Slide 1

Administrivia

- If you haven't turned in a final/working version of Homework 7, don't give up — I'll accept late submissions through Friday at least, and you'll get full or near-full credit if you turned in something plausible earlier.
- There will be a set of extra-credit problems, assigned probably this weekend (I will send mail), to be due toward the end of finals period.

Slide 2

Minute Essay From Last Lecture

- Many people reported that they thought they had learned more about pointers from Homework 7. Many people found it difficult, but a few said "not as bad as I expected". "Hm!"?

  (This is much like what last semester's students said. "Hm!" again?)
- Lots of interest in extra credit.

Slide 3

Full-Screen Text-Mode Programming

- As you know(?), the C standard library defines text input/output via "streams", but these are line-oriented.
- But at least one text-mode programs you've used (`vim`) allows for other kinds of text I/O.
- How do they do that? various ways, none portable in the sense of working with all standard-conforming implementations of C, but the `ncurses` library is fairly standard in UNIXworld.
- (Examples on "sample programs" page.)

Slide 4

Other Useful(?) C Libraries

- `readline` to provide command history and some simple editing. (Probably not standard C but probably widely available. `man readline` for some info.)
- "X11 library" for fairly low-level graphics programs. (I'm not finding a good man-page starting point, but your favorite search engine . . .) Several third-party toolkits that build on it. Nothing completely portable, alas, but remember that C's supposed to be implementable on a very wide range of hardware.

  (Example next time?)

Slide 5

Tools Revisited — Compiling C

- One person asked about "other ways to compile C programs". Not sure what was meant, but . . .
- For non-trivial programs, my guess is that most people use something to automate the build process — makefiles on UNIX-like systems, probably other tools on other platforms, and there are probably some IDEs one could use.
- For small programs, since it's tedious to remember all the useful command-line options, I say either use a makefile (such as the simple one on the "sample programs" page) or a script (an example also under "sample programs").

Slide 6

Tools Revisited — Text Editors

- One person asked about `vim` tips. At this point it may not do you much good, but for future reference maybe . . .
- If mostly you just go into insert mode and type, using the arrow keys to move around and the backspace to delete — well, it's easy to remember but strikes me as doing things the hard way.
- If you ever decide to ramp up your skills — try `vimtutor` again, or the online help (`:help`).

  In particular look for ways of searching and ways of doing cut/copy/paste, and in general `vim`'s notion of "motion commands". You will probably like "visual mode".

Slide 7

Tools Revisited — `vim` Tips

- `%` to find matching parentheses.
- `*` to find next occurrence of "word" under cursor; `n` to find others.
- Re-indentation (tidying things up). == to do the current line; `gg=G` to do the whole file (`gg` says "go to top of file", = says apply the reindent operation, `G` says do that until end of file).

Slide 8

Minute Essay

- None really, unless there are other questions I could answer next time about C or anything else from the class?