Slide 1

Administrivia

- One purpose of the syllabus is to spell out policies (next slides).
- Most other information will be on the Web, either on my home page (here, office hours) or the course Web page (here).

  A request: If you spot something wrong with course material on the Web, please let me know!

Slide 2

Course FAQ

- "What will my grade be based on?" (See syllabus.)
- "What happens if I can't turn in work on time, or I miss a class?" (See syllabus.)
- "What's your policy on collaboration?" (See syllabus.)

Slide 3

Course FAQ, Continued

- "When is the next homework due?" (See "Lecture topics and assignments" page.)
- "When are your office hours?" (See my home page.)

  Note that part of my job is to answer your questions outside class, so if you need help, please ask! in person or by e-mail.
- "Do I have to buy the textbook?"

  No, but it's not a bad idea; we can only cover so much in class, and the alternative reading (a short-ish online tutorial) is also somewhat limited. (Be advised, also, that while there is good information about C to be found in other books and Web sites, *there is also a significant amount of not-so-good information*. So use caution?)

Slide 4

Course FAQ, Continued

- "What computer(s) can I use to do homework?"

  Easiest option may be department's Linux machines. You should have physical access via your TigerCard to all the classrooms and labs (probably not today but soon) any time the building is open. You should also be able to log in remotely to any that are booted into Linux, or to a cluster of Linux-only machines in ITS's server room (names `diasnn`, where `nn` ranges from 01 to 05). To log in from off-campus, we are currently recommending that you use ITS's VDI.

Slide 5

What Is This Course About?

- Back story: Primary goal of our traditional first course (CSCI 1320) is to introduce students to programming and algorithmic problem-solving. Another goal of the course as taught some years ago, however, was to expose students to certain low-level concepts that contribute to a well-rounded education in computer science. Students coming into the major via other routes often did not get this exposure and struggled in later courses.
- CSCI 1120 was added to the curriculum as a way to address this problem — i.e. to cover the parts of CSCI 1320 that might not be covered by alternative introductory courses. A few years ago we switched to a more-abstract language for CSCI 1320, and at that point this course became required for all students.

Slide 6

Course Topics

- Basic C programming, for people who already know how to write programs in some other language.
- (Review of) the Linux/UNIX command-line environment and command-line development tools.
- (Review of) basics of computer arithmetic and data representation.
- More-advanced topics as time permits.

Slide 7

Why Learn C? (For Scala/Java/Python Programmers)

- Scala and Python (and Java, less so) provide a programming environment that's nice in many ways — lots of safety checks, nice features, extensive standard library. But they hide a lot about how hardware actually works.
- C, in contrast, has been called "high-level assembly language" — so it seems primitive in some ways compared to many other languages. What you get (we think!) in return for the annoyances is more understanding of hardware — and if you do low-level work (e.g., operating systems, embedded systems), it may well be in C.

Slide 8

First Things First(?) — Text Editors

- In class I will use `vim` to write programs. I don't insist that you use it too, but it's a good tool for this job, and if you aren't very good with it, no time like the present to get better with it.

  To encourage you, see the first homework.
- (Indeed, I think this class is a good time to get more practice with the command line in general; it's in keeping with the spirit of the course, and you have an instructor who knows it pretty well.)

Slide 9

A Very Short Introduction to C

- As you read (or skim) sections of the textbook you may want to try running the programs yourself. More about all of this next time, but today let's do the "hello world" program . . .
- (Aside: The tradition of having one's first program in a language print "hello, world" was started by the inventors of C.)

Slide 10

A Very Short Introduction to C, Continued

- First write the program using a text editor (e.g., `vim`) and save it with a name ending in `.c` (say `hello.c`). (See the "sample programs" Web page for what it looks like.)
- Next, compile the program (turn it into something the computer can execute). Simplest command for that:

  ```
  gcc hello.c
  ```

  If there are no syntax or other errors, this produces an "executable" file `a.out`.
- Run the program by typing `a.out` at the command prompt. (If that doesn't work, try `./a.out`.)

Slide 11

Minute Essay

- (Most lectures will end with a "minute essay" — as a quick check on your understanding, a way for me to get some information, etc., and also to track attendance. Just put your answer in the body of the message; no Word documents please, and put "minute essay" and the course in the Subject line.)
- Tell me about your background: If you took CSCI 1320 at Trinity, who was your instructor? Do you have other programming experience?
- What are your goals for this course? Anything else you want to tell me?
- Don't forget the reading and homework for next time . . .