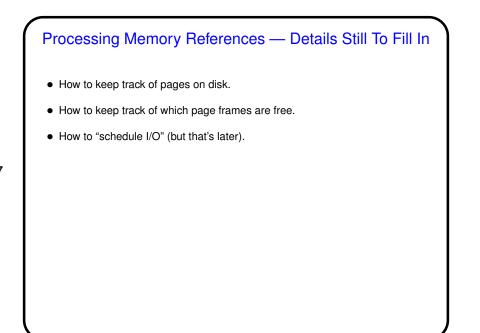
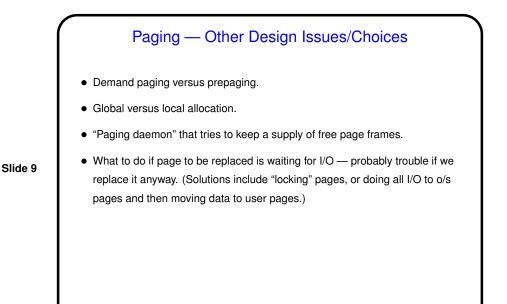


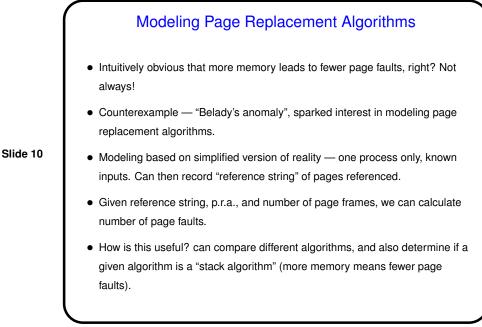
Processing Memory References — Page Fault Interrupt Handler
Is page on disk or invalid (based on entry in process table, or other o/s data structure)? If invalid, error — terminate process.
Is there a free page frame? If not, choose one to steal. If it needs to be saved to disk, start I/O to do that. 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.

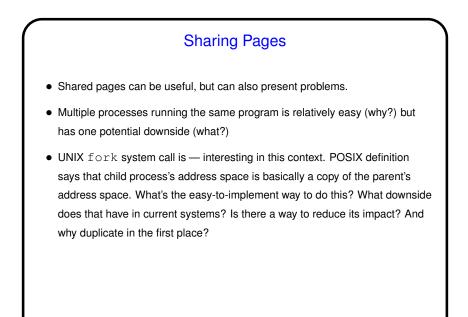
- Update process table, etc., for process that caused the page fault, and restart it at instruction that generated page fault.
- Slide 6



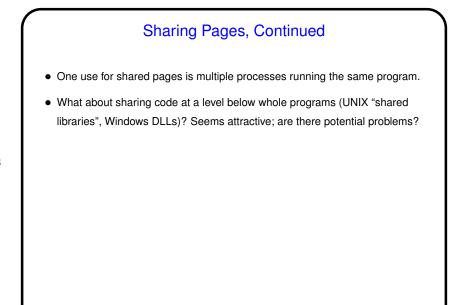
Keeping Track of Pages on Disk To implement virtual memory, need space on disk to keep pages not in main memory. Reserve part of disk for this purpose ("swap space"); (conceptually) divide it into page-sized chunks. How to keep track of which pages are where? One approach — give each process a contiguous piece of swap space. Advantages/disadvantages? Another approach — assign chunks of swap space individually. Advantages/disadvantages? Either way — processes must know where "their" pages are (via page table and some other data structure), operating system must know where free slots are (in memory and in swap space).







Sharing Pages and fork Duplicating pages is easy but inefficient, especially if the child process is going to call execve or something similar right away. Some systems use "copy-on-write" to improve efficiency. Why did the people who designed UNIX require this duplication ... Possibly because it makes some things easy (such as setting up parent/child pipes) and wasn't very costly when designed. Windows's system call for creating processes takes a different approach. Maybe that's better!



Shared Libraries

 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 14

- 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.

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 15

• Could 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 16
- 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 more memory than its author intended.

