Slide 1

Administrivia

- Reminder: Homework 6 due today.
- Homework 7 to be on the Web soon, due after Thanksgiving. I will send mail.

Slide 2

Character-Oriented Terminals — Recap/Review

- Hardware: Keyboard sends a character at a time. Display accepts stream of characters, which may include "escape sequences" (to position cursor, turn on reverse-video mode, etc.).
- Software: Can accept input in "raw" or "cooked" mode (with the latter, device driver can do some simple line editing). Must produce output including any needed escape sequences (which might vary by terminal type — in UNIX-world, "termcap" can be used to hide this from application).

  Example: programs using `ncurses` library. (I will try to put some on the "useful links" page.)

Slide 3

GUI Hardware and Software — Recap/Review

- Hardware: Keyboard and mouse send very low-level events. Display at one point was fairly low-level, but now often contains its own processors.
- Software: Framework for providing graphical interfaces may be integral to O/S (Windows) or an add-on (UNIX/Linux).

Slide 4

Network Terminals — Hardware

- Keyboard, mouse, and display as described previously, plus local processor; connected to remote system.
- Local processor can be very capable (X terminal, or even PC configured to run as one) or more primitive.

Slide 5

GUI-Based Programming

- Input from keyboard and mouse captured by O/S and turned into messages to process owning appropriate window.
- Typical structure of GUI-based program is a loop to receive and dispatch these messages — "event-driven" style of programming.
- Details vary between Windows and X, but overall idea is similar. See example programs in textbook. (I've also written programs using the fairly low-level X11 interface, but — maybe not. But it's doable, even from C, though of course not completely portable.)

Slide 6

I/O in UNIX/Linux

- Access to devices provided by special files (normally in `/dev/*`), to provide uniform interface for callers. Two categories, block and character. Each defines interface (set of functions) to device driver. Major device number used to locate specific function.
- For block devices, buffer cache contains blocks recently/frequently used.
- For character devices, optional line-discipline layer provides some of what we described for text-terminal keyboard driver.
- Streams provide additional layer of abstraction for callers — can interface to files, terminals, etc. (This is what you access with `*scanf`, `*printf`.)

Slide 7

I/O in Windows

- Hardware Abstraction Layer (HAL) attempts to insulate rest of O/S from some low-level details — e.g., I/O using ports versus memory-mapped I/O.
- Standard interface to device drivers — Windows Driver Model. Drivers are passed I/O Request Packet objects.

Slide 8

"Everything's a File" Revisited

- I mentioned the pseudofilesystem `/proc`? which supposedly you can read/write just as if it were a file?
- I wrote some throwaway code to access "files" within it and learned(?) that while C stream I/O (`fopen`, `fgetc`, etc.) didn't work well, the lower-level routines (`open`, `read`, etc.) did.

Slide 9

Minute Essay

- Anything noteworthy about Homework 6?