Slide 1

Administrivia

- (Most lectures will start with some “administrivia”.)
- One purpose of the syllabus is to spell out policies (more about that shortly).
- 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

What Is This Course About?

- It’s an introductory course in programming, with a focus on problem-solving and logic. (It also includes an introduction to Linux and some of its command-line tools, though that’s somewhat secondary.)
- “Programming” — ? solving problems with computers, which requires expressing ideas in a way the computer can understand.

Slide 3

Who Should Take This Course?

- Computer science majors who do not have credit for a similar course.
- Students majoring in subjects for which an understanding of programming is helpful or even necessary.
- Students who want to meet the "Using Scientific Methods" common-curriculum requirement by learning programming.
- Anyone else who wants to understand something about how computers — which more and more are everywhere — do what they do!
- No background in programming is assumed — some students have some, but others do not. (Be prepared to spend some time on homeworks, however. In the words of colleague Dr. Maury Eggen: Programming is not a spectator sport. But it can be fun!)

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.
- It also involves a form of experimentation (which is why this course meets the CC requirement it does).

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. Right now different sections of this course use different languages but the same platform (Linux). In this section we will use Scala; it is somewhat easier for beginners than some of the other choices but also powerful enough to write interesting programs.

Slide 7

Course FAQ

- "What will my grade be based on?" (See syllabus.)
- "When are the exams?" (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 8

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 9

Course FAQ, Continued

- "Why does the bookstore not have a book for this course?"

  (We will be using one that my colleague Dr. Lewis is writing. A current draft is available online (from the Trinity network only). Later chapters may change as the semester goes on.)

  (The first chapter may seem slightly intimidating, with the description of hardware. Don't panic!)

Slide 10

Classroom/Lab Machines

- Trinity's ITS department provides computing facilities for general use. We maintain our own set of computers tailored to the needs of our department (courses and research). Probably the easiest (though not the only) option for doing the assignments is to use these machines.
- To access these computers you need an "account" separate from your main Trinity account . . .

Slide 11

Classroom/Lab Machines, Continued

- Students who have previously taken a CSCI course 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.)
- Accounts have been set up for students who have not taken a CSCI course before. Username is the same as your Windows/ITS username; password has been sent to your Trinity e-mail address.
- We will start using these accounts in the next class, or you're welcome to try them now. The command-line way to change your password is to open a terminal window and type `passwd`.

Slide 12

Classroom/Lab Machines, Continued

- Most of the department's computers live in three classrooms (HAS 227, HAS 228, HAS 340) and two labs (HAS 200, HAS 329). (The others are servers, in ITS's server room.)

  You should have physical access (via your TigerCard) to all of the classrooms and labs any time the building is open. HAS 340 is Linux-only, but the machines in the other rooms dual-boot Linux and Windows.
- You can also access of these machines from other computers on campus (we will talk later about how), provided the computer you want to access is running Linux.

Slide 13

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.)
- Tell me about why you are taking this course — as a prospective CS major? for another major (what)? to meet a CC requirement? what is your major?
- Tell me about your background:

  Have you had any exposure to programming? If so, in what language?

  Have you had any exposure to a Linux/Unix command-line interface?
- What are your goals for this course? Anything else you want to tell me?
- (Don't forget the reading . . .)