Slide 1

## Administrivia

- Notice that readings are assigned via the "Lecture topics and assignments" page.
- Homework 1 to be on Web (also linked from the "Lecture topics and assignments" page) soon.
- *Please* do not reboot the machines in HAS 340; people rely on their being available for remote access.

  Also be careful not to inadvertently shut them down when trying to log off.

  If a previous user has left the machine's screen locked, you can use use control-alt-backspace to restart the graphical subsystem.

Slide 2

## 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 3**

## 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 4**

## 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 5

## `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 6

## 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 7

## File Permissions in UNIX/Linux

- Access to files specified in terms of three categories of users (owner, group, and other) and three kinds of access (read, write, and execute).
- To show permissions, `ls -l`. First character says directory/not, then three groups of three letters each (rwx), one for each category of user.
- To change permissions, `chmod`. Can specify via octal (base 8) numbers, but usually easier to use symbolic mode. Examples:

  `chmod go= foo` to say only owner can access `foo`.

  `chmod go+r foo` to say everyone can read `foo` (but not necessarily write it).

Slide 8

## Minute Essay

- None — sign in.