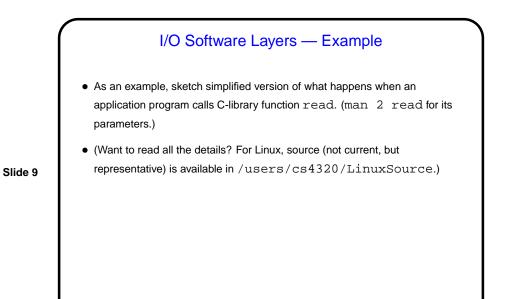


Slide 10



User-Space Software Layer — C-Library read function
Library function called from application program, so executes in "user space".
Sets up parameters — buffer, count, "file descriptor" constructed by previous open (as discussed briefly in the chapter on filesystems) — and issues read system call.
System call generates interrupt (trap), transferring control to system read function.
Eventually, control returns here, after other layers have done their work.
Returns to caller.



- Invoked by interrupt handler for system calls, so executes in kernel mode.
- Checks parameters is the file descriptor okay (not null, open for reading, etc.)? Returns error code if necessary.

Slide 11

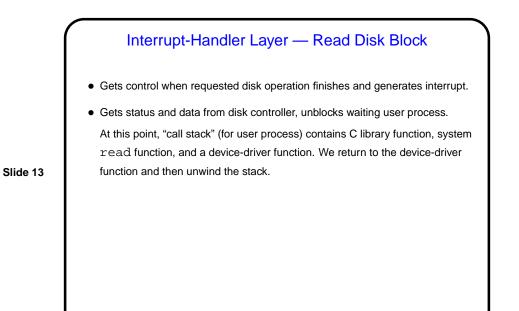
- If buffering, checks to see whether request can be obtained from buffer. If so, copies data and returns.
- If no buffering, or not enough data in buffer, calls appropriate device driver (file descriptor indicates which one to call, other parameters such as block number) to fill buffer, then copies data and returns.



- Contains code to be called by device-independent layer and also code to be called by interrupt handler.
- Maintains list of read/write requests for disk (specifying block to read and buffer).
- Slide 12
- When called by device-independent layer, either adds request to its queue or issues appropriate commands to controller, then blocks requesting process (application program).

(This is where things become asynchronous.)

• When called by interrupt handler, transfers data to memory (unless done by DMA), unblocks requesting process, and if other requests are queued up, processes next one.



Minute Essay
A year or two ago I argued with a Windows person about schemes for representing devices: UNIX uses "special files", normally in /dev but can be anywhere, identifiable as different from normal files; Windows puts them all at the top level, prefix similar to drive letter.
Which seems more logical to you, and why? from the standpoint of end users, application programmers, o/s developers?

Slide 14