Slide 1

Administrivia

- Syllabus updated with information about textbook. Bookstore does *not* have copies (yet?) I am exploring options for quickly getting paper copies for those who want them. Don't order one yourself without checking with me.
- Most of "course Web site" filled in now. Meant to answer many of your administrative questions.
- Everyone already had an account on our computers set up (i.e., none were new as of this semester). If you did not already have your password reset, let's try to sort that out today. (Why didn't I do this last time — ITS kind of prefers that you go through them, but in a pinch I can do it.)

Slide 2

More Administrivia

- For minute essays — no Word files please (unless there's a compelling reason for them).
- Also, if you have an urgent question, please put "urgent" in the subject line, since I routinely set these messages aside to look at later.

Slide 3

Minute Essay From Last Lecture

- Almost everyone has had exposure to Matlab.
- Several have had previous experience with more-traditional programming languages too, but many have not.
- Almost everyone has had a fair amount of math. (Engineers!)
- But almost no one claims too much experience with command-line environments or Linux. (So after a little more intro stuff we'll do that first.)

Slide 4

Solving Problems on Computers

- Appearances (maybe?) to the contrary, computers are not smart. What they do well is perform sequences of simple math/logic operations very fast and very accurately.
- What makes them useful is that people have figured out how to break complicated tasks down into sequences of simple operations — i.e., how to "program" them.
- This requires a mindset not quite like that required for any other activity — and can involve a lot of creativity.

Slide 5

Steps in Solving Problems on a Computer

- Understand the problem — what do you want the computer to do, exactly?
- Design a solution suitable for a computer ("develop an algorithm").
- Implement the solution ("write the program"). This requires expressing your ideas in "a programming language" — of which there are many! Programming languages are similar to human languages in some ways, different in others; they are meant to be (somewhat!) human-readable while still being precise enough for a computer to understand.
- Test your solution. This will involve the use of some tool that translates what you write ("source code") into something the computer hardware can work with.

Slide 6

Solving Problems on a Computer, Continued

- The overall process — understand the problem, develop and test a solution — is mostly independent of the choice of programming language and platform (combination of hardware and operating system, roughly). So once you understand the principles it is relatively easy to learn new languages.
- Opinions about which language to learn first, and on what platform, vary. For this course we will use C, largely at the request of the folks in Engineering Science. It's not as easy to use as some other choices but is more widely used (especially in so-called "embedded systems") and is closer to the hardware. We will also do most work from the command line under Linux.

Slide 7

Programming Basics

- What computers actually execute is *machine language* — binary numbers each representing one primitive operation. Once upon a time, people programmed by writing machine language (!).
- Obviously that was tedious and error-prone. A very early bright idea — write something more human-readable (*source code*) and *have the computer translate it*. Useful even if the source code is just a human-readable version of the primitive operations (*assembler language*). Even better if the source code is less primitive (*high-level language*).
- Source code is simply plain text (as opposed to text plus formatting, as in a word-processor document). Since the hardware doesn't understand it, however, . . .

Slide 8

Programming Basics, Continued

- Source code can be *interpreted* — translated line by line into something the hardware can understand, by another program called an *interpreter*.

  (This is how "scripting languages" work. An example is the command shell's language. !)
- Or it can be *compiled* — translated by a program called a *compiler* into something the hardware can execute directly.

  (This is how traditional "high-level" languages such as C and Fortran work.)
- Or it can be compiled into some intermediate form that can be executed by another program.

  (This is how some recent languages such as Java work.)

Slide 9

Writing Source Code

- How do you get source code? If using an interpreter, you can just type it in. If you want something you can keep and reuse, however, you need a tool that will do that.
- Simplest way to create source code is with a *text editor* — a program for writing and editing plain text. This is what we will do for now.
- (Another way is to use an *IDE* (Interactive Development Environment). We will try one of these later in the semester.)

Slide 10

A Word About Tools

- In this class we use Linux and command-line tools because we believe it is important for budding computer scientists to know how to work with these tools.

  For others — exposure to something new and different?
- (What is Linux? it's an operating system, as Windows and Mac OS X are operating systems. It's one of a family of operating systems descended from UNIX, developed at Bell Labs in the early 1970s. A lot of servers run Linux or some other UNIX-like system. There are also ongoing efforts to develop mainstream desktop systems.)
- A UNIX person's response to claims that UNIX isn't user-friendly: "Sure it is. It's just choosy about its friends."

Slide 11

Getting Started with Linux

- When you log in, you should get a graphical desktop, which should be navigable with what you know from using other graphical environments (though some details are different).
- In Linux, we talk about files and directories; the idea is the same as Windows' files and folders, though again some details are different.
- The graphical system should give you a way to get a terminal window. Once you have that . . . (to be continued)

Slide 12

More About Tools / Administrivia

- Eventually your TigerCard will let you into any of our classroom/lab rooms (CSI 257, 388, and 488). All these machines are dual-boot (Windows 7 and Linux). It may take a few days to get that set up, though.
- In the meantime — since I strongly encourage you to use our machines for trying things out and for doing your homework — you can get the equivalent of terminal window by using ITS's VDI interface and PuTTY. I will e-mail details (since these slides are public).

Minute Essay

- Anything today that was particularly unclear?