Slide 1

Administrivia

- Homework 1 on Web; due next Tuesday. (Should be fairly easy — practice with tool(s).)
- A request: You will turn in most if not all work for this course by e-mail. Please include the name or number of the course in the subject line of your message, plus something about which assignment it is, to help me get it into the correct folder for grading.

Slide 2

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.”

Getting Started with Linux

Slide 3

- 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 . . .

Getting Started With the Command Line

Slide 4

- 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.
- (Why would you want to use a command line? because for some things it's arguably more efficient, and it's "scriptable" in ways that GUIs typically aren't.)
- Let's try some commands . . . (Don't worry if this goes by quickly — you should plan anyway to spend some time outside class trying out what we do in class and what's in the reading.)

Slide 5

Some Commands

- `pwd` shows the current directory. (When you give a filename, it's relative to this directory unless you give a full pathname.)
- `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`.
- `passwd` changes your password. (Not a command you'll want often, but probably now!)

Slide 6

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 is reference information rather than a tutorial, but usually very complete. `man -k foo` will give you a list of commands having something to do with `foo`.

Slide 7

Remote Access

- One of the strengths of a command-line enviroment is that it works well for “remote access” (using the computer when you aren’t sitting in front of it).
- To do this from another UNIX-like computer, use `ssh`. `scp` and `sftp` can be used to copy files.
- From a Windows computer, install either Cygwin (UNIX-like toolkit) or PuTTY (terminal emulator).
- More details in chapter 2 of online book and on the course Web site (“Useful links” to “More useful links for 1320/1321” to “More about remote access”).

Slide 8

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 9

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 10

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 11

Text Editors

- 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 `gedit`.

Slide 12

`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 hello.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. (Homework 1 asks you to try it.)

More Commands

Slide 13

- Now that we have a way of creating files, we can try out some other basic commands.
- `cat` to show contents of a file. `more` or `less` to show it a screenful at a time.
- `cp` to copy one file to another. `-i` to warn about overwrites.
- `mv` to move or rename a file. `-i` to warn about overwrites.
- `rm` to delete a file. (Note — no recycle bin, so use with caution! or `-i` to prompt.)

Minute Essay

Slide 14

- Did anything today seem particularly unclear? (What?)