Slide 1

Administrivia

- Reading Quiz 2 posted. Due in a week. Very short. Homework coming soon I hope!
- A few words about the reading: I had trouble finding readings that provide the right level of detail; often what's listed is reference material that goes into more detail than I'd like. So it's AOK to skim!
- If you're curious about what a shell program might look like inside — look at Homework 1b for CSCI 3323. (I ask the students to fill in some blanks of a very simple one.)

Slide 2

`bash` In-Place Editing of Command Line

- As mentioned last time, `bash` provides several ways to edit a command being typed. Many key bindings based on `emacs`. (Can customize for vi bindings too.)
- One more: ctrl-w deletes the last "word"; ctrl-y pulls it back in. Very useful as a way of duplicating text!

Slide 3

Shell Customizations

- At startup, shell reads in various configuration files (see `man` page for details, under `INVOCATION`). At least one will be in your home directory. For `bash`, `.bashrc` is read for all shells and `.bash_profile` when it's a "login shell" (e.g., `ssh` session, but not terminal window).
- Default `.bashrc` file on our systems reads (or more properly, "sources" — more about that soon) `/etc/bashrc`. Somewhat complicated, but eventually reads files in `/etc/profile.d`. Allows sites to do site-wide customizations. Appears to be somewhat standard practice for Linux, at least the distributions I know.

Slide 4

Shell Customizations — User-Defined Files

- In these files, you can do many things:
- Define/redefine environment variables.
- Set various shell options and variables.
- Define aliases/functions.
- Invoke other commands (e.g., `umask` to set default file permissions, or `module load` (later).

Slide 5

Environment Variables

- Some we've mentioned already (e.g, `PATH`). Others we haven't (e.g., `PS1`).
- For `bash`, be sure to `export` them so they're available to called programs.
- Can also define new ones (I find this useful).

Slide 6

Shell Options and Variables

- `set` and `shopt` let you set various shell variables and options.
- Details in `man` page or manual, but some I find useful:

```
set -o noclobber
set -o ignoreeof
shopt -s histappend
```

Slide 7

Shell Aliases and Functions (`bash`)

- Aliases are simple substitution, no parameters. Examples:

```
alias lt='ls -ltF'
```

- Functions can have positional parameters. Examples:

```
cd-and-show() { cd $1 ; pwd ; ls; }
```

Slide 8

I/O Redirection

- In programming classes I talk about "reading from standard input" (`stdin`) rather than "reading from the keyboard", and "writing to standard output" rather than "writing to the screen". Why?

Slide 9

I/O Redirection, Continued

- `stdin` (standard input) can come from keyboard, file, or inline in shell script.
- `stdout` and `stderr` (standard output, error) can go to terminal or file (overwrite or append), separately or together.
- Syntax depends in part on which shell you're using. All based on idea of associating a number with each I/O stream — 0 for `stdin`, 1 for `stdout`, 2 for `stderr`. Target of redirection can be another stream.

  `bash` examples:

  ```
  ls >out 2>err
  ls >out 2>&1
  ```

Slide 10

I/O Redirection, Continued

- How is this useful? (e.g., in program development? testing?)
- *OR* — remember quotation from first class?

  "Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."

Slide 11

Pipes

- "Pipes" provide one-way communication between programs — output of program A becomes input of program B.
- Key component of "the UNIX philosophy" — emphasis on providing a toolkit of small programs, mechanisms for combining them.
- "Filters" are programs designed to work this way, and there are lots of them (next time). `less` and `more` also useful.

Slide 12

Minute Essay

- Have you made changes to your `.bashrc`, perhaps for another class? If so, what?