Slide 1

Administrivia

- About minute essays: In addition to answering whatever questions I ask, feel free to ask me, well, anything (preferably related in some way to the course or computing).
- Homework 1 on the Web, linked from the "lecture topics and assignments" page. Due next Wednesday (11:59pm). A little more about homeworks later.

Slide 2

Programming Basics — Recap

- First step is to understand the problem.
- Next step is to develop an "algorithm" (roughly, a computer-understandable way to solve the problem).
- After that (or really at the same time), express the algorithm in a program ("source code", which we'll create with a text editor).
- Finally, do what you need to do to get the computer to run your program. Usually on the first try you will get "syntax errors" — computers are far pickier than humans. Once you resolve those you can try the program and observe whether it solves the intended problem.

Slide 3

Text Editors

- Many, many text editors, and people have favorites. Notepad is an example from the Windows world.
- I use and will teach in this class `vi`: It's found on every UNIX/Linux system I know of, and is very powerful, though it takes some getting used to. (`vi` on our Linux machines is actually `vim`, a more featureful "clone" of the original `vi`.)
- Other popular Linux text editors include `emacs`, `pico`, and `gedit`. Advice: Give `vi` a real try first, but if using it is just too painful, use something else!

Slide 4

`vi` Basics

- `vi` has two *modes* — insert mode (where what you type goes into the file) and command mode (where you can type commands to copy, move, delete, save, etc.).
- You start an editing session by typing, e.g., `vi hello.txt`. It starts in command mode. Enter insert mode by typing `i`. Exit by pressing `ESC`. Move around with the arrow keys. (Try entering some text.) Backspace deletes a single character.
- Save by typing `:w`; exit by typing `:q`.
- *Highly recommended:* `vimtutor` brings up an interactive tutorial. (Homework 1 asks you to try it.)

Slide 5

More Commands

- Now that we have a way of creating files, we can try out some other basic commands.
- `cat` to show contents of a file. `more` or `less` to show it a screenful at a time.
- `cp` to copy one file to another. `mv` to move or rename a file. For both, `-i` to warn about overwrites.
- `rm` to delete a file. (Note — no recycle bin, so use with caution! or `-i` to prompt.) For this and `cp` and `mv`, `-v` shows what's being done.
- Other useful/interesting commands in video lectures and next time.

Slide 6

UNIX Filesystem Basics

- Unlike in Windows (and Mac, usually), UNIX filesystems are case-sensitive (so hello and Hello are different files).
- Files have two levels of ownership — "owner" (user) and "group". Groups allow sharing files with some but not all users.
- File access is controlled by "permissions". Three levels (owner, group, and everyone else), three types of access (read, write, execute).
- `ls -l` shows permissions. `chmod` changes them.

Slide 7

Useful Command-Line Tips

- The shell (the application that's processing what you type) keeps a history of commands you've recently typed. Up and down arrows let you cycle through this history and reuse commands. You can even search the history (by pressing ctrl-r and then typing the text to search for).
- The shell offers "tab completion" for filenames — if you type part of a filename and press the tab key, it will try to complete it.
- Together these can save you a lot of typing!
- Pedantic aside: "The shell" here means the one you're most likely to be using. There are other programs with similar functionality you could use instead.
- If you have trouble remembering the commands (which you likely will at first!): In times past beginners got paper "cheat sheets" of commonly-used commands. Maybe make yourself an electronic equivalent?

Slide 8

`vim` Tips

- Biggest hurdle may be the notion of modes. (But you already know about this, sort of? Word processors have insert/overwrite modes.)
- Cut/copy/paste basics:

  `dd` cuts a whole line. `yy` copies a whole line.

  `p` pastes after the current line. `P` pastes before the current line.
- Search by typing `/`, text to search for, Enter. Repeat search with `n`. Search-and-replace using this, `cw`, and `.`

Slide 9

More `vim` Tips

- `:help` brings up online help. `:help visual-mode` describes one feature you may like.
- `u` to undo. `:w` ("write") to save. `:q` to exit. `:q!` to exit without saving.

Slide 10

`vim` Tips — Errors/Mistakes

- If you type just `q` rather than `:q`, `vim` thinks you want to record a macro. Screen will show "recording". Press `q` to make it stop.
- If you type `q:` rather than `:q`, `vim` thinks you want it to display a history of commands and shows them to you in a subwindow. Type `:q` to make that go away.
- If you want to copy-and-paste text using window manager, `:set paste` first to avoid annoying indentation behavior. `:set nopaste` after.

Slide 11

`vim` Tips — Errors/Mistakes, Continued

- If you just close the terminal window when running `vim`, that "crashes" `vim`. So what? Well . . .
- `vim` creates a hidden file that saves information that can help with recovery if it crashes. Deleted on normal exit, otherwise not. And then next time you start `vim` on that file — screenful of messages starting "ATTENTION" and "Found a swap file" and finally asking you whether you want to open it anyway or what. If you respond `R` `vim` will try to recover unsaved changes; otherwise not. To actually delete this hidden file, so you don't get that same screenful of messages next time, respond `D`.

Slide 12

Homeworks

- For each homework you will be asked to turn some files (usually source code, though not for the first assignment), by e-mail, with each file as an attachment. In the assignment I say to send them to me at `cs.trinity.edu`, or you can use my TMail address.
- If you're working in one of our labs this should be easy. If you're working remotely you can try my `mail-files` script, available on the "sample programs" page. This mails from the command line.

Slide 13

Homeworks, Continued

- You will also be asked for two more things, which you can put in the body of the e-mail or in another file you send me (might be helpful if using `mail-files` script):
  - An explicit honor-code pledge and a statement about any collaboration or help (if none, say so).
  - A sentence or two reflecting on what if anything was noteworthy about the assignment.

Slide 14

Minute Essay

- If you've used a command-line environment, how does this one compare to what you've used before?
- Anything surprising or startling today?