Administrivia

- Reminder: Homework 5 due today.

Slide 1

Minute Essay From Last Lecture

- No one's using `make`. Some people are using `gcc -Wall` consistently and finding it useful. Others sometimes forget or use it only if code doesn't work.
- Since with loops it's often useful to also compile with `-std=c99`, it may be time to switch to using `make` . . .

Slide 2

Slide 3

`make` Revisited

- `make` is an old-but-still-useful tool for building programs. It can do a lot of complicated things, but it can also be used just to make it easier to always compile with a selected bunch of flags. To do this:
- Put a copy of `Makefile` from the "sample programs" page in the directory where you compile programs.
- To compile `mypgm.c`, type `make mypgm`. `make` should use `gcc` to compile with a bunch of flags. (If it doesn't, you've done something wrong. Ask.)
- The result is called not `a.out` but `mypgm`. Run it by typing `./mypgm`.

Slide 4

`vim` Tips

- You may have discovered already that if you don't know/remember many of the keyboard shortcuts (and `vim` is pretty much *all* keyboard shortcuts) it's painful to use `vim`. I like text-based editors for this class because they're easy to use remotely. There are others that may be easier to get started with, but . . .
- I think `vim` is a good editor for writing code: It does syntax highlighting of code in any language it "knows about" as well as automatic indentation. (Tidy up indentation by typing == on a line.) It also shows matching parentheses/braces, and if you put the cursor on one of those and press `%` it takes you to the match — or indicates there isn't one. Helpful!
- If you have trouble remembering, try a "cheat sheet" of commands you want to remember.

Slide 5

`vim` Tips, Continued

- Short way to cut/copy/paste: `yy` (“yank”) to copy a line. `dd` to delete a line. Both go into a buffer you can then insert with `p` or `P`. Precede the `yy` or `dd` with a number to get multiple lines. Or . . .
- You will probably like “visual mode”: Put the cursor at the start of text to highlight and press `v`. Move the cursor to the end and then type `y` to copy or `d` to delete, and then use `P` to (re)insert.
- **.** repeats the most recent command (e.g., `dd`).
- You can search for text with `/`. Repeat search with `n`. Use `cw` to change a “word”. Combine with **.** to do a quick repeated search-and-replace.

Slide 6

“Random” Numbers Revisited

- Many situations in computer science where it’s useful to work with a sequence of “random” numbers.
- Examples include simulating physical systems, “Monte Carlo” algorithms. An example is a program to estimate π by simulating throwing darts into a board containing a quarter circle. (Count the number inside the circle and compute the ratio of that to the total number.)
- Often what we want is not something that’s truly unpredictable but something that looks that way and can be reproduced — i.e., we want a “pseudo-random number generator”.

Slide 7

Pseudo-Random Number Generators in C

- C library includes functions `srand()`, `rand()`. `srand()` uses a “seed” to initialize some behind-the-scenes variables, after which you call `rand()` repeatedly to generate a sequence of “random” numbers. If you do this more than once with the same seed you get the same sequence; using different values of the seed gives different sequences.
- (Example — Monte Carlo method for estimating π.)

Slide 8

Character Data

- As mentioned previously, in C we can represent characters as type `char`.
- Simplest way to input/output a single character is with `getchar` and `putchar`. Note that `getchar` returns an `int`; this is so there can be a “special” value `EOF` for “end of file”. (For input from a terminal, signal with something system-dependent, control-D on Linux machines.)
- Functions in `ctype.h` classify characters as alphabetic, digits, etc. Functions `toupper()` and `tolower()` do what their names suggest.

Slide 9

Loops — Another Example

- We could also write some programs that do things with character input. (When we know about files — next topic — this may get more interesting.)
- One example, interesting from a programming perspective, is writing something that "detabifies" input — replaces tab characters with the right number of spaces so that things line up at "tab stops".
- As a warm-up, we could write something that just takes input from the user, as many lines as entered, and echoes it. (How do we know when to stop? signal "end of file" — control-D on Linux/UNIX.)

Slide 10

Minute Essay

- None — quiz.