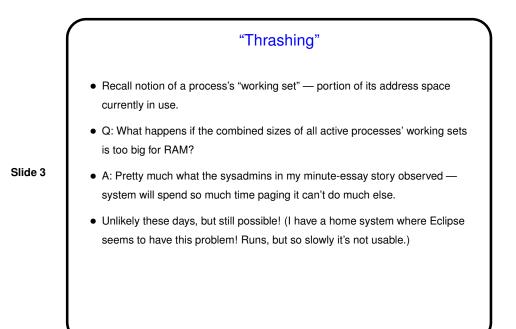
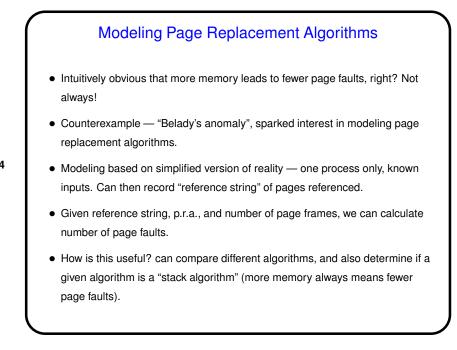
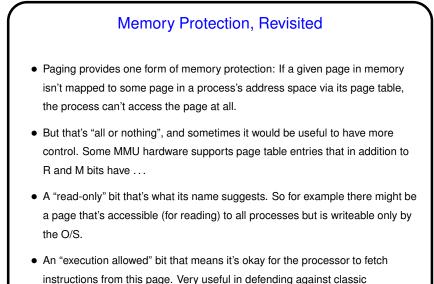


Page Replacement Algorithms — Recap
If there are always free frames to bring in pages from disk, no need to make decisions. Not always guaranteed, hence the need to choose.
Many many ways to choose (no surprise!). Goal is to reduce number of page faults. Often based on observation that recent past predicts near future — notion of "working set".

Slide 2



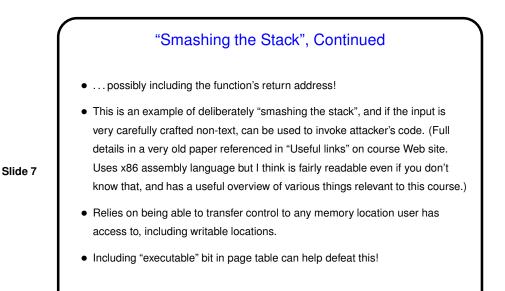




buffer-overflow attacks (by not setting this bit for stack pages)!

Sidebar: "Smashing the Stack"
Usual scheme for memory use within a process puts a stack at high addresses, used in function calls (for parameters and return address) and also for local variables. What happens if an attempt is made to store more data in a local-variable array than will fit? (And in C this is all too easy, no?)
Well, you know from CSCI 1120, no? Whatever is after the array is overwritten ...

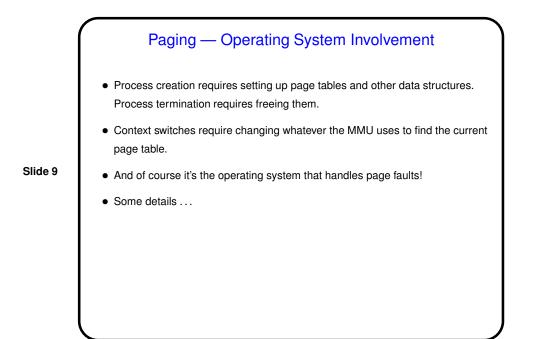
Slide 6

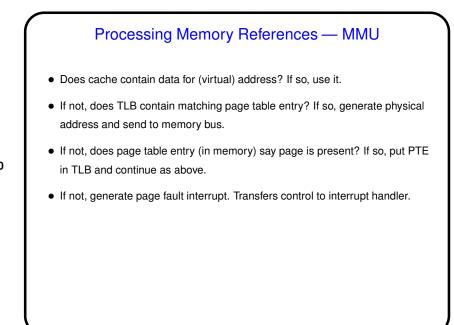


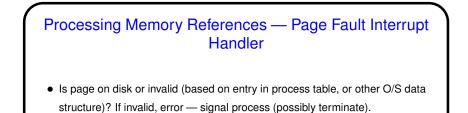
Paging — Operating System Versus MMU

• Some aspects of paging are dealt with by hardware (MMU) — translation of program addresses to physical addresses, generation of page faults, setting of *R* and *M* bits.

• Other aspects need O/S involvement. What/when?







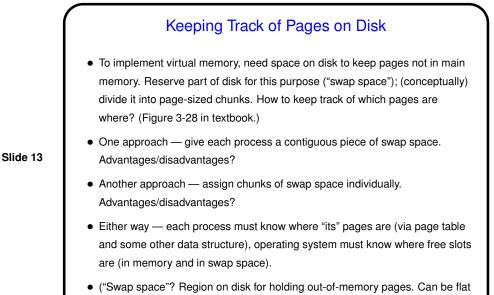
 Is there a free page frame? If not, choose one to steal (using page replacement algorithm). If it needs to be saved to disk, start I/O to do that. Update process table, PTE, etc., for "victim" process. Block process until I/O done.

- 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 at instruction that generated page fault.

Processing Memory References — Details Still To Fill In

- How to keep track of pages on disk.
- How to keep track of which page frames are free.
- How to "schedule I/O" (but that's later).

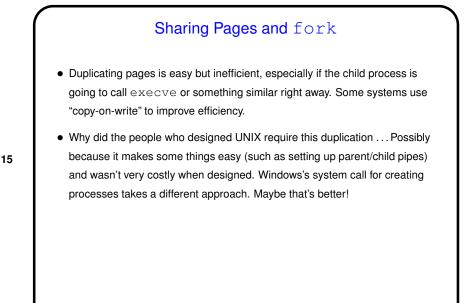
Slide 12

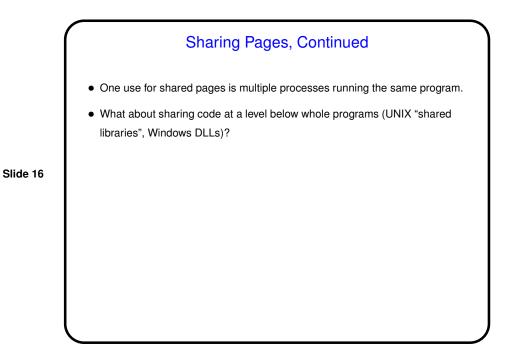


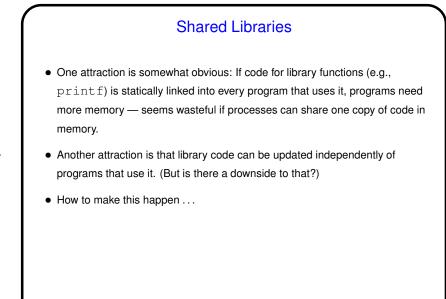
Sharing Pages • (Pause first to try to combat Zoom fatigue ...) • Shared pages can be useful, but can also present problems. • Multiple processes running the same program is relatively easy (why?) but has one potential downside (what?) • UNIX fork system call is - interesting? - in this context. POSIX definition says that child process's address space is basically a copy of the parent's address space. What's the easy-to-implement way to do this? What downside does that have in current systems? Is there a way to reduce its impact? And why duplicate in the first place?

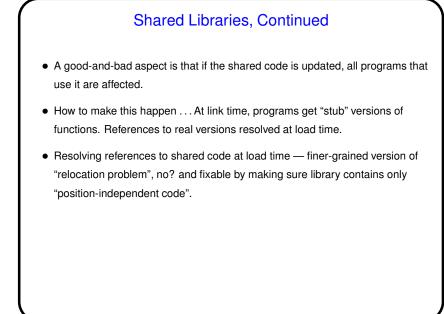
Slide 14

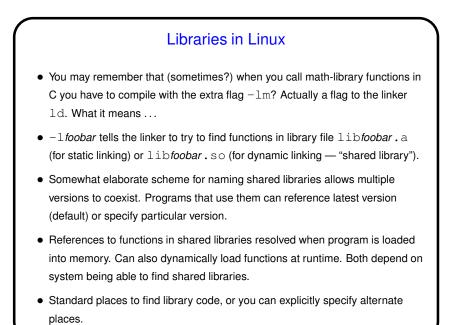
file or separate partition. A.k.a. "backing store".)



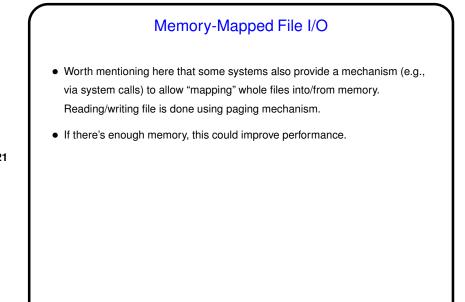








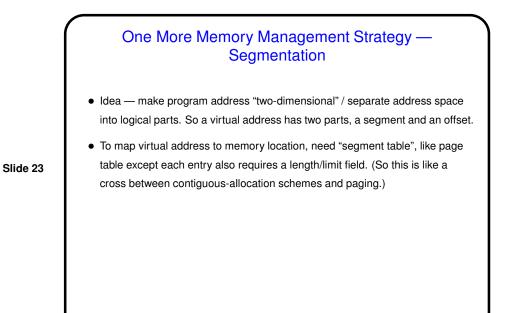
	Libraries in Linux, Continued
	 Creating a static library is relatively straightforward: Compile code as usual and then use ar to combine object code files into library.
Slide 20	 Creating a shared library is less so: Compile code with flag to generate "position-independent code" (why? to
	avoid "relocation problem" previously discussed). Generate shared library and set up symbolic links following naming conventions (in which a library has a "real name", an "soname", and a name by which the linker normally finds it).
	At runtime, must be sure system knows where to find library. Either "hardcode" in executable or use environment variable LD_LIBRARY_PATH.
	• (Example on course Web site.)



Memory-Mapped I/O in Linux

• System calls mmap, etc., allow whole or partial files to be "mapped" to memory. Map can be private to process (essentially a copy of the file, with changes not saved back) or shared among processes.

- Actual file reads happen only as locations are referenced, using more or less the same mechanism as paging. Actual file writes happen only with shared maps, either as pages are swapped in and out of memory or via msync system call.
- (Example on course Web site.)



Segmentation, Continued Benefits? Nice abstraction; nice way to share memory. Flexible use of memory — can have many areas that grow/shrink as required, not just heap and stack — especially if we combine with paging. Slide 24 Drawbacks? External fragmentation possible (can offset by also paging). More complex. "Paging" in/out more complex — issues similar to with contiguous-allocation.

