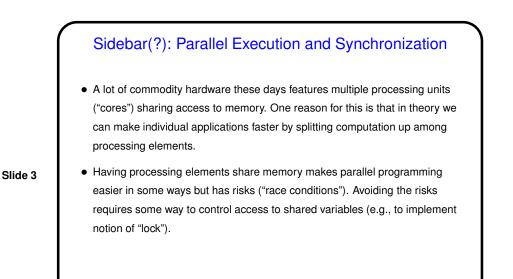
Administrivia
Recurring meetings for M/W office hours created. Information shared by e-mail and reachable from my home page (http://www.cs.trinity.edu/~bmassing).
First of these is today. It will be my first try at using the one-at-a-time feature. If it doesn't work, fallback is Google Chat. chat.google.com seems to work better than clicking "Chat" from TMail. I'm kind of hoping at least a few of you will try it, so I get a better sense of how it will work!)
I just relayed to all of you a message from Dr. Fogarty about Discord, as a possible way to connect a bit more with your classmates. Looks promising?

Exam 1, Revisited
I prefer that you do this during one of the Zoom meeting times:

Thursday March 26, 5pm to 7pm.
Friday March 27, 4pm to 6pm.
I've set up the meetings like office hours, with one-at-a-time access.

Slide 2

I plan to put the exam in our shared folder during those times. If you need to take the exam at another time, let me know and I'll make it available then.
If you have an SAS accommodation allowing you extra time, let me know. My idea is that you'd start the exam during one of the scheduled periods and work for your allowed time.
As mentioned Monday — PDF from me to you (shared Google Drive folder), PDF from you to me (e-mail or your "graded work" folder).
Questions?

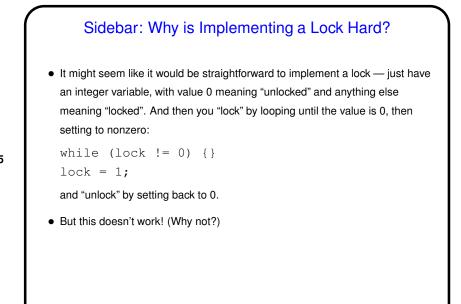


Parallel Execution and Synchronization, Continued

 Most texts on operating systems discuss synchronization issues and present several solutions ("synchronization mechanisms"), some rather high-level and others not.

(Why is this in O/S textbooks? because O/Ss typically have to manage "processes" executing concurrently, either truly at the same time or interleaved.)

• The most primitive can (with some simplifying assumptions) be implemented with no hardware support. But hardware support is very useful.



Slide 5

Instructions for Synchronization
Key goal in designing hardware support for synchronization is to provide "atomic" (indivisible) load-and-store. This allows writing a low-level implementation of "lock" idea.
Many architectures do this with a single instruction (e.g., "test and set" or "compare and swap"). Requires two accesses to memory so may be difficult to implement efficiently.
MIPS approach: Same idea, but using a pair of instructions, 11 ("load linked") and sc ("store conditional").

