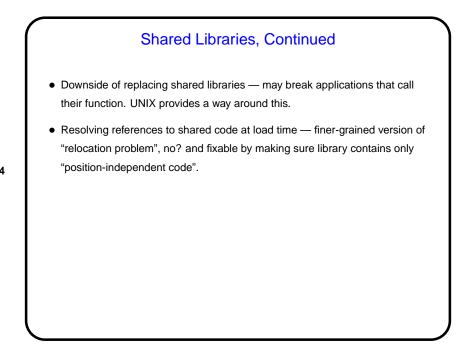


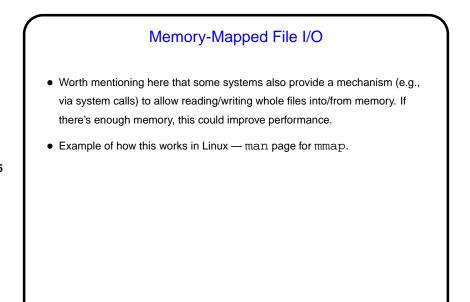


• One attraction is somewhat obvious — if code for library functions (e.g., printf) is statically linked into every program that uses it, programs need more memory — seems wasteful if processes can share one copy of code in memory.

Slide 3

- Another attraction is that library code can be updated independently of programs that use it. (Is there a downside to that?)
- How to make this happen ... At link time, programs get "stub" versions of functions. References to real versions resolved at load time. Does this remind you of anything? and suggest a possible problem? how to fix?





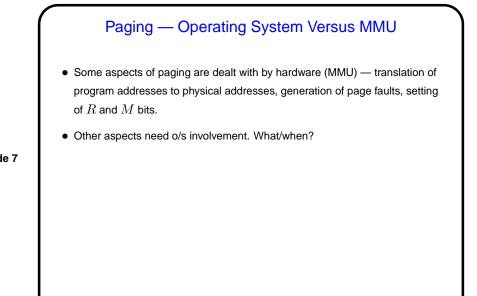
Slide 5

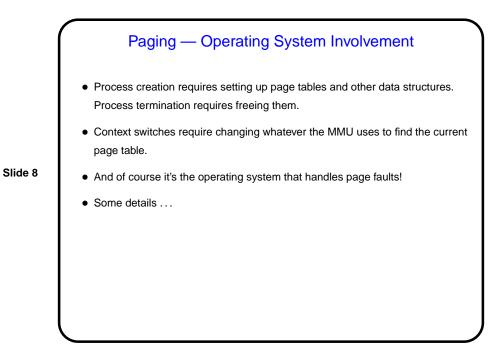
One More Design Issue

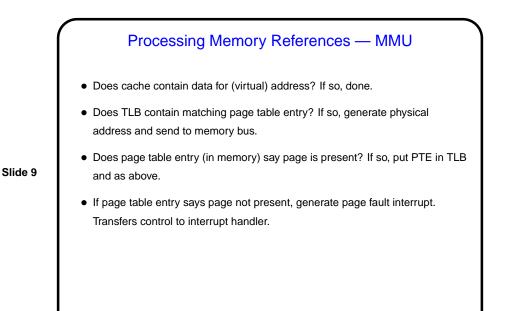
• Page replacement algorithms as discussed all seem based on the idea that we let memory fill up, and then "steal" page frames as needed. Is that really the best way ...

Slide 6

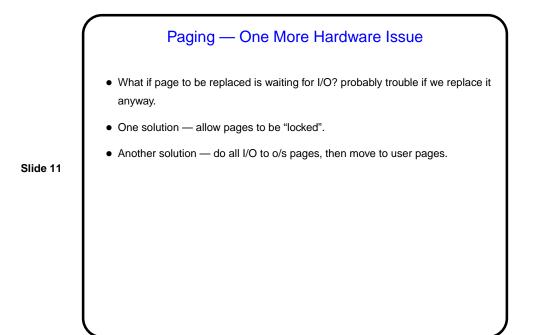
 An alternative — background process ("paging daemon") that tries to keep a supply of free page frames, or at least ones that can be stolen without needing to write out their contents. Can use algorithms similar to page replacement algorithms to do this.

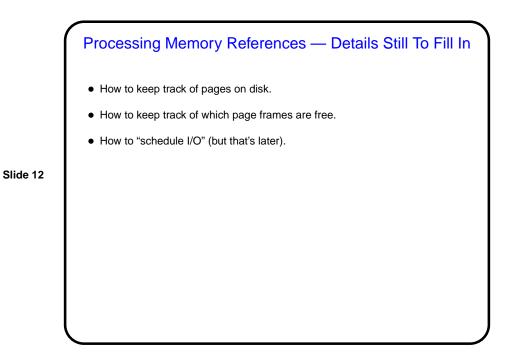


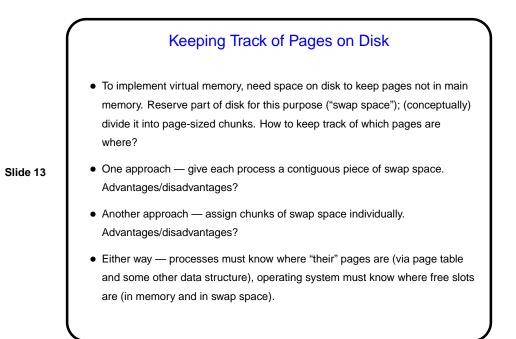




Processing Memory References — Page Fault Interrupt Handler
Is page on disk or invalid (based on entry in process table, or other o/s data structure)? If invalid, error — terminate process.
Is there a free page frame? If not, choose one to steal. If it needs to be saved to disk, start I/O to do that. Update process table, PTE, etc., for "victim" process. Block process until I/O done.
Start I/O to bring needed page in from swap space (or zero out new page). If I/O needed, block process until done.
Update process table, etc., for process that caused the page fault, and restart it at instruction that generated page fault.

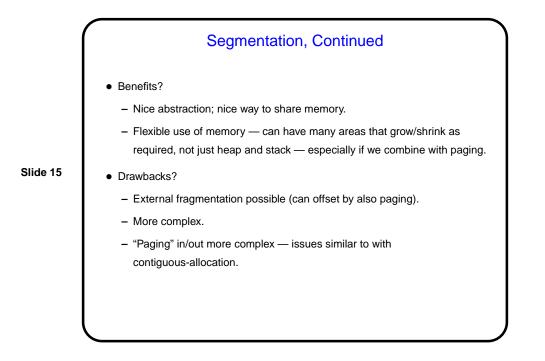


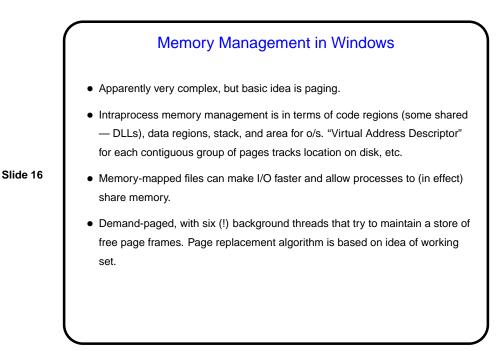


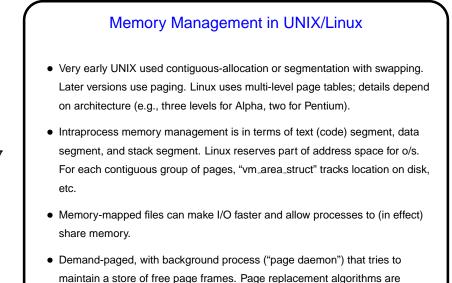


One More Memory Management Strategy — Segmentation

- Idea make program address "two-dimensional" / separate address space into logical parts. So a virtual address has two parts, a segment and an offset.
- To map virtual address to memory location, need "segment table", like page table except each entry also requires a length/limit field. (So this is like a cross between contiguous-allocation schemes and paging.)







mostly variants of clock algorithm.

Minute Essay
At least one early mainframe operating system supported virtual memory (i.e., keeping some pages on disk), but provided only a single address space that all processes shared. How does this compare with giving each process its own address space — with respect to simplicity, efficiency, anything else you can think of? (There appears to be active research into reviving this idea. Hm!)
Anything about memory management you'd like to hear more about / have clarified?

Slide 17

Minute Essay Answer

 A single address space seems like it might be simpler and more efficient could have a single page table. Also it would be easier for processes to share memory. But it seems potentially less secure — how to protect one process's memory for another? — and probably would make it very difficult to implement fork. (You may think of other considerations!)