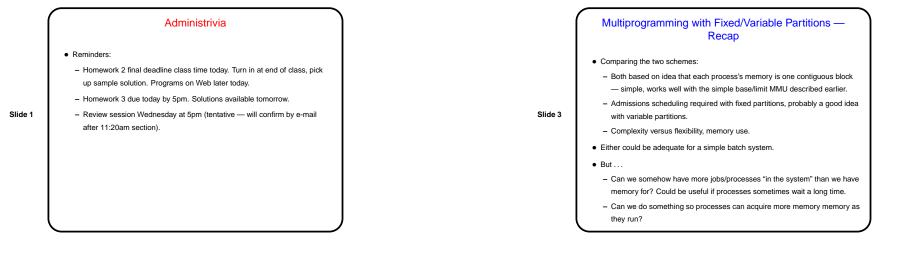
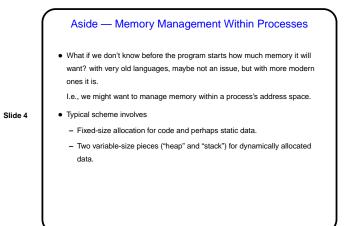
CSCI 4320



### Minute Essay From Last Lecture

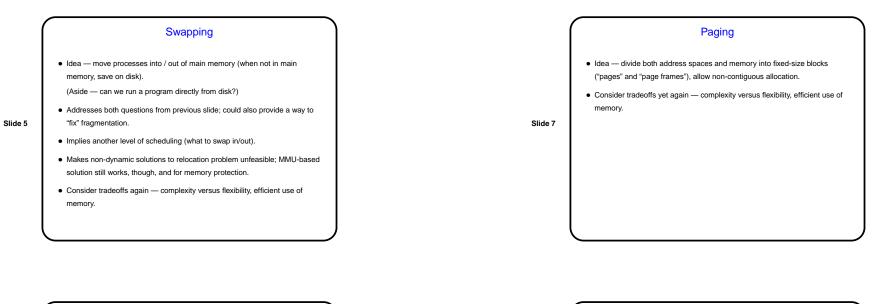
- What if anything did you find difficult about the programming part of the homework? What if anything did you find interesting/useful about it?
   Almost half mentioned rusty programming skills.
- The point of the first problem (together with the first written question) was to provide some exposure to system calls in a real-world context. (Also you probably have a different view now of bash?)



CSCI 4320

Slide 8

October 14, 2003





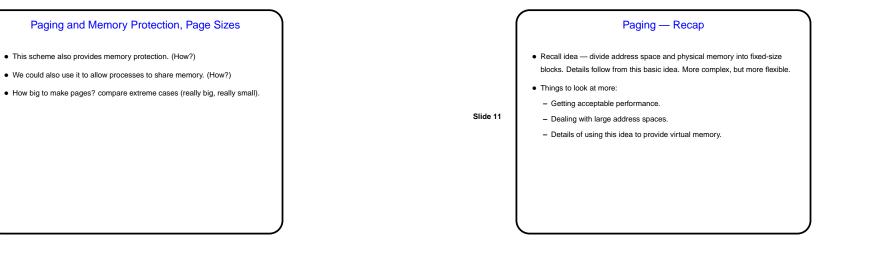
- Contiguous-allocation schemes are simple to understand, implement.
- But they're not very flexible process's memory must be contiguous, swapping is all-or-nothing.
- Can we do better? yes, by relaxing one or both of those requirements "paging".

Slide 6

# Paging — Mapping Program to Physical Addresses One consequence — mapping from program addresses to physical addresses is much more complicated. How? "page table" for each process maps pages of address space to page frames; use this to calculate physical address from program address. (Are there page sizes for which this is easier?) As with base/limit scheme, makes more sense to implement this in MMU. (Notice again interaction between hardware design and o's design.) Could let page table size vary, but easier to make them all the same (i.e., each process has the same size address space), have a bit to indicate valid/invalid for each entry. Attempt to access page with invalid bit — "page fault".

Slide 9

CSCI 4320



## Paging and Virtual Memory

- Idea extend this scheme to provide "virtual memory" keep some pages on disk. Allows us to pretend we have more memory than we really do.
- · Compare to swapping.

Slide 10

# Performance / Large Address Spaces Even with good choice of page size, serious performance implications — page table can still be big, and every memory reference involves page-table access — how to make this feasible/fast? Consider several options — compare access time, cost, context-switch time: Keep page table for current process in registers. Keep whole page table in main memory, pointed to by special register. Use multilevel page tables. (More about this later.) Use inverted page tables (one entry per page frame). (More about this later.) If page tables are in memory, performance improves with "translation lookaside buffer" (TLB) — special-purpose cache.

October 14, 2003

## Minute Essay

- Reminder turn in homework.
- Given a page size of 64K (2<sup>16</sup>), 64-bit addresses, and 4G (2<sup>32</sup>) of main memory, at least how much space is required for a page table?

Slide 13