Slide 1

Administrivia

- One purpose of the syllabus is to spell out policies, especially about:
  - Course requirements and grading.
  - Exam dates.
  - Late work.
  - Academic integrity.
- 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

More Administrivia

- 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.
- Some of my office hours are designated as “open lab”. At those times I will be in one of the classrooms/labs ready to answer questions.

Slide 3

More Administrivia

- Probably the easiest (though not the only) option for doing the assignments is to use the Linux lab machines.
- You should have physical access (via your TigerCard) to four 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.) Accounts have been set up for new students. Username is the same as your Windows/ITS username; password has been mailed to you. To change it, open a terminal window and type `yppasswd`.

Slide 4

What Is This Course About?

- Back story: The primary goal of our traditional first course is to introduce students to programming and algorithmic problem-solving. Another goal of the course, however, is to expose students to certain low-level concepts that contribute to a well-rounded education in computer science. Students who come into the major via other routes may not get this exposure, and they are apt to struggle in later courses.
- CSCI 1120 is a new course intended to cover only the parts of CSCI 1320 that might not be covered by alternative introductory courses.

Slide 5

Course Topics

- Basic C programming, for people who already know how to write programs in some other language (such as Java).
- The Linux/UNIX command-line environment and command-line development tools.
- Basics of computer arithmetic.
- More advanced topis as time permits.

Slide 6

Getting Started with Linux

- (A UNIX person's response to claims that UNIX isn't user friendly: "Sure it is. It's just choosy about its friends.")
- 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 . . .

Slide 7

Getting Started With the Command Line

- What you get when you start a terminal window is a "command shell", similar to Windows' "MS-DOS prompt". Rather than pointing and clicking, you type the name of the program you want to run, plus whatever arguments (parameters) it needs.
- Let's try some:
  - `pwd` shows the current directory.
  - `ls` lists the current directory. Add `-l` to get more information.
  - `cd foo` changes to directory `foo`. Just `cd` goes back to your home directory. Try `cd Local` and then `ls`.

Slide 8

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 9

Text Editors

- "Programming" usually means writing "source code". (More later about how this relates to what the machine can actually execute.) How do you get source code? The simplest way is to create it with a *text editor* — a program for writing and editing plain text.
- 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 a little 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.

Slide 10

`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 example.txt`. It starts in command mode. Enter insert mode by typing `i`. Exit by pressing `ESC`. Move around with the arrow keys. Delete a single character with `x`. (Try entering some text.)
- Save and exit by typing `:wq`.
- *Highly recommended:* `vimtutor` brings up an interactive tutorial.

Slide 11

More Commands

- Now that we have at least one file, we can try out some other basic commands.
- `cp` to copy one file to another.
- `mv` to move or rename a file.
- `rm` to delete a file. (Note — no recycle bin, so use with caution.)

Slide 12

Minute Essay

- Tell me about your background:

  What programming classes have you taken (high school or other), and what language(s) did you use?

  Have you had any exposure to a Linux/UNIX command-line interface? Linux/Unix command-line interface?
- What are your goals for this course? Anything else you want to tell me?