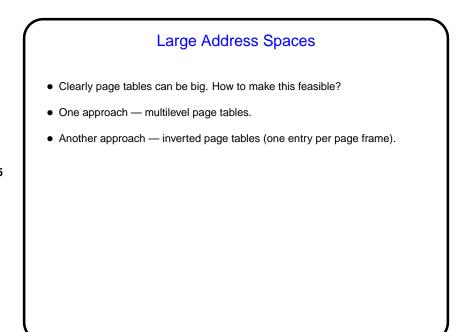


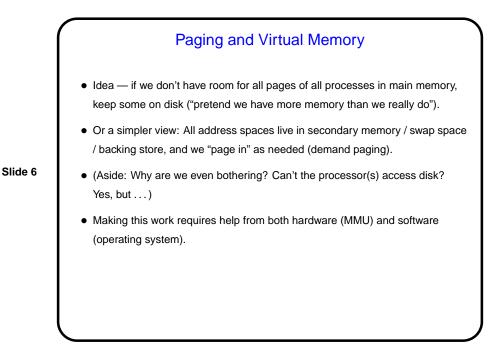


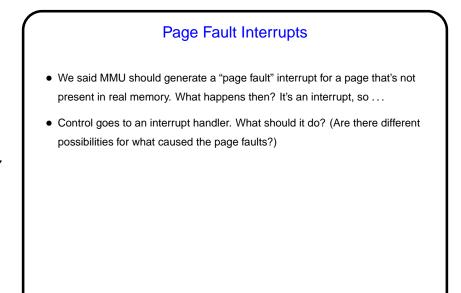
- Idea divide both address spaces and memory into fixed-size blocks ("pages" and "page frames"), allow non-contiguous allocation.
- Makes for a much more flexible system but at a cost in complexity keeping track of a process's memory requires a "page table" to be used by both hardware (MMU) and software (O/S).

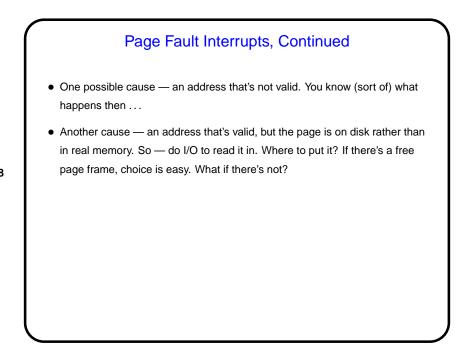
Slide 4

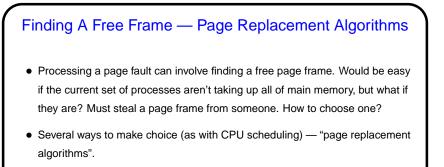
Page Tables — Performance Issues One possibility is to keep the whole page table for the current process in registers. Could possibly use general-purpose registers for this but likely would not. Should make for fast translation of addresses, but — is this really feasible for a large table? and what about context switches? Another possibility is to keep the process table in memory and just have one register (probably a special-purpose one) point to it. Cost/benefit tradeoffs here seem like the opposite of the first scheme, no? The big downside is slow lookup, though, and that can be improved with a "translation lookaside buffer" (TLB) — special-purpose cache.



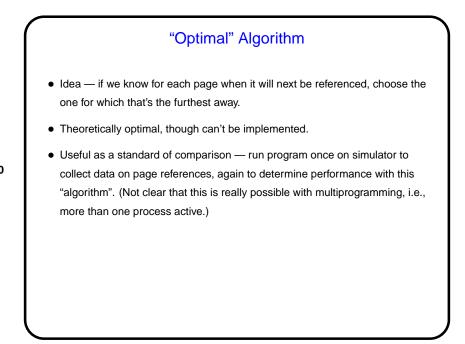


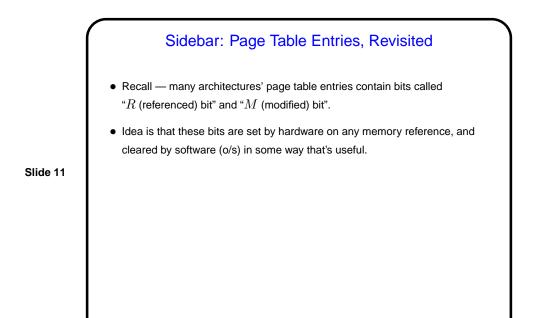






- "Good" algorithms are those that result in few page faults. (What happens if there are many page faults?)
- Choice usually constrained by what MMU provides (though that is influenced by what would help o/s designers).
- Many choices (no surprise, right?)

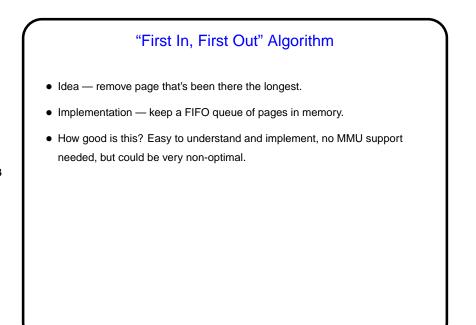




"Not Recently Used" Algorithm
Idea — choose a page that hasn't been referenced/modified recently, hoping it won't be referenced again soon.
Implementation — use page table's R and M bits, group pages into four classes:

R = 0, M = 0.
R = 0, M = 1.
R = 1, M = 0.
R = 1, M = 1.

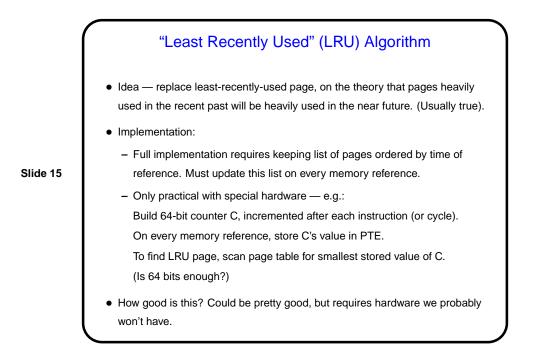
Choose page to replace at random from first non-empty class.
How good is this? Easy to understand, reasonably efficient to implement, often gives adequate performance.

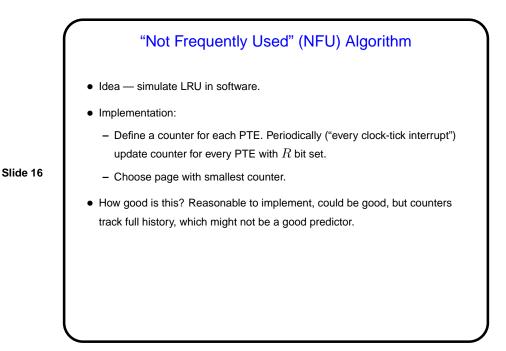


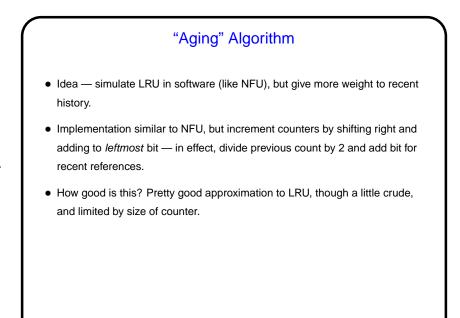
"Second Chance" Algorithm

- Idea modify FIFO algorithm so it only removes the oldest page if it looks inactive.
- Implementation use page table's R and M bits, also keep FIFO queue. Choose page from head of FIFO queue, *but* if its R bit is set, just clear R bit and put page back on queue.
- Variant "clock" algorithm (same idea, keeps pages in a circular queue).
- How good is this? Easy to understand and implement, probably better than FIFO.

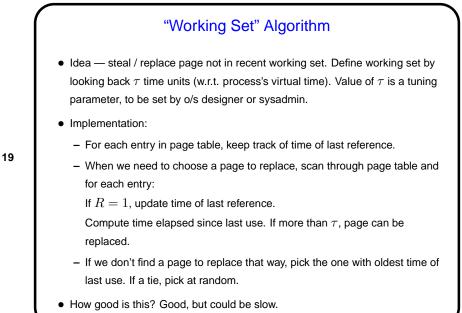


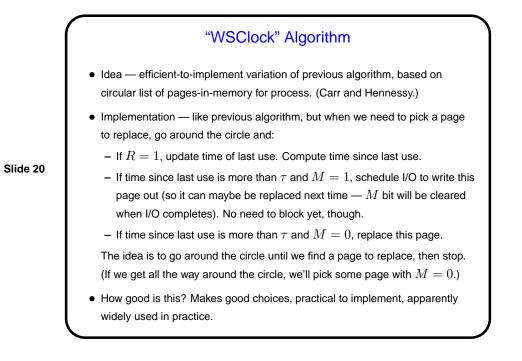






Sidebar: Working Sets Most programs exhibit "locality of reference", so a process usually isn't using all its pages. A process's "working set" is the pages it's using. Changes over time, with size a function of time and also of how far back we look.





Minute Essay

 Another story from long ago: Once upon a time, a mainframe computer was running very slowly. The sysadmins were puzzled, until one of them noticed that one of the disk drives seemed to be very busy and asked "which disk are you using for paging?" The answer made everyone say "aha!" What was wrong (to make the system so slow)?

Slide 21

• Does anything like this still happen?

Minute Essay Answer

- The disk being used for paging was the one that was very busy. So, mostly likely the system was spending so much time paging ("thrashing") that it wasn't able to get anything else done. Usually this means that the system isn't able to keep up with active processes' demand for memory.
- Slide 22
- Memory sizes have increased to a point where the odds aren't as good as they were. But a few years ago we did run into problems with the machines in one of the classrooms trying to run both Eclipse and a Lewis simulation, and then more recently with some of them attempting to run a background program that asked for memory than its author intended.