

Slide 1

Memory Management — Review The problem we're solving — partition physical memory among processes. Two related issues — program relocation and memory protection — both nicely solved by defining "address space" abstraction and implementing with help from hardware (MMU). Contiguous-allocation schemes are simple but not very flexible. Paging is more flexible but more complex.

Slide 2

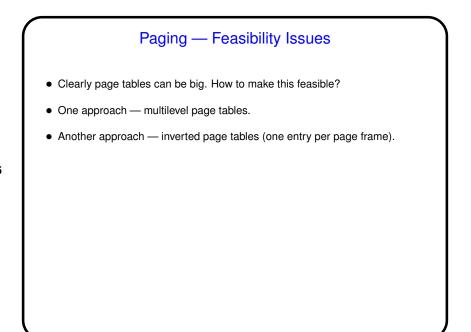


- Idea divide both address spaces and memory into fixed-size blocks ("pages" and "page frames"), allow non-contiguous allocation.
- Makes for a much more flexible system but at a cost in complexity keeping track of a process's memory requires a "page table" to be used by both hardware (MMU) and software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

Slide 3

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. Can be mitigated with a "translation lookaside buffer" (TLB) — special-purpose cache.



Slide 5

