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 or phone.

Slide 4

Course FAQ, Continued

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

  Easiest option may be department’s Linux lab machines. There are others.

  You should have physical access (via your TigerCard) to five rooms containing such machines any time the building is open. You should have remote access to any that are booted into Linux.

  Returning students should already have accounts set up. (If you’ve forgotten your password, go to the ITS help desk and ask for it to be reset.) To change your password, open a terminal window and type `passwd`.

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 up to this year, 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. With the recent shift in language(s) used in CSCI 1320, it is 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.
- More advanced topics as time permits.

Slide 7

Why Learn C? (For Java/Python/Scala 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

Getting Started with Linux (Review)

- (A UNIX person's response to claims that UNIX isn't user friendly: "Sure it is. It's just choosy about its friends.")
- The graphical system should give you a way to get a terminal window, which is what we will use a lot in this class (in keeping with the title!). In theory you know the basics from CSCI 1320. If not, review the relevant chapter of the book Dr. Lewis is writing for POP I/II.

Slide 9

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.

  (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.)
- 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.
- To learn more about command `foo`, type `man foo`. (This also works with C library routines — more about them later.) This is reference information rather than a tutorial, but usually very complete.

Slide 10

Text Editors

- Many, many text editors, and people have favorites. 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 capable "clone" of the original `vi`.) Other popular Linux text editors include `emacs`, `pico`, and various graphical editors that come with "desktop environments" such as GNOME and KDE.
- Tip: If you're struggling with whatever editor you previously used, either spend a little time learning its features, or choose another one! `vim` has `vimtutor`. `emacs` also has built-in tutorial.

Minute Essay

- Tell me about your background: What programming classes have you taken (at Trinity or elsewhere) and what language(s) did you use?
- What are your goals for this course? Anything else you want to tell me?